Model Counting and Sampling via Semiring Extensions

Andreas Goral, Joachim Giesen, Mark Blacher, Christoph Staudt, Julien Klaus

Friedrich Schiller University Jena, Germany
{andreas.goral,joachim.giesen,mark.blacher,christoph.staudt,julien.klaus}@uni-jena.de

Abstract

Many decision and optimization problems have natural extensions as counting problems. The best known example is the Boolean satisfiability problem (SAT), where we want to count the satisfying assignments of truth values to the variables, which is known as the #SAT problem. Likewise, for discrete optimization problems, we want to count the states on which the objective function attains the optimal value. Both SAT and discrete optimization can be formulated as selective marginalize a product function (MPF) queries. Here, we show how general selective MPF queries can be extended for model counting. MPF queries are encoded as tensor hypernetworks over suitable semirings that can be solved by generic tensor hypernetwork contraction algorithms. Our model counting extension is again an MPF query, on an extended semiring, that can be solved by the same contraction algorithms. Model counting is required for uniform model sampling. We show how the counting extension can be further extended for model sampling by constructing yet another semiring. We have implemented the model counting and sampling extensions. Experiments show that our generic approach is competitive with the state of the art in model counting and model sampling.

Introduction

The marginalize a product function (MPF) framework was formally introduced by Aji and McEliece (2000) who had observed that surprisingly many applications in signal processing (Kalman 1960; Viterbi 1967; Kschischang and Frey 1998; Aji 2000), probabilistic and causal inference (Baum 1972; Pearl 1993), and natural language processing (Sutton and McCallum 2012) fall within the same algorithmic message passing framework. The MPF framework is also known as algebraic model counting (Kimmig, Van den Broeck, and De Raedt 2017).

The prototypical MPF query is computing marginals of discrete Markov random fields. A Markov random field is a multivariate probability distribution p on a discrete sample space that factors as

$$p(x_1, \ldots, x_n) = \prod_{j=1}^{m} q_j(x_{|J}),$$

where $x_{|J}$ is the projection of $(x_1, \ldots, x_n)$ onto its elements indexed by $J \subset [n] = \{1, \ldots, n\}$. The marginal of p on the index set I is given by the following sum over a product of functions,

$$p_I(x_{|I}) = \sum_{x' : x'_{|I} = x_{|I}} \prod_{j=1}^{m} q_j\big(x'_{|J}\big).$$

A different, but closely related type of inference query asks for the probability of a most likely element in the sample space, that is,

$$\max_x \prod_{j=1}^{m} q_j(x_{|J}),$$

which, as it turns out, is also an MPF query, when we replace additions by maximizations. In the MPF abstraction, computations are on semirings $(S, \oplus, \otimes)$, which generalize the standard sum-product semiring $(\mathbb{R}, +, \cdot)$. Here, the standard sum-product semiring is used for computing marginals, and the maximization query is using the Viterbi semiring $\big([0, 1], \max, \cdot\big)$.

There are, however, problems that can not be directly solved by MPF queries. For instance, the maximization query for Markov random fields is often not the query we are most interested in. Instead of the maximum value we typically want to get a *model*, that is, an element x from the sample space that maximizes the function. Therefore, we want to answer the query

$$\text{argmax}_x \prod_{j=1}^{m} q_j(x_{|J}),$$

which is not an MPF query. For solving this query, the algorithms for answering MPF queries need to be adapted. The adaptions exploit the fact that the $\max$-operation is *selective*, that is, it always returns one of its arguments.

Here, we show that adapting the algorithms can be avoided. We present a general framework for lifting MPF queries over selective semirings into MPF queries for model counting and model sampling. The framework is sketched in Figure 1. It builds on a construction scheme for semiring extensions that combines a selective semiring on which we want to serve an MPF query with a second semiring that

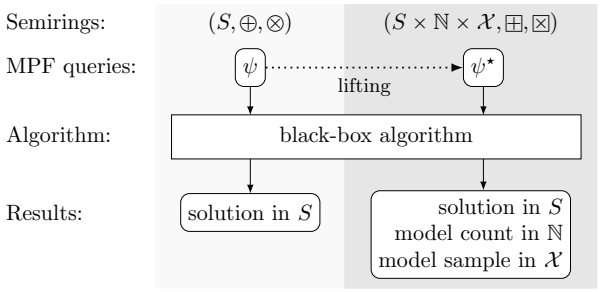

Figure 1: A generic framework for lifting MPF queries ψ over selective semirings $(S, \oplus, \otimes)$ into MPF queries $\psi^\star$ for model counting and sampling. The lifted queries can be answered by the same black box algorithm that is used for answering the original MPF query. Only the semiring S needs to be replaced by the extended semiring $(S \times \mathbb{N} \times \mathcal{X}, \boxplus, \boxtimes)$.

is used to track additional information like sets of models or numbers of models. Model counting and sampling then themselves become MPF queries over extended semirings. Therefore, the algorithms for answering MPF queries can also be used for model counting and sampling without changes. The (black-box) algorithms just need to be instantiated with the appropriate semiring.

For the experimental evaluation of our generic model counting and sampling framework, we have implemented the semiring extension and used it together with a vanilla tensor network contraction algorithm. With this implementation we can improve the virtual best solver on the actively researched model counting and sampling problems for the Boolean satisfiability problem.

Related work. Generalizations of the MPF framework such as *functional aggregate queries* (FAQ) and *aggregations and joins over annotated relations* (AJAR), which use more than one aggregation operation, have been discussed by Khamis, Ngo, and Rudra (2016) and Joglekar, Puttagunta, and Ré (2016), respectively. Our selective semiring extension can be used also with selective aggregation operations in these frameworks. Other frameworks that are parameterized by semirings are *path problems in networks* (Baras and Theodorakopoulos 2010) and *semiring-based constraint satisfaction problems* (Bistarelli, Montanari, and Rossi 1997; Kohlas and Wilson 2008). Both frameworks can also be used with our selective semiring extension, when they are instantiated with a selective semiring, such as the min-norm semiring (Sanmartín, Damrich, and Hamprecht 2022) for path problems.

MPF Queries

We need the following notation to formally introduce MPF queries. Throughout this article, D is a tuple $(d_1, \ldots, d_n)$ of natural numbers, $[d_i]$ denotes the set $\{1, \ldots, d_i\}$, and $[D]$ denotes the Cartesian product $[d_1] \times \cdots \times [d_n]$. For a subset $J \subseteq [n]$, D_J is the projection of D onto its components indexed by J. MPF queries are performed on discrete functions $f : [D] \to S$, where $(S, \oplus, \otimes)$ is a commutative semiring. We call $\oplus$ the *aggregation* and $\otimes$ the *combination* operation on S. A formal definition of commutative semirings is included in the supplemental material. For brevity, in the following, semiring always means commutative semiring.

Definition 1 (MPF Query (Aji and McEliece 2000))**.** *Let $f : [D] \to S$ be a function that decomposes as*

$$f(x) = \bigotimes_{j=1}^{m} f_j(x),$$

where each f_j is a function defined on a projection $[D_J]$ of $[D]$ and x is implicitly projected to $x_{|J}$. For a non-empty subset $I \subseteq [n]$ and $y \in [D_{[n]\setminus I}]$, a marginalize a product function *(MPF) query asks to compute the aggregation*

$$\psi(y) = \bigoplus_{x \in [D_I]} \bigotimes_{j=1}^{m} f_j(y, x).$$

Here, we only consider the case $I = [n]$, which means that the result $\psi = \bigoplus_x \bigotimes_{j=1}^m f_j(x)$ of the MPF query is a scalar, that is, an element in S. We also assume that the functions f_j have unique domains, which is not a restriction, because if two functions have the same domain, then we can just replace them by their combination. Furthermore, we only consider *selective* semirings. The aggregation operation $\oplus$ in selective semirings always satisfies

$$s_1 \oplus s_2 \in \{s_1, s_2\}.$$

For MPF queries on selective semirings, not only the query result ψ is of interest, but also the elements in the domain $[D]$ on which ψ attains this value. These elements are called *models*. Selective MPF queries can have more than one model, which also makes model counting and model sampling interesting problems.

Formally, given a selective MPF query ψ, the *model count* of ψ is given as

$$\mathrm{MC}(\psi) = \sum_x \mathbb{1}\left[\bigotimes_{j=1}^{m} f_j(x) = \bigoplus_{x'} \bigotimes_{j=1}^{m} f_j(x')\right],$$

where the symbol $\mathbb{1}[x]$ denotes the indicator function, which evaluates to 1 if x is a true statement, and to 0 otherwise.

It is instructive to consider the example of model counting for the Boolean satisfiability problem (SAT). A SAT instance $\varphi = \bigwedge_{j=1}^m \varphi_j$ in conjunctive normal form (CNF) is a conjunction of clauses φ_j, which are disjunctions of literals. A literal is a Boolean variable or its negation. Any CNF problem can be phrased as an MPF query over the Boolean semiring, where $S = \{0, 1\}$, $\oplus := \vee$, and $\otimes := \wedge$, as follows

$$\varphi = \bigoplus_x \bigotimes_{j=1}^{m} \mathbb{1}\left[\varphi_j(x)\right] := \bigvee_x \bigwedge_{j=1}^{m} \varphi_j(x),$$

In this special case, the model counting problem can be directly formulated as an MPF query itself (Khamis, Ngo, and Rudra 2016), namely, as

$$\mathrm{MC}(\psi) = \sum_x \prod_{j=1}^{m} \mathbb{1}\left[\varphi_j(x)\right]$$

over the semiring $(\mathbb{N}, +, \cdot)$. This works, because there is only one non-zero element, namely, 1, in the Boolean semiring. In general, however, when there is more than one non-zero element, this construction does not work anymore. A proof is included in the supplemental material.

In this article, we show how to formulate model counting problems and model sampling problems for general selective MPF queries again as MPF queries. For doing so, we introduce a framework for non-trivially combining a selective semiring with any other semiring into a new semiring. We call the framework the *selective semiring extension*.

Selective Semiring Extension

The idea behind the selective semiring extension framework is to combine a selective semiring on which we want to serve MPF queries with a second semiring that is used to track additional information like partial models or the number of partial models. To that end, the first semiring needs to share information with the second semiring. We implement this information sharing in the aggregation operation of the combined semiring.

Definition 2 (Selective Semiring Extension)**.** *For a selective semiring $(S, \oplus_{\mathrm{S}}, \otimes_{\mathrm{S}})$ and a semiring $(R, \oplus_{\mathrm{R}}, \otimes_{\mathrm{R}})$ with neutral elements $0_{\mathrm{R}}, 1_{\mathrm{R}}$, the selective semiring extension defines the following combination operation $\boxtimes$ on the Cartesian product $S \times R$,*

$$(s_1, r_1) \boxtimes (s_2, r_2) = (s_1 \otimes_{\mathrm{S}} s_2, r_1 \otimes_{\mathrm{R}} r_2),$$

and the aggregation operation $\boxplus$,

$$(s_1, r_1) \boxplus (s_2, r_2) = \big(s_1 \oplus_{\mathrm{S}} s_2, \; (r_1 \otimes_{\mathrm{R}} \mathbb{1}[s_1 = s_1 \oplus_{\mathrm{S}} s_2]) \oplus_{\mathrm{R}} (r_2 \otimes_{\mathrm{R}} \mathbb{1}[s_2 = s_1 \oplus_{\mathrm{S}} s_2])\big),$$

where $\mathbb{1}[x] = \begin{cases} 1_{\mathrm{R}} : & x \text{ is a true statement} \\ 0_{\mathrm{R}} : & x \text{ is a false statement.} \end{cases}$

Unfortunately, the selective semiring extension is not always a semiring itself, because the distributive law, which is exploited in efficient MPF query answering algorithms, may not hold. It is, however, a semiring if the aggregation operation on the selective semiring is consistent. The concept of consistency is captured in the following definition.

Definition 3 (Consistent Semirings)**.** *A semiring $(S, \oplus, \otimes)$ is consistent if*

$$s_1 = s_1 \oplus s_2, \text{ if and only if } s_1 \otimes s_3 = (s_1 \oplus s_2) \otimes s_3$$

for all $s_1, s_2, s_3 \in S$ with $s_3 \neq 0$, where 0 is the neutral element for $\oplus$.

Any semiring with multiplicative inverses, such as the sum-product semiring, the Viterbi semiring, and the tropical semiring, is consistent. Examples of consistent semirings without multiplicative inverses are the Boolean semiring and the min-norm semiring (Kim and Choi 2013; Sanmartín, Damrich, and Hamprecht 2022). Consistency proofs are given in the supplement.

For consistent selective semirings S and semirings R in the definition of the selective semiring extension, we can prove that the extension is again a semiring.

Theorem 1. *For consistent selective semirings S, the selective semiring extension is again a semiring.*

Proof. Included in the supplement. □

In the following sections, we will use the selective semiring extension as a template to construct new semirings for model counting, model sets, and model sampling.

Model Counting

The *model counting semiring* $(S \times \mathbb{N}, \boxplus, \boxtimes)$ is an instantiation of the abstract selective semiring extension framework. It keeps the consistent selective semiring S in Definition 2, but instantiates the semiring R by the sum-product semiring $(\mathbb{N}, +, \cdot)$.

We will show that for a given MPF query on S, a corresponding lifted query on $S \times \mathbb{N}$ gives the result of the original MPF query in its first component and the model count in its second component. The lifted query is formally defined as follows.

Definition 4 (Model Counting Query)**.** *For an MPF query*

$$\psi = \bigoplus_x \bigotimes_{j=1}^{m} f_j(x)$$

on a consistent selective semiring S, the lifted query on the model counting semiring is defined as

$$\psi^c = \boxplus_x \boxtimes_{j=1}^{m} f_j^c(x)$$

with $f_j^c : [D_J] \to S \times \mathbb{N}$, $z \mapsto \big(f_j(z), 1\big)$.

The number of component functions f_j and their domains are not affected by lifting the query. Therefore, the structure of the queries ψ and ψ^c is the same. In particular, the treewidth, which determines the asymptotic computational complexity of the queries, remains the same. Also, an optimal junction tree for ψ is also an optimal junction tree for ψ^c.

Theorem 2 (Model Counting as MPF Query)**.** *Let ψ be an MPF query on a consistent selective semiring S and let ψ^c be its lifting to the model counting semiring. Then the first component of ψ^c is the result of ψ, that is, $\bigoplus_x \bigotimes_{j=1}^m f_j(x)$, and the second component of ψ^c is the corresponding model count*

$$MC(\psi) = \sum_x \mathbb{1}\left[\bigotimes_{j=1}^{m} f_j(x) = \psi\right].$$

Proof. By the definition of $\boxtimes$ and of the tuples $f_j^c(x)$ with $(f_j^c(x))_1 = f_j(x)$ and $(f_j^c(x))_2 = 1$, we have

$$\boxplus_x \boxtimes_{j=1}^{m} f_j^c(x) = \boxplus_x \left(\bigotimes_{j=1}^{m} \left(f_j^c(x)\right)_1, \prod_{j=1}^{m} \left(f_j^c(x)\right)_2 \right)$$
$$= \boxplus_x \left(\bigotimes_{j=1}^{m} f_j(x), \prod_{j=1}^{m} 1 \right) = \boxplus_x \left(\bigotimes_{j=1}^{m} f_j(x), 1 \right).$$

From the definition of $\boxplus$ we get for the first component of ψ^c,

$$\left(\bigboxplus_x \left(\bigotimes_{j=1}^m f_j(x), 1\right)\right)_1 = \bigoplus_x \bigotimes_{j=1}^m f_j(x) = \psi,$$

and for the second component,

$$\left(\bigboxplus_x \left(\bigotimes_{j=1}^m f_j(x), 1\right)\right)_2 = \sum_x 1 \cdot \mathbb{1}\left[\bigotimes_{j=1}^m f_j(x) = \psi\right].$$

The second component thus counts the $x \in [D]$ for which $\bigotimes_{j=1}^m f_j(x) = \psi$, that is, the model count of ψ. □

In the supplement, we generalize the construction from model counting to weighted model counting.

Model Set

Another instantiation of the selective semiring extension is the *model set semiring*, which can be used to compute the set of all models for an MPF query. An element $x' \in [D]$ is a model of an MPF query $\psi = \oplus_x \otimes_{j=1}^m f_j(x)$ on a selective semiring S if

$$\bigotimes_{j=1}^m f_j(x') = \bigoplus_x \bigotimes_{j=1}^m f_j(x).$$

Therefore, the *model set* for ψ, that is, the set of all its models, is given as

$$\text{MS}(\psi) = \Big\{x' \in [D] \;\Big|\; \bigotimes_{j=1}^m f_j(x') = \bigoplus_x \bigotimes_{j=1}^m f_j(x)\Big\}.$$

To be able to construct the model set by answering an MPF query, we need to extend S by a semiring that allows us to track partial models. To that end, we first need to construct an appropriate semiring which we then use in the selective semiring extension of S for the model set extension. We call the constructed semiring the *semiring of sets of partial states*.

Semiring of Sets of Partial States

The model set is a subset of the domain $[D]$ of the product function $\bigotimes_{j=1}^m f_j$. We also call the domain $[D]$ the set of *states*. The factors f_j of the product function are defined on projections $[D_J]$ of the domain $[D]$. We call the elements of projected domains *partial states*. The set $\bigcup_{I\subseteq[n]} [D_I]$ contains all possible partial states, and the semiring of sets of partial states is defined on the power set of this set. That is, the semiring is defined on

$$\mathcal{X}_D = \mathcal{P}\Big(\bigcup_{I\subseteq[n]} [D_I]^+\Big),$$

where $[D_I]^+$ is $[D_I]$, with the only difference that each element in $[D_I]^+$ also has a reference to the index set I, which we call *axis identifier*. That is, the axis identifier of $x \in [D_I]$ refers the index set I. The model set is an element $\text{MS}(\psi) \in \mathcal{X}_D$ that only contains states $x \in [D]$, but not partial states. However, while computing the model set, we need to track sets of partial models, that is, sets of partial states in $\mathcal{X}_D$. In the following, whenever the set D is clear from the context, we omit the subscript D.

We need some additional notation, especially for the definition of the combination operation on $X \in \mathcal{X}$. Partial states $x_i, x_j \in X$ can have different axis identifiers, and only share the axes in the intersection of the axis identifiers. Two partial states x_i and x_j are called *compatible*, if they coincide on the intersection $I \cap J$ of their axis identifiers, that is, if $x_{i|I\cap J} = x_{j|I\cap J}$. Remember that $x_{|K}$ denotes the projection of $x \in [D_I]$ onto its elements indexed by $K \subseteq I$, that is, $x_{|K} = (x_k \mid k \in K)$. If $I \cap J = \varnothing$, then x_i and x_j are always compatible.

For defining a combination operation on $\mathcal{X}$, we first define a *fusion* operation on X. The fusion of compatible models $x_i \in [D_I]^+$ and $x_j \in [D_J]^+$ is the unique partial state

$$z = x_i \odot x_j \in [D_{I\cup J}]^+$$

with $z_{|I} = x_i$ and $z_{|J} = x_j$. Since the fusion operation works on partial states, but not on sets of partial states, it is not yet a combination operation on X. However, the fusion operation can be used to define a combination operation on X. This leads us to the following definition.

Definition 5 (Semiring of Partial States)**.** *The semiring of partial states $(\mathcal{X}, \oplus_\mathcal{X}, \otimes_\mathcal{X})$ is given by the aggregation operation*

$$\oplus_\mathcal{X} : \mathcal{X} \times \mathcal{X} \to \mathcal{X},\ (X_1, X_2) \mapsto X_1 \cup X_2,$$

and the combination operation $\otimes_\mathcal{X}$: $\mathcal{X} \times \mathcal{X} \to \mathcal{X}$,

$$(X_1, X_2) \mapsto \big\{x_1 \odot x_2 \mid x_1 \in X_1, x_2 \in X_2 \text{ compatible}\big\}.$$

Of course, we need to prove that $(\mathcal{X}, \oplus_\mathcal{X}, \otimes_\mathcal{X})$ is a semiring, which we do in the following theorem.

Theorem 3. *$(\mathcal{X}, \oplus_\mathcal{X}, \otimes_\mathcal{X})$ is indeed a semiring with neutral elements $0_\mathcal{X} = \varnothing$ and $1_\mathcal{X} = \{()\}$, where $() \in [D_\varnothing]^+$ is the empty tuple.*

Proof. Included in the supplement. □

Model Set Semiring

The model set semiring is now just the selective semiring extension of a consistent selective semiring $(S, \oplus, \otimes)$, where the second semiring R is instantiated by the semiring $(\mathcal{X}, \oplus_\mathcal{X}, \otimes_\mathcal{X})$ of partial states. Similar to the model counting extension, we can again lift an MPF query ψ on S to the model set extension.

Definition 6 (Model Set Query)**.** *For an MPF query*

$$\psi = \bigoplus_x \bigotimes_{j=1}^m f_j(x)$$

on a consistent selective semiring S, the lifted query on the model set semiring is defined as

$$\psi^s = \bigboxplus_x \boxtimes_{j=1}^m f_j^s(x)$$

with $f_j^s : [D_J] \to S \times \mathcal{X},\ z \mapsto \big(f_j(z), \{z\}\big)$.

With this definition, we can prove that the lifted MPF query does indeed provide the answer to the original MPF query and the corresponding model set.

Theorem 4 (Model Set Construction as MPF query)**.** *Let ψ be an MPF query on a consistent selective semiring S and let ψ^s be its lifting to the model set semiring. Then the first component of ψ^s is the result of ψ, that is, $\bigoplus_x \bigotimes_{j=1}^m f_j(x)$, and the second component of ψ^s is the corresponding model set*

$$\Big\{x \in [D] \;\Big|\; \bigotimes_{j=1}^{m} f_j(x) = \psi\Big\}.$$

Proof. Included in the supplement. □

As for the model counting query, the structure of the query ψ is unchanged by lifting it to the model set semiring. However, when there are many models, tracking the model set quickly dominates the computation. In the next section, we tackle this issue by sampling only one model uniformly at random instead of constructing the full model set.

Model Sampling

The model set of a selective MPF query can be so large that enumerating all models explicitly can be computationally infeasible. Therefore, instead of computing the full model set, we compute an element from the model set uniformly at random. For sampling uniformly we need information about the model count and the model set. To this end, we first combine the model counting and the model set extensions into a single extension, which we call the *sampling semiring*.

Model Sampling Semiring

The model sampling semiring is just the selective semiring extension of a consistent selective semiring S and the product of the sum-product semiring $(\mathbb{N}, +, \cdot)$ and the semiring of sets of partial states $(\mathcal{X}, \oplus_\mathcal{X}, \otimes_\mathcal{X})$. That is, R in the definition of the selective semiring extension (Definition 2) is the product semiring of $(\mathbb{N}, +, \cdot)$ and $(\mathcal{X}, \oplus_\mathcal{X}, \otimes_\mathcal{X})$. Therefore, the combination operation $\boxtimes$ on the model sampling semiring is defined as

$$(s_1, w_1, X_1) \boxtimes (s_2, w_2, X_2) = \big(s_1 \otimes s_2, w_1 \cdot w_2, X_1 \otimes_\mathcal{X} X_2\big),$$

and the aggregation operation $\boxplus$ on the model sampling semiring is defined as

$$(s_1, w_1, X_1) \boxplus \big(s_2, w_2, X_2\big) = \big(s_1 \oplus s_2, w, X\big),$$

where

$$w = \big(w_1 \cdot \mathbb{1}_\mathbb{N}\left[s_1 = s_1 \oplus s_2\right]\big) + \big(w_2 \cdot \mathbb{1}_\mathbb{N}\left[s_2 = s_1 \oplus s_2\right]\big)$$

and

$$X = \big(X_1 \cdot \mathbb{1}_\mathcal{X}\left[s_1 = s_1 \oplus s_2\right]\big) \cup \big(X_2 \cdot \mathbb{1}_\mathcal{X}\left[s_2 = s_1 \oplus s_2\right]\big).$$

As we have done before for the model counting semiring and the model set semiring, we can lift an MPF query on a consistent selective semiring S to the model sampling semiring $S \times \mathbb{N} \times \mathcal{X}$.

Definition 7 (Lifted Model Sampling Query)**.** *For an MPF query*

$$\psi = \bigoplus_x \bigotimes_{j=1}^{m} f_j(x)$$

on a consistent selective semiring S, the lifted query on the model sampling semiring $S \times \mathbb{N} \times \mathcal{X}$ is defined as

$$\psi^\star = \boxplus_x \boxtimes_{j=1}^{m} f_j^\star(x)$$

with $f_j^\star : [D_J] \to S \times \mathbb{N} \times \mathcal{X}$, $z \mapsto \big(f_j(z), 1, \{z\}\big)$.

It follows directly from Theorems 2 and 4 that the model sampling query $\psi^\star$ computes the result of the original MPF query in its first component, the model count in its second component, and the model set in its third component.

Uniform Model Sampling

For sampling uniformly at random from the model set, we need to specify the algorithm for evaluating $\psi^\star$. Therefore, we introduce a sampling representation of the elements of the semiring $(\mathcal{X}, \oplus_\mathcal{X}, \otimes_\mathcal{X})$ of sets of partial states. From the sets $X \in \mathcal{X}$ we only consider one *representative element* $x \in X$. In the sampling representation, we need to adapt the combination and aggregation operations. The adapted operations work on the representative elements, and compute representative elements of the results of the corresponding operations on the sets of partial states. This can be accomplished as follows: either one of the representative elements $x_i \in X_i$ and $x_j \in X_j$ is a representative element of $X_i \oplus_\mathcal{X} X_j$. If x_i and x_j are compatible, then $x_i \odot x_j$ is a representative element of the combination $X_i \otimes_\mathcal{X} X_j$. As we have pointed out in the proof of Theorem 4 (see the supplement), when evaluating the lifted MPF query ψ^s, we only combine different projections $\{x_{|J}\}$ of the same element $x \in [D]$, which are always compatible. This observation also holds for $\psi^\star$, and therefore, the requirement of compatibility in the definition of the combination operation on X is not a restriction when evaluating model sampling queries. Thus, we can always choose a representative element of the aggregation and combination operations from the representative elements of their arguments. In the supplemental material, we provide a formal proof of this claim.

Working with representative elements has the advantage that it grants us the freedom to choose them. Here, we want to choose representative elements such that we can make sure to sample models from the model set uniformly at random. This can be achieved through the following probabilistic implementation of the combination and aggregation operations: In the implementation of the combination operation, the two representative elements $x_1 \in X_1$ and $x_2 \in X_2$ are simply fused into a representative of $X_1 \otimes_\mathcal{X} X_2$. The implementation of the aggregation operation is more interesting, where a representative $x \in \{x_1, x_2\}$ of $X_1 \oplus_\mathcal{X} X_2$ is chosen at random depending on the result of the aggregation of s_1, s_2 in S. In the case $s_1 \neq s_2$ we always want $x = x_i$ if $s_i = s_1 \oplus s_2$ since in accordance with the selective semiring extension, the aggregation only keeps the set X_i that corresponds to the selected element s_i. The interesting case is

when $s_1 = s_2$, where x is chosen with probability proportional to the cardinalities w_1, w_2 of X_1 and X_2. Therefore, we have for $i \in \{1, 2\}$,

$$p(x = x_i) = \begin{cases} \mathbb{1}_{\mathbb{N}}\left[s_i = s_1 \oplus s_2\right] & \text{if } s_1 \neq s_2 \\ w_i/(w_1 + w_2) & \text{if } s_1 = s_2. \end{cases}$$

Now, when we use representative elements instead of the sets of partial states, the model sampling query computes in its third element only one representative element of the model set, that is, only one model $x \in \mathrm{MS}(\psi)$. Using the probabilistic implementation of the aggregation operation on $\mathcal{X}$, the lifted model sampling query returns a model from the model set that is chosen uniformly at random. We formalize this claim in the following theorem.

Theorem 5 (Uniform Model Set Sampling)**.** *Let ψ be an MPF query on a consistent selective semiring S and let $\psi^\star$ be its lifting to the model sampling semiring $S \times \mathbb{N} \times \mathcal{X}$. Using the probabilistic implementation of the aggregation operation on $\mathcal{X}$, the third component of $\psi^\star$ is an element $x^\star \in [D]$ such that for all $x \in [D]$ holds*

$$p(x^\star = x) = \begin{cases} 1/\mathit{MC}(\psi) & \textit{if } x \textit{ is a model of } \psi \\ 0 & \textit{otherwise.} \end{cases}$$

That is, $x^\star$ is drawn uniformly from all models of ψ.

Proof. We prove by induction over the size of the domain $[D]$ that $x^\star$ is drawn uniformly at random from the model set of ψ, which is subset of $[D]$.

For the base case, assume that $[D]$ has only one element z (with n components). The lifted query $\psi^\star$ evaluates to

$$\begin{aligned}(s^\star, w^\star, x^\star) &= \bigoplus_{x \in [D]} \bigotimes_{j=1}^{m} f_j^\star(x) = \bigoplus_{x \in \{z\}} \bigotimes_{j=1}^{m} f_j^\star(x) \\ &= \bigotimes_{j=1}^{m} f_j^\star(z) = \bigotimes_{j=1}^{m} \left(f_j(z), 1, z_{|J}\right).\end{aligned}$$

Now consider the combination. By definition, $\boxtimes$ reuses the multiplication on $\mathbb{N}$ in the second component, which gives $w^\star = \mathrm{MC}(\psi) = 1$, and $\boxtimes$ uses the fusion operation in the third component, which gives

$$x^\star = \bigodot_{j=1}^{m} z_{|J} = z.$$

Therefore, $p(x^\star = z) = 1 = 1/\mathrm{MC}(\psi)$, and the claim holds for the base case.

For the induction hypothesis, assume that the claim holds for any query on domains $[D]$ with k elements, that is, for any query

$$\psi_k^\star = \bigoplus_{x \in \{z_1, \ldots, z_k\}} \bigotimes_{j=1}^{m} f_j^\star(x), \text{ where } z_i \in [D], i \in [k].$$

For the inductive step, let $[D] = \{z_1, \ldots, z_k, z_{k+1}\}$ be a domain with $k + 1$ elements. By the induction hypothesis, we have

$$\begin{aligned}\psi_{k+1}^\star &= \underbrace{\bigoplus_{x \in \{z_1, \ldots, z_k\}} \bigotimes_{j=1}^{m} f_j^\star(x)}_{=\psi_k^\star} \boxplus \bigotimes_{j=1}^{m} f_j^\star(z_{k+1}) \\ &= \left(s', w', x'\right) \boxplus (s, 1, z_{k+1})\end{aligned}$$

where s' is the value of the MPF query ψ_k on the selective semiring S, $w' = \mathrm{MC}(\psi_k)$ is the corresponding model count, and, by the induction hypothesis, x' is a sample from the corresponding model set $\mathrm{MS}(\psi_k)$ that has been drawn with probability $p(x_k^\star = x') = 1/\mathrm{MC}(\psi_k) = 1/w'$.
It remains to compute the result of the final aggregation

$$(s^\star, w^\star, x^\star) = \left(s', w', x_k^\star = x'\right) \boxplus (s, 1, z_{k+1}).$$

We distinguish the two cases $s' \neq s$ and $s' = s$. First, let $s' \neq s$. If $s^\star = s'$, then the claim holds by the induction hypothesis. Otherwise, if $s^\star = s$, then the claim is equivalent to the base case. Therefore, the claim holds true in both cases.

Second, let $s' = s = s^\star$. By the definition of $\boxplus$, we have $\mathrm{MC}(\psi_{k+1}) = w^\star = w' + 1$, and

$$p(x^\star = z_{k+1}) = \frac{1}{w' + 1} = 1/\mathrm{MC}(\psi_{k+1})$$

and

$$p(x^\star = x' | x_k^\star = x') = w'/(w' + 1).$$

Since $p(x^\star = x' | x_k^\star \neq x') = 0$ and, by the induction hypothesis, $p(x_k^\star = x') = 1/w'$, we have

$$\begin{aligned}p(x^\star = x') &= \sum_{x : x \neq z_{k+1}} p(x^\star = x' | x_k^\star = x) p(x_k^\star = x) \\ &= p(x^\star = x' | x_k^\star = x') p(x_k^\star = x') \\ &= \frac{w'}{w' + 1} \cdot \frac{1}{w'} = \frac{1}{\mathrm{MC}(\psi_{k+1})}.\end{aligned}$$

Therefore, we get for $x \in \mathrm{MS}(\psi_{k+1})$,

$$p(x^\star = x) = \begin{cases} 1/\mathrm{MC}(\psi_{k+1}) & \text{if } x \in \mathrm{MS}(\psi_k), \\ 1/\mathrm{MC}(\psi_{k+1}) & \text{if } x = z_{k+1}, \\ 0 & \text{otherwise,} \end{cases}$$

and thus $p(x^\star = x) = 1/\mathrm{MC}(\psi_{k+1}^\star)$ if $x \in \mathrm{MS}(\psi_{k+1}^\star)$ and $p(x^\star = x) = 0$, otherwise. Therefore, also in this case, the claim holds true, which concludes the proof. □

The model sampling semiring can be adapted for weighted sampling from a discrete graphical model by implementing the probabilities similar to the variable weights in weighted model counting (Darwiche 2009). Also, the sampling semiring extends naturally to a semiring on $S \times \mathbb{N} \times \mathcal{X}^k$, which can be used to compute k-many independent samples with a single MPF query.

Implementation and Experiments

Here, we compare our semiring extension approach with state-of-the-art methods for the model counting and sampling problems for Boolean formulas, because for these problems, mature implementations of problem specific algorithms are publicly available.

The experiments were performed on an Intel i9-10980XE 18-core processor machine running Ubuntu 20.04.1 LTS with 128 GB of RAM. Each core has a base frequency of 3.0 GHz, a max turbo frequency of 4.6 GHz, and supports the AVX-512 vector instruction set.

Implementation of MPF by Tensor Networks

Aji and McEliece (2000) propose a junction tree algorithm for serving MPF queries. For our experiments, however, we build on the observation that MPF queries are just tensor hypernetwork expressions, which describe the computation of a tensor from its decomposition into smaller tensors. Therefore, generic tensor network contraction algorithms can be used for answering MPF queries. Since tensor expressions are ubiquitous in modern machine learning, many programming languages and frameworks support tensor computations, among them the Python package NumPy (Harris et al. 2020) and the framework PyTorch (Paszke et al. 2019), which both provide an implementation of the *einsum* interface to express tensor computations.

We implemented the semiring extension as a stand-alone datatype in Python 3.9 and Cython 3.0. In our implementation, tensors are represented by NumPy 1.25.1 nd-arrays, and the contraction order of the tensors is computed by `opt_einsum` 3.3.0 (Smith and Gray 2018). We refer to our implementation as `MCSSE` (model counting and sampling semiring extension). The code and more details about the experiments are provided in the supplement.

Model Counting

We evaluated the performance of `MCSSE` on the 200 problems of the 2022 Model Counting Competition for Boolean formulas in conjunctive normal form (CNF)[1]. We preprocessed the problems with `Arjun` (Soos and Meel 2022), which renders 23 of them trivial.

We compared our semiring extension to the direct MPF model counting query from the MPF queries section, and to three state-of-the-art model counting softwares, namely, `SharpSAT-TD` (Korhonen and Järvisalo 2021, 2023), the Python model counter `PySDD` (Darwiche 2011), and `D4` (Lagniez and Marquis 2017). `SharpSAT-TD` with `Arjun` pre-processing is the winner of the 2022 model counting competition. If applicable, we always used default settings for these softwares.

The direct MPF model counting was faster on **18** of the non-trivial **177** instances than any of the three state-of-the-art model counters, while our semiring extension was still faster on **four** instances. That is, both the direct MPF model counter and our semiring extension can improve the virtual best solver, when added to the three state-of-the-art model counters.

[1]https://mccompetition.org/past_iterations

Model Sampling

We have evaluated the performance of `MCSSE` for model sampling by comparing it with the recent `WAPS` model sampler (Gupta et al. 2019), which builds on a connection between knowledge compilation and sampling. Another recent sampler, `DPSampler` (Dudek, Shrotri, and Vardi 2022), uses dynamic programming and algebraic decission diagrams, but was not available in an executable form.

For the comparison we used a data set of 773 CNF formulas, which has been constructed by Gupta et al. (2019) from a wide range of publicly available benchmarks. For each CNF formula in the data set, we measured the time to get one sample. Since `WAPS` is based on knowledge compilation, we measured compile and sample time separately for `WAPS`.

Out of the **773** instances, there were **583** instances where at least one of the algorithms was able to return a sample within our time limit of 30 seconds. On **293** out of the **591** instances, the time used by `MCSSE` to return one sample was less then the total time used by `WAPS`. Furthermore, on **258** out of these **293** instances, `MCSSE` used less time than `WAPS` used only for sampling. On average, `MCSSE` was $\mathbf{49.5}$ times faster than `WAPS` on these instances, which means that the compilation approach does not pay off on these instances. On the remaining instances, `WAPS` was on average $\mathbf{7.5}$ times faster than `MCSSE`.

Discussion

Generic frameworks such as MPF separate the interface from algorithmic details and their implementation. This has the advantage that the same algorithm, even the same implementation, can be used directly for different applications within the same framework. However, the advantage can also be a disadvantage, because it can be difficult to exploit application specific information algorithmically. Nevertheless, our generic tensor network based implementation is able to compete with application specific implementations for model counting and sampling on Boolean formulas.

We observe that our prototypical implementation is slower than the state of the art on two kinds of problem instances. On many small instances the overhead of setting up the tensor network is dominating. And on larger instances, large intermediate tensors can be created during the tensor network evaluation. However, there is significant potential to further improve the performance of MPF based model counting and sampling. The creation of large intermediate tensors can sometimes be avoided by switching to a different tensor contraction order. Moreover, large intermediate tensors tend to be sparse. In our implementation, we used a dense tensor format, because there are no established implementations of *einsum* for sparse tensors.

Conclusion

We have extended the marginalize a product function (MPF) framework for model counting and sampling on selective semirings. Although our extension is generic, it shows competitive performance on the special case of model counting and sampling for Boolean formulas, using a non-optimized implementation.

Acknowledgements

This work was supported by the Carl Zeiss Stiftung within the projects "Interactive Inference" and "A virtual Werkstatt for digitization in the sciences".

References

Aji, S. M. 2000. *Graphical models and iterative decoding*. California Institute of Technology.

Aji, S. M.; and McEliece, R. J. 2000. The generalized distributive law. *IEEE Trans. Inf. Theory*, 46(2).

Baras, J. S.; and Theodorakopoulos, G. 2010. *Path Problems in Networks*. Synthesis Lectures on Communication Networks. Morgan & Claypool Publishers.

Baum, L. E. 1972. An Inequality and Associated Maximization Technique in Statistical Estimation for Probabilistic Functions of Markov Processes. In Shisha, O., ed., *Inequalities III: Proceedings of the Third Symposium on Inequalities*. University of California, Los Angeles: Academic Press.

Bistarelli, S.; Montanari, U.; and Rossi, F. 1997. Semiring-based constraint satisfaction and optimization. *J. ACM*, 44(2).

Darwiche, A. 2009. *Modeling and Reasoning with Bayesian Networks*. Cambridge University Press.

Darwiche, A. 2011. SDD: A New Canonical Representation of Propositional Knowledge Bases. In Walsh, T., ed., *IJCAI 2011, Proceedings of the 22nd International Joint Conference on Artificial Intelligence, Barcelona, Catalonia, Spain, July 16-22, 2011*. IJCAI/AAAI.

Dudek, J. M.; Shrotri, A. A.; and Vardi, M. Y. 2022. DPSampler: Exact Weighted Sampling Using Dynamic Programming. In Raedt, L. D., ed., *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI 2022, Vienna, Austria, 23-29 July 2022*. ijcai.org.

Gupta, R.; Sharma, S.; Roy, S.; and Meel, K. S. 2019. WAPS: Weighted and Projected Sampling. In Vojnar, T.; and Zhang, L., eds., *Tools and Algorithms for the Construction and Analysis of Systems - 25th International Conference, TACAS 2019, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2019, Prague, Czech Republic, April 6-11, 2019, Proceedings, Part I*, volume 11427 of *Lecture Notes in Computer Science*. Springer.

Harris, C. R.; Millman, K. J.; van der Walt, S.; Gommers, R.; Virtanen, P.; Cournapeau, D.; Wieser, E.; Taylor, J.; Berg, S.; Smith, N. J.; Kern, R.; Picus, M.; Hoyer, S.; van Kerkwijk, M. H.; Brett, M.; Haldane, A.; del Río, J. F.; Wiebe, M.; Peterson, P.; Gérard-Marchant, P.; Sheppard, K.; Reddy, T.; Weckesser, W.; Abbasi, H.; Gohlke, C.; and Oliphant, T. E. 2020. Array Programming with NumPy. *CoRR*, abs/2006.10256.

Joglekar, M. R.; Puttagunta, R.; and Ré, C. 2016. AJAR: Aggregations and Joins over Annotated Relations. In Milo, T.; and Tan, W., eds., *Proceedings of the 35th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems, PODS 2016, San Francisco, CA, USA, June 26 - July 01, 2016*. ACM.

Kalman, R. E. 1960. A new approach to linear filtering and prediction problems. *Journal of basic Engineering*, 82(1).

Khamis, M. A.; Ngo, H. Q.; and Rudra, A. 2016. FAQ: Questions Asked Frequently. In Milo, T.; and Tan, W., eds., *Proceedings of the 35th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems, PODS 2016, San Francisco, CA, USA, June 26 - July 01, 2016*. ACM.

Kim, K.; and Choi, S. 2013. Walking on Minimax Paths for k-NN Search. In desJardins, M.; and Littman, M. L., eds., *Proceedings of the Twenty-Seventh AAAI Conference on Artificial Intelligence, July 14-18, 2013, Bellevue, Washington, USA*. AAAI Press.

Kimmig, A.; Van den Broeck, G.; and De Raedt, L. 2017. Algebraic model counting. *Journal of Applied Logic*, 22: 46–62.

Kohlas, J.; and Wilson, N. 2008. Semiring induced valuation algebras: Exact and approximate local computation algorithms. *Artif. Intell.*, 172(11).

Korhonen, T.; and Järvisalo, M. 2021. Integrating Tree Decompositions into Decision Heuristics of Propositional Model Counters (Short Paper). In Michel, L. D., ed., *27th International Conference on Principles and Practice of Constraint Programming, CP 2021, Montpellier, France (Virtual Conference), October 25-29, 2021*, volume 210 of *LIPIcs*, 8:1–8:11. Schloss Dagstuhl - Leibniz-Zentrum für Informatik.

Korhonen, T.; and Järvisalo, M. 2023. SharpSAT-TD in Model Counting Competitions 2021-2023. *CoRR*, abs/2308.15819.

Kschischang, F. R.; and Frey, B. J. 1998. Iterative Decoding of Compound Codes by Probability Propagation in Graphical Models. *IEEE J. Sel. Areas Commun.*, 16(2).

Lagniez, J.; and Marquis, P. 2017. An Improved Decision-DNNF Compiler. In Sierra, C., ed., *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI 2017, Melbourne, Australia, August 19-25, 2017*. ijcai.org.

Paszke, A.; Gross, S.; Massa, F.; Lerer, A.; Bradbury, J.; Chanan, G.; Killeen, T.; Lin, Z.; Gimelshein, N.; Antiga, L.; Desmaison, A.; Köpf, A.; Yang, E. Z.; DeVito, Z.; Raison, M.; Tejani, A.; Chilamkurthy, S.; Steiner, B.; Fang, L.; Bai, J.; and Chintala, S. 2019. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In Wallach, H. M.; Larochelle, H.; Beygelzimer, A.; d'Alché-Buc, F.; Fox, E. B.; and Garnett, R., eds., *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*.

Pearl, J. 1993. Belief Networks Revisited. *Artif. Intell.*, 59(1-2).

Sanmartín, E. F.; Damrich, S.; and Hamprecht, F. A. 2022. The Algebraic Path Problem for Graph Metrics. In Chaudhuri, K.; Jegelka, S.; Song, L.; Szepesvári, C.; Niu, G.; and Sabato, S., eds., *International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA*, volume 162 of *Proceedings of Machine Learning Research*. PMLR.

Smith, D. G. A.; and Gray, J. 2018. opt_einsum - A Python package for optimizing contraction order for einsum-like expressions. *J. Open Source Softw.*, 3(26).

Soos, M.; and Meel, K. S. 2022. Arjun: An Efficient Independent Support Computation Technique and its Applications to Counting and Sampling. In Mitra, T.; Young, E. F. Y.; and Xiong, J., eds., *Proceedings of the 41st IEEE/ACM International Conference on Computer-Aided Design, ICCAD 2022, San Diego, California, USA, 30 October 2022 - 3 November 2022*. ACM.

Sutton, C.; and McCallum, A. 2012. An Introduction to Conditional Random Fields. *Found. Trends Mach. Learn.*, 4(4).

Viterbi, A. J. 1967. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Trans. Inf. Theory*, 13(2).