Algebraic Model Counting

Angelika Kimmig[a,*], Guy Van den Broeck[b], Luc De Raedt[a]

[a] *Department of Computer Science, KU Leuven, Celestijnenlaan 200a – box 2402, 3001 Heverlee, Belgium.*
[b] *UCLA – Computer Science Department, 4531E Boelter Hall, Los Angeles, CA 90095-1596A, USA.*

Abstract

Weighted model counting (WMC) is a well-known inference task on knowledge bases, and the basis for some of the most efficient techniques for probabilistic inference in graphical models. We introduce algebraic model counting (AMC), a generalization of WMC to a semiring structure that provides a unified view on a range of tasks and existing results. We show that AMC generalizes many well-known tasks in a variety of domains such as probabilistic inference, soft constraints and network and database analysis. Furthermore, we investigate AMC from a knowledge compilation perspective and show that all AMC tasks can be evaluated using `sd-DNNF` circuits, which are strictly more succinct, and thus more efficient to evaluate, than direct representations of sets of models. We identify further characteristics of AMC instances that allow for evaluation on even more succinct circuits.

Keywords: Knowledge Compilation, Model Counting, Logic

1. Introduction

Today, some of the most efficient techniques for probabilistic inference employ reductions to weighted model counting (WMC) both for propositional and for relational probabilistic models (Park, 2002; Sang et al., 2005; Darwiche, 2009; Fierens et al., 2011; Van den Broeck et al., 2011). The resulting weighted model counting task is often solved by a single pass over a propositional circuit, which is a compact graphical representation of the models of interest. This approach makes it possible to perform the possibly expensive knowledge compilation step, that is, the construction of the circuit, only once, and to then evaluate this circuit repeatedly, for instance, under different evidence or with different parameters.

On the other hand, it is well-known that probabilistic inference as well as many other tasks can be generalized to a sum of products computation over

[*] Corresponding author
Email addresses: angelika.kimmig@cs.kuleuven.be (Angelika Kimmig), guyvdb@cs.ucla.edu (Guy Van den Broeck), luc.deraedt@cs.kuleuven.be (Luc De Raedt)

models with suitable operators from a semiring structure. This has led to common inference algorithms for a variety of different inference problems in many fields, including parsing (Goodman, 1999), dynamic programming (Eisner et al., 2005), constraint programming (Meseguer et al., 2006), databases (Green et al., 2007), Bayesian inference (Bacchus et al., 2009), propositional logic (Larrosa et al., 2010), networks (Baras & Theodorakopoulos, 2010) and logic programming (Kimmig et al., 2011). The work presented here provides a unified view on these two lines of work by introducing both a general definition of model counting in a semiring setting and a solution approach for this task based on knowledge compilation.

As our first contribution, we introduce the task of *algebraic model counting (AMC)*. AMC generalizes weighted model counting to the semiring setting and supports various types of labels (or weights), including numerical ones as used in WMC, but also sets (e.g., to collect relevant variables), Boolean formulae (e.g., to obtain explicit representations of models), polynomials (e.g., for sensitivity analysis in probabilistic models), and many more. It thus provides a framework that covers many different tasks from a variety of different fields. As our second contribution, we investigate how to solve AMC problems using knowledge compilation. As AMC is defined in terms of the set of models of a propositional logic theory, we can exploit the succinctness results of the knowledge compilation map of Darwiche & Marquis (2002). We show that AMC can in general be evaluated using `sd-DNNF` circuits, which are more succinct, and thus more efficient to evaluate, than a direct representation of the set of models. Furthermore, we identify a number of characteristics of AMC tasks that allow for evaluation on even more succinct types of circuits. Our results provide a unified view on existing results, which also allows us to generalize well-known insights for satisfiability and model counting in circuits to broad classes of AMC tasks and to extend the task classification in algebraic Prolog (Kimmig et al., 2011) to more succinct types of circuits. As our third contribution, we further broaden the applicability of the AMC framework by linking it to semiring sums of products defined over derivations, that is, sequences of possibly repeated variables, instead of over models.

This paper is organized as follows. We introduce algebraic model counting in Section 2. Section 3 provides task characteristics that allow for correct evaluation on specific classes of circuits and shows how these generalize previous results. We discuss future work and conclude in Section 4.

2. Algebraic Model Counting

Our definition of algebraic model counting builds upon the well-known task of weighted model counting for propositional logic theories. Given a propositional logic theory T over a set of variables $\mathcal{V}$, an *interpretation* of $\mathcal{V}$ assigns a truth value from the set $\{true, false\}$ to every variable in $\mathcal{V}$. The set $\mathcal{M}(T)$ of *models* of theory T contains exactly those interpretations of $\mathcal{V}$ for which T evaluates to true. We here view interpretations (and models) as sets of literals, that is, for each variable $v \in \mathcal{V}$, an interpretation contains either the positive literal v or the negative literal $\neg v$. We use $\mathcal{L}$ to denote the set of literals for the variables in $\mathcal{V}$. In weighted model counting, non-negative real-valued weights are associated with all literals, and the weighted model count of a propositional theory is obtained by multiplying these weights for each model of the theory, and summing the results for all models.[1]

Definition 1 (Weighted Model Counting (WMC))**.** Given

- a *propositional logic theory* T over a set of variables $\mathcal{V}$ and
- a *weight function* $w : \mathcal{L} \to \mathbb{R}_{\geq 0}$, mapping literals of the variables in $\mathcal{V}$ to non-negative real-valued weights,

the task of *weighted model counting (WMC)* is to compute

$$\mathbf{WMC}(T) = \sum_{I \in \mathcal{M}(T)} \prod_{l \in I} w(l). \tag{1}$$

Algebraic model counting generalizes multiplication and summation of real-valued weights to corresponding operations from an arbitrary commutative semiring. It thus extends WMC to more general classes of weights, which are not necessarily real-valued. To emphasize the latter, we use the terms *labeling function* and *label* in the context of algebraic model counting.

Definition 2 ((Commutative) Semiring)**.** A *semiring* is a structure $(\mathcal{A}, \oplus, \otimes, e^{\oplus}, e^{\otimes})$, where

- *addition* $\oplus$ is an associative and commutative binary operation over the set $\mathcal{A}$,
- *multiplication* $\otimes$ is an associative binary operation over the set $\mathcal{A}$,
- $\otimes$ distributes over $\oplus$,
- $e^{\oplus} \in \mathcal{A}$ is the neutral element of $\oplus$, i.e., for all $a \in \mathcal{A}$, $a \oplus e^{\oplus} = a$,
- $e^{\otimes} \in \mathcal{A}$ is the neutral element of $\otimes$, i.e., for all $a \in \mathcal{A}$, $a \otimes e^{\otimes} = a$, and
- $e^{\oplus}$ is an annihilator for $\otimes$, i.e., for all $a \in \mathcal{A}$, $e^{\oplus} \otimes a = a \otimes e^{\oplus} = e^{\oplus}$.

In a *commutative semiring*, $\otimes$ is commutative as well.

Examples of commutative semirings can be found in columns 2–6 of Table 1; these will be discussed below. Generalizing weighted model counting to labeling functions defined over commutative semirings, we now define algebraic model counting as follows:

Definition 3 (AMC Problem)**.** Given

[1] The case where weights are associated with joint assignments to groups of variables, as in factor graphs, can be mapped to the case considered here; cf. Section 2.2.

- a *propositional logic theory* T over a set of variables $\mathcal{V}$,
- a *commutative semiring* $(\mathcal{A}, \oplus, \otimes, e^{\oplus}, e^{\otimes})$, and
- a *labeling function* $\alpha : \mathcal{L} \rightarrow \mathcal{A}$, mapping literals $\mathcal{L}$ of the variables in $\mathcal{V}$ to values from the semiring set $\mathcal{A}$,

the task of *algebraic model counting (AMC)* is to compute

$$\mathbf{A}(T) = \bigoplus_{I \in \mathcal{M}(T)} \bigotimes_{l \in I} \alpha(l). \tag{2}$$

That is, starting from semiring values associated with individual literals via the labeling function, AMC assigns a value to each interpretation of the variables by combining the values of corresponding literals with semiring multiplication, and a value to the theory by combining values of all its models with semiring addition.

2.1. Examples of AMC Tasks

To provide a better idea of the variety of tasks covered by this general definition, we next discuss examples of AMC tasks based on semirings and labeling functions found in the literature, as summarized in Table 1. Probably the most basic instance of AMC is the evaluation of a Boolean formula for a given interpretation (Bool), where the labeling function assigns the truth values given by the interpretation to the literals, i.e., labels are Boolean, $\alpha(v) = true$ if v is true in the given interpretation (and $\alpha(v) = false$ else), negative literals $\neg v$ are labeled $\neg\alpha(v)$, and those labels are combined using the usual Boolean semiring with $\oplus = \vee$ and $\otimes = \wedge$. In fact, this instance can be seen as providing the basis for the evaluation of AMC tasks via knowledge compilation as discussed in Section 3. Using the same Boolean semiring, but in combination with a labeling function that assigns *true* to all literals, we obtain an instance of AMC that corresponds to the satisfiability task of propositional logic (SAT).

As can already be seen from their definitions, weighted model counting (WMC) itself is another instance of AMC, which combines non-negative real numbers as labels with ordinary multiplication and addition. The well-known task of model counting (#SAT) corresponds to the special case where all literal weights are 1 (and counts thus restricted to the natural numbers), whereas probabilistic inference (Prob) in a setting where all variables are independently assigned truth values at random restricts the labeling function of WMC to values from $[0, 1]$ such that labels of positive and negative literals for each variable sum to one, i.e., for every variable v, $\alpha(v) \in [0, 1]$ and $\alpha(\neg v) = 1 - \alpha(v)$.

We can extend the Prob setting to an AMC task to perform sensitivity analysis (Sens) by allowing the use of variables instead of constant probabilities as labels, i.e., a positive literal v can be labeled with a value in $[0, 1]$ as before, or with variable v, and negative literals are still labeled $\alpha(\neg v) = 1 - \alpha(v)$. The corresponding semiring uses summation and multiplication of polynomials as $\oplus$ and $\otimes$, respectively. That is, the algebraic model count is an explicit function

task	$\mathcal{A}$	$e^{\oplus}$	$e^{\otimes}$	$\oplus$	$\otimes$	$\alpha(v)$	$\alpha(\neg v)$	ref	idp. $\oplus$	neut. $(\oplus, \alpha)$	icp. $(\otimes, \alpha)$
BOOL	$\{true, false\}$	$false$	$true$	$\vee$	$\wedge$	$\in \{true, false\}$	$\neg\alpha(v)$		✓	✓	✓
SAT	$\{true, false\}$	$false$	$true$	$\vee$	$\wedge$	$true$	$true$	B, BT, G, GK, K, L, M	✓	✓	–
WMC	$\mathbb{R}_{\geq 0}$	0	1	$+$	$\cdot$	$\in \mathbb{R}_{\geq 0}$	$\in \mathbb{R}_{\geq 0}$		–	–	–
#SAT	$\mathbb{N}$	0	1	$+$	$\cdot$	1	1	B, G, GK, K, L	–	–	–
PROB	$\mathbb{R}_{\geq 0}$	0	1	$+$	$\cdot$	$\in [0,1]$	$1-\alpha(v)$	B, BT, E, G, K	–	✓	–
SENS	$\mathbb{R}[\mathcal{V}]$	0	1	$+$	$\cdot$	v or $\in [0,1]$	$1-\alpha(v)$	K	–	✓	–
GRAD	$\mathbb{R}_{\geq 0} \times \mathbb{R}$	$(0,0)$	$(1,0)$	Eq. (5)	Eq. (6)	Eq. (3)	Eq. (4)	E, K	–	✓	–
MPE	$\mathbb{R}_{\geq 0}$	0	1	max	$\cdot$	$\in [0,1]$	$1-\alpha(v)$	B, BT, G, K, L, M	✓	–	–
S-PATH*	$\mathbb{N}^{\infty}$	∞	0	min	$+$	$\in \mathbb{N}$	0	BT, GK, K	✓	✓	–
W-PATH*	$\mathbb{N}^{\infty}$	0	∞	max	min	$\in \mathbb{N}$	∞	BT	✓	✓	–
FUZZY	$[0,1]$	0	1	max	min	$\in [0,1]$	$\in [0,1]$	GK, M	✓	–	–
kWEIGHT	$\{0,\ldots,k\}$	k	0	min	$+^k$	$\in \{0,\ldots,k\}$	$\in \{0,\ldots,k\}$	M	✓	–	–
$\texttt{OBDD}_<$	$\texttt{OBDD}_<(\mathcal{V})$	$\texttt{OBDD}_<(0)$	$\texttt{OBDD}_<(1)$	$\vee$	$\wedge$	$\texttt{OBDD}_<(v)$	$\neg\texttt{OBDD}_<(v)$	K	✓	✓	✓
WHY*	$\mathcal{P}(\mathcal{V})$	$\emptyset$	$\emptyset$	$\cup$	$\cup$	$\{v\}$	$\emptyset$	GK	✓	–	–
$\mathcal{RA}^{+*}$	$\mathbb{N}[\mathcal{V}]$	0	1	$+$	$\cdot$	v	1	GK	–	–	–

Table 1: Examples of AMC tasks based on commutative semirings and labeling functions that have been used in various contexts in the literature. Tasks marked with * are examples of algebraic derivation count tasks mapped to AMC, see Section 3.5 for details. Reference key: B (Bacchus et al., 2009), BT (Baras & Theodorakopoulos, 2010), E (Eisner, 2002), G (Goodman, 1999), GK (Green et al., 2007), K (Kimmig et al., 2011), L (Larrosa et al., 2010), M (Meseguer et al., 2006); more examples can be found in these references. The last three columns classify the tasks based on the semiring properties studied in Sections 3.2 and 3.3: idempotent $\oplus$ (Def. 11), neutral $(\oplus, \alpha)$ (Def. 12), and idempotent and consistency-preserving $(\otimes, \alpha)$ (Def. 13).

of the probabilities of the literals labeled with variables, which can directly be evaluated for various choices of these model parameters. Still within the same probabilistic setting, calculating the gradient with respect to one variable, which is an important subtask in many parameter learning approaches, can be formulated as AMC task as well (Grad). In this case, literal labels are tuples (p_i, g_i) with $p_i \in [0,1]$ the probability of the literal and g_i the gradient with respect to the kth variable[2]:

$$\alpha(v_i) = \begin{cases} (p_i, 1) & \text{if } i = k \\ (p_i, 0) & \text{if } i \neq k \end{cases} \tag{3}$$

$$\alpha(\neg v_i) = \begin{cases} (1-p_i, -1) & \text{if } i = k \\ (1-p_i, 0) & \text{if } i \neq k \end{cases} \tag{4}$$

$$(a_1, a_2) \oplus (b_1, b_2) = (a_1 + b_1, a_2 + b_2) \tag{5}$$

$$(a_1, a_2) \otimes (b_1, b_2) = (a_1 \cdot b_1, a_1 \cdot b_2 + a_2 \cdot b_1) \tag{6}$$

If the second element of the label denotes a cost, the GRAD semiring calculates expected costs.

Another well-known task in the probabilistic setting is finding the probability of the most likely model (MPE), which is formulated as AMC task by using the Prob setting except for $\oplus$, which now is maximization rather than summation.

The next two settings, finding the length of the shortest path (S-Path) and finding the width of the widest path (W-Path), are inspired by optimization tasks in weighted networks.[3] In both cases, positive literals are labeled with a natural number, and negative literals with the neutral element $e^{\otimes}$ of the corresponding semiring multiplication, which ensures that the latter are not taken into account when calculating labels of models. Furthermore, the choice of semiring operators ensures that optimization $\oplus$ always selects the value of a model containing a minimal set (w.r.t. cardinality) of positive literals (corresponding to the edges on a path). In the case of shortest path, $\otimes$ sums labels of literals and $\oplus$ minimizes over these sums, whereas in the case of widest path, $\otimes$ minimizes over labels of literals (thus finding the narrowest part or bottleneck of a path), and $\oplus$ maximizes over those.

The Fuzzy AMC task is closely related to W-Path, also using $\oplus = \max$ and $\otimes = \min$, but assigns values from the interval $[0,1]$ to literals, reflecting their degree of membership in a fuzzy model. The algebraic model count then corresponds to the highest minimal degree of membership of a literal in a model. Similarly, kWeight is closely related to S-Path and also uses $\oplus = \min$, but imposes an upper bound on the value of any model by restricting literal weights to integers $\{0, \ldots, k\}$ and using bounded addition $+^k$ as $\otimes$. The algebraic model count then is the minimal value of a model, or k if no model has a smaller value.

As illustrated by the `OBDD`$_<$ task, labels in AMC can also be complex struc-

[2]By using $(n+1)$-tuples, this can directly be extended to calculate gradients with respect to n variables in parallel.

[3]We will discuss the relationship between algebraic path problems and AMC in Section 3.5.

tures. $\texttt{OBDD}_<$ circuits, which are canonical representations of Boolean functions, are a popular data structure in many fields of computer science, ranging from hardware verification to artificial intelligence. To use them for AMC, we label each literal with its $\texttt{OBDD}_<$ circuit, and set $\oplus$ and $\otimes$ to disjunction and conjunction on $\texttt{OBDD}_<$, respectively.

The last two tasks in the table originate from probabilistic databases under the positive relational algebra $\mathcal{RA}^+$ and are thus defined in terms of (possibly repeated) positive literals only. We will discuss such *algebraic derivation count* (ADC) tasks and their relation to AMC in Section 3.5. In contrast to S-Path and W-Path, which also are instances of ADC originally, expressing Why and $\mathcal{RA}^+$-provenance requires to bring the propositional theory into a specific form; we will come back to the details in Section 3.5. Negative literals are again labeled with the neutral element $e^{\otimes}$ of semiring multiplication. Why-provenance (Why) collects the set of identifiers of all tuples an answer depends on. It labels positive literals with $\alpha(v) = \{v\}$, and uses set union as both $\oplus$ and $\otimes$. $\mathcal{RA}^+$-provenance constructs polynomials that also take into account the number of times the tuples are used. Positive literals are labeled with $\alpha(v) = v$, and $\oplus$ and $\otimes$ are summation and multiplication on polynomials, respectively. Both settings thus provide insight into the way answers to database queries have been derived, and can be used for instance to understand unexpected answers and to identify possible causes of wrong answers.

As a summary of this discussion, we obtain:

Theorem 1. *Evaluation of Boolean formulae (*Bool*), satisfiability (*SAT*), model counting (*#SAT*), weighted model counting (*WMC*), probabilistic inference (*Prob*), sensitivity analysis (*Sens*), gradient (*Grad*), probability of most likely states (*MPE*), shortest (*S-Path*) and widest (*W-Path*) paths, fuzzy (*Fuzzy*) and k-weighted (*kWeight*) constraints, and* $\texttt{OBDD}_<$ *construction are instances of AMC, with the semirings and labeling functions provided in Table 1.*

While all tasks listed in Table 1 are representative examples from the literature, cf. the references given in the table, this is by no means an exhaustive list of semirings and labeling functions that can be used for AMC.

2.2. Related Work

As the examples discussed above illustrate, the AMC task shares its use of semirings with a number of other tasks. The class of sum-of-products problems generalizes factor graphs to the algebraic setting, but uses factors over discrete valued variables as basic building blocks (Bacchus et al., 2009), that is, the task is to compute $\bigoplus_{I(\mathcal{V})} \bigotimes_{i=1}^{n} f_i(E_i)$, where $I(\mathcal{V})$ are all possible value assignments to the set of variables $\mathcal{V}$, and the f_i are functions on sets of variables $E_i \subseteq \mathcal{V}$ taking values from the underlying semiring. In this context, affine algebraic decision diagrams (Sanner & McAllester, 2005) and AND/OR multi-valued decision diagrams (Mateescu et al., 2008) have been used for inference with real-valued semirings. It has been shown before that the factor representation can be transformed into a propositional logic representation by introducing additional variables corresponding to factors (Chavira et al., 2006; Sang et al., 2005). Consider for example an algebraic factor $f(x_1, x_2)$ associating a label with each of the four joint assignments of Boolean variables x_1 and x_2:

x_1	x_2	$f(x_1, x_2)$
0	0	α_{00}
0	1	α_{01}
1	0	α_{10}
1	1	α_{11}

This factor could for instance represent the conditional probabilities $\Pr(x_2|x_1) = f(x_1, x_2)$ for the edge $x_1 \to x_2$ in a Bayesian network. In this case, the equivalent AMC Prob task would consist of one logical equivalence for every row of the table, e.g., $\theta_{01} \equiv \neg x_1 \wedge x_2$ for the second row, and the labeling function would assign $\alpha(\theta_{ij}) = \alpha_{ij}$, and $\alpha(l) = 1$ for all other literals l. This principle generalizes to arbitrary AMC tasks, where we use the neutral element of multiplication $e^{\otimes}$ instead of 1 as the label of all other literals. For more details on the transformation, we refer to Chavira et al. (2006); Sang et al. (2005). The restriction to two-valued variables allows us to directly compile AMC tasks to propositional circuits without adding constraints on legal variable assignments to the theory.

In soft constraint programming, additional constraints are imposed on the semiring, which ensure that addition optimizes the degree of constraint satisfaction (Meseguer et al., 2006). Wilson (2005) provides an algorithm that compiles semiring-based systems into semiring-labelled decision diagrams, which are closely related to unordered binary decision diagrams (also known as free binary decision diagrams or FBDDs), to compute valuations. Semiring-induced propositional logic labels clauses with semiring elements with a weight associated to their falsification and is restricted to semirings whose induced pre-order is partial (Larrosa et al., 2010). In algebraic Prolog (aProbLog), a semiring-labeled logic program is reduced to AMC for inference (Kimmig et al., 2011).

In the context of knowledge compilation and algebraic frameworks, Fargier & Marquis (2007) introduce *valued negation normal form (VNNF)*, a generalization of `NNF` circuits from the Boolean domain to valuation structures over ordered sets, and study the complexity of a range of queries and transformations on different subclasses of VNNFs. In contrast, we generalize a single task (model counting) to the semiring setting and relate it to well-established subclasses of `NNF`. In both cases, properties of the circuit classes such as determinism and decomposability (cf. Sec. 3) as well as of the valuation structure (such as distributivity) play key roles.

While AMC sums over models, other tasks sum over sequences of possibly repeated variables. Examples include algebraic path problems (Baras & Theodorakopoulos, 2010), semiring parsing (Goodman, 1999), provenance semirings for positive relational algebra queries in databases (Green et al., 2007), and semiring-weighted dynamic programs (Eisner et al., 2005). We will discuss the difference between such derivation-based settings and AMC in more detail in Section 3.5.

3. AMC using Knowledge Compilation

Propositional circuits represent Boolean formulae as rooted acyclic graphs where terminal nodes are labeled with literals and inner nodes with Boolean operators applied to their child nodes, cf. Figure 1 for examples. Given such a representation, the underlying formula can be evaluated (as in the Bool task) by a single bottom-up pass from the terminal nodes to the root, which first assigns truth values to literals and then combines truth values of subcircuits at each inner node. In their knowledge compilation map, Darwiche & Marquis (2002) provide an overview of succinctness relationships between various types of propositional circuits. Furthermore, they show which other reasoning tasks in propositional logic, such as (weighted) model counting (#SAT/WMC) or satisfiability checking (SAT), can be evaluated on which circuits in time polynomial in the size of the circuit. In these cases, the operations performed during circuit evaluation are adapted according to the problem at hand, for instance, assigning weights to literals and replacing disjunction by summation and conjunction by multiplication for WMC. Propositional circuits are often used as a representation language in weighted model counting and similar tasks, including for instance probability calculation and sensitivity analysis in probabilistic databases (Jha & Suciu, 2011; Kanagal et al., 2011) and inference in probabilistic and algebraic Prolog (Fierens et al., 2011; Kimmig et al., 2011).

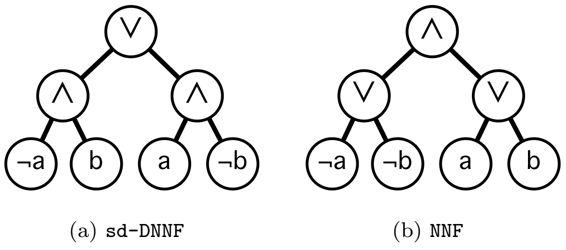
In the following, we extend this approach to AMC, stating a single generic evaluation algorithm that operates on a propositional circuit. Using knowledge compilation for AMC allows one to only perform the expensive compilation step once and to then evaluate the resulting circuit many times, for instance, to repeatedly calculate gradients during parameter learning, to explore different parameter combinations (by keeping some variables' values fixed and varying others) for sensitivity analysis, or to even perform different AMC tasks for the same theory.

In this section, we use conjunction ($\wedge$), disjunction ($\vee$), true ($\top$), false ($\bot$), and propositional literals to denote generic labels of propositional circuits. Given an AMC task, evaluation interprets these as semiring multiplication ($\otimes$), semiring addition ($\oplus$), the neutral element of semiring multiplication ($e^{\otimes}$), the neutral element of semiring addition ($e^{\oplus}$), and the labels $\alpha(l)$ of these literals, respectively. We first repeat the relevant knowledge compilation concepts, closely following Darwiche & Marquis (2002).

Definition 4 (`NNF`)**.** A sentence in *negation normal form* (`NNF`) over a set of propositional variables $\mathcal{V}$ is a rooted, directed acyclic graph where each leaf node is labeled with true ($\top$), false ($\bot$), or a literal of a variable in $\mathcal{V}$, and each internal node with disjunction ($\vee$) or conjunction ($\wedge$).

Definition 5 (Decomposability)**.** An `NNF` is *decomposable* if for each conjunction node $\bigwedge_{i=1}^{n} \phi_i$, no two children ϕ_i and ϕ_j share any variable.

Definition 6 (Determinism)**.** An `NNF` is *deterministic* if for each disjunction node $\bigvee_{i=1}^{n} \phi_i$, each pair of different children ϕ_i and ϕ_j is logically inconsistent.



(a) `sd-DNNF` (b) `NNF`

Figure 1: Example of an `sd-DNNF` and `NNF` circuit.

Definition 7 (Smoothness)**.** An `NNF` is *smooth* if for each disjunction node $\bigvee_{i=1}^{n} \phi_i$, each child ϕ_i mentions the same set of variables.

`DNNF`, `d-NNF`, `s-NNF`, `sd-NNF`, `d-DNNF`, `s-DNNF`, and `sd-DNNF` are the subsets of `NNF` satisfying (combinations of) these properties, where `D` stands for decomposable, `d` for deterministic, and `s` for smooth. For instance, the circuit in Figure 1a is in `sd-DNNF`, while the one in Figure 1b has none of the three properties. `DNF` (disjunctive normal form) is the subset of `NNF` where every sentence is a disjunction of conjunctions, and `MODS` is the subset of `DNF` where every sentence is deterministic and smooth.

A key characteristic when comparing different subsets of `NNF` is their ability to compactly represent propositional sentences, which is captured by the notion of *succinctness.*

Definition 8 (Succinctness (Darwiche & Marquis, 2002))**.** Let L_1 and L_2 be two subsets of `NNF`. L_1 is *at least as succinct* as L_2 iff there exists a polynomial p such that for every sentence $\phi_2 \in L_2$, there exists an equivalent sentence $\phi_1 \in L_1$ with $|\phi_1| \leq p(|\phi_2|)$, where $|\phi_i|$ is the size of ϕ_i.

The algebraic model count $\mathbf{A}(T)$ is defined as a summation over the set of models $\mathcal{M}(T)$ of a propositional theory T, which corresponds to the `MODS` language in the knowledge compilation map. However, as `MODS` is exponentially less succinct than any other representation of T included in the map, converting to `MODS` in order to evaluate Equation (2) directly is undesirable. In the following, we therefore establish a connection between characteristics of AMC tasks and properties of the `NNF` circuits they can be evaluated on, resulting in the classification scheme summarized in Table 2. The last three columns of Table 1 indicate for each example task which of the semiring characteristics it satisfies; the tasks are also included in the corresponding field of Table 2.

The key idea underlying `NNF` evaluation is to perform a bottom-up pass over the circuit, labeling each node with the value of the subcircuit rooted at that node. For disjunction nodes, the values of all their children are combined using $\oplus$, for conjunction nodes using $\otimes$.

Definition 9 (`NNF` Evaluation)**.** The function Eval specified in Algorithm 1

	general $\otimes$		idempotent and consistency-pres. $(\otimes, \alpha)$	
	neutral $(\oplus, \alpha)$	non-neutral $(\oplus, \alpha)$	neutral $(\oplus, \alpha)$	non-neutral $(\oplus, \alpha)$
idempotent $\oplus$	`DNNF` (Th. 5) SAT, S-Path, W-Path	`s-DNNF` (Th. 3) MPE, Fuzzy, kWeight	`NNF` (Th. 7) Bool, $\texttt{OBDD}_<$	`s-NNF` (Th. 7)
non-idempotent $\oplus$	`d-DNNF` (Th. 4) Prob, Sens, Grad	`sd-DNNF` (Th. 2) #SAT, WMC	`d-NNF` (Th. 7)	`sd-NNF` (Th. 6)

Table 2: Semiring characteristics and corresponding circuits that allow for correct AMC evaluation, with example tasks from Table 1.

Algorithm 1 Evaluating an `NNF` circuit N for a commutative semiring $(\mathcal{A}, \oplus, \otimes, e^{\oplus}, e^{\otimes})$ and labeling function α.

```
1: function Eval($N, \oplus, \otimes, e^{\oplus}, e^{\otimes}, \alpha$)
2:   if $N$ is a true node $\top$ then return $e^{\otimes}$
3:   if $N$ is a false node $\bot$ then return $e^{\oplus}$
4:   if $N$ is a literal node $l$ then return $\alpha(l)$
5:   if $N$ is a disjunction $\bigvee_{i=1}^{m} N_i$ then
6:     return $\bigoplus_{i=1}^{m}$ Eval($N_i, \oplus, \otimes, e^{\oplus}, e^{\otimes}, \alpha$)
7:   if $N$ is a conjunction $\bigwedge_{i=1}^{m} N_i$ then
8:     return $\bigotimes_{i=1}^{m}$ Eval($N_i, \oplus, \otimes, e^{\oplus}, e^{\otimes}, \alpha$)
```

evaluates an `NNF` circuit for a commutative semiring $(\mathcal{A}, \oplus, \otimes, e^{\oplus}, e^{\otimes})$ and labeling function α.

Consider for example #SAT for the two circuits in Figure 1, which both represent an exclusive OR of two variables. Evaluation of the `sd-DNNF` in Figure 1a, which in fact is a `MODS` representation, assigns label 1 to each leaf, $1 \cdot 1 = 1$ to each conjunction node, and $1 + 1 = 2$ to the disjunction node at the root and thus the entire circuit, which is correct. On the other hand, evaluation on the general `NNF` in Figure 1b assigns $1 + 1 = 2$ to each disjunction node and $2 \cdot 2 = 4$ to the conjunction node at the root. This overestimation is due to models shared by the children of the same disjunction node and variables shared by the children of the conjunction node, as we will see in more detail in Section 3.2 and 3.3.

Definition 10 (Correctness)**.** Evaluating an `NNF` representation N_T of a propositional theory T for a semiring $(\mathcal{A}, \oplus, \otimes, e^{\oplus}, e^{\otimes})$ and labeling function α is a *correct AMC computation* iff $\textsc{Eval}(N_T, \oplus, \otimes, e^{\oplus}, e^{\otimes}, \alpha) = \mathbf{A}(T)$.

In the following, we establish a general correctness result for AMC evaluation on `sd-DNNF` circuits as well as properties of AMC tasks that guarantee correctness for various other subclasses of `NNF`. Given correctness, we inherit the polynomial complexity results of the knowledge compilation map (Darwiche & Marquis, 2002) for semiring operators with constant cost, cf. also Section 3.4. Note however that there are semirings with more expensive operators. For instance, labels in $\texttt{OBDD}_<$ may grow exponentially in the circuit size.

3.1. `sd-DNNF` *Evaluation*

We show that AMC evaluation is correct on `sd-DNNF` circuits. As these are strictly more succinct than `MODS` representations, they allow for more efficient inference.

Theorem 2 (`sd-DNNF` Evaluation)**.** *Evaluating an* `sd-DNNF` *representation of the propositional theory T is a correct AMC computation.*

Proof. We show that $\textsc{Eval}(N_T, \oplus, \otimes, e^{\oplus}, e^{\otimes}, \alpha)$ for an `sd-DNNF` representation N_T of the theory T computes $\mathbf{A}(T)$ with respect to all variables in N_T:

1. Line 2: $\mathbf{A}(\top) = \bigoplus_{I\in\{\emptyset\}} \bigotimes_{l\in I} \alpha(l) = e^{\otimes}$
2. Line 3: $\mathbf{A}(\bot) = \bigoplus_{I\in\{\}} \bigotimes_{l\in I} \alpha(l) = e^{\oplus}$
3. Line 4: $\mathbf{A}(l) = \bigoplus_{I\in\{\{l\}\}} \bigotimes_{k\in I} \alpha(k) = \alpha(l)$

Due to associativity and commutativity of the semiring operators, operands of each summation and each multiplication can be evaluated in arbitrary order. We therefore restrict ourselves to binary disjunction and conjunction nodes here. In the following, we assume we have already correctly evaluated the two subcircuits ϕ_1 and ϕ_2 over sets of variables $\mathcal{V}_1$ and $\mathcal{V}_2$, respectively. Let the circuits' sets of models with respect to those variables be $\mathcal{M}_1$ and $\mathcal{M}_2$. We now obtain:

4. Lines 5-6: Disjunction node $\phi_1 \vee \phi_2$: For $\phi_1 = \bot$, we have $\mathbf{A}(\bot) \oplus \mathbf{A}(\phi_2) = e^{\oplus} \oplus \mathbf{A}(\phi_2) = \mathbf{A}(\phi_2) = \mathbf{A}(\bot \vee \phi_2)$ by neutrality of $e^{\oplus}$ (and similarly for $\phi_2 = \bot$ by commutativity). As the circuit is deterministic, the ϕ_i cannot be $\top$. We now consider the case where none of the ϕ_i is $\bot$ or $\top$. As the circuit is smooth, we have $\mathcal{V}_1 = \mathcal{V}_2$, and this is also the set of variables used by the circuit rooted at the disjunction. The set of models of this circuit is thus $\mathcal{M}(\phi_1 \vee \phi_2) = \mathcal{M}_1 \cup \mathcal{M}_2$, which is a disjoint union due to determinism. Therefore, $\mathbf{A}(\phi_1) \oplus \mathbf{A}(\phi_2) = \left(\bigoplus_{I\in\mathcal{M}_1} \bigotimes_{l\in I} \alpha(l)\right) \oplus \left(\bigoplus_{I\in\mathcal{M}_2} \bigotimes_{l\in I} \alpha(l)\right) = \bigoplus_{\mathcal{M}_1\cup\mathcal{M}_2} \bigotimes_{l\in I} \alpha(l) = \mathbf{A}(\phi_1 \vee \phi_2)$.
5. Lines 7-8: Conjunction node $\phi_1 \wedge \phi_2$: For the case of $\phi_1 = \top$ (or symmetrically $\phi_2 = \top$), we have $\mathbf{A}(\top) \otimes \mathbf{A}(\phi_2) = e^{\otimes} \otimes \mathbf{A}(\phi_2) = \mathbf{A}(\phi_2) = \mathbf{A}(\top \wedge \phi_2)$ by neutrality of $e^{\otimes}$. For the case of $\phi_1 = \bot$ (or symmetrically $\phi_2 = \bot$), we have $\mathbf{A}(\bot) \otimes \mathbf{A}(\phi_2) = e^{\oplus} \otimes \mathbf{A}(\phi_2) = e^{\oplus} = \mathbf{A}(\bot) = \mathbf{A}(\bot \wedge \phi_2)$ as $e^{\oplus}$ is an annihilator for $\otimes$. We now consider the case involving neither $\top$ nor $\bot$ as one of the disjuncts. As $\mathcal{V}_1 \cap \mathcal{V}_2 = \emptyset$ due to decomposability, we obtain all models of the conjunction by combining each model of ϕ_1 with each model of ϕ_2, that is, $\mathcal{M}(\phi_1 \wedge \phi_2) = \{I_1 \cup I_2 | I_1 \in \mathcal{M}_1, I_2 \in \mathcal{M}_2\}$. Together with distributivity, we get $\mathbf{A}(\phi_1) \otimes \mathbf{A}(\phi_2) = \left(\bigoplus_{I\in\mathcal{M}_1} \bigotimes_{l\in I} \alpha(l)\right) \otimes \left(\bigoplus_{I\in\mathcal{M}_2} \bigotimes_{l\in I} \alpha(l)\right) = \bigoplus_{\mathcal{M}(\phi_1\wedge\phi_2)} \bigotimes_{l\in I} \alpha(l) = \mathbf{A}(\phi_1 \wedge \phi_2)$.

□

Clearly, the correctness of AMC evaluation on `sd-DNNF` depends on all three properties of this subclass of `NNF`. On the other hand, circuits without these properties may be exponentially smaller and thus allow for more efficient inference. In the following, we therefore analyze evaluation in the absence of these properties, which allows us to identify characteristics of the semiring and labeling function that ensure correct evaluation on the corresponding classes of circuits.

3.2. Evaluation on other Decomposable Circuits

If a circuit is not deterministic, children of a disjunction node may have common models, in which case evaluation sums over such shared models multiple times. For instance, consider the circuit in Figure 1b with PROB, $\alpha(a) = 0.6$ and $\alpha(b) = 0.3$. Evaluation on this circuit results in $0.6 + 0.3 = 0.9$ for the right disjunction node, while $\mathbf{A}(a \vee b) = 0.6 \cdot 0.3 + (1 - 0.6) \cdot 0.3 + 0.6 \cdot (1 - 0.3) = 0.72$.

Definition 11 (Idempotent Operator)**.** A binary operator $\odot$ over a set $\mathcal{A}$ is *idempotent* iff $\forall a \in \mathcal{A} : a \odot a = a$.

Theorem 3 (`s-DNNF` Evaluation)**.** *Evaluating an* `s-DNNF` *representation of the propositional theory T for a semiring with idempotent $\oplus$ (a dioid) is a correct AMC computation.*

Proof. Reconsider point (4) of the proof of Theorem 2. Without determinism, $\phi_1 = \top$ is possible, and with smoothness, we have $\phi_2 = \top$ in this case. Then, $\mathbf{A}(\top) \oplus \mathbf{A}(\top) = e^{\otimes} \oplus e^{\otimes} = e^{\otimes} = \mathbf{A}(\top) = \mathbf{A}(\top \vee \top)$ as $\oplus$ is idempotent. The proof for $\phi_1 = \bot$ still holds. If both subformulae involve variables, with smoothness, but without determinism, $\mathcal{M}_1(\phi_1) \cup \mathcal{M}_2(\phi_2)$ is no longer a union of disjoint sets, and $\mathbf{A}(\phi_1) \oplus \mathbf{A}(\phi_2) = \bigoplus_{i=1,2} \bigoplus_{\mathcal{M}_i} \bigotimes_{l \in I} \alpha(l)$ sums over the models in $\mathcal{M}_1(\phi_1) \cap \mathcal{M}_2(\phi_2)$ twice. Due to associativity and commutativity, this is correct for idempotent $\oplus$. □

If a circuit is not smooth, the children of a disjunction node may use different sets of variables. Each model of a child node corresponds to a set of models for the full set of variables, but evaluation on a non-smooth circuit ignores the labels of unmentioned variables. For instance, consider the circuit in Figure 1b with MPE, $\alpha(a) = 0.6$ and $\alpha(b) = 0.3$. Evaluating the right disjunction node of this circuit results in $\max(0.6, 0.3) = 0.6$, while $\mathbf{A}(a \vee b) = \max(0.6 \cdot 0.3, (1 - 0.6) \cdot 0.3, 0.6 \cdot (1 - 0.3)) = 0.42$.

Definition 12 (Neutral $(\oplus, \alpha)$)**.** A semiring addition and labeling function pair $(\oplus, \alpha)$ is *neutral* iff $\forall v \in \mathcal{V} : \alpha(v) \oplus \alpha(\neg v) = e^{\otimes}$.

For instance, the semiring addition and labeling function of the PROB task in Table 1 are neutral, as for all $v \in \mathcal{V}$, we have $\alpha(v) \oplus \alpha(\neg v) = \alpha(v) + (1 - \alpha(v)) = 1 = e^{\otimes}$ in this case.

Theorem 4 (`d-DNNF` Evaluation)**.** *Evaluating a* `d-DNNF` *representation of the propositional theory T for a semiring and labeling function with neutral $(\oplus, \alpha)$ is a correct AMC computation.*

Proof. Reconsider point (4) of the proof of Theorem 2. The cases involving $\top$ or $\bot$ still hold. For the remaining case of both subformulae involving variables to be correct, the sum of the AMCs computed by the children over their sets of variables $\mathcal{V}_i$ has to be equal to the AMC of the entire disjunction over the full set of variables $\mathcal{V}_1 \cup \mathcal{V}_2$. Given the child AMC $\mathbf{A}_{\mathcal{V}_i}(\phi_i)$, adding a variable v to $\mathcal{V}_i$ replaces each model I of ϕ_i by two models $I^+ = I \cup \{v\}$ and $I^- = I \cup \{\neg v\}$. Due to distributivity, commutativity and the neutral sum property, the algebraic sum of these two models equals the AMC of the original model:

$$\begin{aligned}
\mathbf{A}_{\mathcal{V}_i \cup \{v\}}(I) &= \mathbf{A}_{\mathcal{V}_i \cup \{v\}}(I^+) \oplus \mathbf{A}_{\mathcal{V}_i \cup \{v\}}(I^-) \\
&= (\alpha(v) \oplus \alpha(\neg v)) \otimes \bigotimes_{l \in I} \alpha(l) \\
&= \bigotimes_{l \in I} \alpha(l) = \mathbf{A}_{\mathcal{V}_i}(I)
\end{aligned}$$

Evaluation therefore computes $\mathbf{A}_{\mathcal{V}_1}(\phi_1) \oplus \mathbf{A}_{\mathcal{V}_2}(\phi_2) = \mathbf{A}_{\mathcal{V}_1 \cup \mathcal{V}_2}(\phi_1) \oplus \mathbf{A}_{\mathcal{V}_1 \cup \mathcal{V}_2}(\phi_2)$, which due to determinism is equal to $\mathbf{A}_{\mathcal{V}_1 \cup \mathcal{V}_2}(\phi_1 \vee \phi_2)$. $\square$

Note that from a complexity point of view, non-neutral $(\oplus, \alpha)$ does not influence the tractability of inference, as any `NNF` can be smoothed in polytime preserving determinism and decomposability (Darwiche & Marquis, 2002).

The previous two results can directly be combined for `DNNF` circuits that are neither smooth nor deterministic.

Theorem 5 (`DNNF` Evaluation)**.** *Evaluating a* `DNNF` *representation of the propositional theory T for a semiring and labeling function with idempotent and neutral $(\oplus, \alpha)$ is a correct AMC computation.*

Proof. Reconsider point (4) of the proof of Theorem 2. Due to neutral $(\oplus, \alpha)$, values for all children of a disjunction node (including $\top$) are correct with respect to the full set of variables (cf. proof of Theorem 4). Due to idempotent $\oplus$, multiple occurrences of the same model do not influence the result (cf. proof of Theorem 3). $\square$

This completes the left part of Table 2, where no conditions are imposed on semiring multiplication.

3.3. Evaluation on Non-Decomposable Circuits

If a circuit is not decomposable, the children of a conjunction node may share variables. In this case, simply combining results for all pairs of their models may produce results corresponding to (multi-)sets of literals that are not models, because they either contain contradicting literals, or several copies of the same literal, which results in erroneous extra multiplications. For instance, in Figure 1b, $\{\neg a, b\}$ is a model of both disjunction nodes, and $\{a, b\}$ of the right one only. The conjunction node sums among others the products $\alpha(\neg a) \otimes \alpha(b) \otimes \alpha(a) \otimes \alpha(b)$, which does not correspond to a model, and $\alpha(\neg a) \otimes \alpha(b) \otimes \alpha(\neg a) \otimes \alpha(b)$, where labels of both literals are multiplied twice.

Definition 13 (Consistency-Preserving $(\otimes, \alpha)$)**.** A semiring multiplication and labeling function pair $(\otimes, \alpha)$ is *consistency-preserving* iff $\forall v \in \mathcal{V} : \alpha(v) \otimes \alpha(\neg v) = e^{\oplus}$.

For instance, Boolean evaluation (BOOL) is consistency-preserving, as $\alpha(v) \wedge \neg\alpha(v) = \mathit{false} = e^{\oplus}$.

Theorem 6 (`sd-NNF` Evaluation)**.** *Evaluating an* `sd-NNF` *representation of the propositional theory T for a semiring and labeling function with idempotent and consistency-preserving $(\otimes, \alpha)$ is a correct AMC computation.*

Proof. Reconsider point (5) of the proof of Theorem 2. The set of models of $\phi_1 \wedge \phi_2$ contains exactly all pairwise combinations of models of its parts that agree on all shared variables. The circuit evaluates the AMC of $\phi_1 \wedge \phi_2$ as $\mathbf{A}(\phi_1) \otimes \mathbf{A}(\phi_2)$, which due to distributivity is the sum over all pairwise combinations of models. Without decomposability, each such combination I contains two literals for each $v \in \mathcal{V}_1 \cap \mathcal{V}_2$. As $\otimes$ is associative, commutative and idempotent, repeated occurrences of a literal l in $\otimes_{i \in I} \alpha(i)$ do not affect the result. If $\{l, \neg l\} \subseteq I$, $\otimes_{i \in I} \alpha(i)$ includes a multiplication by $\alpha(l) \otimes \alpha(\neg l) = e^{\oplus}$, which in a semiring means $\otimes_{i \in I} \alpha(i) = e^{\oplus}$. Such inconsistent I thus do not contribute to the semiring sum. $\square$

Theorem 6 affects only conjunction nodes, whereas Theorems 3, 4 and 5 only affect disjunction nodes. Their combination thus extends our results to non-decomposable circuits that do not satisfy (one of) the other two properties either:

Theorem 7 (`s-NNF`, `d-NNF`, and `NNF` Evaluations)**.** *For a semiring and labeling function with idempotent and consistency preserving $(\otimes, \alpha)$, evaluating the following representation of the propositional theory T is a correct AMC computation:*

- `s-NNF` *if $\oplus$ is idempotent*
- `d-NNF` *if $(\oplus, \alpha)$ is neutral*
- `NNF` *if $(\oplus, \alpha)$ is idempotent and neutral*

This completes the right part of Table 2, where using non-decomposable circuits is possible as a consequence of restrictions on semiring multiplication. Given a new AMC instance, this table allows one to immediately choose the appropriate type of circuit for efficient evaluation. Table 2 provides this classification for the examples discussed earlier, cf. Table 1.

	neutral $(\oplus, \alpha)$	non-neutral $(\oplus, \alpha)$
idempotent $\oplus$	`DNF`	`s-DNF`
non-idempotent $\oplus$	`d-DNF` `OBDD`	`sd-DNF` `s-OBDD`

Table 3: Overview of AMC settings used in algebraic Prolog (Kimmig et al., 2011).

3.4. Discussion

The results summarized in Table 2 provide a unified view on a number of known results. They generalize the complexity results for evaluation of SAT and #SAT using knowledge compilation to broad classes of tasks, provide more succinct types of circuits for inference in algebraic Prolog, and show that all circuits that are practically relevant for AMC are well-studied in the knowledge compilation map. We now address these points in more detail.

First, Darwiche & Marquis (2002) show that SAT is correctly evaluated in polynomial time on `DNNF`, while #SAT is correctly evaluated in polynomial time on `d-DNNF`, as the required smoothing to obtain an `sd-DNNF` is possible in polynomial time. We generalize these results to broad classes of semirings, always ensuring correctness, and, as discussed above, preserving polynomial time complexity as long as each semiring operation has constant cost. More specifically, our Theorem 5 generalizes correctness of `DNNF` evaluation from SAT to all commutative semirings and labeling functions with idempotent and neutral $(\oplus, \alpha)$, and our Theorem 2 generalizes correctness of `sd-DNNF` evaluation from #SAT to arbitrary commutative semirings and labeling functions. As part of their tractability study of inference in weighted bases, that is, propositional theories with penalties on unsatisfied formulae, Darwiche & Marquis (2004) have shown that the weight of a weighted base in normal form, that is, with all penalties on literals, can efficiently be computed on its `DNNF` representation. This directly translates to an instance of AMC with $\oplus = \min$ and $\otimes = +$ that satisfies our criteria for `DNNF` evaluation.

Second, Kimmig et al. (2011) reduce inference in algebraic Prolog (aProbLog) to AMC evaluation on disjunctive normal form (`DNF`). For non-idempotent addition, a `DNF` whose conjunctions are not mutually exclusive is then compiled into an ordered binary decision diagram (`OBDD`). For non-neutral $(\oplus, \alpha)$, circuits are smoothed before evaluation. This results in the settings listed in Table 3. Table 2 uses the same characteristics of semiring operators and labeling function, but does not require to transform the theory to a `DNF` as starting point. The left half of Table 2 directly generalizes the aProbLog scheme to strictly more succinct superclasses of circuits, namely (`s-`)`DNNF` instead of (`s-`)`DNF`, and (`s`)`d-DNNF` instead of (`s`)`d-DNF` or (`s-`)`OBDD`. As probabilistic inference in ProbLog has recently been improved by replacing `OBDD`-based approaches with weighted model counting on `sd-DNNF` (Fierens et al., 2011), our results promise practical improvements for aProbLog as well.

Third, we observe that while there are interesting inference tasks that are correctly evaluated on the more succinct class of `DNNF` instead of the general `sd-DNNF` evaluation, the conditions for correct evaluation on non-decomposable circuits are too strict in most practical cases. This is in line with the knowledge compilation map, which excludes non-decomposable circuits (with the exception of the most general class `NNF`) as they do not support any of the studied tasks in polytime (Darwiche & Marquis, 2002).

Fourth, extending weighted model counting towards negative probabilities makes it possible to use existential quantification when defining probabilistic models based on first order logic (Van den Broeck et al., 2014; Beame et al., 2015). The correctness of this generalization follows immediately from our results.

Finally, we note that our results imply general complexity bounds for AMC in terms of the size of the original logical theory T. For example, when T is given in conjunctive normal form (`CNF`), an equivalent `sd-DNNF` can be compiled in time polynomial in the size of T and exponential in its *treewidth* (Darwiche, 2001). Hence, all AMC tasks in `CNF` with constant-cost $\oplus$ and $\otimes$ can be evaluated in time polynomial in the size and exponential in the treewidth of T, and are thus tractable for bounded treewidth. Similar results for (`d-`)`NNF` state that these circuits can be compiled starting from `CNF` or `DNF` in time polynomial in the size of T (Darwiche & Marquis, 2002, Lemma A.8).

3.5. AMC and Algebraic Derivation Counting

While AMC is a sum over models, or sets of literals, many other semiring-based tasks require a sum over derivations, that is, sequences of possibly repeated variables. Examples include algebraic path problems (Baras & Theodorakopoulos, 2010), semiring parsing (Goodman, 1999), provenance semirings for positive relational algebra queries in databases (Green et al., 2007), and semiring-weighted dynamic programs (Eisner et al., 2005). We refer to this type of task as *algebraic derivation counting* (ADC), and restrict the discussion to the case of finite, distinct sequences.

Definition 14 (ADC Problem)**.** Given

- a set S of finite sequences of variables from a set $\mathcal{V}$,
- a *commutative semiring* $(\mathcal{A}, \oplus, \otimes, e^{\oplus}, e^{\otimes})$, and
- a *labeling function* $\delta : \mathcal{V} \rightarrow \mathcal{A}$, mapping variables in $\mathcal{V}$ to values from the semiring set $\mathcal{A}$,

the task of *algebraic derivation counting (ADC)* is to compute

$$\mathbf{D}(S) = \bigoplus_{(v_1,\ldots,v_n)\in S} \bigotimes_{i=1}^{n} \delta(v_i). \tag{7}$$

For instance, consider a graph with three nodes s, t and r, and three directed edges $e_1 = (s,t)$, $e_2 = (s,r)$ and $e_3 = (r,t)$. A derivation in this context is a path in the graph represented as its sequence of edges, e.g., (e_2, e_3) represents the path from s to t via r. The ADC of the set of all paths from s to t in the graph is then given by

$$\mathbf{D}(\{(e_1), (e_2, e_3)\}) = \delta(e_1) \oplus (\delta(e_2) \otimes \delta(e_3)) \tag{8}$$

For instance, if δ assigns a cost to every edge, we could use the S-Path or W-Path semirings to compute the cost of the shortest or widest path, respectively.

As a second example, consider the following context free grammar, where we add the variable we will use to refer to an application of the rule in a derivation in parentheses:

S $\rightarrow$ aS	(s_1)	A $\rightarrow$ AA	(a_1)
S $\rightarrow$ AA	(s_2)	A $\rightarrow$ a	(a_2)
S $\rightarrow \epsilon$	(s_3)		

The word aa has two leftmost derivations in this grammar: $S \rightarrow aS \rightarrow aaS \rightarrow aa$, using the rule sequence (s_1, s_1, s_3), and $S \rightarrow AA \rightarrow aA \rightarrow aa$, using (s_2, a_2, a_2). The ADC of this set of derivations is given by

$$\mathbf{D}(\{(s_1, s_1, s_3), (s_2, a_2, a_2)\}) = (\delta(s_1) \otimes \delta(s_1) \otimes \delta(s_3)) \oplus (\delta(s_2) \otimes \delta(a_2) \otimes \delta(a_2)) \tag{9}$$

For instance, using the probability semiring, this corresponds to computing the probability of aa in a probabilistic context free grammar with these rules.[4]

While both ADC and AMC are semiring sums over semiring products, their key difference is that computing ADC is based on the *structure* of a set of sequences of positive variables, whereas computing the AMC is based on the *models* of a propositional theory represented as an arbitrary `NNF`. Nevertheless, it is possible to map each of the tasks onto the other, as we will show now.

Reducing AMC to ADC. Reducing an AMC task to an ADC task is straightforward. The set of variables in the ADC task contains one variable for each literal in the AMC task, labeled with that literal's label. For each model of the AMC theory T, the ADC derivation set S contains one sequence enumerating exactly the variables corresponding to the literals in the model. Then, Equation (7) and Equation (2) have the same structure, and the ADC of the translation thus equals the original AMC.

However, this reduction is clearly not desirable from a complexity point of view, as it requires bringing T into `MODS` form. Alternatively, one could adapt the semiring used. For instance, Baras & Theodorakopoulos (2010) provide an ADC encoding of network reliability under probabilistic edge failure, that is, the Prob AMC task for a positive propositional formula. They essentially modify multiplication to filter repeated literals from derivations (i.e., repeated edges from cyclic paths), and addition to subtract shared models of its operands

[4]Note that the input to ADC is a set of finite derivations, not the grammar, i.e., ADC does not solve the parsing problem.

(i.e., subgraphs containing multiple paths), which, while operating on the ADC structure, drastically increases complexity of these operations. Under which general conditions such transformations are possible is an open question.

Reducing ADC to AMC. To reduce an arbitrary ADC task to AMC, we construct a propositional theory that has one model for every sequence in the ADC task. The construction (a) systematically uses different variables for repeated occurrences of the same variable in a derivation, and (b) makes explicit that (renamed) variables not appearing in a derivation do not contribute to the semiring product by adding their negation, labeled with $e^{\otimes}$, to the derivation. That is, if variable v occurs at most k times in any sequence $s \in S$, we introduce k copies $v^1, \ldots, v^k$, all labeled with $\delta(v)$. Thus, Equation (9) becomes

$$\mathbf{D}(\{(s_1^1, s_1^2, s_3^1), (s_2^1, a_2^1, a_2^2)\}) = (\delta(s_1^1) \otimes \delta(s_1^2) \otimes \delta(s_3^1)) \oplus (\delta(s_2^1) \otimes \delta(a_2^1) \otimes \delta(a_2^2))$$

Clearly, this construction in general does not change the ADC. Second, we expand derivations to all variables by adding negative literals, labeled with $e^{\otimes}$[5]:

$$\begin{aligned}\mathbf{D}(\{(s_1^1, s_1^2, s_3^1, \neg s_2^1, \neg a_2^1, \neg a_2^2), (s_2^1, a_2^1, a_2^2, \neg s_1^1, \neg s_1^2, \neg s_3^1)\})&\\ =(\delta(s_1^1) \otimes \delta(s_1^2) \otimes \delta(s_3^1) \otimes \delta(\neg s_2^1) \otimes \delta(\neg a_2^1) \otimes \delta(\neg a_2^2))&\\ \oplus (\delta(s_2^1) \otimes \delta(a_2^1) \otimes \delta(a_2^2) \otimes \delta(\neg s_1^1) \otimes \delta(\neg s_1^2) \otimes \delta(\neg s_3^1))&\end{aligned}$$

This step again maintains the ADC, as can easily be verified using the properties of commutative semirings. Furthermore, as we now have a 1-1-correspondence between derivations and models, setting α to δ and T to the disjunction of the conjunctions of elements in derivations, i.e., $(s_1^1 \wedge s_1^2 \wedge s_3^1 \wedge \neg s_2^1 \wedge \neg a_2^1 \wedge \neg a_2^2) \vee (s_2^1 \wedge a_2^1 \wedge a_2^2 \wedge \neg s_1^1 \wedge \neg s_1^2 \wedge \neg s_3^1)$, completes the reduction to AMC.

While this two-step reduction from ADC to AMC on a `MODS` representation is possible for any commutative semiring and ADC labeling function δ, for some semirings, AMC effectively selects the desired model for a derivation without need to make this explicit in the propositional theory.

First, for commutative semirings and labeling functions with idempotent, consistency-preserving $(\otimes, \alpha)$ and idempotent, neutral $(\oplus, \alpha)$ such as for instance $\texttt{OBDD}_<$, neither variable renaming nor model restriction are required. The reason is that repetition within derivations is handled by idempotent $\otimes$, whereas idempotent, neutral $(\oplus, \alpha)$ implies that summing out unmentioned variables maps them to the label of the negative literal, $e^{\otimes}$ (cf. Theorem 5). However, as noted above, such tasks are rare.

Second, if derivations do not contain repeated variables, there is no need to rename variables. If furthermore $\oplus$ is idempotent and $(\oplus, \alpha)$ is neutral, as in the first case, we do not need to explicitly restrict models. This is for instance the case for the S-PATH and W-PATH semirings in Table 1. Intuitively, if we do not restrict models, instead of directly summing over the products for all (acyclic)

[5]Strictly speaking, ADC would require new, positive variables for those negative literals, but we use negation for ease of exposition.

paths, AMC additionally sums for each path over all subgraphs containing that path, but the properties of the semiring and labeling function ensure that the results coincide. For instance, as absent edges are labeled with the minimal value 0 in shortest path, among all graphs containing the same path, semiring addition (using min) effectively selects the one containing no additional edges.

On the other hand, the Why and $\mathcal{RA}^+$ semirings also listed in Table 1 do not fall into this category (as can be verified by inspecting the properties of their definitions), but require the full translation. Intuitively, Why collects all variables appearing in derivations, or, in its original database context, all tuples contributing to a query answer. Evaluating this on all models instead of the models constructed in the translation would result in the full set of variables (all tuples in the database) instead of just the relevant ones. $\mathcal{RA}^+$ refines Why by constructing a polynomial that keeps track of how often each variable (positively) contributes to each derivation. Again, summing over all extensions of a derivation would add too much information to the result.

4. Conclusions and Future Work

We have introduced the task of algebraic model counting, which generalizes weighted model counting to a semiring setting and thus to various types of labels, including numerical ones as used in WMC, but also sets, polynomials, or Boolean formulae. We have shown that evaluating AMC is correct on `sd-DNNF` circuits, which are known to be more succinct than the `MODS` language used in the problem definition. Furthermore, we have provided characteristics of AMC tasks that guarantee correct evaluation on more succinct classes of circuits, which provides a means of directly choosing a circuit type that allows for efficient inference given a new AMC task. AMC also provides a unified view on a number of known results as well as a framework to connect algebraic derivation counts to AMC tasks.

Given the results presented here, it is worth investigating which other algebraic representations can be reduced to algebraic model counting. Another line of future work concerns the introduction of additional operators that would make it possible to express additional tasks, for instance, partial MAP, which requires a maximization operator in addition to summation and multiplication.

Acknowledgements. This work has been supported by the Research Foundation Flanders (FWO Vlaanderen).

References

Bacchus, F., Dalmao, S., & Pitassi, T. (2009). Solving #SAT and Bayesian inference with backtracking search. *Journal of Artificial Intelligence Research*, *34*, 391–442.

Baras, J. S., & Theodorakopoulos, G. (2010). *Path Problems in Networks* volume 3 of *Synthesis Lectures on Communication Networks*. Morgan & Claypool Publishers.

Beame, P., Van den Broeck, G., Gribkoff, E., & Suciu, D. (2015). Symmetric weighted first-order model counting. In *Proceedings of the 34th ACM Symposium on Principles of Database Systems (PODS)* (pp. 313–328).

Chavira, M., Darwiche, A., & Jaeger, M. (2006). Compiling relational Bayesian networks for exact inference. *International Journal of Approximate Reasoning*, *42*, 4 – 20.

Darwiche, A. (2001). On the tractable counting of theory models and its application to truth maintenance and belief revision. *Journal of Applied Non-Classical Logics*, *11*, 11–34.

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

Darwiche, A., & Marquis, P. (2002). A knowledge compilation map. *Journal of Artificial Intelligence Research*, *17*, 229–264.

Darwiche, A., & Marquis, P. (2004). Compiling propositional weighted bases. *Artificial Intelligence*, *157*, 81–113.

Eisner, J. (2002). Parameter estimation for probabilistic finite-state transducers. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL)* (pp. 1–8). Philadelphia.

Eisner, J., Goldlust, E., & Smith, N. (2005). Compiling Comp Ling: Weighted dynamic programming and the Dyna language. In *Proceedings of the Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing (HLT/EMNLP)* (pp. 281–290).

Fargier, H., & Marquis, P. (2007). On valued negation normal form formulas. In *IJCAI* (pp. 360–365).

Fierens, D., Van den Broeck, G., Thon, I., Gutmann, B., & De Raedt, L. (2011). Inference in probabilistic logic programs using weighted CNF's. In *Proceedings of the 27th Conference on Uncertainty in Artificial Intelligence (UAI)* (pp. 211–220).

Goodman, J. (1999). Semiring parsing. *Computational Linguistics*, *25*, 573–605.

Green, T. J., Karvounarakis, G., & Tannen, V. (2007). Provenance semirings. In *Proceedings of the 26th ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems (PODS)* (pp. 31–40).

Jha, A. K., & Suciu, D. (2011). Knowledge compilation meets database theory: compiling queries to decision diagrams. In *Proceedings of the 14th International Conference on Database Theory (ICDT)* (pp. 162–173).

Kanagal, B., Li, J., & Deshpande, A. (2011). Sensitivity analysis and explanations for robust query evaluation in probabilistic databases. In *Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD)* (pp. 841–852).

Kimmig, A., Van den Broeck, G., & De Raedt, L. (2011). An algebraic Prolog for reasoning about possible worlds. In *Proceedings of the 25th AAAI Conference on Artificial Intelligence (AAAI)* (pp. 209–214).

Larrosa, J., Oliveras, A., & Rodríguez-Carbonell, E. (2010). Semiring-induced propositional logic: Definition and basic algorithms. In E. M. Clarke, & A. Voronkov (Eds.), *Revised Selected Papers of the 16th International Conference on Logic for Programming, Artificial Intelligence, and Reasoning (LPAR)* (pp. 332–347). Springer volume 6355 of *Lecture Notes in Computer Science*.

Mateescu, R., Dechter, R., & Marinescu, R. (2008). And/or multi-valued decision diagrams (AOMDDs) for graphical models. *Journal of Artificial Intelligence Research*, *33*, 465–519.

Meseguer, P., Rossi, F., & Schiex, T. (2006). Soft constraints. In F. Rossi, P. Van Beek, & T. Walsh (Eds.), *Handbook of constraint programming* (pp. 281–328). Elsevier Science.

Park, J. D. (2002). Using weighted MAX-SAT engines to solve MPE. In *Proceedings of the 18th AAAI Conference on Artificial Intelligence (AAAI)* (pp. 682–687).

Sang, T., Beame, P., & Kautz, H. (2005). Solving Bayesian networks by weighted model counting. In *Proceedings of the 20th AAAI Conference on Artificial Intelligence (AAAI)* (pp. 475–482).

Sanner, S., & McAllester, D. A. (2005). Affine algebraic decision diagrams (AADDs) and their application to structured probabilistic inference. In *Proceedings of the 19th International Joint Conference on Artificial Intelligence (IJCAI)* (pp. 1384–1390).

Van den Broeck, G., Meert, W., & Darwiche, A. (2014). Skolemization for weighted first-order model counting. In *Proceedings of the 14th International Conference on Principles of Knowledge Representation and Reasoning (KR)* (pp. 1–10).

Van den Broeck, G., Taghipour, N., Meert, W., Davis, J., & De Raedt, L. (2011). Lifted probabilistic inference by first-order knowledge compilation. In *Proceedings of the 22nd International Joint Conference on Artificial Intelligence (IJCAI)* (pp. 2178–2185).

Wilson, N. (2005). Decision diagrams for the computation of semiring valuations. In *Proceedings of the 19th International Joint Conference on Artificial Intelligence (IJCAI)* (pp. 331–336).