Journal of Artificial Intelligence Research 77 (2023) 207-293 Submitted 06/2022; published 05/2023

Semiring Reasoning Frameworks in AI and Their Computational Complexity

Thomas Eiter THOMAS.EITER@TUWIEN.AC.AT
Rafael Kiesel RAFAEL.KIESEL@WEB.DE
Knowledge Based Systems Group
Institute of Logic and Computation
Vienna University of Technology (TU Wien)
Favoritenstraße 9-11
1040 Vienna, Austria

Abstract

Many important problems in AI, among them #SAT, parameter learning and probabilistic inference go beyond the classical satisfiability problem. Here, instead of finding a solution we are interested in a quantity associated with the set of solutions, such as the number of solutions, the optimal solution or the probability that a query holds in a solution. To model such quantitative problems in a uniform manner, a number of frameworks, e.g. Algebraic Model Counting and Semiring-based Constraint Satisfaction Problems, employ what we call the *semiring paradigm*. In the latter the abstract algebraic structure of the semiring serves as a means of parameterizing the problem definition, thus allowing for different modes of quantitative computations by choosing different semirings. While efficiently solvable cases have been widely studied, a systematic study of the computational complexity of such problems depending on the semiring parameter is missing. In this work, we characterize the latter by $\mathrm{NP}(\mathcal{R})$, a novel generalization of NP over semiring $\mathcal{R}$, and obtain $\mathrm{NP}(\mathcal{R})$-completeness results for a selection of semiring frameworks. To obtain more tangible insights into the hardness of $\mathrm{NP}(\mathcal{R})$, we link it to well-known complexity classes from the literature. Interestingly, we manage to connect the computational hardness to properties of the semiring. Using this insight, we see that, on the one hand, $\mathrm{NP}(\mathcal{R})$ is always at least as hard as NP or $\mathrm{Mod}_p\mathrm{P}$ depending on the semiring $\mathcal{R}$ and in general unlikely to be in $\mathrm{FPSpace(poly)}$. On the other hand, for broad subclasses of semirings relevant in practice we can employ reductions to NP, $\mathrm{Mod}_p\mathrm{P}$ and #P. These results show that in many cases solutions are only mildly harder to compute than functions in NP, $\mathrm{Mod}_p\mathrm{P}$ and #P, give us new insights into how problems that involve counting on semirings can be approached, and provide a means of assessing whether an algorithm is appropriate for a given class of problems.

1. Introduction

Especially in the last decades, there has been a big influx of new quantitative problems in AI as well as a largely increased interest in quantitative versions of well-known qualitative problems and their solution. Prototypical problems involve probabilistic reasoning over Bayesian networks (Pearl, 1985, 2014; Heckerman, 2008; Niedermayer, 2008) or propositional theories (De Raedt, Kimmig, & Toivonen, 2007; Baral, Gelfond, & Rushton, 2009; Lee & Yang, 2017; Sato & Kameya, 1997), counting the models of a logical theory (Valiant, 1979), learning the entailment relation of a logical theory (Khardon & Roth, 1997) or finding

an optimal solution to a logical theory (Brailsford, Potts, & Smith, 1999; Erdem, Gelfond, & Leone, 2016; Kautz & Selman, 1996; Li & Manyà, 2021). More recent advances go even further than that by computing not only probabilities but also expected utilities (Van den Berg, Van Bremen, Derkinderen, Kimmig, Schrijvers, & De Raedt, 2021; Van den Broeck, Thon, Van Otterlo, & De Raedt, 2010) or by learning parameterized probability distributions (Manhaeve, Dumancic, Kimmig, Demeester, & De Raedt, 2019; Skryagin, Stammer, Ochs, Dhami, & Kersting, 2021) in neuro-symbolic reasoning. Others introduced advanced mechanisms to formulate preferences over models (Brewka, Delgrande, Romero, & Schaub, 2015; Ruttkay, 1994) or the ability to capture the lineage of data (Cui, 2002).

This list could be continued and already its length posed the question of how these problems could be approached in a uniform manner. At this point *semirings* come into play. These are algebraic structures $\mathcal{R} = (R, \oplus, \otimes, e_\oplus, e_\otimes)$ consisting of a nonempty set R of values together with two binary operations $\oplus$ and $\otimes$ called addition and multiplication, respectively, having neutral elements $e_\oplus$ and $e_\otimes$, respectively, that satisfy a set of axioms. They provide an abstract, algebraic perspective of looking at quantitative computations. As such, they are a well-suited basis for defining quantitative frameworks that uniformly capture a multitude of problems by introducing semantics depending on a semiring parameter. Indeed, Kimmig, Van den Broeck, and De Raedt (2017) and Belle and De Raedt (2020) provide an extensive list of problems, including the ones mentioned above, that are easily formulated by instantiating a semiring framework with different semirings. To name a few, SAT corresponds to the Boolean semiring $\mathbb{B}$; #SAT corresponds to the semiring of the natural numbers $\mathbb{N}$; parameter learning corresponds to the gradient semiring (Manhaeve et al., 2019); and expected utility computations are handled by the expectation semiring (Eisner, 2002).

This semiring paradigm, which allows us to introduce a multitude of quantitative versions of a qualitative problem in a uniform manner, was fruitfully used in a manifold of ways, including but not limited to:

- Algebraic Model Counting (AMC) (Kimmig et al., 2017), which generalizes Weighted Model Counting to work with semiring values as weights;
- Semiring-based Constraint Satisfaction Problems (SCSPs), which were shown to encompass Fuzzy CSPs, Probabilistic CSPs, Weighted CSPs and more, each corresponding to SCSPs over some semiring (Bistarelli, Montanari, Rossi, Schiex, Verfaillie, & Fargier, 1999);
- Answer Set Programming (ASP) with Algebraic Constraints (Eiter & Kiesel, 2020), which is an extension of ASP that captures and generalizes many previous quantitative extensions of ASP in a uniform manner;
- Provenance Semirings (Green, Karvounarakis, & Tannen, 2007), which were shown to be a useful tool to express which-, why- and bag-why-provenance in the context of positive relational algebra queries, by employing polynomial semirings;
- algebraic Prolog (Kimmig, Van den Broeck, & De Raedt, 2011), which is not only available as an implementation for probabilistic logic programming, but also capable of parameter learning and offers most probable explanation inference by employing specialized semirings.

This naturally begs the question, already posed by Belle and De Raedt (2020), of how such general frameworks with a semiring parameter be efficiently implemented.

Obtaining a solution that *works* for general semirings turns out to be feasible via knowledge compilation (Mengel, 2021; Darwiche & Marquis, 2002). Kimmig et al. (2017) showed that if the underlying logical theory is compiled into sd-DNNF representation TC, then we can solve weighted model counting instances over *any* semiring using polynomially (in $|TC|$) many additions and multiplications. This is also the strategy that current implementations like ProbLog (De Raedt et al., 2007), aspmc (Eiter, Hecher, & Kiesel, 2021) and ProvSQL (Senellart, Jachiet, Maniu, & Ramusat, 2018) predominantly employ. As Kimmig et al. (2017) also noted, depending on the specific semiring in use, this is clearly far from the most efficient solution. Knowledge compilation to sd-DNNF allows one to solve #SAT in polynomial time after compilation and is thus #P-hard. While this is less of an issue when we are evaluating a problem over the semiring $\mathbb{N}$ of the natural numbers, which has been shown to lead to #P-hardness, it is for instance an issue for the Boolean semiring $\mathbb{B}$ leading to instances that are in NP. Toda's Theorem (Toda, 1989) shows that not only NP but even the whole polynomial hierarchy PH is contained in $\mathrm{P}^{\#\mathrm{P}[1]}$, i.e., each problem in PH can be solved in polynomial time with a single call to an #P oracle (tantamount to solving a single #SAT instance). This gives us the intuition that there is likely a large gap between NP and #P and, thus, also between the semirings $\mathbb{B}$ and $\mathbb{N}$.

We see that to answer the question of what an efficient implementation must look like, we first need to understand how the computational complexity behaves depending on the semiring parameter. Here, according to the best of our knowledge, current complexity results for general semirings are still preliminary.[1] Completeness results are only known for a few specific semirings (Stearns & III, 1996), NP-hardness is only known over *idempotent* semirings (Bistarelli et al., 1999) and the EXPTIME upper bound that follows from the results of Kimmig et al. (2017) assumes that multiplication and addition can be done in constant time.

Problems studied. In this paper, we therefore take a closer look at the computational complexity of reasoning frameworks that depend on a semiring parameter such as the ones above. For this purpose, we first introduce a new machine model called Semiring Turing Machines (SRTMs) that can solve sum-of-product like problems over semirings while using the semiring in a black-box fashion. Based on SRTMs we introduce $\mathrm{NP}(\mathcal{R})$, the class of problems solvable by a polynomial time SRTM over semiring $\mathcal{R}$. As the name suggests, $\mathrm{NP}(\mathcal{R})$ is a generalization of NP to computations with semirings. It also features $\mathrm{SAT}(\mathcal{R})$, a generalization of SAT over semirings, as a canonical complete problem. Apart from shedding light on how computations over semirings relate to the classical model of computation using (non-deterministic) Turing machines, $\mathrm{NP}(\mathcal{R})$ and $\mathrm{SAT}(\mathcal{R})$ are basic tools for the characterization of the computational complexity associated with evaluation over semirings.

Following the introduction of $\mathrm{NP}(\mathcal{R})$ and $\mathrm{SAT}(\mathcal{R})$, we immediately make use of these tools to characterize the complexity of a wide variety of semiring frameworks depending on their semiring parameter. Interestingly, while most frameworks turn out to be

1. This does not extend to the field of fixed-parameter-tractability. There is a wide range of results for semiring frameworks in this area, cf. (Ganian, Kim, Slivovsky, & Szeider, 2022; Friesen & Domingos, 2016; Bacchus, Dalmao, & Pitassi, 2009)

NP($\mathcal{R}$)-complete as expected and thus in a sense equally hard, provenance-queries for datalog (Green et al., 2007) are NP($\mathcal{R}$)-hard but provably not complete. This already shows the utility of our toolbox for the comparison of the intrinsic difficulty of different semiring problems.

However, since SRTMs are a purely hypothetical machine model, this alone only serves for a comparative analysis but does not give us any tangible insights into the computational complexity we need to expect in practice. Therefore, we follow up by relating SAT($\mathcal{R}$) to classical complexity classes, such as NP (Cook, 1971), #P (Valiant, 1979), OptP (Krentel, 1988), GapP (Fenner, Fortnow, & Kurtz, 1994) and Mod_pP (Hertrampf, 1990). This allows us to make statements about how difficult SAT($\mathcal{R}$) is at least by showing $\mathcal{C}$-hardness, which among other insights may allow one to assess how far at most a given algorithm is from a worst-case optimal one. Furthermore, by showing membership in some complexity class $\mathcal{C}$ we can also bound the resource requirements of an algorithm to solve SAT($\mathcal{R}$).

Naturally, an interesting question is where the hardness associated with a semiring comes from. The intuition conveyed by our results is that it depends mostly on two factors. The first factor is the encoding of the values of the semiring. This phenomenon is well known for the Knapsack problem, which is *pseudo-polynomial*, i.e., polynomial for the unary encoding of the integers but NP-hard for the binary encoding of integers. The second factor is the amount and kind of information that persists in values of the given semiring when they are added and multiplied; this was already shown to be a discerning factor for the complexity of conjunctive query containment over data annotated with semiring values (Green, 2011). For the Boolean semiring $\mathbb{B}$, this information is merely whether there exists a solution; for the semiring of natural numbers $\mathbb{N}$, it is the number of solutions; and even harder semirings like polynomial semirings $\mathcal{R}[(x_i)_\alpha]$ can amalgamate the result of many queries to an NP- or #P-oracle into one value, when the coefficients are from $\mathbb{B}$ and $\mathbb{N}$, respectively.

The difficulty caused by the encoding is not inherently connected to the difficulty associated with the semiring since different encodings can lead to different complexities over the same semiring. Therefore, we focus on the second complexity source of a semiring, i.e., the amount of information captured by its values. In order to avoid interference of the complexity due to the encoding, we assume efficiency restrictions on the encoding when necessary. By conducting an analysis of structural properties of semirings and their implications on information preservation, we manage to provide a tetrachotomy for the complexity of non-trivial semirings, i.e., we split them into four distinct classes that can all be related to (although not shown to be complete for) the same complexity class. Intuitively, we show that for non-trivial semirings the problems over them are either NP-like, Mod_pP-like, $\text{Mod}_p\text{P} \cup$ NP-like or #P-like. This $\mathcal{C}$-likeness is reflected in a variety of results, ranging from $\mathcal{C}$-hardness to non-reducibility to polynomially many $\mathcal{C}$-oracle calls in the general case, to reducibility to polynomially many $\mathcal{C}$-oracle calls in other, more restricted cases.

Contributions. The main contributions in this paper are briefly summarized as follows:

- We introduce Semiring Turing Machines and show that SAT($\mathcal{R}$), SCSPs, AMC, weighted first-order formula evaluation over $\mathcal{R}$, as well as some other problems are NP($\mathcal{R}$)-complete for every semiring $\mathcal{R}$, where NP($\mathcal{R}$) is a novel analog of NP over semirings. This not only characterizes the computational complexity of these problems but also provides the base of a toolbox for analyzing the complexity of semiring

and related quantitative reasoning frameworks, because it provides a collection of various kinds of problems that we can reduce from/to in order to prove NP($\mathcal{R}$)-hardness respectively membership in NP($\mathcal{R}$).

- We formalize, inspired by (Green, 2011), the intuition that the complexity associated with a semiring depends on the information its values can preserve. This is achieved by connecting the algebraic concept of *epimorphism* (a surjective mapping $f : \mathcal{R}_1 \to \mathcal{R}_2$ compatible with the operations), viewed as a measure of this information preservation, and reductions between semiring problems. Namely, we show that having an epimorphism with some additional restrictions allows us to solve problems over $\mathcal{R}_2$ by solving them over $\mathcal{R}_1$, leading to an "informational-hardness-map" presented in Section 6 (see Figure 2).
- Depending on the semiring properties of *periodicity* and *offset*, we give tetrachotomy results showing that non-trivial countable, commutative semirings fall into four complexity categories, meaning they are either NP-like, Mod_pP-like, $\text{Mod}_p\text{P} \cup$ NP-like, or #P-like. Furthermore, we derive the following specific properties:
  - *General Hardness:* For every non-trivial countable, commutative semiring $\mathcal{R}$, the problem SAT($\mathcal{R}$) is NP- or Mod_pP-hard.
  - *Zero-One Bounds:* When only the weights zero ($e_\oplus$) and one ($e_\otimes$) are used, the difference between the complexity of a $\mathcal{C}$-like problem and the complexity class $\mathcal{C}$ is limited given some additional reasonable restrictions. This shows that very hard instances require the usage of weights.
  - *Polynomial Semiring Hardness:* For a $\mathcal{C}$-like semiring $\mathcal{R}$ we show that $\mathcal{R}[(x_i)_\infty]$, the semiring of the polynomials with countably many variables and coefficients in $\mathcal{R}$, can intuitively be used to solve exponentially many $\mathcal{C}$-problems at the same time. This implies that SAT($\mathcal{R}[(x_i)_\infty]$) is not in FPSpace(poly), for any reasonable encoding unless the polynomial hierarchy collapses, which is widely assumed to be unlikely. Furthermore, for in a sense reasonable encodings, this question is e.g. for $\mathcal{R} = \mathbb{N}$ and $\mathcal{R} = \mathbb{B}$ equivalent to the open complexity-theoretic question whether #P $\subseteq$ FP/poly (resp. NP $\subseteq$ P/poly).
  - *Upper Bounds:* On the other hand, we show that for a broad subclass of $\mathcal{C}$-like problems over semirings, SAT($\mathcal{R}$) is counting-reducible to $\text{FP}_\parallel^{\mathcal{C}}$, i.e. solvable in polynomial time with polynomially many parallel queries to a $\mathcal{C}$-oracle.

Apart from the knowledge gain that our results provide directly, they have immediate further uses. First, they make the complexity analysis for specific semirings significantly easier. One example of this is the gradient semiring GRAD, which is used for parameter learning (Kimmig et al., 2017; Manhaeve et al., 2019). By applying our results, we demonstrate in Section 8 how one may easily prove that it leads to #P-complete problems. Second, we recall the question of Belle and De Raedt (2020) regarding the efficient implementation of semiring frameworks depending on the semiring parameter. The authors suspected that it is neither practical to provide a specific solver for each semiring nor efficient to have one solver that does not differentiate between the semirings. For this reason, Belle and De Raedt proposed a hybrid approach, in which the evaluation strategies are adapted for a few different *classes* of semirings. Indeed, our tetrachotomy results provide

tangible evidence that the hybrid approach is preferable. On top of that, they furthermore provide

- a set of properties that separates the semirings into reasonable classes, and
- concrete reductions that can be used as a starting point for an implementation.

Outline. In the following, we first provide in Section 2 some necessary preliminaries. We then provide in Section 3 some examples that motivate the paradigm of computations with semirings as parameter and give some guidance for its development. Next, we lay in Section 4 the basis of a methodical complexity analysis of semiring-based formalisms by introducing NP($\mathcal{R}$), a semiring version of NP, and SAT($\mathcal{R}$) as a canonical complete problem. Afterwards, we immediately employ in Section 5 the NP($\mathcal{R}$)-completeness result for SAT($\mathcal{R}$), by providing a variety of NP($\mathcal{R}$)-completeness and -hardness results for different semiring frameworks. In the subsequent Section 6 the relation of NP($\mathcal{R}$) to classical complexity classes in the form of different upper and lower bounds is explored. In Section 7 we consider related work and discuss our results, while in the final Section 8 conclude with issues for future research.

Full proofs for all theorems that are only sketched in the main body of the paper can be found in the appendix.

2. Preliminaries

We start by giving the necessary preliminaries. Throughout this paper, we assume knowledge of the basics of propositional logic. In this context, we consider propositional theories T over a set $\mathcal{V}$ of Boolean variables (i.e., propositional atoms) using the Boolean connectives $\vee$, $\wedge$, and $\neg$ for disjunction, conjunction, and negation, respectively; further connectives, e.g. material implications $\rightarrow$, may be defined as usual. Furthermore, $\bot$ and $\top$ are shorthands for an unsatisfiable (false) and a tautologic (true) formula, respectively. For interpretations of T, we consider subsets $\mathcal{I}$ of $\mathcal{V} \cup \{\neg v \mid v \in \mathcal{V}\}$ such that $\neg v \in \mathcal{I}$ iff $v \notin \mathcal{V}$. Satisfaction of formulas, theories ϕ etc. by an interpretation $\mathcal{I}$ is then defined as usual and denoted by $\mathcal{I} \models \phi$.

Apart from that, we need semirings.

Definition 1 (Semiring)**.** *A* semiring $\mathcal{R} = (R, \oplus, \otimes, e_\oplus, e_\otimes)$ *consists of a nonempty set* R *equipped with two binary operations* $\oplus$ *and* $\otimes$*, called addition and multiplication, where*

- $(R, \oplus)$ *is a commutative monoid with identity element* $e_\oplus$*,*
- $(R, \otimes)$ *is a monoid with identity element* $e_\otimes$*,*
- *multiplication left and right distributes over addition, and*
- $e_\oplus$ *annihilates* R*, i.e.* $\forall r \in R : r \otimes e_\oplus = e_\oplus = e_\oplus \otimes r$*.*

Here, a monoid $\mathcal{M} = (M, \odot, e_\odot)$ *consists of a nonempty set* M*, with a binary operation* $\odot$ *and an element* $e_\odot$ *such that*

- $\odot$ *is associative, i.e.,*
  for all $m_1, m_2, m_3 \in M$ *it holds that* $m_1 \odot (m_2 \odot m_3) = (m_1 \odot m_2) \odot m_3$*;*

- $e_\odot$ *is a neutral element for* $\odot$*, i.e.,*
  for all $m \in M$ *it holds that* $m \odot e_\odot = m = e_\odot \odot m$.

Furthermore, a semiring $\mathcal{R}$ *is*

- commutative *if for all* $r, r' \in R$ *it holds that* $r \otimes r' = r' \otimes r$
- idempotent *if for all* $r \in R$ *it holds that* $r \oplus r = r$
- periodic *if for all* $r \in R$ *there exist* $n, m \in \mathbb{N}$ *such that* $\bigoplus_{i=1}^{n} r = \bigoplus_{i=1}^{m} r$

Some examples of well-known semirings are

- $\mathbb{F} = (\mathbb{F}, +, \cdot, 0, 1)$, for $\mathbb{F} \in \{\mathbb{N}, \mathbb{Z}, \mathbb{Q}, \mathbb{R}\}$, the semiring of the numbers in $\mathbb{F}$ with addition and multiplication. These semirings are all commutative but not idempotent.
- $\mathcal{R}_{\max} = (\mathbb{N} \cup \{-\infty\}, \max, +, -\infty, 0), \mathcal{R}_{\min} = (\mathbb{N} \cup \{\infty\}, \min, +, \infty, 0)$, the tropical semirings. These semirings are both commutative and idempotent.
- $\mathbb{B} = (\{0, 1\}, \vee, \wedge, 0, 1)$, the Boolean semiring, which is both commutative and idempotent.
- $\mathcal{R}[(x_i)_n] = (R[(x_i)_n], \oplus, \otimes, e_\oplus, e_\otimes)$, for $n \in \mathbb{N}$, is the semiring of polynomials with variables $x_1, \ldots, x_n$ and coefficients from the semiring $\mathcal{R}$.
- $\mathcal{R}[(x_i)_\infty] = (\bigcup_{n \in \mathbb{N}} R[(x_i)_n], \oplus, \otimes, e_\oplus, e_\otimes)$ is the semiring of polynomials with variables $x_1, x_2, \ldots$ and coefficients from the semiring $\mathcal{R}$. The semirings $\mathcal{R}[(x_i)_\alpha]$ for $\alpha \in \mathbb{N} \cup \{\infty\}$ are commutative (resp. idempotent) if $\mathcal{R}$ is commutative (resp. idempotent).
- $\mathbb{T} = (\{e\}, \odot, \odot, e, e)$, the trivial semiring, where $e \odot e = e$.

Furthermore, we make use of generated semirings.

Definition 2 (Generated Semiring)**.** *Let* $\mathcal{R} = (R, \oplus, \otimes, e_\oplus, e_\otimes)$ *be a semiring. For any* $R^\star \subseteq R$*, the semiring generated by* $R^\star$*, denoted* $\langle R^\star \rangle_\mathcal{R}$*, is the least (w.r.t.* $\subseteq$*) semiring* $(R', \oplus, \otimes, e_\oplus, e_\otimes)$ *s.t.* $R^\star \subseteq R'$.

Example 1. $\mathbb{N}$*, the semiring of the natural numbers, is generated by* $\{1\}$ *or even by the empty set. The same holds for the Boolean semiring. On the other hand, the semiring* $\mathbb{Q}$ *of the rational numbers needs more elements to be generated.* $\{1/n \mid n \in \mathbb{N}\}$ *is a possible generator.*

For our complexity considerations we need problem reductions. Since we are studying functional complexity, we use the following notions of reductions.

Definition 3 (Metric, Counting & Karp Reduction)**.** *Let* $f_i : \Sigma^* \to \Sigma^*, i = 1, 2$ *be functions, then*

- *a* metric reduction from f_1 to f_2 *is a pair* T_1, T_2 *of polynomial time computable functions* $T_1 : \Sigma^* \to \Sigma^*$*,* $T_2 : \Sigma^* \times \Sigma^* \to \Sigma^*$ *such that* $f_1(x) = T_2(x, f_2(T_1(x)))$ *for every* $x \in \Sigma^*$.

- *a* counting reduction *is a metric reduction such that* $T_2(x, y) = T_2(x', y)$ *for all* x, x', y.
- *a* Karp reduction *is a metric reduction such that* $T_2(x, y) = y$ *for all* x, y.

Also, we use some well-known complexity classes.

- #P (Valiant, 1979) (resp. GapP, Fenner et al., 1994): the functions definable as the number of accepting paths (resp. minus the number of rejecting paths) of a nondeterministic polynomial time Turing Machine (NTM);
- OptP (Krentel, 1988): the functions definable as the maximum output of a polynomial time NTM;
- FP: the functions computable in polynomial time;
- $\mathrm{FP}_{\parallel}^{\mathcal{C}}$ (Jenner & Torán, 1995): the functions computable in polynomial time with parallel queries to a $\mathcal{C}$ oracle, where $\mathcal{C}$ is a complexity class;
- FPSpace(poly) (resp. FPSpace(exp)): the functions computable in polynomial space with polynomial (resp. unrestricted and thus exponential) size output;
- $\mathcal{C}/poly$ (Karp & Lipton, 1982): the problems solvable in $\mathcal{C}$ with polynomial advice, where $\mathcal{C}$ is a complexity class, i.e., there is a function f such that we can solve the problem for all inputs x in $\mathcal{C}$ given $x, f(|x|)$, where $|x|$ denotes the length of the input and $|f(x)| \in \mathcal{O}(|x|^k)$ for constant $k \in \mathbb{N}$;
- $\mathrm{Mod}_p\mathrm{P}$ (Hertrampf, 1990): the languages L described as $x \in L$ iff $f(x) \not\equiv 0 \mod p$ for some f in #P;
- $\mathcal{C}_1^{\mathcal{C}_2[f(n)]}$: the problems solvable in $\mathcal{C}_1$ with at most $f(n)$ calls to a $\mathcal{C}_2$ oracle, where $\mathcal{C}_1$ and $\mathcal{C}_2$ are complexity classes. Here, $\mathcal{C}_1$ is a Turing machine based complexity class (such that an oracle makes sense);
- BPP (Gill, 1977): the languages L for which some NTM T exists such that if $x \in L$ at least $2/3$ of the computation paths of T on x accept and if $x \notin L$ at most $1/3$ of the computation paths of T on x accept.

The exact relationship between the different complexity classes is often unclear. However, it is known that #P, OptP, $\mathrm{Mod}_p\mathrm{P}$ and NP are contained in FPSpace(poly). Apart from that, #P is considered to be at least as hard as NP and $\mathrm{Mod}_p\mathrm{P}$, as we can obtain the solution of a problem in NP or $\mathrm{Mod}_p\mathrm{P}$ via one #P-oracle call. As for the relationship between NP and $\mathrm{Mod}_p\mathrm{P}$, it is known that $\mathrm{NP} \subseteq \mathrm{BPP}^{\mathrm{Mod}_p\mathrm{P}}$ (Bennett & Gill, 1981). In regards to the power of Turing machines with advice, we note that the advice function does not need to be computable and thus, P/poly contains some undecidable problems. Nevertheless, the power of advice is limited: if $\mathrm{NP} \subseteq \mathrm{P/poly}$ or $\#\mathrm{P} \subseteq \mathrm{FP/poly}$ then the polynomial hierarchy collapses to a finite level (Karp & Lipton, 1980), which is considered to be unlikely. On the other hand, it is known that $\mathrm{BPP} \subseteq \mathrm{P/poly}$ (Bennett & Gill, 1981).

Hardness and completeness are defined as usual:

Definition 4 (Hardness, Completeness)**.** *A problem* P *is* $\mathcal{C}$-hard *for a complexity class* $\mathcal{C}$ *under X-reductions, if every problem* $P' \in \mathcal{C}$ *can be reduced to* P *by some X-reduction;* P *is* $\mathcal{C}$-complete *under X-reductions, if in addition* $P \in \mathcal{C}$.

3. The Semiring Paradigm

Before we introduce new complexity classes to capture the computational complexity of frameworks that depend on a semiring parameter, we first consider two prominent examples of such frameworks. This is, on the one hand, to further motivate the interest in a complexity study. On the other hand, it provides an insight into the structure that is shared by different such frameworks. The latter is also important as it tells us more about the nature of the problems at hand and thus guides us to an appropriate definition of a machine model for computations involving semirings.

Algebraic Model Counting The first framework we consider is Algebraic Model Counting (AMC), introduced by Kimmig et al. (2017). Intuitively, it is a generalization of weighted model counting for propositional formulas, with weights that can be values from a semiring.

Definition 5 (AMC)**.** *Given a propositional theory T over propositional variables $\mathcal{V}$, a commutative semiring $\mathcal{R}$, and a labeling function $\alpha : L \to R$ that maps the literals L over $\mathcal{V}$ to R,* AMC *is to compute the value*

$$A(T) = \bigoplus_{\mathcal{I},\ s.t.\ \mathcal{I} \models T} \bigotimes_{v \in \mathcal{I}} \alpha(v) \otimes \bigotimes_{\neg v \in \mathcal{I}} \alpha(\neg v).$$

That is, we take a sum over all interpretations that satisfy a propositional theory T, where each addend is a product of the weights of the literals satisfied by the interpretation. Contrary to weighted model counting, where sum and product need to be the usual addition and multiplication over the reals, they can be from any semiring here. The same holds for the weights, which also do not necessarily need to be reals.

Besides the standard applications in SAT, #SAT, and probabilistic inference, the authors showed that AMC can be used to perform sensitivity analysis of probabilistic inference w.r.t. a parameter by using the semiring of the polynomials with coefficients in $[0, 1]$. For more semirings that allow additional applications, like the construction of tractable circuit representations, we refer the reader to Kimmig et al. (2017).

We see that regardless of the specific semiring, some of the structure stays the same: Similarly to SAT, MaxSAT and #SAT, we guess an interpretation, obtain a value based on the truth of the literals and their relation to the propositional theory and in a last step perform some form of aggregation operation over all interpretations. In the case of SAT, this corresponds to checking whether a given assignment satisfies the theory and aggregating these values by quantifying existentially over all possible interpretations. For MaxSAT on the other hand, we associate a number with a given interpretation and aggregate these values by taking the maximum.

Semiring-based Constrained Satisfaction Problems A similar structure can be observed for Semiring-based Constrained Satisfaction Problems (SCSPs). Bistarelli et al. (1999) introduced them as a generalization of constraint satisfaction problems parameterised with *c-semirings* $\mathcal{R}$, which are idempotent commutative semirings such that the axiom $\forall r \in R : r \oplus e_\otimes = e_\otimes$ holds. This restriction is due to the fact that SCSPs were defined to capture semantics for the levels of consistency of Constraint Satisfaction Problems (CSPs), rather than semantics for general quantitative reasoning.

Definition 6 (Constraint System, Constraint Problem)**.** *A* constraint system *is a tuple* $CS = \langle \mathcal{R}, D, V \rangle$*, where* $\mathcal{R}$ *is a c-semiring,* D *is a finite domain, and* V *is an ordered set of variables. A constraint over* CS *is a pair* $\langle def, con \rangle$*, where* $con \subseteq V$ *and* $def : D^{con} \to R$ *is the* value *of the constraint. Here,* D^{con} *denotes the set* $\{(d_x)_{x \in con} \mid \forall x \in con : d_x \in D\}$*.*

A constraint problem P *over* CS *is a pair* $P = \langle C, con \rangle$*, where* C *is a multiset of constraints over* CS *and* $con \subseteq V$*.*

Using D^{con} instead of $D^{|con|}$ is helpful, since it allows us to associate inputs and their variable.

Bistarelli et al. (1999) showed that SCSPs correspond to classical CSP, probabilistic CSP, weighted CSP and fuzzy CSP when the c-semiring is respectively chosen as $\mathbb{B}, ([0,1], \max, \cdot, 0, 1), \mathcal{R}_{\min}$ and $([0,1], \max, \min, 0, 1)$.

Example 2. *The following are constraints over* $CS = \langle \mathcal{R}_{\min}, \{a, b\}, \{x, y\} \rangle$*:*

$$c_1 = \left\langle \begin{array}{l} aa \mapsto 1 \\ ab \mapsto 2 \\ ba \mapsto 3 \\ bb \mapsto 4 \end{array} , \{x, y\} \right\rangle, c_2 = \left\langle \begin{array}{l} a \mapsto 5 \\ b \mapsto 6 \end{array} , \{y\} \right\rangle.$$

Together they define the constraint problem $C = \langle \{c_1, c_2\}, \{x, y\} \rangle$*.*

The two main operations on constraints are combination $*$ and projection $\Downarrow$.

Definition 7 (Combination, Projection)**.** *For* $t = (d_x)_{x \in con}$ *and* $con' \subseteq con$ *the projection* $t \downarrow_{con'}^{con}$ *is equal to* $(d_x)_{x \in con'}$*.*

The combination $c_1 * c_2$ *of two constraints* $c_i = \langle def_i, con_i \rangle, i = 1, 2$ *is the constraint* $c = \langle def, con_1 \cup con_2 \rangle$*, where*

$$def(t) = def_1(t \downarrow_{con_1}^{con_1 \cup con_2}) \otimes def_2(t \downarrow_{con_2}^{con_1 \cup con_2})$$

and$\otimes$ *is the multiplication of the semiring.*

The projection $c \Downarrow_{con'}$ *of a constraint* $c = \langle def, con \rangle$ *to* $con' \subseteq con$ *is* $\langle def', con' \rangle$ *with* $def'(t') = \bigoplus_{\{t \in D^{con} \mid t \downarrow_{con'}^{con} = t'\}} def(t)$*, where* $\bigoplus$ *is the addition of the semiring.*

Intuitively, combination $*$ is the product ($\otimes$) of constraint values and projection $\Downarrow_{con'}$ is the sum ($\oplus$) over all assignments to the variables in $con \setminus con'$. Combination, as the name says, combines multiple constraints into one, and projection partially evaluates a constraint thus removing variables from the constraint.

Example 3. *The combination* $c_1 * c_2$ *of* c_1, c_2 *is*

$$\left\langle \begin{array}{l} aa \mapsto 1 \otimes 5 \\ ab \mapsto 2 \otimes 6 \\ ba \mapsto 3 \otimes 5 \\ bb \mapsto 4 \otimes 6 \end{array} , \{x, y\} \right\rangle = \left\langle \begin{array}{l} aa \mapsto 1 + 5 \\ ab \mapsto 2 + 6 \\ ba \mapsto 3 + 5 \\ bb \mapsto 4 + 6 \end{array} , \{x, y\} \right\rangle = \left\langle \begin{array}{l} aa \mapsto 6 \\ ab \mapsto 8 \\ ba \mapsto 8 \\ bb \mapsto 10 \end{array} , \{x, y\} \right\rangle.$$

The projection $c_1 \Downarrow \{x\}$ *of* c_1 *down to* $\{x\}$ *is*

$$\left\langle \begin{array}{l} a \mapsto 1 \oplus 2 \\ b \mapsto 3 \oplus 4 \end{array} , \{x\} \right\rangle = \left\langle \begin{array}{l} a \mapsto \min(1, 2) \\ b \mapsto \min(3, 4) \end{array} , \{x\} \right\rangle = \left\langle \begin{array}{l} a \mapsto 1 \\ b \mapsto 3 \end{array} , \{x\} \right\rangle.$$

Using $*$ and $\Downarrow$, the consistency-level of an SCSP is defined as follows.

Definition 8 (Consistency-Level)**.** *Given an SCSP problem $P = \langle C, con\rangle$, the* best level of consistency *of P is defined as $blevel(P) = (\Pi_{c\in C}\ c) \Downarrow_\emptyset$. Here, Π in the expression $\Pi_{c\in C}\ c$ is used for application of $*$ to all constraints c in C.*

So the consistency level can be computed by first combining all constraints (using $*$) and then projecting onto the empty set (using $\Downarrow_\emptyset$).

Example 4. *The consistency level of $C = \langle\{c_1, c_2\}, \{x, y\}\rangle$ is $(c_1 * c_2) \Downarrow_\emptyset$, i.e.,*

$$\left\langle \begin{array}{c} aa \mapsto 6 \\ ab \mapsto 8 \\ ba \mapsto 8 \\ bb \mapsto 10 \end{array} , \{x, y\} \right\rangle \Downarrow_\emptyset = \left\langle\ \varepsilon \mapsto \min(6, 8, 8, 10)\ , \emptyset \right\rangle .$$

Here, the last expression evaluates to 6, meaning that the consistency level $blevel(C)$ is 6.

Again, we observe a similar structure regardless of the semiring: we can compute the *blevel* of a constraint problem $P = \langle C, con\rangle$ by guessing an assignment to the variables in *con*, by evaluating the constraints under the given assignment, taking the product, $\otimes$, of their values for that assignment and aggregating the results for each assignment using the sum, $\oplus$, of the semiring. This structure is strikingly similar to the approach we can use to evaluate AMC instances. While the guessed part here is an assignment to multi-valued variables instead of an interpretation, for both frameworks, we can evaluate an instance by

1. *Guessing* some form of assignment;
2. *Evaluating* the instance for the given assignment; and
3. *Aggregating* the values of all guesses.

Using this insight into the structure of the evaluation problems associated with semiring frameworks, we can now approach the characterization of their complexity.

4. Semiring Complexity Classes and a Complete Problem

In this section, we develop a toolbox that first and foremost allows completeness results for arbitrary commutative semirings for the first time. For this, we first introduce a prototypical problem called $\mathrm{SAT}(\mathcal{R})$ using the insights from the previous sections on what the evaluation problems over semiring frameworks typically look like. The reason for introducing a new problem instead of using AMC or SCSPs as a canonical complete problem is that $\mathrm{SAT}(\mathcal{R})$ in our opinion better shows the power of semiring frameworks. Rather than only allowing that we compute the value of an assignment as a product of weights, it allows us to specify the value of an assignment as a complex arithmetic expression that depends on a propositional interpretation.

In order to capture this guess-evaluate-aggregate pattern over different semirings, we introduce Semiring Turing Machines (SRTMs), a novel machine model that allows restricted non-deterministic computation of a semiring value in a black box fashion, whose final output

is then defined as the sum of the values of all computation paths. Based on SRTMs, we can then define NP($\mathcal{R}$), a generalization of NP over semirings.

Putting things together, we prove SAT($\mathcal{R}$) to be NP($\mathcal{R}$)-complete under Karp reductions, thus allowing us to use it as a canonical problem to reduce from, for NP($\mathcal{R}$)-hardness proofs.

Furthermore, we consider other alternative ways to generalize NP to a quantitative setting. We highlight the differences in computational power caused already by small changes to our model and, by this, justify the definition of NP($\mathcal{R}$) that we chose.

4.1 Weighted Quantified Boolean Formulas and SAT($\mathcal{R}$)

The most natural way to define SAT($\mathcal{R}$) is as a special case of weighted Quantified Boolean Formulas (QBFs).

We define weighted QBFs similarly to other weighted logics (Droste & Gastin, 2007; Mandrali & Rahonis, 2015).

Definition 9 (Syntax)**.** *Let $\mathcal{V}$ be a set of propositional variables and $\mathcal{R} = (R, \oplus, \otimes, e_\oplus, e_\otimes)$ be a commutative semiring. A* weighted *QBF over $\mathcal{R}$ is of the form α given by the grammar*

$$\alpha ::= k \mid v \mid \neg v \mid \alpha + \alpha \mid \alpha * \alpha \mid \Sigma v \alpha \mid \Pi v \alpha$$

where $k \in R$ and $v \in \mathcal{V}$. A variable $v \in \mathcal{V}$ is free *in a weighted QBF α, if α has v is not in the scope of a quantifier Σv or Πv. A weighted* fully quantified Boolean Formula *is a weighted QBF without free variables.*

Definition 10 (Semantics)**.** *Given a weighted QBF α over a commutative semiring $\mathcal{R} = (R, \oplus, \otimes, e_\oplus, e_\otimes)$ and variables from $\mathcal{V}$ as well as a propositional interpretation $\mathcal{I}$ of $\mathcal{V}$, the semantics $[\![\alpha]\!]_\mathcal{R}(\mathcal{I})$ of α over $\mathcal{R}$ w.r.t. $\mathcal{I}$ is defined as follows:*

$$\begin{aligned}
[\![k]\!]_\mathcal{R}(\mathcal{I}) &= k \\
[\![l]\!]_\mathcal{R}(\mathcal{I}) &= \begin{cases} e_\otimes & l \in \mathcal{I} \\ e_\oplus & \textit{otherwise.} \end{cases} \quad (l \in \{v, \neg v\}) \\
[\![\alpha_1 + \alpha_2]\!]_\mathcal{R}(\mathcal{I}) &= [\![\alpha_1]\!]_\mathcal{R}(\mathcal{I}) \oplus [\![\alpha_2]\!]_\mathcal{R}(\mathcal{I}) \\
[\![\alpha_1 * \alpha_2]\!]_\mathcal{R}(\mathcal{I}) &= [\![\alpha_1]\!]_\mathcal{R}(\mathcal{I}) \otimes [\![\alpha_2]\!]_\mathcal{R}(\mathcal{I}) \\
[\![\Sigma v \alpha]\!]_\mathcal{R}(\mathcal{I}) &= [\![\alpha]\!]_\mathcal{R}(\mathcal{I}_v) \oplus [\![\alpha]\!]_\mathcal{R}(\mathcal{I}_{\neg v}) \\
[\![\Pi v \alpha]\!]_\mathcal{R}(\mathcal{I}) &= [\![\alpha]\!]_\mathcal{R}(\mathcal{I}_v) \otimes [\![\alpha]\!]_\mathcal{R}(\mathcal{I}_{\neg v})
\end{aligned}$$

where $\mathcal{I}_v = \mathcal{I} \setminus \{\neg v\} \cup \{v\}$ and $\mathcal{I}_{\neg v} = \mathcal{I} \setminus \{v\} \cup \{\neg v\}$.

Weighted QBFs generalize QBFs in negation normal form (NNF), as negation is only allowed in front of variables. Intuitively, allowing negation in front of complex formulas would add the ability to test whether the value of a weighted QBF is zero. We can only test whether an atomic formula is false. Therefore, our variant is or at least seems less expressive. However, it fits better into the context of the problems we consider, where it also is not possible to perform such "zero-tests" for complex expressions.

Here, we further focus on ΣBFs, i.e., the weighted fully quantified BFs that contain only sum quantifiers (i.e. Σv) and we introduce their evaluation problem as

$$\mathrm{SAT}(\mathcal{R})\text{: given a } \Sigma\text{BF } \alpha \text{ over } \mathcal{R} \text{ compute } [\![\alpha]\!]_{\mathcal{R}}(\emptyset).$$

Even though we restrict ourselves to ΣBFs, they at first glance seem to be different from the propositional formulas describing a SAT-instance. Namely, for SAT the quantifier Σ is implicit since all variables that occur in the SAT-instance are existentially quantified. For $\mathrm{SAT}(\mathcal{R})$, however, this is not the case. Here, we can quantify a variable using Σ in subexpressions of $+$ and $*$, e.g., formulas like $\alpha + \Sigma v \beta$ and $\Sigma v \alpha * \Sigma w \beta$ are allowed. Requiring all quantifiers to occur outside of subexpressions of $+$ and $*$ is not a semantic restriction however:

Lemma 11 (Prefix Normal Form)**.** *For every ΣBF α over a commutative semiring $\mathcal{R}$ there exists a ΣBF β over $\mathcal{R}$ such that*

(i) $\beta = \Sigma v_1 \dots \Sigma v_n \gamma$*, where γ is quantifier free,*

(ii) β can be constructed from α in polynomial time, and

(iii) $[\![\alpha]\!]_{\mathcal{R}}(\emptyset) = [\![\beta]\!]_{\mathcal{R}}(\emptyset)$*, i.e., α and β evaluate to the same value.*

Proof (Sketch). This statement can be proved by induction on the number of sum quantifiers Σ that occur in the subexpression of some other connective $+$ or $*$. The base case is clear, for the induction step, we show that $(\Sigma v \alpha_1) * \alpha_2$ is equivalent to $\Sigma v(\alpha_1 * \alpha_2)$ by assuming w.l.o.g. that α_2 does not contain v and that $(\Sigma v \alpha_1) + \alpha_2$ is equivalent to $\Sigma v(\alpha_1 + \alpha_2 * v)$ under the same assumption. □

This way the connection between SAT and $\mathrm{SAT}(\mathcal{R})$ becomes clearer. In fact, as with AMC, many versions of SAT can be seen as special cases of $\mathrm{SAT}(\mathcal{R})$ for different semirings $\mathcal{R}$.

Example 5. *Over $\mathbb{B}$, the Boolean semiring,* $\mathrm{SAT}(\mathbb{B})$ *is equivalent to* SAT*: On the one hand, given a* $\mathrm{SAT}(\mathcal{R})$*-instance $\beta = \Sigma v_1 \dots \Sigma v_n \gamma$, where γ is quantifier free, we see that $[\![\beta]\!]_{\mathbb{B}}(\emptyset) = 1$ iff the propositional formula ϕ obtained from γ by replacing every $0, 1, +, *$ with $\bot, \top, \vee, \wedge$, respectively, is a yes instance of* SAT*, i.e., satisfiable.*

On the other hand, given a SAT*-instance ϕ, we see that a propositional formula ϕ in NNF is satisfiable if the* $\mathrm{SAT}(\mathbb{B})$*-instance $\beta = \Sigma v_1 \dots \Sigma v_n \gamma$ fulfills $[\![\beta]\!]_{\mathbb{B}}(\emptyset) = 1$. Here, γ is obtained from ϕ by replacing every $\bot, \top, \vee, \wedge$ with $0, 1, +, *$, respectively, and $\{v_1, \dots, v_n\}$ is the set of variables occurring in ϕ.*

Another, more interesting example is LexMaxSAT*, the problem of obtaining the lexicographically maximum satisfying assignment for a propositional formula ϕ. W.l.o.g. we assume that ϕ is in NNF. We can reformulate* LexMaxSAT *as a* $\mathrm{SAT}(\mathcal{R}_{\max})$*-instance β over the max-tropical semiring $\mathcal{R}_{\max}$, where the corresponding instance β is given by*

$$\Sigma v_1 \dots \Sigma v_n \gamma * (v_1 * 2^{n-1} + \neg v_1) * \dots * (v_n * 2^0 + \neg v_n).$$

As above, γ is obtained from ϕ by replacing every $\bot, \top, \vee, \wedge$ with $-\infty, 0, \max, +$, respectively. Then $[\![\beta]\!]_{\mathcal{R}_{\max}}(\emptyset) = e_{\oplus} = -\infty$ iff ϕ is unsatisfiable and $[\![\beta]\!]_{\mathcal{R}_{\max}}(\emptyset) = m = \sum_{i=1}^{n} b_i 2^{n-i}$ iff the lexicographically maximum satisfying assignment for ϕ sets v_i to true whenever $b_i = 1$.

These examples show that we can use SAT($\mathcal{R}$) over different semirings $\mathcal{R}$ to express different versions of SAT. Moreover, we observe that problems described by a ΣBF of the form

$$\beta = \Sigma v_1 \dots \Sigma v_n \gamma, \text{ where } \gamma \text{ is quantifier free,}$$

are again structurally similar to the other problems following the semiring paradigm. That is, the value $[\![\beta]\!]_{\mathcal{R}}(\emptyset)$ is obtainable by *guessing* an interpretation $\mathcal{I}$ of the variables $v_1, \dots, v_n$, *evaluating* γ using the interpretation, and *aggregating* the values $[\![\gamma]\!]_{\mathcal{R}}(\mathcal{I})$ using the sum, $\oplus$, of the semiring.

This guess-evaluate-aggregate pattern is also similar to definition of complexity classes such as NP, #P and OptP in terms of non-deterministic Turing machines. Based on this intuition we will next provide NP($\mathcal{R}$) a generalized version of NP, which also depends on a semiring parameter, to abstractly characterize the complexity of SAT($\mathcal{R}$).

4.2 Semiring Turing Machines and NP($\mathcal{R}$)

We generalize NTMs to Semiring Turing Machines (SRTMs) to characterize the complexity of SAT($\mathcal{R}$). At this point, we do not want the time and space needed to perform operations on semiring values that are encoded in a finite alphabet to influence our definition. Thus, we need SRTMs to be capable of

- performing semiring operations atomically in a black box manner, irrespective of encodings of values;
- summing up values generated by nondeterministic computations; and
- using semiring values in the input in calculations.

On the other hand, too much power should be avoided; to this end, we relegate semiring computations to weighted transitions.

Definition 12 (SRTM)**.** *A* Semiring Turing Machine *is a 7-tuple* $M = (\mathcal{R}, R', Q, \Sigma, \iota, \sqcup, \delta)$*, where*

- $\mathcal{R}$ *is a commutative semiring,*
- $R' \subseteq R$ *is a finite set of semiring values, (intuitively, this is a set of fixed values that are "known" to* M*)*
- Q *is a finite set of states,*
- Σ *is a finite set of symbols (the tape alphabet),*
- $\iota \in Q$ *is the initial state,*
- $\sqcup \in \Sigma$ *is the blank symbol,*
- $\delta \subseteq (Q \times (\Sigma \cup R)) \times (Q \times (\Sigma \cup R)) \times \{-1, 1\} \times R$ *is a weighted transition relation, where the last entry of the tuple is the weight. For each* $((q_1, \sigma_1), (q_2, \sigma_2), d, r) \in \delta$ *the following holds:*

1. M cannot write or overwrite semiring values:
   if $\sigma_1 \in R$ *or* $\sigma_2 \in R$, *then* $\sigma_1 = \sigma_2$,
2. M can only make a transition with weight r when r is from R' or under the head:
   $r \in R'$ *or* $r = \sigma_1 \in R$
3. M cannot discriminate semiring values:
   if $\sigma_1 \in R$, *then*
   (a) *for all* $\sigma_1' \in R$ *we have* $((q_1, \sigma_1'), (q_2, \sigma_1'), d, \sigma_1') \in \delta$ *or*
   (b) *for all* $\sigma_1' \in R$ *we have* $((q_1, \sigma_1'), (q_2, \sigma_1'), d, r) \in \delta$.

As usual, -1 and 1 move the head to the left and right. Intuitively, the combination of the second and third condition ensures that when an SRTM reads a semiring value r in a given state then it always, independently of the value at hand, makes a transition with weight r or with a constant value from the finite set R'. Note that this "or" is not exclusive. Having access to some constants is necessary, since every transition is assumed to have a weight. Additionally, these constants allow us to use semiring values that do not occur in the input. This is important since otherwise we cannot always multiply partial solutions by constant factors or even return a constant value.

Importantly, this allows us to always represent the transition function finitely by grouping the transitions at state q_1 when a semiring value is read to

$$((q_1, X), (q_2, X), d, X) \text{ and } ((q_1, X), (q_2, X), d, r),$$

where X is a placeholder representing any (although the same in all occurrences in the expression) semiring value and $r \in R'$.

We remark that in terms of control flow, i.e., "if"-statements, loops, recursive definitions, etc., SRTMs feature the same possibilities as non-deterministic Turing machines, since the difference between the two models of computation regarding the transition function δ is that for SRTMs it is weighted and has some restrictions on the weights. Therefore, we can use the typical control flow instructions also with SRTMs.

The output of a computation is as follows.

Definition 13 (SRTM function)**.** *The* value $v(c)$ *of an SRTM* M *on a configuration* $c = (q, x, n)$, *where* $q \in Q$ *is a state,* $x \in (\Sigma \cup R)^*$ *is the string on the tape, and* $n \in \mathbb{N}$ *is the head position, is recursively defined by* $v(c) = \bigoplus_{c \xrightarrow{r} c'} r \otimes v(c')$, *where* $c \xrightarrow{r} c'$ *denotes that* M *can transit from* c *to* c' *with weight* r*; the empty sum has value* $e_\otimes$. *The* output *of* M *on input* x *is* $v(\iota, x, 0)$.

In other words, this means that the output of an SRTM M is calculated by aggregating the value v_π of each non-deterministic computation path π using the addition $\oplus$ of the semiring $\mathcal{R}$. Here, the value v_π of a non-deterministic computation path π along configurations $c_1 \xrightarrow{r_1} \dots \xrightarrow{r_{n(\pi)-1}} c_{n(\pi)}$ is given by $r_1 \otimes \dots \otimes r_{n(\pi)-1}$, i.e., by multiplying the weights of the single transition steps.

Example 6 (SRTM Computation)**.** *Consider Algorithm 1, which sketches in pseudocode the working of an SRTM that solves* $\mathrm{SAT}(\mathcal{R})$. *The computation tree*[2] *of* $\textsc{Eval}_{\mathbb{N}}(\alpha, \emptyset)$ *with*

2. Naturally, we left out transitions with weight 1 for the checks of the if and switch statements.

Algorithm 1 An SRTM algorithm for SAT($\mathcal{R}$)

Input A ΣBF α over semiring $\mathcal{R}$ and an interpretation $\mathcal{I}$.
Output $[\![\alpha]\!]_{\mathcal{R}}(\mathcal{I})$.

function $\textsc{Eval}_{\mathcal{R}}(\alpha, \mathcal{I})$
  switch α **do**
    case $\alpha = k$:
      transition with k
    case $\alpha = l, l \in \{v, \neg v\}$:
      if $l \in \mathcal{I}$ **then**:
        transition with $e_\otimes$
      else:
        transition with $e_\oplus$
    case $\alpha = \alpha_1 + \alpha_2$:
      Guess $i \in \{1, 2\}$
      Execute $\textsc{Eval}_{\mathcal{R}}(\alpha_i, \mathcal{I})$
    case $\alpha = \alpha_1 * \alpha_2$:
      Execute $\textsc{Eval}_{\mathcal{R}}(\alpha_1, \mathcal{I})$
      Execute $\textsc{Eval}_{\mathcal{R}}(\alpha_2, \mathcal{I})$
    case $\alpha = \Sigma v \alpha'$:
      Guess $\mathcal{I}' \in \{\mathcal{I} \setminus \{\neg v\} \cup \{v\}, \mathcal{I} \setminus \{v\} \cup \{\neg v\}\}$
      Execute $\textsc{Eval}_{\mathcal{R}}(\alpha', \mathcal{I}')$

$\alpha = \Sigma v v * 3$ is given in Figure 1. Its initial configuration c_0 is the root node of the tree. Since the SRTM is over the natural number semiring $\mathbb{N}$, the leaves $c_{0.0.0.0}, c_{0.1.0.0}$, i.e., the configurations without a successor, have the value $e_\otimes = 1$.

The values of their predecessors $c_{0.0.0}$ and $c_{0.1.0}$ are both $3 \cdot 1 = 1$. It is calculated as the value of their successor (1) and the weight of the transition, which is 3 in both bases since the SRTM reads the subformula $\alpha = 3$ in either case. At their predecessors $c_{0.0}$ and $c_{0.1}$, we observe a difference in values, i.e., $v(c_{0.0}) = 0$ and $v(v_{0.1}) = 3$, since the SRTM reads the subformula $\alpha = v$ here and thus transitions either with weight $e_\otimes$ or $e_\oplus$ depending on the guess of the value of v in the respective branch. $c_{0.0}$ and $c_{0.1}$ have as a common predecessor the initial configuration c_0. Here, a formula of the form $\alpha = \Sigma v \alpha$ is read and thus both transitions from c_0 to $c_{0.0}$ and to $c_{0.1}$, have weight 1. They correspond to guessing v to be false and true, respectively. As such, the output value is given by 3.

As we can see, this definition of SRTMs also follows the aforementioned guess-evaluate-aggregate pattern. With this in place, we define an analog of NP.

Definition 14 (NP($\mathcal{R}$))**.** *NP($\mathcal{R}$) is the class of all functions computable in polynomial time by an SRTM over $\mathcal{R}$.*

Here, we choose the name NP($\mathcal{R}$) to highlight the tight connection to NP, which stems from the fact that (as we will see later) NP can be seen as NP($\mathbb{B}$). Indeed, the underlying idea of SRTMs and NP($\mathcal{R}$) is to proceed in the same manner as when generalizing a Boolean logic to a weighted logic. That is, we add weights and replace disjunction by addition and

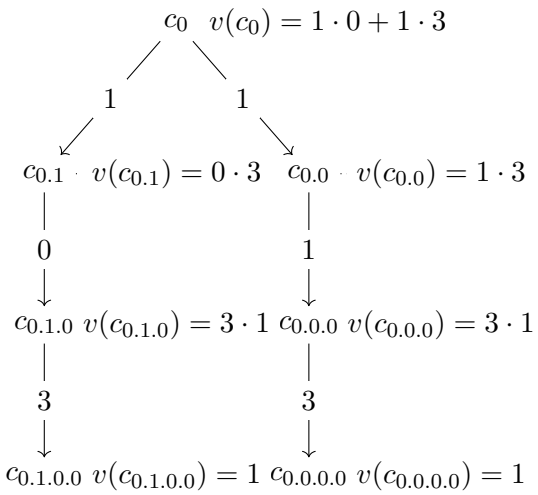

Figure 1: A computation tree over $\mathbb{N}$. Each transition $c \xrightarrow{r} c'$ is annotated with its weight r and each configuration c is annotated with its value $v(c)$.

conjunction by multiplication over the semiring. Since over the Boolean semiring $\mathbb{B}$ the addition is disjunction and the multiplication is conjunction, solving a problem in NP($\mathbb{B}$) is to check whether there exists a non-deterministic computation path with non-zero weight, i.e., which accepts. The same holds for problems in NP.

Note that NP($\mathcal{R}$) as it follows from our definition of SRTMs is only *some* analog of NP over semirings, and we do not claim that it is the only reasonable one. However, first and foremost, our goal with this definition is to obtain a complexity class that characterizes the computational complexity of SAT($\mathcal{R}$) and other related problems. This succeeds, as indeed SAT($\mathcal{R}$) is NP($\mathcal{R}$)-complete. However, in order to formally state the result we first need to establish how reductions work in this context, given that we allow semiring values on the tape explicitly.

Definition 15 (Surrogate Alphabet)**.** *Let $\mathcal{R}$ be a semiring and let Σ be some alphabet. Then the* surrogate alphabet $s_{S,V,E}(\Sigma)$ *is set of words over Σ extended with surrogates (for semiring variables) defined by*

$$s_{S,V,E}(\Sigma) = (\Sigma \cup \{SV^nE \mid n \in \mathbb{N}\})^*,$$

where $S, V, E \notin \Sigma$ are distinguished letters such that in SV^nE the letter S denotes the start, the string V^n denotes the index n, and E denotes the end of a surrogate (for a semiring value). For a substitution $\sigma : \mathbb{N} \to R$ and a word $x \in s_{S,V,E}(\Sigma)$, we denote by $\sigma(x)$ the word in $(\Sigma \cup R)^*$ obtained by replacing SV^nE with $\sigma(n)$.

The idea here is that we refer to surrogates $SVE, \ldots, SV^nE$ for variable semiring values instead of actual semiring values $r_1, \ldots, r_n$. Thus, we know of the presence of semiring values but not which semiring value is represented exactly by the surrogate.

Using the surrogate alphabet, we adapt Karp reductions to the context of SRTMs.

Definition 16 (Karp Surrogate-Reduction)**.** *Let f_1 and f_2 be functions $f_i : (\Sigma \cup R)^* \to R, i = 1, 2$. Then a Karp* surrogate $s_{S,V,E}$ reduction *from f_1 to f_2 is a polynomial time computable function $T : s_{S,V,E}(\Sigma) \to s_{S,V,E}(\Sigma)$ with a finite set $R' = \{r_1, \ldots, r_k\} \subseteq R$, $k \geq 0$, of semiring values such that for all $x \in s_{S,V,E}(\Sigma)$ and $\sigma : \mathbb{N} \to R$ such that $\sigma(i-1) = r_i$, $1 \leq i \leq k$, it holds that $f_1(\sigma(x)) = f_2(\sigma(T(x)))$.*

Since we only use Karp surrogate $s_{S,V,E}$ reductions with surrogate alphabet S, V, E we omit $s_{S,V,E}$ and write "Karp s-reduction".

Intuitively, the idea behind surrogate-reductions is that as with SRTMs, we cannot modify semiring values, make decisions based on them etc. Therefore, also surrogate reductions must not be able to modify semiring values, produce different outputs based on them etc., as this would mean that reductions would have more power than SRTMs, which we need to avoid if $\mathrm{NP}(\mathcal{R})$ should be closed under reductions. The set R' serves to distinguish semiring values that should be constant. Instead we can only modify the instance specification that tells us how we should combine (using sum and product) the semiring values to obtain a solution from one that is expected by f_1 to one that is expected by f_2.

Nevertheless, a reduction T at least needs to be able to transfer the semiring values that are used in the original instance x to the instance $T(x)$. For this reason, we give T access to surrogates SV^nE for semiring value occurrences such that T can intuitively "copy" semiring values to another position but cannot do more.

While this is a strong restriction, we will show in Theorem 18 that this suffices to reduce any problem in $\mathrm{NP}(\mathcal{R})$ to $\mathrm{SAT}(\mathcal{R})$ for any commutative semiring $\mathcal{R}$.

Surrogate-reductions are appropriate for our setting:

Lemma 17. *Let $\mathcal{R}$ be a semiring and $f_i : (\Sigma \cup R)^* \to R, i = 1, 2$. If $f_2 \in \mathrm{NP}(\mathcal{R})$ and f_1 is Karp s-reducible to f_2 then $f_1 \in \mathrm{NP}(\mathcal{R})$.*

Proof. Let T be the reduction function and $R' = \{r_1, \ldots, r_k\}$ the associated set of semiring constants. Let $M = (\mathcal{R}, R'_M, Q, \Sigma, \iota, \sqcup, \delta)$ be an SRTM that solves f_2. We construct an SRTM $M' = (\mathcal{R}, R'_M \cup R', Q', \Sigma', \iota', \sqcup', \delta')$ that solves f_1. Note that we give M' access to both R'_M, the set of semiring constants from M, and R', the set of semiring constants, fixed in the reduction.

While SRTMs cannot distinguish different semiring values, they can check whether there is a semiring value on the tape in a given position. Thus, given some input $x \in (\Sigma \cup R)^*$ we can obtain in polynomial time a string $x' \in s_{S,V,E}(\Sigma)$ from x by using $SV^{n+k}E$ for the n^{th} semiring value occurrence in x (from left to right). E.g. the string $x = x_1x_2x_3r_1x_4x_5r_2 \in (\Sigma \cup R)^*$, where $x_1, \ldots, x_4 \in \Sigma$ and $r_1, r_2 \in R$, leads to the string $x' = x_1x_2x_3SVEx_4x_5SVVE \in s_{S,V,E}(\Sigma)$ when $k = 0$. Then, we can apply T to x' in polynomial time. Finally, we can execute the algorithm for f_2 on $T(x')$. Here, we only need polynomial extra time to look up the $(n-k)^{\text{th}}$ semiring value in the original input when we need to make a transition with a semiring value under the head and read SV^nE for $n > k$. For $n \leq k$ we cannot transition with a weight under the head, since r_n does not necessarily occur in x. However, since we added R' to the set of constants that M' has access to, we can simply specify a transition with weight r_n directly. □

With this in mind, we can finally state the main result of the section.

Theorem 18. $\mathrm{SAT}(\mathcal{R})$ *is* $\mathrm{NP}(\mathcal{R})$*-complete w.r.t. Karp s-reductions for every commutative semiring* $\mathcal{R}$*.*

Proof (sketch). We prove membership constructively, using the SRTM Algorithm 1. We need to prove that $\textsc{Eval}_{\mathcal{R}}(\alpha, \emptyset) = [\![\alpha]\!]_{\mathcal{R}}(\emptyset)$, for any ΣBF α. When considering the formula α we could exhaustively apply the distributive law for every addition $(+, \Sigma a)$ that occurs inside a multiplication $(*)$. Then one obtains a sum of products of values from the semiring and literals. In general this would, however, lead to a formula that is exponentially bigger than α. Instead, the idea of $\textsc{Eval}_{\mathcal{R}}$ is to non-deterministically generate every summand s that we *would* have after having exhaustively applied the distributive law. For each non-deterministically generated s consisting of factors $f_1, \ldots, f_m$, $\textsc{Eval}_{\mathcal{R}}$ has a computation path that includes a transition with weight f_i for each $i = 1, \ldots, m$. This happens in such a way that each sequence of choices leads to a different s and such that every s is covered by a sequence of choices. Thus, the sum, using $\oplus$, of the value of all execution paths for a call to $\textsc{Eval}_{\mathcal{R}}(\alpha, \emptyset)$ is equal to $[\![\alpha]\!]_{\mathcal{R}}(\mathcal{I})$.

To prove the $\mathrm{NP}(\mathcal{R})$-hardness, we can generalize the well-known Cook-Levin Theorem, cf. (Gary & Johnson, 1979). Here, we need to rewrite disjunctions $a \vee b$ into formulas $a + \neg a * b$ s.t. these constraints can only be equal to either $e_\oplus$ or $e_\otimes$ but not $e_\otimes \oplus e_\otimes$. Apart from that we only need to add the weights of transitions to the formula. □

Nevertheless, before we prove a wide range of semiring formalism to be $\mathrm{NP}(\mathcal{R})$-complete, we discuss in the following possible alternative choices in the definition of SRTMs and their effect on the computational power of SRTMs.

4.2.1 Alternative SRTM Definitions

There are multiple possible small changes in the definition of SRTMs that would lead to an alternative model of non-deterministic computations over semirings and therefore also to an alternative complexity class $\mathrm{NP}(\mathcal{R})$. We discuss some of these possible changes and their implications on the power of SRTMs, to explain our rationale for choosing this definition of SRTMs.

Allowing Semiring Operations on the Tape One alternative would be to remove the weights from the transition functions and instead to let the machine multiply and add values on a tape explicitly. We see two main options that would enable this change. The first is to depart from the idea to use semiring values on the tape and to encode them in some finite alphabet instead. This would have the positive effect of getting a definition that is very close to the definition of typical non-deterministic TMs. However, this comes with two significant drawbacks.

First, it implies that we need to choose an encoding of the semiring values. As we will see later on, in Examples 11 and 12, the choice of the encoding can strongly influence the complexity of semiring operations. We want to avoid this, since we want to use SRTMs to characterize the complexity associated with a semiring, rather than the complexity associated with the encoding of its values.

Second, it would allow for the computation of functions that cannot be Karp-reduced to $\mathrm{SAT}(\mathcal{R})$.

Example 7 (Separating Function I)**.** *The function $f : \{0,1\}^* \to \mathbb{N}[(x_i)_\infty]$, $f(w) = x_1 + x_2 + \cdots + x_{2^{|w|}}$ cannot be Karp reduced to $\mathrm{NP}(\mathbb{N}[(x_i)_\infty])$ using an encoding $e : \mathbb{N}[(x_i)_\infty] \to \Sigma^*$ in some appropriate alphabet $\Sigma = \{0,1\} \cup \{\dots\}$, where coefficients, exponents and indices are encoded in binary, and a polynomial is encoded as a list of the encodings of the monomials. Assume that on the contrary, α_w is the $\mathrm{SAT}(\mathbb{N}[(x_i)_\infty])$-instance corresponding to input w to f. Then α_w needs to contain each of the monomials $x_1, \dots, x_{2^{|w|}}$ and is therefore necessarily of size exponential in $|w|$. Since Karp reductions must be computable in polynomial time, this is impossible.*

On the other hand, when we explicitly write semiring values in encoded form on the alphabet of a TM, we can guess $1 \leq i \leq 2^{|w|}$ and return x_i. Then the sum of all computation paths is $f(w)$.

Alternatively, we could allow for the computation of sums/products of pairs of semiring values, which can be written to the tape (e.g. using multiple tapes or heads). Without additional restrictions, this would also give SRTMs more power than what can be expressed using $\mathrm{SAT}(\mathcal{R})$.

Example 8 (Separating Function II)**.** *Consider the machine over the semiring $\mathbb{N}$ that takes as input a pair (k, x), where $k \in \mathbb{N}$ and $x \in \{0,1\}^*$ has length $|x| = n$, and computes a semiring value as output as follows: first, it sets $x_1 = k$ and then iteratively sets $x_{i+1} = x_i * x_i$ for $i = 2, \dots, n$. Then the output of the machine is $x_n = k^{2^n}$. This computation is possible if we allow computations on the tape as described above. On the other hand, every Karp s-reduction from (k, x) to a $\mathrm{SAT}(\mathbb{N})$-instance α satisfies that $[\![\alpha]\!]_{\mathbb{N}}(\emptyset)$, the result of evaluating α, is in $\mathcal{O}(k^{\mathcal{O}(n^c)} \cdot 2^{\mathcal{O}(n^c)})$, where c is a constant. This is because we can only use the semiring value k in α when using s-reductions and since we know that any $\mathrm{SAT}(\mathbb{N})$-instance α has a value in $\mathcal{O}(\max_{r \in \alpha} r^{|\alpha|} \cdot 2^{|\alpha|})$.*

Note that interestingly, in a circuit version of $\mathrm{SAT}(\mathbb{N})$ we *are* able to express the value k^{2^n} for $k \in \mathbb{N}$ using an instance of size polynomial in n as they also allow reuse of partial results multiple times. This also implies that even though CircuitSAT and SAT are Karp-reducible to one another, the same is in general not the case for $\mathrm{SAT}(\mathcal{R})$ and its circuit version.

Nevertheless, by restricting the semiring computations on the tape to be *non-recursive*, i.e., by only allowing the use of intermediate results only once, this problem is avoided. With this restriction in place the two definitions would result in two machine models with the same power that can simulate each other. Since, however, the non-recursiveness restriction is a semantic rather than a syntactic one (and thus harder to verify and enforce) we prefer our original definition.

Allowing Decisions Based on Semiring Values Recall that Condition 3. on the weighted transition function δ requires that for each $((q_1, \sigma_1), (q_2, \sigma_2), e, r) \in \delta$,

$$\sigma_1 \in R \Rightarrow \ [\forall \sigma_1' \in R : ((q_1, \sigma_1'), (q_2, \sigma_1'), e, \sigma_1') \in \delta] \ \vee \ [\forall \sigma_1' \in R : ((q_1, \sigma_1'), (q_2, \sigma_1'), e, r) \in \delta].$$

Therefore, we cannot distinguish different semiring values that are on the tape *during* the computation. That is, we cannot draw conclusions about the semiring value on the tape based on the state q_2 we transition to or based on the symbols on the tape that are non-semiring values. Furthermore, we cannot distinguish semiring values on the tape based

on the weight of the transitions they are involved in, if the weight is not the semiring value itself but a constant from R'. SRTMs without this property may not necessarily be finitely specified. Apart from that, having the ability to take different transitions based on the semiring value would allow an SRTM over semiring values from $\{0,1\}^*$ to distinguish strings that encode halting Turing machines from strings that do not, in constant time. This is obviously not desirable.

Allowing Semiring Values to be Written Also the first restriction on the weighted transition relation, which constitutes that we cannot write new semiring values to the tape or change existing ones, may seem questionable at first glance. Allowing to write or change semiring values in an unrestricted manner would allow us to distinguish semiring values. We already described previously why this would be undesirable. However, it is possible to allow for writing and changing of semiring values on the tape under restrictions such as only allowing to copy values or write values from a finite set. This would neither lead to additional power nor would it make the specification of SRTM machines simpler. Therefore, we chose this simpler version of the definition of SRTMs and only note that this restricted form of writing semiring values can be simulated.

5. Completeness Results for Semiring Frameworks

SAT($\mathcal{R}$) and NP($\mathcal{R}$) generalize SAT and NP to the semiring setting. As we discussed above their definitions equip SAT($\mathcal{R}$) and NP($\mathcal{R}$) with appropriate power. This is underlined also by the fact that as one expects SAT($\mathcal{R}$) is NP($\mathcal{R}$)-complete. Therefore, we now have all the necessary prerequisites for a complexity analysis of frameworks that are defined dependent on a semiring parameter.

In the following, we apply the results from above to different semiring formalisms in AI.

5.1 Sum-Of-Products Problems

Another prototypical problem that is similar to SAT($\mathcal{R}$) is the Sum-of-Products Problem SumProd($\mathcal{R}$) (Bacchus et al., 2009). It is defined as follows

Definition 19 (SumProd($\mathcal{R}$))**.** *Given a finite domain $\mathcal{D}$ and functions $f_i : \mathcal{D}^{j_i} \to R, i = 1, \ldots, n$, compute*

$$\bigoplus_{X_1,\ldots,X_m \in \mathcal{D}} \bigotimes_{i=1}^{n} f_i(\vec{Y_i}), \tag{1}$$

where $\vec{Y_i}$ is a vector with entries from a set $\{X_1, \ldots, X_m\}$ of variables.

To solve a problem instance, we need to compute the sum of the products of the functions f_i for all assignments of the variables X_i. Here, the "sum" and the "product" are the addition $\oplus$ and multiplication $\otimes$ of the semiring $\mathcal{R}$, respectively.

There are two main differences between SAT($\mathcal{R}$) and SumProd($\mathcal{R}$). The first is that SumProd($\mathcal{R}$) does not admit sums that occur under products, whereas for SAT($\mathcal{R}$) sums and products can alternate arbitrarily. Second, SumProd($\mathcal{R}$) is not propositional but more related to first-order logic, since we are not summing up values using a quantifier $\Sigma a \alpha$ by evaluating α under both truth values of a. Instead, we are summing up values over all assignments to variables over a finite domain.

Nevertheless, if the functions f_i in the expression (1) are given explicitly as key value pairs of variable assignments and semiring values, we obtain:

Theorem 20. SAT($\mathcal{R}$) *is Karp s-reducible to* SumProd($\mathcal{R}$) *and vice versa for every commutative semiring* $\mathcal{R}$. *Thus,* SumProd($\mathcal{R}$) *is* NP($\mathcal{R}$)*-complete w.r.t. Karp s-reductions.*

In the Boolean case, i.e. $\mathcal{R} = \mathbb{B}$, SumProd($\mathcal{R}$) essentially corresponds to Constraint Satisfaction Problems (CSP). Here, the reduction from SAT($\mathcal{R}$) to SumProd($\mathcal{R}$) borrows from the fact that 3SAT is NP-complete. However, for semirings in general establishing this result is more difficult, since the Tseitin-transformation (Tseitin, 1983), which is used for the proof that 3SAT is NP-complete, does not work anymore in the same way. The idea of the Tseitin-transformation is to introduce a new variable a_β for a complex subformula β such that a_β takes the value of β. In the Boolean case, we recall that this is established by means of the formula $a_\beta \leftrightarrow \beta$. In our case, the value of a subformula β may not only be $\top$ or $\bot$ as in the Boolean case but can take one of exponentially many different values in the size of β. So, while we could introduce variables $a_{\beta,r}$ that are true iff β evaluates to r, this would lead to an exponential explosion and is, thus, not helpful for proving the existence of a Karp s-reduction.

Instead, we need to exploit the distributive law. While naive application of it also leads to a formula of exponential size, we can avoid this by exploiting the distributivity implicitly during evaluation as in Algorithm 1. Intuitively, the idea here is to introduce new variables a_{β_1}, a_{β_2} for each non-deterministic choice introduced by a subformula $\alpha = \beta_1 + \beta_2$, which tell us whether we include β_1 or β_2. A formal proof is included in Appendix B.

5.2 Weighted First-Order Logic

Weighted first-order logics were introduced by Mandrali and Rahonis (2015) for expressivity characterizations and by Eiter and Kiesel (2020) for a quantitative extension of ASP.

They are defined over a *signature* $\sigma = \langle \mathcal{D}, \mathcal{P}, \mathcal{X} \rangle$ with predicates $p \in \mathcal{P}$ that have a fixed arity $r(p) \in \mathbb{N}$ over a domain $\mathcal{D}$ and variables in $\mathcal{X}$.

Definition 21 (Syntax)**.** *Let* $\sigma = \langle \mathcal{D}, \mathcal{P}, \mathcal{X} \rangle$ *be a signature and* $\mathcal{R} = (R, \oplus, \otimes, e_\oplus, e_\otimes)$ *be a commutative semiring. The* weighted σ-formulas *over* $\mathcal{R}$ *are of the form* α *given by the grammar*

$$\alpha ::= k \mid p(\vec{x}) \mid \neg p(\vec{x}) \mid \alpha + \alpha \mid \alpha * \alpha \mid \Sigma x \alpha \mid \Pi x \alpha.$$

Here $k \in R$, $p \in \mathcal{P}$, $\vec{x} \in (\mathcal{D} \cup \mathcal{X})^{r(p)}$ *and* $x \in \mathcal{X}$. *A* weighted σ-sentence *is a weighted* σ*-formula that does not contain free variables.*

Note that we again only allow for negation in front of atoms $p(\vec{x})$.

Definition 22 (Semantics)**.** *A* σ-interpretation *is a subset* $\mathcal{I}$ *of* $\{p(\vec{x}), \neg p(\vec{x}) \mid p \in \mathcal{P}$ *and* $\vec{x} \in \mathcal{D}^{r(p)}\}$ *s.t.* $\neg p(\vec{x}) \in \mathcal{I} \Leftrightarrow p(\vec{x}) \notin \mathcal{I}$ *for all* $p \in \mathcal{P}, \vec{x} \in \mathcal{D}^{r(p)}$. *Given a weighted* σ*-sentence* β *and a* σ*-interpretation* $\mathcal{I}$ *the semantics* $[\![\beta]\!]_\mathcal{R}(\mathcal{I})$ *of* β *over* $\mathcal{R}$ *w.r.t.* $\mathcal{I}$ *is defined as*

$$[\![\Sigma x \beta]\!]_\mathcal{R}(\mathcal{I}) = \bigoplus_{d \in \mathcal{D}} [\![\beta\{x \mapsto d\}]\!]_\mathcal{R}(\mathcal{I})$$
$$[\![\Pi x \beta]\!]_\mathcal{R}(\mathcal{I}) = \bigotimes_{d \in \mathcal{D}} [\![\beta\{x \mapsto d\}]\!]_\mathcal{R}(\mathcal{I})$$

The rest of the cases are as in Definition 10, where we identify $p(\vec{x})$ with a propositional variable $v_{p(\vec{x})}$.

We consider the evaluation of ΣFO σ-formulas, which only use sum quantifiers (i.e. Σx):

ΣFO-Eval($\mathcal{R}$): Given a weighted ΣFO σ-sentence α over the commutative semiring $\mathcal{R}$ and a σ-interpretation $\mathcal{I}$, compute $[\![\alpha]\!]_{\mathcal{R}}(\mathcal{I})$.

Quantitative first-order evaluation problems are present for example in probabilistic inference from Bayesian networks (Van den Broeck, Meert, & Darwiche, 2014); further, Eiter and Kiesel (2020) showed that aggregation and other extensions of Answer Set Programming can be modeled as ΣFO-Eval($\mathcal{R}$) over different semirings $\mathcal{R}$. This opens up a perspective of optimization based on algebraic rewriting paired with heuristics, as common e.g. for query languages in databases.

The ΣFO-Eval($\mathcal{R}$) problem is very similar to SAT($\mathcal{R}$). Indeed, under the assumption that $\mathcal{I}$ is given explicitly as key value pairs of predicates with variable assignments and truth values, we obtain:

Theorem 23. *Problem* ΣFO-Eval($\mathcal{R}$) *is Karp s-reducible to* SAT($\mathcal{R}$)*, and vice versa, and thus* NP*($\mathcal{R}$)-complete w.r.t. Karp s-reductions for every commutative semiring $\mathcal{R}$.*

Proof (sketch). $\Rightarrow$: Let α be a ΣBF over $\mathcal{R}$. We choose $\sigma = \langle\{\bot, \top\}, \{t(.)\}, \{x_{v_1}, \ldots, x_{v_n}\}\rangle$ and $\mathcal{I} = \{t(\top), \neg t(\bot)\}$ and replace every propositional variable v in α by $t(x_v)$.
$\Leftarrow$: We replace $p(\vec{x})$ by $\Sigma_{p(\vec{d})\in\mathcal{I}}\Pi_{x_i\in\vec{x}} v_{x_i,d_i}$, where $\vec{x} = x_1, \ldots, x_{r(p)}$, $\vec{d} = d_1, \ldots, d_{r(p)}$, and $v_{x,d}$ means that variable x has value d. We add constraints such that when both v_{x_i,d_i} and $v_{x_i,d_i'}$ are true, then $d_i = d_i'$ and add a quantifier $\Sigma v_{x_i,d_i}$ for each pair (x_i, d_i). □

5.3 Semiring-based Constraint Satisfaction Problems

Recall the definition of SCSPs from Section 3. We see that the computation of $blevel(P)$ is a sum of the products of the values of the constraints in P over all possible variable assignments. If def is given explicitly as key value pairs of variable assignments and semiring values, we obtain:

Theorem 24. *Computing blevel(.) over $\mathcal{R}$ is Karp s-reducible to* SumProd*($\mathcal{R}$) and vice versa, and thus* NP*($\mathcal{R}$)-complete w.r.t. Karp s-reductions for every commutative semiring $\mathcal{R}$.*

Proof (sketch). The variables of the constraint problem are the ones of SumProd($\mathcal{R}$) and for each constraint $\langle def_i, con_i\rangle$ the function def_i is a function f_i in SumProd($\mathcal{R}$); $blevel(.)$ is the solution of SumProd($\mathcal{R}$). □

5.4 Algebraic Model Counting

Recall the definition of AMC from Section 3. Since AMC also follows the guess-evaluate-aggregate pattern, we can show:

Theorem 25. *AMC over $\mathcal{R}$ is Karp s-reducible to* SAT($\mathcal{R}$) *and vice versa, and thus* NP*($\mathcal{R}$)-complete w.r.t. Karp s-reductions for every commutative semiring $\mathcal{R}$.*

The main differences between AMC and SAT($\mathcal{R}$) are that, on the one hand, in AMC the logical part and the part that weights an interpretation are explicitly separated, whereas for SAT($\mathcal{R}$) they are intertwined. Second, while AMC only allows for weight functions that can be expressed as a product of weights, SAT($\mathcal{R}$) allows for complex arithmetic expressions over the semiring.

Proof (sketch). $\Rightarrow$: We can translate T into a ΣBF β and weight it with α using $(v_i * \alpha(v_i) + \neg v_i * \alpha(\neg v_i))$ for $v_i \in \mathcal{V}$.
$\Leftarrow$: As we noticed before, if we were to apply the distributive law exhaustively on a SAT($\mathcal{R}$)-instance α, then the expression would be a sum of products of the semiring values r that occur in the input ΣBF and the literals. We add for each occurrence r_i of r a variable v_r^i with $\alpha(v_r^i) = r, \alpha(\neg v_r^i) = e_\otimes$. Then we use T to specify which semiring values are included in the sum. □

5.5 Datalog Semiring Provenance

Green at al. (2007) introduced a semiring-based semantics for relational algebra queries. Its main use case as described in the paper is to allow queries for the provenance of answers to standard queries. The semantics is capable of expressing bag semantics, why-provenance and more. Here intuitively, bag semantics tell us not only *whether* we can derive a query but also *how often* we can derive it. Why-provenance tells us instead the different reasons *why* we have to derive a query. There are more possibilities but generally, the semiring semantics allows us to obtain more, often quantitative, information about the derivations of a query.

For positive logic programs, i.e. datalog programs, their semantics over a commutative semiring $(R, \oplus, \otimes, e_\oplus, e_\otimes)$ is as follows: the label of a query result $q(\overline{x})$ is the sum (using $\oplus$) of the labels of derivation trees for $q(\overline{x})$, where the label of a derivation tree is the product (using $\otimes$) of the labels of the leaf nodes (i.e. extensional atoms). As the number of derivation trees may be countably infinite, Green et al. used ω-continuous semirings, where countable sums have a value. Examples of such semirings are $\mathbb{N}_\infty = (\mathbb{N} \cup \{\infty\}, +, \cdot, 0, 1)$, the natural number semiring extended with infinity, $\mathbb{B}$, and other idempotent semirings.

Example 9 (Bag Semantics)**.** *Consider the program*

$$r_1\colon q(X,Y) \leftarrow r(X,Y) \qquad\qquad r_2\colon q(X,Y) \leftarrow q(X,Z), q(Z,Y)$$

over $\mathbb{N}_\infty$ *and the semiring-weighted extensional database (edb)*

$$\{(r(a,b),2),(r(b,c),3),(r(b,a),4)\}.$$

Here, $\mathbb{N}_\infty$ *corresponds to bag semantics, which means that the label of a query* $q(\overline{x})$ *is the number of derivations of* $q(\overline{x})$*. The labels of the atoms in the semiring-weighted edb thus intuitively mean that there are* $2, 3$*, and* 4 *ways to derive* $r(a,b), r(b,c)$*, and* $r(b,a)$*, respectively.*

It follows that the label of $q(b,c)$ *under bag semantics is* 3 *due to the derivation* $q(b,c) \leftarrow r(b,c)$ *from* $r(b,c)$*, which itself has* 3 *derivations according to the edb, and the fact that this is the only derivation for* $q(b,c)$*. On the other hand, for* $q(a,a)$ *the label under bag semantics*

is ∞. This can be seen as follows: we can derive $q(a,a)$ from $r(a,b), r(b,a)$ in 2×3 ways according to their labels in the edb. But there exist infinitely many derivations of $q(a,a)$ from itself using rule r_2 instantiated as $q(a,a) \leftarrow q(a,a), q(a,a)$, leading to the label ∞.

For the problem of computing the label of a query under provenance semantics it is not immediately clear how it relates to the other problems discussed in this section. Kimmig et al. (2017) touched upon the relation of this and other so called *Algebraic Derivation Counting* (ADC) formalisms to AMC. They include not only provenance queries but also semiring parsing (Goodman, 1999) and semiring-weighted dynamic programming (Eisner, Goldlust, & Smith, 2005). They showed that it is possible to reduce an AMC-instance T, α to ADC by computing all models of the theory T and viewing them as derivations. As they noteed, this is obviously ineffective in general as the number of models of T can be exponential. The explicit computation of all the models is, however, not necessary for a reduction.

Theorem 26. *Given a positive, single-rule datalog program*

$$\Pi = \{q \leftarrow r_1(\vec{Y_1}), \ldots, r_n(\vec{Y_n})\}$$

and a semiring-weighted extensional database D over a commutative semiring $\mathcal{R}$ as input, it is $\mathrm{NP}(\mathcal{R})$-complete w.r.t. Karp s-reductions to compute the label of q.

We assume here, as usual, that the database D is stored explicitly as a set of tuples $(r(\overline{X}), k)$, where r is a predicate and k is its semiring weight.

Proof. Hardness: We provide a reduction from SumProd($\mathcal{R}$). Let $\mathcal{D}$ be a finite domain, $f_i : \mathcal{D}^{k_i} \to R, i = 1, \ldots, n$ be functions with the set of semiring values as co-domain, and let

$$\bigoplus_{X_1,\ldots,X_m \in \mathcal{D}} \bigotimes_{i=1}^{n} f_i(\vec{Y_i}), \tag{2}$$

be the SumProd($\mathcal{R}$)-instance at hand. Here, $\vec{Y_i}$ is a tuple consisting of k_i elements from $\{X_1, \ldots, X_m\}$, the set of variables of the SumProd($\mathcal{R}$)-instance, for $i = 1, \ldots, n$.

Consider the following positive, single-rule datalog program

$$\Pi = \{q \leftarrow r_1(\vec{Y_1}), \ldots, r_n(\vec{Y_n})\}$$

and the semiring-weighted extensional database

$$D = \bigcup_{i=1,\ldots,n} \{(r_i(\vec{y_i}), f_i(\vec{y_i})) \mid \vec{y_i} \in \mathcal{D}^{k_i}\}.$$

Both D and Π can be constructed in polynomial time from the SumProd($\mathcal{R}$)-instance. Furthermore, the label of the query q is equal to the result of evaluating the SumProd($\mathcal{R}$)-instance. Since SumProd($\mathcal{R}$) is NP($\mathcal{R}$)-complete w.r.t. Karp s-reductions, also provenance queries are NP($\mathcal{R}$)-hard.

NP($\mathcal{R}$)-membership: We can reduce to SumProd($\mathcal{R}$) analogously to obtain that provenance queries of the above form over $\mathcal{R}$ are in NP($\mathcal{R}$). This implies NP($\mathcal{R}$)-completeness overall. □

Note that we only showed NP($\mathcal{R}$)-completeness for a restricted case of provenance computations here, where the datalog program has only one rule. In fact, it is not possible to evaluate provenance queries in NP($\mathcal{R}$) in general. This is not because of hard special cases caused by the possibility of countably infinite sums, as already Green et al. (2007) gave an algorithm to recognize this case. The hardness comes from higher expressivity in two regards instead. First, in the Boolean setting, it is known that the evaluation of a datalog program is EXPTIME-complete, even for a fixed extensional database D (Dantsin, Eiter, Gottlob, & Voronkov, 2001). Thus, it seems unlikely that we can stay in NP($\mathcal{R}$), even when $\mathcal{R} = \mathbb{B}$.

Second, for other semirings, provenance queries are even provably more expressive than ΣBFs, already for ground programs Π, i.e., programs where only nullary predicates are used. The latter allow us to express circuit-like terms.

Example 10. *Let $n \in \mathbb{N}$ and consider the positive datalog program*

$$\Pi = \{a_{i+1} \leftarrow a_i, a_i \mid i = 1, \ldots, n-1\}$$

and the semiring-weighted extensional database $D = \{(a_1, 2)\}$. Then the label l_i of a_i is given by the recurrence relation $l_1 = 2$ and $l_{i+1} = a_i^2$ for $i > 0$. Therefore, the label of a_n is $(\ldots(2^2)^2\ldots)^2 = 2^{2^n}$, which is exponential in the size of the input.

5.6 Semiring-induced Propositional Logic

Larrosa et al. (2010) introduced an extension of propositional logic with a semiring-based semantics. They confined to CNF formulas and associated a weight from the semiring with each of the clauses. Formally:

Definition 27 (SRPL)**.** *Given a* semiring-labeled CNF $F = \{(C_1, w_1) \ldots, (C_n, w_n)\}$ *over a commutative semiring $\mathcal{R}$, where each C_i is a clause and $w_i \in R$ is a value from $\mathcal{R}$, the semantics $\phi_i(\mathcal{I})$ of clause C_i under interpretation $\mathcal{I}$ is*

$$\phi_i(\mathcal{I}) = \begin{cases} w_i & \text{if } \mathcal{I} \not\models C_i \\ e_\otimes & \text{otherwise.} \end{cases} .$$

Here, $\models$ is the satisfaction relation of classical propositional logic.

Furthermore, $\phi_F(\mathcal{I})$, the semantics of F under interpretation $\mathcal{I}$ is given by

$$\phi_F(\mathcal{I}) = \bigotimes_{i=1}^{n} \phi_i(\mathcal{I}).$$

This is reminiscent of the evaluation of a SumProd($\mathcal{R}$)-instance for one assignment $\vec{x}$ to the variables $\vec{X}$: both the term under the assignment $\vec{x}$ and $\phi_F(\mathcal{I})$ are products of values functionally determined by $\vec{x}$ and $\mathcal{I}$, respectively.

As with SumProd($\mathcal{R}$), one is not interested in the value for a single assignment but in the sum over all assignments, called *marginal* here, which is defined as follows:

Definition 28 (Marginalization Problem)**.** *Given a semiring-labeled CNF F over a commutative semiring $\mathcal{R}$ as input, the* marginalization problem over $\mathcal{R}$ *is to compute*

$$mrg(F) = \bigoplus_{\mathcal{I} \in \mathrm{Int}(F)} \phi_F(\mathcal{I}),$$

where $\mathrm{Int}(F)$ *denotes for a semiring-labeled CNF F the set of all interpretations of variables in F.*

Larossa et al. (2010) noted that many typical problems, like SAT, #SAT and MAX-SAT can be expressed as marginalization problems over different semirings. We show that moreover the marginalization problem is NP($\mathcal{R}$)-complete.

Theorem 29. *The marginalization problem over $\mathcal{R}$ is* NP*($\mathcal{R}$)-complete w.r.t. Karp s-reductions for every commutative semiring $\mathcal{R}$.*

Proof (Sketch). NP($\mathcal{R}$)-membership is easy to establish by a simple guess and evaluate algorithm.

We can prove NP($\mathcal{R}$)-hardness by a reduction from AMC over $\mathcal{R}$. W.l.o.g. we assume that the propositional theory T of the given AMC instance is a CNF with clauses $C_1, \ldots, C_n$ and the labeling function is α. Then the result of the marginalization problem for

$$F = \{(C_1, e_\oplus), \ldots, (C_n, e_\oplus)\} \cup \{(\neg v, \alpha(v)) \mid v \in V\} \cup \{(v, \alpha(\neg v)) \mid v \in V\},$$

where V is the set of variables of T, is the algebraic model count of T with respect to α. □

5.7 Other Frameworks

There are also other frameworks, such as algebraic Prolog (Kimmig et al., 2011), algebraic measures (Eiter et al., 2021), and semiring induced valuation algebras (Kohlas & Wilson, 2008) that we can show to be NP($\mathcal{R}$)-complete using our methodology and the previous reductions. We refrain from doing so, since the corresponding evaluation problems either already come with reductions to AMC (Kimmig et al., 2017; Eiter et al., 2021) or are very similar to other frameworks as in the case of semiring induced valuation algebras and SCSPs.

6. Relation to Well-known Classical Complexity Classes

We were able to show in the previous section that NP($\mathcal{R}$) can be used to characterize the complexity of many different semiring formalisms that each in a way use the guess-evaluate-aggregate pattern. However, these results only characterize the complexity over different semirings on a relatively abstract level by using SRTMs as a machine model. In order to gain more *tangible* insights into their complexity in a usual setting, we relate NP($\mathcal{R}$) to well-known classical complexity classes defined on Turing machines in this section.

For this purpose, we must encode semiring values in a finite tape alphabet. As we will see, the choice of encoding can strongly influence the complexity of computations over a semiring. To address this, we introduce conditions for *efficient* encodings that we can use to limit the influence of the encoding.

Equipped with this, we first link well known complexity classes such as NP, #P, and OptP with NP($\mathcal{R}$) for specific semirings, thus showing that NP($\mathcal{R}$) is in a sense a proper generalization of existing quantitative complexity classes.

We then turn to the more general case of commutative semirings as a whole and subclasses thereof. Here, we prove on the one hand general lower bounds, upper bounds and a tetrachotomy result that separates the non-trivial semirings into four distinct types of problems each of which is strongly connected to either NP, Mod_pP, NP $\cup$ Mod_pP or #P.

6.1 Encoding Semirings

To represent semiring values on classical Turing machines, we need to encode their values using an *encoding function*. We introduce the necessary notions and consider the implications that the choice of the encoding has on the complexity

Definition 30 (Encoding Function, Encoded Semiring)**.** *Let $\mathcal{R} = (R, \oplus, \otimes, e_\oplus, e_\otimes)$ be a semiring. Then an injective function $\tau : R \to \{0,1\}^*$ is an* encoding function*.*

We let $\tau(\mathcal{R}) = (\tau(R), \oplus, \otimes, \tau(e_\oplus), \tau(e_\otimes))$ denote the encoded semiring *given by $\tau(R) = \{\tau(r) \mid r \in R\}$ and $\odot$ on $\tau(R)$, s.t. $\tau(r_1) \odot \tau(r_2) = \tau(r_1 \odot r_2)$ for $\odot = \oplus, \otimes$.*

Given an encoded value $\tau(r)$ we define $\|r\|_\tau$, the size of r w.r.t. τ*, as the length of the bitstring $\tau(r)$, i.e. $|\tau(r)|$.*

Now, we can use classical machines to solve $\mathrm{SAT}(\tau(\mathcal{R}))$ and consider its complexity in terms of classical complexity classes. Naturally, it depends on the complexity of addition and multiplication. While for $\mathbb{N}, \mathbb{B}$ these operations are "easy", i.e., feasible in polynomial time given a binary encoding, this is not the case for arbitrary semirings. A single multiplication may even be undecidable.

Example 11 (Arbitrary Complexity Semirings)**.** *Given $M \subseteq \{0,1\}^*$ and $\succ$, the lexicographical order on $\{0,1\}^*$, we define the semiring $\mathcal{R}_M = (\{0,1\}^* \cup \{\mathbf{0}, \mathbf{1}\}, \max_\succ, \otimes, \mathbf{0}, \mathbf{1})$, where $\mathbf{1} \succ m \succ \mathbf{0}$ for $m \in \{0,1\}^*$ and*

$$m_1 \otimes m_2 := \begin{cases} \min_\succ(m_1, m_2) & m_1, m_2 \in M \cup \{\mathbf{0}\} = S \\ m_1 & m_1 \in S, m_2 \notin S \\ m_2 & m_2 \in S, m_1 \notin S \\ \min_\succ(m_1, m_2) & \textit{otherwise.} \end{cases}$$

Then multiplication requires deciding $m_i \in M$. When M corresponds to the Halting problem, we have undecidability.

However, the difficulty stems from the encoding.

Example 12 (Example 11 cont.)**.** *If the encoding τ maps $m \in \{0,1\}^*$ to $(m, 1)$ if $m \in M$ and to $(m, 0)$ if $m \notin M$, then multiplication and addition in $\tau(\mathcal{R}_M)$ are computable in linear time.*

Our intuition is that there are two sources of complexity. One is the encoding and the other seems to be the amount of information that the weighted semantics gives us about the formula. The latter is determined by the non-collapsing terms in the semiring. For illustration purposes, consider that over $\mathbb{N}[(x_i)_k]$ the coefficients c_1, c_2 are retained by the sum $c_1x_1 + c_2x_2$, over $\mathbb{N}$ only the sum $c_1 + c_2$ is retained after addition, and over $\mathbb{B}$ the value $c_1 \vee c_2$ only tells us if at least one of the values was 1. As a consequence $\mathrm{SAT}(\mathbb{N}[(x_i)_k])$-instances are at least as hard as $\mathrm{SAT}(\mathbb{N})$-instances, since $\mathbb{N}$ is a subset of $\mathbb{N}[(x_i)_k]$ with the same addition and multiplication on $\mathbb{N}$. Additionally, $\mathrm{SAT}(\mathbb{N})$-instances seem to be strictly harder than $\mathrm{SAT}(\mathbb{B})$-instances as NP can easily be reduced to #P.

We focus on the second source of complexity and address the first by introducing the notion of an efficient encoding.

Definition 31 (Efficiently Encoded Semiring)**.** *Let $\tau(\mathcal{R})$ be an encoded semiring. Then $\tau(\mathcal{R})$ is* efficiently encoded *if there exists a polynomial $p(x)$ s.t. for all $\tau(r_1), \ldots, \tau(r_n) \in \tau(R)$ it holds that*

1. $\|\bigotimes_{i=1}^{n} r_i\|_\tau \leq p(n) \max_{i=1,\ldots,n} \|r_i\|_\tau$,
2. $\|\bigoplus_{i=1}^{n} r_i\|_\tau \leq p(\log_2(n)) \max_{i=1,\ldots,n} \|r_i\|_\tau$, *and*
3. $\tau(r), \tau(r') \mapsto \tau(r \odot r')$ *is in* FP *for* $\odot = \oplus, \otimes$.

Conditions 1) and 2) ensure that successive multiplications resp. additions do not cause space explosion, even for sums with exponentially many terms. Condition 3) is necessary since we at least need single additions and multiplications to be tractable if we want to solve problems over a semiring efficiently. The idea behind these conditions is to separate encodings that behave "efficiently" both with respect to space and time and those that do not. For this, we use restrictions that mirror and slightly relax the properties that the prototypical binary encoding "bin(.)" of integers satisfies.

As desired and expected, for integers, a binary representation satisfies the restrictions, whereas a unary representation does not.

Example 13. *With binary representation* $\mathrm{bin}(n) = b_0 \ldots b_m$ *s.t.* $n = \sum_{i=1}^{m} b_i 2^i$*, the semiring* $\mathbb{N}$ *of the natural numbers is efficiently encoded. With the unary representation* $\mathrm{unary}(n)$*, which represents a natural number as a string of* n *characters, the natural numbers are* not *efficiently encoded and in fact do not satisfy any of the conditions 1)-3).*

Furthermore, also $\mathbb{N}[(x_i)_\infty]$ *is not efficiently encoded when polynomials are encoded as lists of monomials and every monomial of the form* $c_{\vec{i}} x_{i_1}^{e_{i_1}} \ldots x_{i_n}^{e_{i_n}}$ *is encoded using a unary or binary representation of the coefficients* $c_{\vec{i}}$*, the exponents* e_{i_j}*, and variable indices* i_j*.*

The conditions of Definition 31 are mild in practice, as besides $\mathbb{N}$ many common semirings, e.g. $\mathbb{Z}, \mathbb{Q}, \mathcal{R}_{\max}$, are efficiently encodable.

When a semiring is efficiently encoded, we can establish a polynomial space upper bound on the complexity of ΣBF evaluation.

Proposition 32 (FPSpace(poly) Upper-Bound)**.** *If $\tau(\mathcal{R})$ is an efficiently encoded commutative semiring, then* $\mathrm{SAT}(\tau(\mathcal{R}))$ *is in* FPSpace(poly)*.*

Proof (Sketch). As we have seen before, the value of an $\mathrm{SAT}(\tau(\mathcal{R}))$-instance α can be expressed as a sum of products of the semiring values in α by implicitly applying the distributive law on α during evaluation as in Algorithm 1. The length of the encoding of each addend of this sum is polynomially bounded in the size of α due to condition 1 of efficient encodedness. Since there are only single exponentially many such addends, it follows from condition 2 of efficient encodedness that the final result also has polynomial size in the size of the biggest addend and is therefore also polynomial in the size of the input. By generating the addends one after the other we, thus, stay in polynomial space. □

6.2 Results for Specific Semirings

Before we consider the complexity in dependence on the semiring parameter in general, we relate well-known classical complexity classes to $\mathrm{NP}(\tau(\mathcal{R}))$ by showing that they share $\mathrm{SAT}(\tau(\mathcal{R}))$ as a complete problem for different specific semirings.

Theorem 33. *For* $(\mathcal{R}, \mathcal{C}) = (\mathbb{B}, \mathrm{NP})$, $(\mathbb{N}, \#\mathrm{P})$, $(\mathbb{Z}, \mathrm{GapP})$, $(\mathcal{R}_{\max}, \mathrm{OptP})$ *and the binary representation* bin *of the integers,* $\mathrm{SAT}(\mathrm{bin}(\mathcal{R}))$ *is* $\mathcal{C}$*-complete w.r.t. Karp reductions.*

Proof (sketch). Membership holds as $\mathrm{bin}(n)$ satisfies Definition 31 and we can, thus, interpret Algorithm 1 as a non-deterministic Turing machine algorithm that generates the correct polynomial size outputs. For hardness we use reductions from SAT, #SAT, computing the permanent of an integer matrix, and LexMaxSAT, respectively. □

Note that even though every function in OptP can be reduced to a $\mathrm{SAT}(\mathrm{bin}(\mathcal{R}_{\max}))$-instance, there are functions in OptP that cannot be computed in $\mathrm{NP}(\mathrm{bin}(\mathcal{R}_{\max}))$, e.g., $x \mapsto 2^{|x|}$. Informally, this is because SRTMs can only generate semiring values by multiplying numbers from $\{e_\oplus, e_\otimes\}$ or the input. The fact that multiplication in $\mathcal{R}_{\max}$ is $+$ implies that we can only generate numbers that are polynomial in the numbers in the input. For NP,#P and GapP this effect does not occur.

6.3 Results for Classes of Semirings

Apart from completeness results for specific semirings, we also care about intuition on why some semirings come with a higher complexity than others, and about results that help to characterize new semirings based on their properties. Thus, we consider the complexity of different classes of semirings that satisfy some property, which allows us to make non-trivial claims about the complexity entailed by evaluating problems over them.

6.3.1 General Lower Bound and Tetrachotomy

First, we consider how hard $\mathrm{SAT}(\mathcal{R})$ has to be at least in general. For this we show the following result:

Theorem 34 (General Lower Bound)**.** *Let* $\tau(\mathcal{R})$ *be an encoded commutative semiring. Then one of the following holds:*

1. $\tau(\mathcal{R}) = \mathbb{T}$*, i.e., the semiring is trivial*
2. $\mathrm{SAT}(\tau(\mathcal{R}))$ *is* $\mathrm{Mod}_p\mathrm{P}$*-hard with respect to counting reductions for some* $p \in \mathbb{N}$ *or*
3. $\mathrm{SAT}(\tau(\mathcal{R}))$ *is* NP*-hard with respect to counting reductions.*

This means that the problem is either trivial (case 1), or expected not to be solvable in FP under common complexity theoretic assumptions such as the Exponential Time Hypothesis (Impagliazzo & Paturi, 2001), otherwise.

Before we continue with the proof, we first make the following remark that we will use throughout the rest of the paper. Let $C = l_1 \vee l_2 \vee l_3$ be a clause and let $\overline{l}$ for a literal l denote the opposite of l, i.e., $\overline{a} = \neg a$ and $\overline{\neg a} = a$ for any propositional variable a. Then:

Lemma 35. *For every non-trivial semiring* $\mathcal{R}$*, an interpretation* $\mathcal{I}$ *satisfies* C *iff for*

$$d(C) := l_1 + \overline{l_1} * l_2 + \overline{l_1} * \overline{l_2} * l_3$$

it holds that $[\![d(C)]\!]_{\mathcal{R}}(\mathcal{I}) = e_\otimes$*. Furthermore, if* $\mathcal{I}$ *does not satisfy* C*, then* $[\![d(C)]\!]_{\mathcal{R}}(\mathcal{I}) = e_\oplus$*.*

Proof of Lemma 35. We proceed by case distinction. Let $\mathcal{I}$ be an interpretation.

Case $\mathcal{I} \models l_1$: Then

$$[\![d(C)]\!]_{\mathcal{R}}(\mathcal{I}) = e_\otimes \oplus e_\oplus \oplus e_\oplus = e_\otimes$$

since $[\![l_1]\!]_{\mathcal{R}}(\mathcal{I}) = e_\otimes$ and $[\![\overline{l_1}]\!]_{\mathcal{R}}(\mathcal{I}) = e_\oplus$.

Case $\mathcal{I} \not\models l_1$, $\mathcal{I} \models l_2$: Then

$$[\![d(C)]\!]_{\mathcal{R}}(\mathcal{I}) = e_\oplus \oplus e_\otimes \oplus e_\oplus = e_\otimes.$$

Case $\mathcal{I} \not\models l_1$, $\mathcal{I} \not\models l_2$, $\mathcal{I} \models l_3$: Then

$$[\![d(C)]\!]_{\mathcal{R}}(\mathcal{I}) = e_\oplus \oplus e_\oplus \oplus e_\otimes = e_\otimes.$$

Case $\mathcal{I} \not\models l_1$, $\mathcal{I} \not\models l_2$, $\mathcal{I} \not\models l_3$: Then

$$[\![d(C)]\!]_{\mathcal{R}}(\mathcal{I}) = e_\oplus \oplus e_\oplus \oplus e_\oplus = e_\oplus.$$

This covers all cases and we see that when $\mathcal{I} \models C$, then $[\![d(C)]\!]_{\mathcal{R}}(\mathcal{I}) = e_\otimes$ and otherwise $[\![d(C)]\!]_{\mathcal{R}}(\mathcal{I}) = e_\oplus$. □

Furthermore, for $n \in \mathbb{N}$ and $r \in R$, we let

$$n \cdot r = \begin{cases} e_\oplus & n = 0 \\ ((n-1) \cdot r) \oplus r & n > 0 \end{cases}.$$

Proof of Theorem 34. Assume $\tau(\mathcal{R}) \neq \mathbb{T}$, since otherwise we are done.

We distinguish the following cases. If for all $n \in \mathbb{N}$ it holds that $n \cdot \tau(e_\otimes) = \tau(e_\oplus)$ implies $n = 0$, then $\mathrm{SAT}(\tau(\mathcal{R}))$ is NP-hard with respect to counting reductions. For the proof, consider a 3CNF $\phi = \bigwedge_{i=1}^{n} C_i$ with variables $v_1, \ldots, v_m$. Then ϕ is satisfiable iff for $\alpha = \Sigma v_1 \ldots \Sigma v_m \Pi_{i=1}^{n} d(C_i)$ it holds that $[\![\alpha]\!]_{\tau(\mathcal{R})}(\emptyset) \neq \tau(e_\oplus)$.

Otherwise, let $M = \langle \tau(e_\otimes) \rangle$ and

$$p = \min\{n \in \mathbb{N} \mid n > 0, n \cdot \tau(e_\otimes) = e_\oplus\}.$$

We claim that $M = \{n \cdot \tau(e_\otimes) \mid n \in \mathbb{N}\}$. This can be seen as follows. By definition of generated semirings, $\langle \tau(e_\otimes) \rangle$ is the smallest semiring with addition $\oplus$ and multiplication $\otimes$ that contains $\tau(e_\otimes)$. We go over the relevant semiring axioms one by one. First, $\tau(e_\oplus) = 0 \cdot \tau(e_\otimes)$ and thus of the desired form. Second, $\tau(e_\otimes) = 1 \cdot \tau(e_\otimes)$ and thus of the desired form. Next, M must be closed under $\oplus$. So let $n \cdot \tau(e_\otimes), m \cdot \tau(e_\otimes) \in M$. Then,

$$(n \cdot \tau(e_\otimes)) \oplus (m \cdot \tau(e_\otimes)) = ((n+1) \cdot \tau(e_\otimes)) \oplus ((m-1) \cdot \tau(e_\otimes)) = \cdots = (n+m) \cdot \tau(e_\otimes),$$

since $\oplus$ is commutative. Last but not least, M must be closed under $\otimes$. So let $n \cdot \tau(e_\otimes), m \cdot \tau(e_\otimes) \in M$. Then,

$$\begin{aligned} (n \cdot \tau(e_\otimes)) \otimes (m \cdot \tau(e_\otimes)) &= (n \cdot \tau(e_\otimes)) \otimes ((m-1) \cdot \tau(e_\otimes)) \oplus (n \cdot \tau(e_\otimes)) \otimes \tau(e_\otimes) \\ &= \ldots \\ &= (n \cdot m) \cdot \tau(e_\otimes) \otimes \tau(e_\otimes) \\ &= (n \cdot m) \cdot \tau(e_\otimes), \end{aligned}$$

since $\otimes$ distributes over $\oplus$ and $\tau(e_\otimes)\otimes\tau(e_\otimes) = \tau(e_\otimes\otimes e_\otimes) = \tau(e_\otimes)$. This proves the claim.

Furthermore, for $n \geq p$ it holds that $n \cdot \tau(e_\otimes) = (n-p) \cdot \tau(e_\otimes)$. Therefore,

$$M = \{\tau(e_\oplus), 1 \cdot \tau(e_\otimes), \ldots, (p-1) \cdot \tau(e_\otimes)\} \equiv \mathbb{Z}_p.$$

It follows that $\mathrm{SAT}(\tau(\mathcal{R}))$ is $\mathrm{MOD}_p\mathrm{P}$-hard with respect to counting reductions. We give a reduction from $\mathrm{MOD}_p\mathrm{3CNF}$, which is the problem of deciding whether the number of models n that a given 3CNF $\phi = \bigwedge_{i=1}^{n} C_i$ with variables $v_1, \ldots, v_m$ has is not equal to zero modulo p; this problem is $\mathrm{MOD}_p\mathrm{P}$-complete (Beigel & Gill, 1992). For the proof of hardness, we again take $\alpha = \Sigma v_1 \ldots \Sigma v_m \Pi_{i=1}^{n} d(C_i)$ and observe that $[\![\alpha]\!]_{\tau(\mathcal{R})}(\emptyset) = n^* \cdot \tau(e_\otimes)$, where $n^* \equiv N \mod p$. Since the cardinality of M is finite, we can decide in constant time what the unique value $n^* \in \{0, \ldots, p-1\}$ is. The claim follows. □

If we restrict ourselves to the case where we cannot have weights and consider an efficiently encoded commutative semiring, we obtain a tetrachotomy result. This means, we split the complexity associated with a commutative semiring into four distinct cases depending on its properties. The properties we use are periodicity and offset of a periodic semiring.

Definition 36 (Periodicity, Offset)**.** *Let $\mathcal{R}$ be a periodic semiring. Then the* periodicity *of $\mathcal{R}$ is the smallest number $p > 0$ such that some $o \geq 0$ with $o \cdot e_\otimes = (o+p) \cdot e_\otimes$ exists. Let p be the periodicity of $\mathcal{R}$. Then the* offset *of $\mathcal{R}$ is the smallest number $o \geq 0$ such that $o \cdot e_\otimes = (o+p)e_\otimes$.*

Both periodicity and offset are well defined, i.e., there are unique numbers p, o that satisfy the definition for periodic semirings.

Note that a periodicity of p does not imply that $(p+1) \cdot e_\otimes = e_\otimes$.

Example 14. *Consider the semiring*

$$\mathbb{N}_{\leq o} = (\{0, 1, \ldots, o\}, +, \cdot, 0, 1),$$

where $i + j := \min(o, i+j)$ and $i \cdot j := \min(o, i \cdot j)$, i.e., the semiring over the natural numbers less than or equal to o. $\mathbb{N}_{\leq o}$ is periodic with periodicity 1 but $1+1 \neq 1$ when $o > 1$.

Corollary 37. *Let $\mathcal{R}$ be a periodic semiring with periodicity p and offset o. Then for all $r \in R$ and $o' \geq o$ it holds that $o' \cdot r = (o'+p) \cdot r$.*

Proof. This can be shown by exploiting that $e_\otimes$ is the neutral element for multiplication and the distributive law:

$$\begin{aligned}\textstyle\bigoplus_{i=1}^{o'} r = \bigoplus_{i=1}^{o'} r \otimes e_\otimes = r \otimes \bigoplus_{i=1}^{o'} e_\otimes &= r \otimes \left(\bigoplus_{i=1}^{o'-o} e_\otimes \oplus \bigoplus_{i=1}^{o} e_\otimes\right) \\ &= r \otimes \left(\bigoplus_{i=1}^{o'-o} e_\otimes \oplus \bigoplus_{i=1}^{o+p} e_\otimes\right) = r \otimes \bigoplus_{i=1}^{o'+p} e_\otimes = \bigoplus_{i=1}^{o'+p} r.\end{aligned}$$

□

With this in mind, we can show that the complexity of evaluating $\mathrm{SAT}(\tau(\langle e_\otimes \rangle_{\tau(\mathcal{R})}))$-instances, i.e., instances over some efficiently encoded semiring $\tau(\mathcal{R})$ that only use weights of the form $k \cdot \tau(e_\otimes)$, falls into one of four distinct cases. Furthermore, we show that the category a semiring falls into is determined by whether it is periodic, and if so, by its periodicity and offset.

Theorem 38 (Tetrachotomy)**.** *Let $\tau(\mathcal{R}) \neq \mathbb{T}$ be an efficiently encoded commutative semiring. Then we have exactly one of the four following situations:*

- *$\tau(\mathcal{R})$ is periodic with periodicity 1 and $\mathrm{SAT}(\tau(\langle e_\otimes \rangle))$ is NP-hard w.r.t. counting reductions and in $\mathrm{FP}^{\mathrm{NP}[\mathcal{O}(1)]}$;*
- *$\tau(\mathcal{R})$ is periodic with periodicity $p \geq 2$ and offset 0 and $\mathrm{SAT}(\tau(\langle e_\otimes \rangle))$ is $\mathrm{Mod}_p\mathrm{P}$-hard w.r.t. counting reductions and in $\mathrm{FP}^{\mathrm{Mod}_p\mathrm{P}[\mathcal{O}(1)]}$;*
- *$\tau(\mathcal{R})$ is periodic with periodicity $p \geq 2$ and offset $o > 0$ and $\mathrm{SAT}(\tau(\langle e_\otimes \rangle))$ is NP-hard, $\mathrm{Mod}_p\mathrm{P}$-hard w.r.t. counting reductions and in $\mathrm{FP}^{(\mathrm{NP} \cup \mathrm{Mod}_p\mathrm{P})[\mathcal{O}(1)]}$;*
- *$\tau(\mathcal{R})$ is not periodic and $\#\mathrm{SAT}$ is in $\mathrm{FNP}^{\mathrm{SAT}(\tau(\langle e_\otimes \rangle))[1]}$. Furthermore, if additionally there is a polynomial p such that for all $k \geq 0$ it holds that $2^{p(\|k \cdot e_\otimes\|)} \geq k$, then $\mathrm{SAT}(\tau(\langle e_\otimes \rangle))$ is in $\mathrm{FP}^{\#\mathrm{P}[1]}$.*

Note that practically for efficient evaluation also parallel oracle calls to NP and $\mathrm{Mod}_p\mathrm{P}$ could be used.

This result shows that if we only consider instances without semiring values, the already known complexity classes NP, $\mathrm{Mod}_p\mathrm{P}$ and #P suffice to bound the complexity in a reasonable fashion, even if we allow general commutative countable semirings. Unfortunately, the next subsection shows that this does not simply generalize to the case when we have non-trivial weights.

We also would like to draw special attention to the last case, where $\tau(\mathcal{R})$ is not periodic. Here, the bounds we can give are weaker than in the other three cases. This is because if a semiring is periodic, then $\tau(\langle e_\otimes \rangle)$ is finite, which implies that we can compute for any $r \in \tau(\langle e_\otimes \rangle)$ in constant time a value $k \in \mathbb{N}$ such that $r = k \cdot e_\otimes$. On the other hand, if $\tau(\mathcal{R})$ is not periodic, it is not clear whether given $r \in \tau(\langle e_\otimes \rangle)$ it is possible even in polynomial time to compute $k \in \mathbb{N}$ such that $r = k \cdot e_\otimes$. The restriction that $\tau(\mathcal{R})$ is efficiently encoded helps us, since it allows us to check in polynomial time whether $r = k \cdot e_\otimes$, given $\tau(r)$ and k. However, as described by Sharma and Singh (2016), it is an open problem whether easy to compute functions (like $f(k) = k \cdot e_\otimes$) have an easy to compute inverse (like $f^{-1}(\tau(r))$). This explains the weaker lower bound on the hardness of $\mathrm{SAT}(\tau(\langle e_\otimes \rangle))$.

For the upper bound, our proof for the reduction of $\mathrm{SAT}(\tau(\langle e_\otimes \rangle))$ to #SAT requires that for all $r \in \tau(\langle e_\otimes \rangle)$ it holds that the value $k \in \mathbb{N}$ such that $r = k \cdot e_\otimes$ is single exponential in the size of the encoding $\|r\|_\tau$. We leave the question of the existence of an efficiently encoded semiring, where this does not hold open.

Proof. Clearly, $\tau(\mathcal{R})$ is exactly one of

1. periodic with periodicity 1
2. periodic with periodicity $p \geq 2$ and offset 0
3. periodic with periodicity $p \geq 2$ and offset $o > 0$
4. not periodic

What is left to show is the corresponding claim for the complexity of $\mathrm{SAT}(\tau(\langle e_\otimes \rangle))$.

Case 1. So let $\tau(\mathcal{R})$ have periodicity 1 and let $\alpha = \Sigma v_1 \dots \Sigma v_l \gamma$, where γ is quantifier-free, be a $\mathrm{SAT}(\tau(\langle e_\otimes \rangle))$-instance. Since γ only contains weights from $\tau(\langle e_\otimes \rangle)$, we know that for any interpretation $\mathcal{I}$ the semantics of γ is $m \cdot \tau(e_\otimes)$ for some $m \geq 0$. Further, let o be the offset of $\tau(\mathcal{R})$. Then we know that $(o + m) \cdot \tau(e_\otimes) = o \cdot \tau(e_\otimes)$ for all $m \geq 0$. Thus, $[\![\gamma]\!]_{\tau(\mathcal{R})}(\mathcal{I}) \in \{0 \cdot \tau(e_\otimes), \dots, o \cdot \tau(e_\otimes)\}$ and also $[\![\alpha]\!]_{\tau(\mathcal{R})}(\emptyset) \in \{0 \cdot \tau(e_\otimes), \dots, o \cdot \tau(e_\otimes)\}$.

Thus, we can use o^2 queries to an NP-oracle to determine for each value $i = 1, \dots, o$ whether there are $1, \dots, o$ or more interpretations with value $i \cdot \tau(e_\otimes)$. From this we can derive the value of α. Thus, $\mathrm{SAT}(\tau(\langle e_\otimes \rangle))$ is in $\mathrm{FP}^{\mathrm{NP}[\mathcal{O}(1)]}$, since o only depends on the semiring and not on the instance. For NP-hardness we can use the same construction as in the proof of Theorem 34.

Case 2. So let $\tau(\mathcal{R})$ have periodicity $p \geq 2$ and offset 0 and let $\alpha = \Sigma v_1 \dots \Sigma v_l \gamma$, where γ is quantifier-free, be a $\mathrm{SAT}(\tau(\langle e_\otimes \rangle))$-instance. Since γ only contains weights from $\tau(\langle e_\otimes \rangle)$, we know that for any interpretation $\mathcal{I}$ the semantics of α is $m \cdot \tau(e_\otimes)$ for some $0 \leq m \leq p$.

Now, let $f \in \#\mathrm{P}$ be a function that evaluates $\mathrm{SAT}(\mathrm{bin}(\mathbb{N}))$-instances. We can use f on α by replacing every value $\tau(k \cdot e_\otimes)$ by $\mathrm{bin}(k)$. Since $k \equiv 0 \mod p$ iff $k \cdot \tau(e_\otimes) = 0 \cdot e_\otimes$, checking whether the semantics of α is $0 \cdot \tau(e_\otimes)$ is in $\mathrm{Mod}_p\mathrm{P}$ by definition.

Additionally, since $f \in \#\mathrm{P}$ implies $f + i \in \#\mathrm{P}$, we can use p queries to a $\mathrm{Mod}_p\mathrm{P}$-oracle to determine for each value $i = 1, \dots, p$ whether the semantics of α is $i \cdot \tau(e_\otimes)$. Thus, $\mathrm{SAT}(\tau(\langle e_\otimes \rangle))$ is in $\mathrm{FP}^{\mathrm{Mod}_p\mathrm{P}[\mathcal{O}(1)]}$, since p only depends on the semiring and not on the instance. For $\mathrm{Mod}_p\mathrm{P}$-hardness we can use the same construction as in the proof of Theorem 34.

Case 3. So let $\tau(\mathcal{R})$ have periodicity $p \geq 2$ and offset $o > 0$ and let $\alpha = \Sigma v_1 \dots \Sigma v_l \gamma$, where γ is quantifier-free, be a $\mathrm{SAT}(\tau(\langle e_\otimes \rangle))$-instance. Since γ only contains weights from $\tau(\langle e_\otimes \rangle)$, we know that for any interpretation $\mathcal{I}$ the semantics of α is $i \cdot \tau(e_\otimes)$ for some $0 \leq i \leq o + p$.

The rest is a combination of case 1. and 2.: we first check whether $i < o$ using o^2 calls to an NP-oracle. If $i < o$ we are done, otherwise, we use p calls to a $\mathrm{Mod}_p\mathrm{P}$-oracle to obtain the value i. For hardness we again use the same construction as in the proof of Theorem 34.

Case 4. So let $\tau(\mathcal{R})$ not be periodic and let $\alpha = \Sigma v_1 \dots \Sigma v_l \gamma$, where γ is quantifier-free, be a $\mathrm{SAT}(\tau(\langle e_\otimes \rangle))$-instance. Since γ only contains weights from $\tau(\langle e_\otimes \rangle)$, we know that for any interpretation $\mathcal{I}$ the semantics of α is $m \cdot \tau(e_\otimes)$ for some $m \in \mathbb{N}$. If we can find out for each r that occurs in γ the natural number k_r such that $r = k_r \cdot e_\otimes$, we can reduce the evaluation of α to evaluating α over $\mathrm{bin}(\mathbb{N})$, as this will have result m. Given m we can then compute $m \cdot \tau(e_\otimes)$ in polynomial time. This establishes the existence of a counting reduction from $\mathrm{SAT}(\tau(\langle e_\otimes \rangle))$ to #SAT, if we can compute given r the natural number k_r such that $r = k_r \cdot e_\otimes$.

If we know that for $k \cdot \tau(e_\otimes)$ it holds that $2^{p(\|k \cdot e_\otimes\|)} \geq k$ for some polynomial p, then, given the encoding $\tau(r)$ we can guess all numbers $0 \leq k \leq 2^{p(\|r\|_\tau)}$ and check whether $k \cdot \tau(e_\otimes) = \tau(r)$ in non-deterministic polynomial time. By doing this for all weights $\tau(r)$ that occur in α before evaluating α non-deterministically over $\mathbb{N}$ and rejecting if a guess was wrong, we can reduce $\mathrm{SAT}(\tau(\langle e_\otimes \rangle))$ to #SAT.

As for the containment of #SAT in $\mathrm{FNP}^{\mathrm{SAT}(\tau(\langle e_\otimes \rangle))[1]}$, consider the following algorithm: given a #SAT-instance ϕ construct a $\mathrm{SAT}(\tau(\langle e_\otimes \rangle))$-instance α such that the semantics of

α is $m \cdot \tau(e_\otimes)$, where m is the number of models of ϕ. This is possible in polynomial time by first constructing a 3CNF $\psi = \bigwedge_{i=1}^{n} C_i$ with variables $v_1, \ldots, v_m$ and the same number of models and using $\alpha = \Sigma v_1 \ldots \Sigma v_m \Pi_{i=1}^{n} d(C_i)$, where $d(C_i)$ is as in Lemma 35. Then, evaluate α, guess $0 \leq n \leq 2^{|Vars(\phi)|}$, where $Vars(\phi)$ denotes the set of variables in ϕ, and compute $n \cdot \tau(e_\otimes)$. If $n \cdot \tau(e_\otimes)$ is equal to the semantics of α accept and return n, otherwise reject. □

6.3.2 Upper Bounds using Polynomial Semirings

We have seen results on the hardness of commutative semirings in general and some containment results for the case when we only use weights of the form $k \cdot e_\otimes$ for some $k \in \mathbb{N}$ in the evaluation of an instance. In order to also obtain containment results for the case when we do have arbitrary weights, we need to apply a different kind of strategy.

The problem with proper weights is that they intuitively allow us to preserve more information when they are added or multiplied. We start with a formalization of this intuition by showing that *epimorphisms*, which can be seen to map a semiring $\mathcal{R}_1$ to an at most as informative semiring $\mathcal{R}_2$, can be used to reduce $\mathrm{SAT}(\mathcal{R}_2)$ to $\mathrm{SAT}(\mathcal{R}_1)$.

This makes polynomial semirings such as $\mathbb{N}[(x_i)_\infty]$ very interesting for us since they are, as we will see, the most information preserving countable commutative semirings and can thus always be reduced to. We show that in some cases this strategy is unlikely to work but also prove that there are still large subclasses of semirings where it does.

We start by defining epimorphisms.

Definition 39 (Homomorphism, Epimorphism)**.** *Let $\mathcal{R}_i = (R_i, \oplus_i, \otimes_i, e_{\oplus_i}, e_{\otimes_i}), i = 1, 2$ be two semirings. A* homomorphism *from $\mathcal{R}_1$ to $\mathcal{R}_2$ is a function $f : R_1 \to R_2$ s.t. for $\odot = \oplus, \otimes$*

$$f(r \odot_1 r') = f(r) \odot_2 f(r') \text{ and } f(e_{\odot_1}) = e_{\odot_2}.$$

If f is in addition surjective, then it is an epimorphism*.*

We can use them similarly to reduce problems over one semiring to problems over another semiring. Formally:

Theorem 40. *Let $\tau_i(\mathcal{R}_i), i = 1, 2$ be two encoded commutative semirings, such that*

1. *there exists a polynomial time computable epimorphism $f : \tau_1(R_1) \to \tau_2(R_2)$, and*
2. *for each $\tau_2(r_2) \in \tau(R_2)$ one can compute in polynomial time $\tau_1(r_1)$ s.t. $f(\tau_1(r_1)) = \tau_2(r_2)$ from $\tau_2(r_2)$.*

Then $\mathrm{SAT}(\tau_2(\mathcal{R}_2))$ is counting-reducible to $\mathrm{SAT}(\tau_1(\mathcal{R}_1))$.

Proof (Sketch). Given a $\mathrm{SAT}(\tau_2(\mathcal{R}_2))$-instance α, we can replace all weights $\tau_2(r_2)$ by a weight $\tau_1(r_1)$ such that $f(\tau_1(r_1)) = \tau_2(r_2)$, which is possible in polynomial time by condition 2. We can evaluate the resulting $\mathrm{SAT}(\tau_1(\mathcal{R}_1))$-instance and apply the epimorphism f. □

Theorem 40 formalizes the intuition which Green and Tannen (2017) also discussed: if a semiring $\mathcal{R}_2$ is the homomorphic image of another semiring $\mathcal{R}_1$, then it is at most as

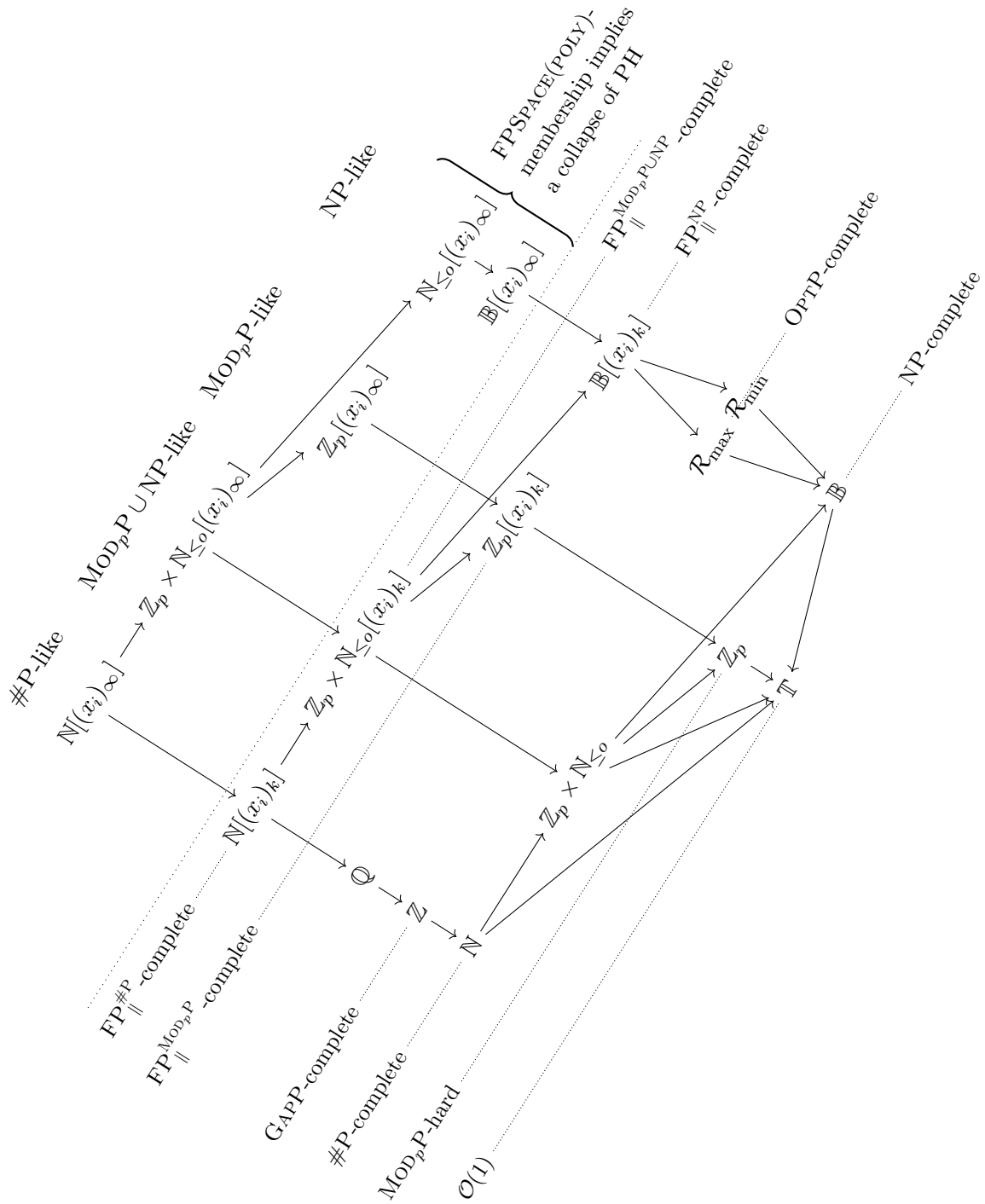

Figure 2: Epimorphisms $f : \mathcal{R}_1 \to \mathcal{R}_2$ between semirings, indicated by arrows $\mathcal{R}_1 \to \mathcal{R}_2$. Relation of complexity classes $\mathcal{C}$ and semirings $\mathcal{R}$, indicated by dotted lines $\mathcal{C}$ ····· $\mathcal{R}$.

information preserving as $\mathcal{R}_1$ and therefore at least as hard structurally. This is because a homomorphism f may assign different values $r \neq r' \in R_1$ the same value $f(r) = f(r') \in R_2$. Then, while we could distinguish these values in the context of $\mathcal{R}_1$, we cannot do the same in $\mathcal{R}_2$. However, clearly it cannot happen that $f(r) \neq f(r') \in R_2$ but $r = r' \in R_1$.

Note that the existence of such an epimorphism does not necessarily mean that the computational complexity of $\mathrm{SAT}(\mathcal{R}_2)$ is lower than $\mathrm{SAT}(\mathcal{R}_1)$ unless the other conditions of Theorem 40 are also satisfied!

Example 15. *The semiring $\mathcal{R}_M$ from Example 11 is a homomorphic image of $\mathbb{B}[(x_i)_\infty]$ using the epimorphism defined by $f(x_i) = \mathrm{bin}(i)$. But $\mathrm{SAT}(\mathbb{B}[(x_i)_\infty])$ is in $\mathrm{FPSpace}(\mathrm{exp})$, while, as noted before already a single multiplication in $\mathcal{R}_M$ may be undecidable. This does not contradict Theorem 40 since we cannot apply it as computing $f(x_i \cdot x_j)$ is not possible in polynomial time when M corresponds to the Halting problem.*

Figure 2 depicts between which semirings we can hope to apply Theorem 40. There, an arrow $\mathcal{R}_1 \to \mathcal{R}_2$ indicates that there exist encoding functions τ_1, τ_2 such that the conditions of Theorem 40 hold. In this map, we see that $\mathbb{N}[(x_i)_\infty]$, $\mathbb{N}_{\leq o} \times \mathbb{Z}_p[(x_i)_\infty]$, $\mathbb{Z}_p[(x_i)_\infty]$ and $\mathbb{N}_{\leq o}[(x_i)_\infty]$ are the most information preserving and therefore in a sense hardest #P-like, $\mathrm{Mod}_p\mathrm{P} \cup$NP-like, Mod_pP-like and NP-like commutative countable semirings, respectively. Among the four classes, we observe that the #P-like problems are the hardest and that, naturally, the $\mathrm{Mod}_p\mathrm{P} \cup$ NP-like problems are harder than both Mod_pP-like and NP-like problems. Furthermore, we see that by restricting ourselves to polynomials with finitely many variables, the complexity decreases (presumably) a bit for all types of problems. Finally, by further epimorphisms, we arrive at semirings that are not over polynomials, which then often correspond to some classical complexity class, as we already observed in Theorem 33.

In the following, we further formalize many intuitions, such as the $\mathcal{C}$-likeness, presented in Figure 2 and present further results on the complexity associated with semirings and classes thereof, often by using the existence of epimorphisms.

We start by formalizing the intuition that the above mentioned polynomial semirings are the hardest ones that satisfy a given set of properties.

Lemma 41 (Hardest Semirings)**.** *There exists an encoding τ^* for*

1. $\mathbb{N}_{\leq o}[(x_i)_\infty]$;
2. $\mathbb{Z}_p[(x_i)_\infty]$;
3. $\mathbb{N}_{\leq o} \times \mathbb{Z}_p[(x_i)_\infty]$;
4. $\mathbb{N}[(x_i)_\infty]$

such that for any commutative efficiently encoded commutative semiring $\tau(\mathcal{R})$ that is in addition

1. *periodic with periodicity* 1;
2. *periodic with periodicity $p \geq 2$ and offset* 0;

3. *periodic with periodicity* $p \geq 2$ *and offset* $o > 0$*;*

4. *not periodic*

it holds that $\mathrm{SAT}(\tau(\mathcal{R}))$ *is counting reducible to*

1. $\mathrm{SAT}(\tau^*(\mathbb{N}_{\leq o}[(x_i)_\infty]))$*;*

2. $\mathrm{SAT}(\tau^*(\mathbb{Z}_p[(x_i)_\infty]))$*;*

3. $\mathrm{SAT}(\tau^*(\mathbb{N}_{\leq o} \times \mathbb{Z}_p[(x_i)_\infty]))$*;*

4. $\mathrm{SAT}(\tau^*(\mathbb{N}[(x_i)_\infty]))$*, respectively.*

Proof (Sketch). The idea is as follows. Instead of performing the evaluation explicitly by performing all the semiring operations on the input values immediately, we replace each semiring value $\tau(r)$ in the given ΣBF α by $x_{\tau(r)}$, where we associate the binary encoding of r with a natural number corresponding to a variable index. Then we evaluate the transformed instance over the corresponding polynomial semiring, where we obtain a polynomial as the result. Finally we use the fact that we can encode the values of the polynomial semiring in such a way that there exists a polynomial time computable epimorphism from the polynomial semiring to $\tau(\mathcal{R})$, which corresponds to evaluating the polynomial with $x_{\tau(r)}$ replaced by $\tau(r)$. □

Thus, if we find an efficient algorithm for $\mathrm{SAT}(\tau^*(\mathcal{S}[(x_i)_\infty]))$, with $\mathcal{S} \in \{\mathbb{N}_{\leq o}, \mathbb{Z}_p, \mathbb{N}_{\leq o} \times \mathbb{Z}_p, \mathbb{N}\}$ with the encoding function τ^* from the proof, we can use it to solve $\mathrm{SAT}(\tau(\mathcal{R}))$ for any efficiently encoded commutative semiring $\tau(\mathcal{R})$ with the corresponding periodicity and offset. Unfortunately, already the result of evaluating a $\mathrm{SAT}(\tau^*(\mathbb{B}[(x_i)_\infty]))$-instance α can be exponential in the size of α. Therefore, any algorithm for $\mathrm{SAT}(\tau^*(\mathcal{S}[(x_i)_\infty]))$ necessarily has exponential running time.

One might be tempted to think that this can be avoided by choosing an encoding function τ that is better than τ^*. However, such an encoding τ is unlikely to exist, as the following result shows.

Theorem 42. *Let* $\mathcal{R} = \mathbb{N}[(x_i)_\infty]$ *(resp.* $\mathcal{R} = \mathbb{B}[(x_i)_\infty]$*). If there is an encoding function* τ *for* $\mathcal{R}$ *s.t.*

1) $\|[\![\alpha]\!]_\mathcal{R}(\emptyset)\|_\tau$ *is polynomial in the size of* α*,*

2) we can extract the binary representation of the coefficient $n \in \mathbb{N}$ *(resp.* $b \in \mathbb{B}$*) of* $x_{i_1}^{j_1} \ldots x_{i_n}^{j_n}$ *from* $\tau(r)$ *in time polynomial in* $\|r\|_\tau$*, and*

3) $\|x_i\|_\tau$ *is polynomial in* i*,*

then $\#\mathrm{P} \subseteq \mathrm{FP/poly}$ *(resp.* $\mathrm{NP} \subseteq \mathrm{P/poly}$*).*

This would imply $\Sigma_2^p = \mathrm{PH}$ (Karp & Lipton, 1982), which is considered to be unlikely. Hence, for any reasonable encoding τ, $\mathrm{SAT}(\tau(\mathbb{N}[(x_i)_\infty])$ and $\mathrm{SAT}(\tau(\mathbb{B}[(x_i)_\infty]))$ are unlikely to have polynomial output and are, therefore, unlikely to be in FPSpace(poly).

Condition 3) of Theorem 42 allows that indices i_k are encoded in unary and imposes no restriction on the encoding of exponents j_k. Requiring the exponents to be encoded in binary puts even already $\mathrm{SAT}(\tau(\mathbb{B}[x]))$ outside of FPSpace(poly), unless NP $\subseteq$ P/poly.

Theorem 43. *Let $\mathcal{R} = \mathbb{N}[x]$ (resp. $\mathcal{R} = \mathbb{B}[x]$). If there is an encoding function τ for $\mathcal{R}$ s.t.*

1) *$\|[\![\alpha]\!]_{\mathcal{R}}(\emptyset)\|_\tau$ is polynomial in the size of α,*
2) *we can extract the binary representation of the coefficient $n \in \mathbb{N}$ (resp. $b \in \mathbb{B}$) of x^i from $\tau(r)$ in time polynomial in $\|r\|_\tau$, and*
3) *$\|x^i\|_\tau$ is polynomial in $\log_2(i)$,*

then #P $\subseteq$ FP/poly *(resp.* NP $\subseteq$ P/poly*).*

Proof (sketch) for Theorems 42 and 43. For each $n \in \mathbb{N}$, we can construct a ΣBF formula α of polynomial size s.t. the solution of every #3SAT (resp. 3SAT)-instance with n variables is obtainable from $\|[\![\alpha]\!]_{\mathcal{R}}(\mathcal{I})\|_\tau$. By the methodology of (Cadoli, Donini, & Schaerf, 1996) to assess compilability, we then infer #P $\subseteq$ FP/poly (resp. NP $\subseteq$ P/poly). □

Notably, the converse of Theorem 43 also holds:

Theorem 44. *If* #P $\subseteq$ FP/poly *(resp.* NP $\subseteq$ P/poly*), then there exist encodings τ_∞ and τ_1 for $\mathbb{N}[(x_i)_\infty]$ and $\mathbb{N}[x]$ (resp. $\mathbb{B}[(x_i)_\infty]$ and $\mathbb{B}[x]$) such that the preconditions 1) - 3) of Theorems 42 and 43 are satisfied.*

Proof (sketch). The idea here is as follows: we know that we can obtain the coefficient of a monomial from $[\![\alpha]\!]_{\mathcal{R}}(\emptyset)$ in #P (resp. NP). Thus, if #P $\subseteq$ FP/poly (resp. NP $\subseteq$ P/poly), we can also obtain it in FP (resp. P) given polynomial advice.

Using this insight, we can construct an encoding that satisfies the desired properties, by representing polynomials as a tuple consisting of a $\mathrm{SAT}(\tau_b(\mathcal{R}))$-instance and the polynomial advice(s) necessary to solve the queries for coefficients of monomials in polynomial time. Here, we only need to make sure that the base encoding τ_b satisfies condition 3). Such encodings are easy to find though. □

Thus, the existence of an encoding τ as in Theorem 43 for $\mathcal{R} = \mathbb{N}[x]$ (resp. $\mathcal{R} = \mathbb{B}[x]$) *equivalent* to the pure complexity-theoretic question whether #P $\subseteq$ FP/poly (resp. NP $\subseteq$ P/poly), which is a well-known open problem in complexity theory.

We can obtain similar results to Theorems 42 and 43 in a more general setting for polynomials with coefficients from any non-trivial commutative semiring.

Theorem 45. *Let $\mathcal{R} \neq \mathbb{T}$ be a commutative semiring. If there is an encoding function τ for $\mathcal{R}[(x_i)_\infty]$ s.t.*

1) *$\|[\![\alpha]\!]_{\mathcal{R}[(x_i)_\infty]}(\emptyset)\|_\tau$ is polynomial in the size of α,*
2) *we can extract the encoding $\tau(r') \in \tau(R)$ of the coefficient of $x_{i_1}^{j_1}...x_{i_n}^{j_n}$ from $\tau(r)$ in time polynomial in $\|r\|_\tau$, and*

3) $\|x_i\|_\tau$ *is polynomial in* i*,*

then either NP $\subseteq$ P/poly *or* $\text{Mod}_p\text{P} \subseteq$ P/poly *for some* $p \in \mathbb{N}$*.*

Theorem 46. *Let* $\mathcal{R} \neq \mathbb{T}$ *be a commutative semiring. If there is an encoding function* τ *for* $\mathcal{R}[x]$ *s.t.*

1) $\|[\![\alpha]\!]_{\mathcal{R}[x]}(\emptyset)\|_\tau$ *is polynomial in the size of* α*,*

2) we can extract the encoding $\tau(r') \in \tau(R)$ *of the coefficient of* x^i *from* $\tau(r)$ *in time polynomial in* $\|r\|_\tau$*, and*

3) $\|x^i\|_\tau$ *is polynomial in* $\log_2(i)$*,*

then either NP $\subseteq$ P/poly *or* $\text{Mod}_p\text{P} \subseteq$ P/poly *for some* $p \in \mathbb{N}$*.*

As already mentioned before, if NP $\subseteq$ P/poly, then the polynomial hierarchy collapses to the second level. As a matter of fact, this also holds if $\text{Mod}_p\text{P} \subseteq$ P/poly since:

Lemma 47. *If* $\text{Mod}_p\text{P} \subseteq$ P/poly *for* $p \in \mathbb{N}, p > 1$*, then* NP $\subseteq$ P/poly*.*

Proof (sketch). Valiant and Vazirani (1986) showed that $\text{NP} \subseteq \text{RP}^{\text{Unique-SAT}}$. Using RP $\subseteq$ BPP we get $\text{RP}^{\text{Unique-SAT}} \subseteq \text{BPP}^{\text{Unique-SAT}}$ and from Unique-SAT $\in \text{Mod}_p\text{P}$ for any $p \in \mathbb{N}, p > 1$ it follows that $\text{NP} \subseteq \text{BPP}^{\text{Mod}_p\text{P}}$.

It remains to show that $\text{BPP}^{\text{Mod}_p\text{P}} \subseteq$ P/poly follows from $\text{Mod}_p\text{P} \subseteq$ P/poly. We already know that BPP $\subseteq$ P/poly (Bennett & Gill, 1981), even without the assumption that $\text{Mod}_p\text{P} \subseteq$ P/poly. We establish that $\text{P}^{\text{P/poly}}/\text{poly} \subseteq$ P/poly and $\text{BPP}^{\text{Mod}_p\text{P}} \subseteq \text{P}^{\text{P/poly}}/\text{poly} \subseteq$ P/poly, which proves the result. □

It follows that if there is any non-trivial commutative semiring $\mathcal{R}$ with an encoding τ as required by Theorems 45 or 46 then NP $\subseteq$ P/poly. This would imply a collapse of the polynomial hierarchy and is therefore considered to be unlikely.

Summing up, we see that reducing $\text{SAT}(\tau(\mathcal{R}))$ to the polynomial semirings is not practical for encoded semirings $\tau(\mathcal{R})$ in general[3]. However, we can obtain positive results when restricting ourselves to polynomials with a limited number of variables by using a different encoding.

Theorem 48. *Let* $\tau(\mathcal{R})$ *be a commutative semiring that is efficiently encoded. Then* $\text{SAT}(\tau(\mathcal{R}[(x_i)_k]))$ *is* $\text{FP}_{\|}^{\text{NP}^{(\tau(\mathcal{R}))}}$*-complete for metric reductions, if we extend* τ *to* $\mathcal{R}[(x_i)_k]$ *by representing polynomials as lists of monomials with exponents in unary and coefficients encoded by* τ*.*

As mentioned before the semirings of polynomials with coefficients in $\tau(\mathcal{R})$ allow one to solve multiple $\text{SAT}(\tau(\mathcal{R}))$-instances at the same time. Theorem 48 shows that when we choose a specific encoding and allow at most k variables then the number of instances that can be solved using one instance over the corresponding semiring $\tau(\mathcal{R}[(x_i)_k])$ is exactly polynomial.

3. Note that Knowledge Compilation (Darwiche & Marquis, 2002) intuitively does something very similar but optimizes the size of the representation of the polynomials such that it can be bounded using the treewidth of a given formula.

Proof (Sketch). We can obtain the coefficient of the polynomially many monomials in the solution of an $\mathrm{SAT}(\tau(\mathcal{R}[(x_i)_k]))$-instance with one independent query to an $\mathrm{NP}(\tau(\mathcal{R}))$-oracle each. On the other hand we can solve n $\mathrm{SAT}(\tau(\mathcal{R}))$-instances at the same time by solving one $\mathrm{SAT}(\tau(\mathcal{R}[(x_i)_k]))$-instance, by multiplying each of them with a different monomial x_1^i. □

As an immediate consequence, we see that $\mathrm{SAT}(\mathrm{bin}(\mathbb{Q}[(x_i)_k]))$ and $\mathrm{SAT}(\mathrm{bin}(\mathbb{B}[(x_i)_k]))$ are metrically reducible to $\mathrm{FP}_{\|}^{\#\mathrm{P}}$ (which is equal to $\mathrm{FP}^{\#\mathrm{P}[1]}$) and $\mathrm{FP}_{\|}^{\mathrm{NP}}$, respectively, and thus not significantly harder than #P and NP, respectively.

The same can be said when a $\mathrm{SAT}(\tau(\mathcal{R}))$-instance only contains few values from the semiring.

Lemma 49. *Let $k \in \mathbb{N}$ be fixed, $\tau(\mathcal{R})$ be an efficiently encoded commutative semiring. If $\tau(\mathcal{R})$ is*

1. *periodic with periodicity 1;*
2. *periodic with periodicity $p \geq 2$ and offset 0;*
3. *periodic with periodicity $p \geq 2$ and offset $o > 0$;*
4. *not periodic*

then $\mathrm{SAT}(\tau(\mathcal{R}))$ for instances that contain at most k different semiring values is in

1. $\mathrm{FP}_{\|}^{\mathrm{NP}}$;
2. $\mathrm{FP}_{\|}^{\mathrm{Mod}_p\mathrm{P}}$;
3. $\mathrm{FP}_{\|}^{\mathrm{Mod}_p\mathrm{P}\cup\mathrm{NP}}$;
4. $\mathrm{FP}_{\|}^{\#\mathrm{P}}$

respectively.

Proof. Let α be a $\mathrm{SAT}(\tau(\mathcal{R}))$-instance that contains at most k different semiring values $\tau(r_1), \ldots, \tau(r_k)$. We define an epimorphism $f : \mathrm{bin}(\mathbb{N}[(x_i)_k]) \to \tau(\mathcal{R})$ by defining it on the monomials $x_1^{i_1} \ldots x_k^{i_k}$ as $f(x_1^{i_1} \ldots x_k^{i_k}) = \bigotimes_{j=1}^{k} \bigotimes_{l_j=1}^{i_j} \tau(r_j)$. Since $\tau(\mathcal{R})$ is efficiently encoded, and the exponents i_j are encoded in unary, f is computable in polynomial time. We can thus apply Theorem 40 with semirings $\mathrm{bin}(\mathbb{N}[(x_i)_k])$ and $\tau(\mathcal{R})$, since the second condition is trivially satisfied by using x_j as an element that maps to $\tau(r_j)$ for $j = 1, \ldots, k$.

Thus, we have a counting reduction from $\mathrm{SAT}(\tau(\mathcal{R}))$ to $\mathrm{SAT}(\mathrm{bin}(\mathbb{N}[(x_i)_k]))$. By applying Theorem 48 we then obtain a metric reduction to a problem in $\mathrm{FP}_{\|}^{\#\mathrm{P}}$. Since metric reductions are polynomial time functions, we can derive that $\mathrm{SAT}(\tau(\mathcal{R}))$ is in $\mathrm{FP}^{\mathrm{FP}_{\|}^{\#\mathrm{P}}[1]}$, which is equal to $\mathrm{FP}_{\|}^{\#\mathrm{P}}$.

The proof for the semirings that are periodic is analogous by performing the same reduction to

1. $\mathrm{SAT}(\mathrm{bin}(\mathbb{N}_{\leq o}[(x_i)_k]))$, when $\tau(\mathcal{R})$ has periodicity 1 and offset o;

2. $\mathrm{SAT}(\mathrm{bin}(\mathbb{Z}_p[(x_i)_k]))$, when $\tau(\mathcal{R})$ has periodicity $p \geq 2$ and offset 0;
3. $\mathrm{SAT}(\mathrm{bin}(\mathbb{Z}_p \times \mathbb{N}_{\leq o}[(x_i)_k]))$, when $\tau(\mathcal{R})$ has periodicity $p \geq 2$ and offset $o > 0$.

□

However, not all relevant $\mathrm{SAT}(\mathcal{R})$-instances have few different semiring values. We can take this idea even further by considering instances that have many different semiring values that can be expressed as "reasonably small" sums and products of few different semiring values. For this, we consider finitely generated semirings.

Definition 50 (Finitely Generated Semiring)**.** *Let $\mathcal{R} = (R, \oplus, \otimes, e_\oplus, e_\otimes)$ be a semiring. We call $\mathcal{R}$* finitely generated*, if $\mathcal{R} = \langle R^\star \rangle_\mathcal{R}$ for some finite $R^\star \subseteq R$.*

Commutative semirings that are finitely generated using k elements $r_1, \dots r_k$ can be seen as reduced versions of the semiring of polynomials with variables $x_1, \dots, x_k$. We thus obtain:

Theorem 51. *Let $\tau(\mathcal{R})$ be an efficiently encoded commutative semiring that is generated by $\{r_1, \dots, r_k\}$. Suppose every $r \in R$ is of the form $r = \bigoplus_{i=1}^{n} a_i \cdot \bigotimes_{j=1}^{k} r_j^{e_{i,j}}$ for some $a_i, e_{i,j} \in \mathbb{N}$ such that*

- *$\max\{e_{i,j}, \log_2(a_i), n\}$ is polynomial in $\|r\|_\tau$, and*
- *we can obtain $a_i, e_{i,j}$ from $\tau(r)$ in polynomial time.*

If $\tau(\mathcal{R})$ is

1. *periodic with periodicity 1;*
2. *periodic with periodicity $p \geq 2$ and offset 0;*
3. *periodic with periodicity $p \geq 2$ and offset $o > 0$;*
4. *not periodic;*

then $\mathrm{SAT}(\tau(\mathcal{R}))$ is in

1. $\mathrm{FP}_{\|}^{\mathrm{NP}}$;
2. $\mathrm{FP}_{\|}^{\mathrm{Mod}_p\mathrm{P}}$;
3. $\mathrm{FP}_{\|}^{\mathrm{Mod}_p\mathrm{P} \cup \mathrm{NP}}$;
4. $\mathrm{FP}_{\|}^{\#\mathrm{P}}$;

respectively.

Note that k in Theorem 51 may be zero. For example, the natural number semiring $\mathbb{N}$ is generated by the empty set, since every semiring over a subset over the natural numbers with the same addition and multiplication needs to contain zero and one. Clearly, $\mathbb{N} = \langle 0, 1 \rangle_\mathbb{N}$. If indeed $k = 0$, the product $\bigotimes_{j=1}^{k} r_j^{e_{i,j}}$ takes values $e_\otimes$ and the Theorem also holds.

Proof (Sketch). As in the proof of Lemma 49, we can find a polynomial time epimorphism from $\tau(\mathcal{R})$ to $\mathrm{bin}(\mathcal{S}[(x_i)_k])$ for the semiring $\mathcal{S} \in \{\mathbb{N}_{\leq o}, \mathbb{Z}_p, \mathbb{Z}_p \times \mathbb{N}_{\leq o}, \mathbb{N}\}$ with the same periodicity as $\tau(\mathcal{R})$. However, contrary to Lemma 49, we do not identify each semiring value in a given instance with a different variable but identify the values $r_1, \ldots, r_k$ that the semiring is generated from with the variables $x_1, \ldots, x_k$. Due to the preconditions of the Theorem, we can then apply Theorem 40 to show that we can counting-reduce $\mathrm{SAT}(\tau(\mathcal{R}))$ to $\mathrm{SAT}(\mathrm{bin}(\mathcal{S}[(x_i)_k])$. By application of Theorem 48 we then prove our claim.

Again note that in case $k = 0$, the polynomial semiring $\mathrm{bin}(\mathcal{S}[(x_i)_k])$ is simply the semiring over the constant polynomials, i.e., $\mathrm{bin}(\mathcal{S})$. □

Note that when $\tau(\mathcal{R})$ is periodic, we know that the coefficients a_i can always be chosen such that they are bounded by a constant.

Furthermore, compare Theorem 51 to Theorem 38 for the case when the encoded semiring $\tau(\mathcal{R})$ is not periodic. Clearly, for any semiring $\tau(\langle e_\otimes \rangle)$ is finitely generated. Thus, one might be tempted to assume that one can always apply Theorem 51 to derive that $\mathrm{SAT}(\tau(\langle e_\otimes \rangle))$ is in $\mathrm{FP}_{\|}^{\#\mathrm{P}}$. This is not necessarily the case, however, since it is not clear that the additional preconditions of Theorem 51 are given, even when the semiring is efficiently encoded.

7. Discussion and Related Work

In this section, we consider related models of computation with algebraic structures, describe some further applications of our results and provide more discussion on the effects of encodings.

7.1 Other Machine Models

In order to capture the complexity of quantitative reasoning frameworks that are defined over semirings $\mathcal{R}$, we introduced Semiring Turing Machines. SRTMs are non-deterministic Turing machines whose result is determined by their transitions that are weighted with semiring values; $\mathrm{SAT}(\mathcal{R})$ and $\mathrm{NP}(\mathcal{R})$, the versions of SAT and NP that are generalized to semirings are in the expected relation, viz. that $\mathrm{SAT}(\mathcal{R})$ is $\mathrm{NP}(\mathcal{R})$-complete with respect to (adjusted) Karp-reductions.

In the literature, other extensions of machine models with means to deal with algebraic structures have been considered.

Weighted Automata The machine model most related to our SRTMs is that of weighted automata, which are finite state automata that have a (semiring)-weighted transition function (Droste & Gastin, 2007). SRTMs are a generalization of weighted automata and can simulate the latter with ease; informally, they equip weighted automata with memory and the full computational power of a Turing machine under the restriction that semiring values are opaque. The connection to weighted automata is especially of interest due to their connection to descriptive complexity, which opens up future work on the same topic for SRTMs.

Blum-Shub-Smale Machines Next, our SRTMs are related to Blum-Shub-Smale (BSS) machines, which were introduced by Blum et al. (1989, 1998) to model algebraic computa-

tion over the reals and general rings, respectively. Roughly speaking, BSS machines operate over a tape whose cells can hold elements from a ring R, and computation steps are performed by the evaluation of polynomials over R that modify finite sequences of cells; a BSS machine is a connected directed graph which has input, output, and computation nodes, and possibly shift and branching nodes, where the former allow for shifting tape contents and the latter for testing a condition such as $r \geq 0$ (if the ring R is ordered) or $r = 0$ based on the ring elements on the tape. Notably, Blum et al. remarked that a classical Turing machine is a BSS-machine over $\mathbb{Z}_2$ (using a binary alphabet).

As for complexity, BSS studied in their model ring operations having unit cost, i.e., independent of the concrete ring elements used; merely for $\mathbb{Z}$ and $\mathbb{Q}$, also bit cost, i.e. $\log m$ where m is the largest number in the computation, was considered. Based on this time measure, Blum et al. defined then polynomial (P-) time and NP computations over rings, where in the latter a guess of values from R can be made; notably, over $\mathbb{Z}$ we have $\mathrm{P} \neq \mathrm{NP}$ under unit cost.

Compared to SRTMs, BSS-machines do not operate over semirings, thus e.g. the natural numbers $\mathbb{N}$ are not covered, as they do not form a ring. While the definition might be adapted to semirings, a distinctive feature is that BSS-machines work over (strings of) ring values only, while our SRTMs allow for semiring values in addition to a tape alphabet as usual. The latter is convenient and provides flexibility for problem representation, which the BSS model lacks. E.g., for solving the classical SAT problem in the BSS model, propositional formulas must be first converted into an algebraic form over a ring that can be processed by a BSS machine; to this end, Blum et al. use a transformation (assuming that input formulas are already in CNF) into a system of polynomials over the ring $\mathbb{Z}_2$.

Furthermore, in order to ensure efficient (polynomial time) computation, the processing of numbers must be controlled with additional restrictions, such that e.g. recursive multiplication of values, as discussed in Section 4.2, is prohibited.

Summarizing, the BSS model has been devised with computations over algebraic structures in mind, where the input is already in a specific form and the cost of algebraic operations is disregarded respectively for $\mathbb{Z}$ and $\mathbb{Q}$ estimated by binary number size; semirings like $\mathbb{N}$ are not covered. Efficient computation is hard to ensure, and the linkage to conventional complexity classes is limited to few cases; different by missing or fixed encodings. This makes the use of BSS machines for problem solving and complexity analysis in the context of AI reasoning frameworks as we consider less attractive.

Real Counting Computations Major motivations of Blum et al. to develop their machine model were to have an idealized computation model for scientific computation with real numbers, in which the cost of multiplication is independent of the size of numbers (which on a Turing machine depends on the encoding), and to bring the theory of computation into the domain of analysis, geometry and topology, such that the mathematics of these subjects can then be put to use in the systematic analysis of algorithms (Blum et al., 1989). Building on the BSS machine model, Meer (2000) developed a theory of counting problems over the real numbers. To this end, he introduced the class $\#\mathrm{P}_{\mathbb{R}}$ as the real analogue of #P, which also counts accepting paths but of a non-deterministic BSS machine, and then embarked on giving a logical characterization of $\#\mathrm{P}_{\mathbb{R}}$ following descriptive complexity theory over the reals, which has been initiated in (Grädel & Meer, 1995). Meer's class $\#\mathrm{P}_{\mathbb{R}}$

is different from our NP($\mathcal{R}$) instantiated for $\mathcal{R} = \mathbb{R}$, in which we sum up (i.e., "count") *weighted* acceptance paths. However, for SRTMs the numbers involved in a computation are deliberately restricted compared to BSS machines, in order not to produce large result values (which does not matter in the BSS-model) such that space efficient storage on a conventional machine is feasible. Detailing the relationship between NP($\mathbb{R}$) and an analogue class that combines weighted acceptance counting with BSS machines is an interesting issue for future research, especially when it comes to generalizations for other (semi)rings.

Real Counting with Arbitrarily Approximatable Weights A different definition of $\#\mathrm{P}_{\mathbb{R}}$ and $\#\mathrm{P}_S$ for $S \subseteq \mathbb{R}$ was provided by De Campos, Stamoulis, and Weyland (2020). While they intuitively also define the output of a computation as the sum over weighted computation paths like we do, they do not assume the values to be given explicitly but require functions that compute them up to a variable precision of 2^{-p} in polynomial time in p. This allows them to work with a finite representation of real numbers. Interestingly, this different way of encoding numbers leads to undecidability of the question whether a computation in $\#\mathrm{P}_{\mathbb{Q}}$ leads to a value greater than zero. Therefore, we argue that while this matter of encoding numbers is interesting to allow computations with uncountable semirings such as $\mathbb{R}$ on classical Turing machines, the explicit encoding in our setting is of more interest when the semiring has only countably many values, as it leads to more intuitive results.

Relational Machines SRTMs are further related to the relational machine of Abiteboul and Vianu (1991, 1995), which was introduced in order to compensate for a mismatch between computations by a Turing machine and the evaluation of logical formulas over finite relational structures, which is of central importance for query answering over relational databases. The generic treatment of elements in such structures, which model relational databases, by logic and the fact that relations are unordered sets of tuples is in contrast to string-based representations where in an encoding elements might be distinguished and by the inherent order of the string's letter an order of tuples induced; the latter may make it easy to answer certain queries (e.g., whether a database has an even number of records) that can not be expressed in a query language without order information (e.g., pure SQL). To this end, Abiteboul and Vianu extended Turing machines with a "relational storage" and the ability to perform relational operations (which underlie the evaluation of logical formulas) directly on the relational storage rather than on the encoding of a database on a machine's tape. Emerging relational complexity classes and their relationships allowed for a precise characterization of the expressive powers of certain database query languages and to translate relationships of the latter into equivalent questions in standard complexity theory. Notably, our notion of Karp surrogate-reduction for problems over semirings aims at a genericity condition similar to the one for databases, but it does not allow for exploiting in the reduction that different semiring surrogates SV^nE and SV^mE (i.e., $n \neq m$), which corresponds to having different elements in a database, will amount to different semiring values.

7.2 Applying the Results for Problem Analysis

Just like the quantitative frameworks that are defined over semirings $\mathcal{R}$ capture many different quantitative problems when instantiated with the correct semiring, we showed that the same is the case on the complexity level. Theorem 33 showed that for $(\mathcal{R}, \mathcal{C}) = (\mathbb{B}, \mathrm{NP})$, $(\mathbb{N}, \#\mathrm{P})$, $(\mathbb{Z}, \mathrm{GapP})$, $(\mathcal{R}_{\max}, \mathrm{OptP})$ and the binary representation bin of the integers, $\mathrm{SAT}(\mathrm{bin}(\mathcal{R}))$ is $\mathcal{C}$-complete w.r.t. Karp reductions. Thus, many well-known complexity classes can be seen as instantiations of $\mathrm{NP}(\mathcal{R})$ with the appropriate semiring under the usual representation of numbers.

Using $\mathrm{NP}(\mathcal{R})$, we are able to characterize the complexity of many semiring frameworks. As our results show

- Sum-Of-Products Problems (Bacchus et al., 2009),
- Weighted First-Order Logic Evaluation (Mandrali & Rahonis, 2015; Eiter & Kiesel, 2020),
- Semiring-based Constrained Satisfaction Problems (Bistarelli et al., 1999),
- Algebraic Model Counting (Kimmig et al., 2017), and
- Semiring-induced Propositional Logic (Larrosa et al., 2010)

are all $\mathrm{NP}(\mathcal{R})$-complete with respect to Karp-reductions. Furthermore, we saw that Semiring Provenance (Green et al., 2007) for Datalog is $\mathrm{NP}(\mathcal{R})$-hard but not complete and therefore even harder than other semiring formalisms.

As showcases, we consider some specific semirings to illustrate how our previous results can be used to analyze the complexity of further specific semirings.

As a consequence of Theorems 40 and 48, or directly from Theorem 51 for the finitely generated $\mathbb{N}, \mathbb{Z}$, we obtain:

Theorem 52. *Let $\mathbb{S} = \mathbb{N}, \mathbb{Z}, \mathbb{Q}$. For $\mathbb{S}^n$, $n \in \mathbb{N}$, the semiring $\mathbb{S}$ over multiple dimensions, we have that $\mathrm{SAT}(\mathrm{bin}(\mathbb{S})^n)$ is $\mathrm{FP}_{\parallel}^{\#\mathrm{P}}$-complete with respect to metric reductions, where $\mathrm{bin}(\mathbb{Q})$ represents $r \in \mathbb{Q}$ as pair $(\mathrm{bin}(p), \mathrm{bin}(q))$ such that $p/q = r$, $p \in \mathbb{Z}$, $q \in \mathbb{N}$ and the greatest common divisor of $|p|$ and q is 1.*

As promised in the introduction, we consider in more detail the semiring

$$\mathrm{GRAD} = (\mathbb{Q}_{\geq 0} \times \mathbb{Q}, +, \otimes, (0,0), (1,0)),$$

where addition is coordinate-wise and $(a_1, b_1) \otimes (a_2, b_2) = (a_1 \cdot a_2, a_2 \cdot b_1 + a_1 \cdot b_2)$. It was introduced by (Eisner, 2002) and shown to be useful for parameter optimization (Kimmig et al., 2017; Manhaeve et al., 2019). Intuitively, we can compute gradient values with GRAD and for (a, b) the value a to the computed result of some function and b corresponds to its gradient with respect to some parameter.

There is an epimorphism f from $\mathbb{Q}[x]$ to GRAD, which can be seen by defining f on the monomials qx^i as

$$f(qx^i) = \left\{ \begin{array}{cc} q(0,1)^i & \text{if } i > 0, \\ |q| & \text{otherwise.} \end{array} \right\} = \left\{ \begin{array}{cc} (0,q) & \text{if } i = 1, \\ (0,0) & \text{if } i > 1, \\ (|q|,0) & \text{otherwise.} \end{array} \right.$$

This means that we can see the elements in GRAD as elements in $\mathbb{Q}[x]$ by identifying $(1,0)$ and $(0,1)$ with 1 and x, respectively. The elements in GRAD are however only reduced versions of the polynomials, i.e., there are additional equalities between values in GRAD that do not hold in $\mathbb{Q}[x]$. An example is x^2, because $(0,1) \otimes (0,1) = (0,0)$ but $x^2 \neq 0$.

Theorem 53. SAT(GRAD) *is* $\mathrm{FP}_{\parallel}^{\#\mathrm{P}}$*-complete with respect to metric reductions.*

Here, the membership follows from Theorems 48 and 51 while the hardness follows from the $\mathrm{FP}_{\parallel}^{\#\mathrm{P}}$-hardness of bin($\mathbb{N}$) and the fact that $f : \mathrm{GRAD}_{\mathbb{N}} \to \mathbb{N}, (a,b) \mapsto a$, where $\mathrm{GRAD}_{\mathbb{N}}$ is the semiring GRAD restricted to elements from $\mathbb{N} \times \mathbb{N}$, is an epimorphism. The latter implies that we can use Theorem 40 to prove the hardness part.

7.3 Effects of the Encoding

By considering the complexity of $\mathrm{SAT}(\tau(\mathcal{R}))$ for encoded semirings $\tau(\mathcal{R})$ in terms of classical complexity classes, we are able to gain a broad overview of how the complexity depends on the semiring. The main intuition that we gain is that the more information semiring values can preserve in addition and multiplication the harder it is to evaluate $\mathrm{SAT}(\tau(\mathcal{R}))$, which we formalized using epimorphisms in Theorem 40. In the simplest case, where we consider $\mathrm{SAT}(\tau(\langle e_\otimes \rangle))$, instances only have weights of the form $\tau(k \cdot e_\otimes)$ and can thus only retain very limited information. Here, there are exactly four different types of non-trivial problems that can occur. Namely, NP-, Mod_pP-, $\mathrm{Mod}_p\mathrm{P} \cup \mathrm{NP}$- or #P-like problems, which are not only similarly hard as the corresponding complexity class but can also be solved (under some additional restrictions on the encoding function τ) with a corresponding oracle.

Interestingly, the separation by problem type is already determined by the periodicity and offset of the semiring and also applies to the case where general weights are allowed, if the semiring is efficiently encoded. Namely, there exists an encoding τ^* such that all NP-, Mod_pP-, $\mathrm{Mod}_p\mathrm{P} \cup \mathrm{NP}$- and #P-like problems can, respectively, be counting reduced to $\mathrm{SAT}(\tau^*(\mathcal{R}[(x_i)_\infty]))$ for $\mathcal{R} \in \{\mathbb{N}_{\leq o}, \mathbb{Z}_p, \mathbb{N}_{\leq o} \times \mathbb{Z}_p, \mathbb{N}$. This means that these semirings are in a sense the hardest ones in this class of problems. In terms of information, the values of these polynomial semirings intuitively carry the results of multiple $\mathcal{C}$-queries, by handling one $\mathcal{C}$-query per monomial, where $\mathcal{C}$ is the corresponding complexity class of the problem type.

Unfortunately, this is likely too much information to store in a "reasonable" polynomial size output. For each $n \in \mathbb{N}$ we are able to construct a $\mathrm{SAT}(\tau(\langle e_\otimes \rangle_{\mathcal{R}[(x_i)_\infty]})$-instance of size polynomial in n, whose result contains as information the solution of every $\mathrm{SAT}(\tau(\langle e_\otimes \rangle_{\mathcal{R}}))$-instance without weights of size at most n. Since there are exponentially many such instances, this would mean we would need to compute in some way the solution of exponentially many $\mathcal{C}$-instances. Using this insight we showed that the existence of a "reasonable" encoding τ such that $\mathrm{SAT}(\tau(\mathcal{R}[(x_i)_\infty]))$ is in $\mathrm{FPSpace}(\mathrm{poly})$ would imply a collapse of the polynomial hierarchy and is, thus, considered to be unlikely.

While this strategy of reducing all $\mathcal{C}$-like problems to the corresponding polynomial semiring $\tau(\mathcal{R}[(x_i)_\infty])$ does not seem recommendable, we see that with appropriate encodings τ the semiring $\tau(\mathcal{R}[(x_i)_k])$ of the polynomials with at most k variables only allows us to retrieve information about polynomially many calls to a $\mathcal{C}$-oracle. We can exploit this result

to show that all $\mathcal{C}$-like problems with few weights from the semiring as well as all $\mathcal{C}$-like problems over a finitely generated semiring with a suitable encoding can be solved in $\mathrm{FP}_{\|}^{\mathcal{C}}$.

Summing up, our results can be interpreted to give the following intuition: Putting aside the difficulties caused by the encoding τ, the non-deterministic evaluation problems over non-trivial semirings $\mathcal{R}$ are generally hard, as they are strongly intertwined with one of NP, $\mathrm{Mod}_p\mathrm{P}$, $\mathrm{Mod}_p\mathrm{P} \cup \mathrm{NP}$ and #P based on the periodicity and the offset of $\mathcal{R}$. Within the class of $\mathcal{C}$-like problems P the number of queries to an $\mathcal{C}$-oracle, whose results can be obtained from the solution of P, in relation to the size of the P-instance, seems to be crucial. When it is polynomial an $\mathrm{FP}_{\|}^{\mathcal{C}}$-computation is sufficient, when it is exponential, an $\mathrm{FPSpace(exp)}$-computation seems to be necessary.

8. Summary and Outlook

In this paper, we have presented a model of computation that equips Turing machines with weighted transitions using values from a semiring, and we have defined with $\mathrm{NP}(\mathcal{R})$ an analogue of NP over generic semirings $\mathcal{R}$; specific instances can be used to characterize the complexity of a number of problems in quantitative reasoning frameworks in AI. Furthermore, we have linked the novel semiring complexity classes to standard complexity classes, by looking into the encoding of a semiring for conventional, string-based computing.

Our investigation of semiring complexity motivated by AI problems is by no means exhaustive. On the contrary, there are a number of further research issues that emerge from it that remain for future work.

Other Classes of Semirings In this paper, we have focused on commutative semirings and restrictions on the addition of the semiring. Dropping commutativity should lead to problems that are at least as hard as in presence of commutativity according to Theorem 40, since we can find epimorphisms from non-commutative to commutative semirings but in general not the other way around. While some of our results do not make use of commutativity, others such as Theorem 51 rely on it. Furthermore, periodicity and offset are properties of the addition of the semiring. If we impose additional conditions on multiplication, we may be able to exploit them for better upper bounds on the complexity of the evaluation problem.

BSS Machines and SRTMs BBS machines and SRTMs both provide a way to perform computations over algebraic structures but SRTMs are more restricted. A characterization of when the machine models differ in power depending on which additional restrictions are put on the BSS machines will help us better understand the effect of limitations on the computational model. Especially, a closer look at the difference in power attained by allowing SRTMs to perform recursive computations with semiring values on the tape is of interest. While we have seen that over natural numbers $\mathbb{N}$ this leads to strictly more powerful machines –as we can compute a value that is double exponential in the size of the input– this is not clear for other semirings in general. For example, over the Boolean semiring $\mathbb{B}$ recursive reuse of semiring values in computations does not increase the hardness since CircuitSAT is Karp-reducible to SAT and vice versa. This begs the question of which properties a semiring has to satisfy such that recursive semiring computations are harder than non-recursive ones.

Semiring Counting Hierarchy Another issue of computational models over algebraic structures that would deserve investigation is an analogue of the polynomial hierarchy for general semirings, going beyond the one for counting problems over the natural numbers as in the work of Allender and Wagner (1993). For example, the problem MajMajSAT requires the solution of a counting problem over the natural numbers $\mathbb{N}$ on the second level, cf. (Oztok, Choi, & Darwiche, 2016). Further problems such as computing the maximum expected utility of a logical theory, probabilistic reasoning over stable models, Maximum A Posteriori (MAP) inference and some decision theoretic problems are also intrinsically second-level problems (Kiesel, Totis, & Kimmig, 2022) but use different semirings at the different levels. The FAQ problem (Khamis, Ngo, & Rudra, 2016) lifts this to problems at an arbitrary level, by allowing for sequences of semiring aggregates. It would thus be interesting to enhance the power of SRTMs such that we can solve and characterize the complexity of such multi-level semiring reasoning frameworks. A promising starting point for this direction of research is the approach of Ladner (1989) who introduced a counting version of the polynomial hierarchy based on alternating Turing Machines.

Descriptive Complexity Apart from the computational complexity of logics over semirings, there is also the question of descriptive complexity. Here, Droste and Gastin (2007) showed that a fragment of weighted monadic second order logic was shown to capture the sentences recognizable by weighted automata over semirings. This result is a generalization of the Büchi, Elgot-Trakhtenbrot Theorem (Büchi, 1960), which states the same in the unweighted (i.e. Boolean) case. This suggests that we may be able to do the same for Fagin's seminal theorem (Fagin, 1974) in descriptive complexity, which states that the existential fragment of second-order logic characterizes the complexity class NP. Recently, second-order logics over specific semirings were already used to capture the complexity of quantitative complexity classes such as FP and #P (Arenas, Muñoz, & Riveros, 2020). A general theorem that relates $\mathrm{NP}(\mathcal{R})$ and second-order logic over semirings $\mathcal{R}$ would be suggestive. Given that SRTMs generalize weighted automata with their semiring weighted transition functions, our work provides a basis for a generalized version of Fagin's Theorem over arbitrary semirings. Such a result and others on descriptive complexity would be valuable to assess the expressiveness of AI reasoning frameworks over varying input data when formulas respectively queries are fixed.

Polynomial Semirings for Knowledge Compilation Another interesting avenue of research is the application of polynomial semirings in the context of Knowledge Compilation. Intuitively, many contemporary approaches in that area compile a circuit that represents a value from $\mathbb{B}[(x_i)_\infty]$ as succinctly as possible without losing information, cf. (Darwiche & Marquis, 2002; Kimmig et al., 2017; Dudek, Phan, & Vardi, 2020). In an application setting such as probabilistic reasoning or most probable explanation inference, this however may be way more information than necessary. A way to mitigate this problem may be given by semirings $\mathcal{R}[(x_i)_k]$ with finitely many variables, which our results suggest may allow for representations that are lot smaller. This would not only be of theoretical benefit but may even lead to more efficient inference in practical applications.

Acknowledgments

This work has been supported by FWF project W1255-N23. We thank the reviewers for their feedback and comments to improve this article.

A preliminary version of this work has been presented at AAAI 2021 (Eiter & Kiesel, 2021). This article contains full details as well as refined and extended results.

Appendix A. Prefix Normal Form ↪ Lemma 11

Lemma 11 (Prefix Normal Form)**.** *For every ΣBF α over a commutative semiring $\mathcal{R}$ there exists a ΣBF β over $\mathcal{R}$ such that*

(i) $\beta = \Sigma v_1 \dots \Sigma v_n \gamma$*, where γ is quantifier free,*

(ii) β *can be constructed from* α *in polynomial time, and*

(iii) $[\![\alpha]\!]_{\mathcal{R}}(\emptyset) = [\![\beta]\!]_{\mathcal{R}}(\emptyset)$*, i.e.,* α *and* β *evaluate to the same value.*

Proof. Let α be a ΣBF over semiring $\mathcal{R}$. We show that we can always push out the sum quantifiers.
Let $\alpha = (\Sigma v \alpha_1) * \alpha_2$. W.l.o.g. we can assume that α_2 does not contain v. Then

$$\begin{aligned}[\![\alpha]\!]_{\mathcal{R}}(\emptyset) &= [\![(\Sigma v \alpha_1) * \alpha_2]\!]_{\mathcal{R}}(\emptyset) \\ &= [\![\Sigma v \alpha_1]\!]_{\mathcal{R}}(\emptyset) \otimes [\![\alpha_2]\!]_{\mathcal{R}}(\emptyset) \\ &= ([\![\alpha_1]\!]_{\mathcal{R}}(\{v\}) \oplus [\![\alpha]\!]_{\mathcal{R}}(\{\neg v\})) \otimes [\![\alpha_2]\!]_{\mathcal{R}}(\emptyset)\end{aligned}$$

Since addition distributes over multiplication and α_2 does not contain v, we get

$$\begin{aligned}[\![\alpha]\!]_{\mathcal{R}}(\emptyset) &= [\![\alpha_1]\!]_{\mathcal{R}}(\{v\}) \otimes [\![\alpha_2]\!]_{\mathcal{R}}(\emptyset) \oplus [\![\alpha_1]\!]_{\mathcal{R}}(\{\neg v\}) \otimes [\![\alpha_2]\!]_{\mathcal{R}}(\emptyset) \\ &= [\![\alpha_1]\!]_{\mathcal{R}}(\{v\}) \otimes [\![\alpha_2]\!]_{\mathcal{R}}(\{v\}) \oplus [\![\alpha_1]\!]_{\mathcal{R}}(\{\neg v\}) \otimes [\![\alpha_2]\!]_{\mathcal{R}}(\{\neg v\}) \\ &= [\![\alpha_1 * \alpha_2]\!]_{\mathcal{R}}(\{v\}) \oplus [\![\alpha_1 * \alpha_2]\!]_{\mathcal{R}}(\{\neg v\}) \\ &= [\![\Sigma v(\alpha_1 * \alpha_2)]\!]_{\mathcal{R}}(\emptyset)\end{aligned}$$

So we can choose $\beta = \Sigma v(\alpha_1 * \alpha_2)$.
Let $\alpha = \alpha_1 * (\Sigma v \alpha_2)$. This case works analogously to the previous one.
Let $\alpha = (\Sigma v \alpha_1) + \alpha_2$. Then $\beta = \Sigma v(\alpha_1 + \alpha_2 * v)$ has the following semantics w.r.t $\emptyset$

$$\begin{aligned}[\![\beta]\!]_{\mathcal{R}}(\emptyset) &= [\![\Sigma v(\alpha_1 + \alpha_2 * v)]\!]_{\mathcal{R}}(\emptyset) \\ &= [\![\alpha_1 + \alpha_2 * v]\!]_{\mathcal{R}}(\{v\}) \oplus [\![\alpha_1 + \alpha_2 * v]\!]_{\mathcal{R}}(\{\neg v\}) \\ &= [\![\alpha_1 + \alpha_2]\!]_{\mathcal{R}}(\{v\}) \oplus [\![\alpha_1]\!]_{\mathcal{R}}(\{\neg v\}) \\ &= [\![\alpha_1]\!]_{\mathcal{R}}(\{v\}) \oplus [\![\alpha_1]\!]_{\mathcal{R}}(\{\neg v\}) \oplus [\![\alpha_2]\!]_{\mathcal{R}}(\{v\})\end{aligned}$$

Since w.l.o.g. v does not occur in α_2 we have

$$[\![\beta]\!]_{\mathcal{R}}(\emptyset) = [\![\alpha_1]\!]_{\mathcal{R}}(\{v\}) \oplus [\![\alpha_1]\!]_{\mathcal{R}}(\{\neg v\}) \oplus [\![\alpha_2]\!]_{\mathcal{R}}(\{v\})$$

$$= [\![\Sigma v \alpha_1]\!]_{\mathcal{R}}(\emptyset) \oplus [\![\alpha_2]\!]_{\mathcal{R}}(\emptyset)$$
$$= [\![(\Sigma v \alpha_1) + \alpha_2]\!]_{\mathcal{R}}(\emptyset)$$

So we can choose $\beta = \Sigma v(\alpha_1 + \alpha_2 * v)$.
The case $\alpha = \alpha_1 + \Sigma x \alpha_2$ works analogously.
This shows that we can always push the Σ quantifier to the top in polynomially many steps in the number of occurrences of $*, +$ in α. Furthermore, as we only add $*v$ to the formula at most once for every combination of Σ and $+$ in α, the size of β is also polynomial in α. □

Appendix B. NP($\mathcal{R}$)-completeness and Karp Reducibility

We prove NP($\mathcal{R}$)-completeness of SAT($\mathcal{R}$) and show that SAT($\mathcal{R}$), SumProd($\mathcal{R}$), AMC, SCSP and ΣFO-Eval($\mathcal{R}$) are Karp-reducible to one another.

B.1 NP($\mathcal{R}$)-completeness of SAT($\mathcal{R}$) ↪ Theorem 18

Theorem 18. SAT($\mathcal{R}$) *is* NP*($\mathcal{R}$)-complete w.r.t. Karp s-reductions for every commutative semiring* $\mathcal{R}$.

Proof. Containment follows from Algorithm 1 and Proposition B.1.

For hardness we generalize the Cook-Levin Theorem. So let $M = (\mathcal{R}, R', Q, \Sigma, \iota, \sqcup, \delta)$ be a polynomial time SRTM and $x \in (\Sigma \cup R)^*$ be the input for which we want to compute the output of M.

We define the following propositional variables:

- $T_{i,j,k}$, which is true if tape cell i contains symbol j at step k of the computation.
- $H_{i,k}$, which is true if the M's read/write head is at tape cell i at step k of the computation.
- $Q_{q,k}$, which is true if M is in state q at step k of the computation.

Furthermore, we need the following semiring values

- r_i, which is the i^{th} semiring value of the input x, and
- $\{r_{s+1}, \ldots, r_m\} = R'$, which are the constant semiring values that M has access to. Here, s is the number of semiring values in x.

Since M is a polynomial time SRTM, we can assume the existence of a polynomial p such that $p(n)$ bounds the number of transitions of M on any input of length n.

Given a finite set I and a family $(\beta_i)_{i \in I}$ of weighted formulas, we use the following shorthand:

$$\Sigma_{i \in I} \beta_i = \begin{cases} e_\oplus & \text{if } I = \emptyset \\ \beta_{i^*} + \Sigma_{i \in I \setminus \{i^*\}} \beta_i & \text{if } i^* \in I \end{cases}.$$

Note that this is well defined, since I is finite and addition is commutative and associative.

We define a weighted QBF $\Sigma\mathbf{T}\ \Sigma\mathbf{H}\ \Sigma\mathbf{Q}\ \alpha$, where $\Sigma\mathbf{T}$, $\Sigma\mathbf{H}$, and $\Sigma\mathbf{Q}$ correspond to the (sum) quantification of all variables $T_{i,j,k}$, $H_{i,k}$, and $Q_{q,k}$, respectively, and α is the product (i.e. connected with $*$) of the following subformulas

1. $T_{i,j,0}$
   Variable ranges: $0 \leq i < n$
   For each tape cell i that initially contains symbol $j \in \Sigma$ or $j = r$ when it contains the semiring value r.

2. $T_{i,\sqcup,0}$
   Variable ranges: $-p(n) \leq i < 0$ or $n \leq i \leq p(n)$
   Each tape cell i outside of the ones that contain the input contains $\sqcup$.

3. $Q_{\iota,0}$
   The initial state of M is ι.

4. $H_{0,0}$
   The initial position of the head is 0.

5. $\neg T_{i,j,k} + T_{i,j,k} * \neg T_{i,j',k}$
   Variable ranges: $-p(n) \leq i \leq p(n), j \in \Sigma \cup \{r_1, \ldots, r_s\}, 0 \leq k \leq p(n)$
   There is at most one symbol per tape cell.

6. $\Sigma_{j \in \Sigma \cup \{r_{i_1}, \ldots, r_{i_m}\}} T_{i,j,k}$
   Variable ranges: $-p(n) \leq i \leq p(n), 0 \leq k \leq p(n)$
   There is at least one symbol per tape cell.

7. $\neg T_{i,j,k} + T_{i,j,k} * \neg T_{i,j',k+1} + T_{i,j,k} * T_{i,j',k+1} * H_{i,k}$
   Variable ranges: $-p(n) \leq i \leq p(n), j \neq j' \in \Sigma \cup \{r_1, \ldots, r_s\}, 0 \leq k < p(n)$
   Tape remains unchanged unless written.

8. $\neg Q_{q,k} + Q_{q,k} * \neg Q_{q',k}$
   Variable ranges: $q \neq q' \in Q, 0 \leq k \leq p(n)$
   There is at most one state at a time.

9. $\neg H_{i,k} + H_{i,k} * \neg H_{i',k}$
   Variable ranges: $i \neq i', -p(n) \leq i \leq p(n), -p(n) \leq i' \leq p(n), 0 \leq k \leq p(n)$
   There is at most one head position at a time.

10. $\neg H_{i,k} + H_{i,k} * \neg Q_{q,k} + H_{i,k} * Q_{q,k} * \neg T_{i,\sigma,k} +$
    $H_{i,k} * Q_{q,k} * T_{i,\sigma,k} * \Sigma_{((q,\sigma),(q',\sigma'),d,r) \in \delta'} H_{i+d,k+1} * Q_{q',k+1} * T_{i,\sigma',k+1} * r$
    Variable ranges: $-p(n) \leq i \leq p(n), 0 \leq k < p(n)$ and $q \in Q$, $\sigma \in \Sigma \cup \{r_{i_1}, \ldots, r_{i_m}\}$ s.t. there exist q', σ', d, r s.t. $((q,\sigma),(q',\sigma'),d,r) \in \delta$.
    Possible transitions at computation step k when head is at position i with their respective weight.
    Here, we use

$$\delta' = \{((q,\sigma),(q',\sigma'),d,r) \in \delta \mid \sigma, \sigma' \in \Sigma \cup \{r_1, \ldots, r_m\}\}$$

since we only need to take into account the transitions for letters from the alphabet Σ, the semiring values $\{r_1, \ldots, r_s\}$ in the input and the constants $\{r_{s+1}, \ldots, r_m\}$. With this restriction $|\delta'|$ is finite and therefore the size of the sum is bounded by a constant that only depends on the SRTM.

11. $\neg H_{i,k} + H_{i,k} * \neg Q_{q,k} + H_{i,k} * Q_{q,k} * \neg T_{i,\sigma,k}$
$+H_{i,k} * Q_{q,k} * T_{i,\sigma,k} * H_{i,k+1} * Q_{q,k+1} * T_{i,\sigma,k+1}$
Variable ranges: $-p(n) \leq i \leq p(n), 0 \leq k < p(n)$ and $q \in Q$, $\sigma \in \Sigma \cup \{r_1, \ldots, r_m\}$ s.t. there do not exist q', σ', d, r s.t. $((q, \sigma), (q', \sigma'), d, r) \in \delta$.
The machine computation has ended. This is included so that when the machine has reached a final state it stays the same until k reaches $p(n)$.

In order to prove correctness, i.e., that the value of M on x is equal to $[\![\Sigma\mathbf{T}\ \Sigma\mathbf{H}\ \Sigma\mathbf{Q}\ \alpha]\!]_{\mathcal{R}}(\emptyset)$, we prove two claims.

(i) For each interpretation $\mathcal{I}$ of the variables $T_{i,j,k}, H_{i,k}, Q_{q,k}$ for $-p(n) \leq i \leq p(n), 0 \leq k < p(n)$ and $q \in Q$ such that $\mathcal{I}$ does not correspond to a computation path of M on x, it holds that $[\![\alpha]\!]_{\mathcal{R}}(\mathcal{I}) = e_{\oplus}$.

(ii) For each interpretation $\mathcal{I}$ of the variables $T_{i,j,k}, H_{i,k}, Q_{q,k}$ for $-p(n) \leq i \leq p(n), 0 \leq k < p(n)$ and $q \in Q$ such that $\mathcal{I}$ corresponds to a computation path π of M on x along configurations $c_1^{\pi} \overset{r^{(\pi_1)}}{\to} \ldots \overset{r^{(\pi_{n(\pi)-1})}}{\to} c_{n(\pi)}^{\pi}$ it holds that

$$[\![\alpha]\!]_{\mathcal{R}}(\mathcal{I}) = \bigoplus_{\pi',\ \text{s.t.} c_i^{\pi} = c_i^{\pi'}} r^{(\pi'_1)} \otimes \ldots \otimes r^{(\pi'_{n(\pi')-1})}.$$

If both claims hold, then it follows that $[\![\Sigma\mathbf{T}\ \Sigma\mathbf{H}\ \Sigma\mathbf{Q}\ \alpha]\!]_{\mathcal{R}}(\emptyset)$ is equal to the sum of $[\![\alpha]\!]_{\mathcal{R}}(\mathcal{I})$ over all interpretations $\mathcal{I}$ such that $\mathcal{I}$ corresponds to a computation path. For each of them, we know that the weight of the path is the product of the weights of the taken transitions, according to (ii). Since the value of M on x is equal to the sum of the weights of the paths, this implies correctness of the reduction.

We proceed to prove the claims. For this, we first generally show that the added subformulas 1. to 11. enforce their given purpose.

For 1. to 4. this is clear: In order for $[\![\alpha]\!]_{\mathcal{R}}(\mathcal{I})$ to be unequal to $e_{\oplus}$, the variables need to be included in the interpretation.

5. and 6. together ensure that at each time step there is exactly one symbol in each tape cell. So assume that $T_{i,j,k}$ and $T_{i,j',k}$ are in $\mathcal{I}$. Then

$$[\![\neg T_{i,j,k} + T_{i,j,k} * \neg T_{i,j',k}]\!]_{\mathcal{R}}(\mathcal{I}) = e_{\oplus} \oplus e_{\oplus} = e_{\oplus},$$

and so $[\![\alpha]\!]_{\mathcal{R}}(\mathcal{I}) = e_{\oplus}$. Assume alternatively that there are i, k such that for no j the variable $T_{i,j,k}$ is in $\mathcal{I}$. Then

$$[\![\Sigma_{j \in \Sigma \cup \{r_1, \ldots, r_s\}} T_{i,j,k}]\!]_{\mathcal{R}}(\mathcal{I}) = e_{\oplus}.$$

7. ensures that the tape remains unchanged unless written, i.e., unless the head is at position i at step k the value of the tape cell i stays the same. So assume that $H_{i,k}$ is not in $\mathcal{I}$ but $T_{i,j,k}$ and $T_{i,j',k}$ for $j \neq j'$ are in $\mathcal{I}$. Then

$$[\![\neg T_{i,j,k} + T_{i,j,k} * \neg T_{i,j',k+1} + T_{i,j,k} * T_{i,j',k+1} * H_{i,k}]\!]_{\mathcal{R}}(\mathcal{I}) = e_{\oplus} \oplus (e_{\otimes} \otimes e_{\oplus}) \oplus (e_{\otimes} \otimes e_{\otimes} \otimes e_{\oplus}) = e_{\oplus}.$$

8. ensures that there is at most one state at a time by analogous reasoning to 5.

9. ensures that there is at most one head position at a time by analogous reasoning to 5.

10. models the possible transitions at computation step k when the head is at position i including their respective weights, *if there is a possible transition* for the given state and tape cell entry. Otherwise, this subformula is not added but the one in 11. is. So assume that $H_{i,k}, Q_{q,k}, T_{i,\sigma,k} \in \mathcal{I}$. Then the value of the subformula is

$$\begin{aligned} & [\![\neg H_{i,k} + H_{i,k} * \neg Q_{q,k} + H_{i,k} * Q_{q,k} * \neg T_{i,\sigma,k}]\!]_{\mathcal{R}}(\mathcal{I}) \oplus \\ & [\![H_{i,k} * Q_{q,k} * T_{i,\sigma,k} * \Sigma_{((q,\sigma),(q',\sigma'),d,r)\in\delta'} H_{i+d,k+1} * Q_{q',k+1} * T_{i,\sigma',k+1} * r]\!]_{\mathcal{R}}(\mathcal{I}) \\ = \quad & e_{\oplus} \oplus e_{\otimes} \otimes [\![\Sigma_{((q,\sigma),(q',\sigma'),d,r)\in\delta'} H_{i+d,k+1} * Q_{q',k+1} * T_{i,\sigma',k+1} * r]\!]_{\mathcal{R}}(\mathcal{I}) \\ = \quad & [\![\Sigma_{((q,\sigma),(q',\sigma'),d,r)\in\delta'} H_{i+d,k+1} * Q_{q',k+1} * T_{i,\sigma',k+1} * r]\!]_{\mathcal{R}}(\mathcal{I}) \end{aligned}$$

This means that the expression evaluates to the sum of all weights of the transitions we take. Note that in order for two transitions $((q,\sigma),(q'_1,\sigma'_1),d_1,r_1)$, $((q,\sigma),(q'_2,\sigma'_2),d_2,r_2)$ to be different at least one of $q'_1 \neq q'_2$, $\sigma'_1 \neq \sigma'_2$, $d_1 \neq d_2$ or $r_1 \neq r_2$ needs to hold. If $q'_1 \neq q'_2$, $\sigma'_1 \neq \sigma'_2$ or $d_1 \neq d_2$ then one of 5., 8., or 9. is falsified. Thus, in this case, we can take multiple transitions if they differ in the weights only. On the other hand, we must take at least one transition, since otherwise the whole sum evaluates to $e_{\oplus}$. It follows that we transition to exactly one new configuration to obtain a non-zero value for α. In that case, we have $H_{i+d,k+1}, Q_{q',k+1}, T_{i,\sigma',k+1} \in \mathcal{I}$ for the corresponding transition(s) $((q,\sigma),(q',\sigma'),d,r) \in \delta'$ and

$$[\![\Sigma_{((q,\sigma),(q',\sigma'),d,r)\in\delta'} H_{i+d,k+1} * Q_{q',k+1} * T_{i,\sigma',k+1} * r]\!]_{\mathcal{R}}(\mathcal{I}) = \bigoplus_{((q,\sigma),(q',\sigma'),d,r)\in\delta'} r.$$

11. models that if at computation step k when the head is at position i *there is no possible transition* for the given state q and tape cell entry σ, then the head position, state and tape cell entries stay the same. Otherwise, this subformula is not added but one in 10. is. So assume that $H_{i,k}, Q_{q,k}, T_{i,\sigma,k} \in \mathcal{I}$. Then the value of the subformula is

$$\begin{aligned} & [\![\neg H_{i,k} + H_{i,k} * \neg Q_{q,k} + H_{i,k} * Q_{q,k} * \neg T_{i,\sigma,k}]\!]_{\mathcal{R}}(\mathcal{I}) \oplus \\ & [\![H_{i,k} * Q_{q,k} * T_{i,\sigma,k} * H_{i,k+1} * Q_{q,k+1} * T_{i,\sigma,k+1}]\!]_{\mathcal{R}}(\mathcal{I}) \\ = \quad & e_{\oplus} \oplus e_{\otimes} \otimes [\![H_{i,k+1} * Q_{q,k+1} * T_{i,\sigma,k+1}]\!]_{\mathcal{R}}(\mathcal{I}) \end{aligned}$$

This means that the expression evaluates to $e_{\otimes}$, if $H_{i,k+1}, Q_{q,k+1}, T_{i,\sigma,k+1} \in \mathcal{I}$ and evaluates to $e_{\oplus}$, otherwise. Thus, the formula enforces the desired constraint: if we are in a configuration without further transitions we stay in it until the time limit $p(n)$ is reached. Notably, this does not influence the weight, since we always obtain a factor of $e_{\otimes}$.

Putting things together, we see that 8. and 10./11. together ensure that there is exactly one state at each time. Similarly, 9. and 10./11. together ensure that there is exactly one head position at each time. This together with the constraints associated originally with 1. to 11. show that the desired claims and therefore also the theorem hold. □

Proposition B.1. *The sum, using $\oplus$, of the results of all execution paths for a call to* $\textsc{Eval}_{\mathcal{R}}(\alpha, \mathcal{I})$ *is equal to* $[\![\alpha]\!]_{\mathcal{R}}(\mathcal{I})$.

Proof. We proceed by structural induction on α.

- Case $\alpha = k$:
  The algorithm returns k, therefore the statement is true.

- Case $\alpha = l, l \in \{v, \neg v\}$:
  The algorithm returns $e_\oplus, e_\otimes$ when l is false or true w.r.t. $\mathcal{I}$, therefore the statement is true.

- Case $\alpha = \alpha_1 + \alpha_2$:
  The algorithm nondeterministically returns $\textsc{Eval}_\mathcal{R}(\alpha_1, \mathcal{I})$ or $\textsc{Eval}_\mathcal{R}(\alpha_2, \mathcal{I})$. By the induction hypothesis, we know that the sum of all the execution paths of $\textsc{Eval}_\mathcal{R}(\alpha_i, \mathcal{I})$ is equal to $[\![\alpha_i]\!]_\mathcal{R}(\mathcal{I})$. Since we guess $i \in \{1, 2\}$ nondeterministically $\textsc{Eval}_\mathcal{R}(\alpha, \mathcal{I})$ has all the execution paths of $\textsc{Eval}_\mathcal{R}(\alpha_1, \mathcal{I})$ and $\textsc{Eval}_\mathcal{R}(\alpha_2, \mathcal{I})$ and therefore the sum of all values produced by execution paths of $\textsc{Eval}_\mathcal{R}(\alpha, \mathcal{I})$ is equal to
  $$[\![\alpha_2]\!]_\mathcal{R}(\mathcal{I}) \oplus [\![\alpha_2]\!]_\mathcal{R}(\mathcal{I}) = [\![\alpha]\!]_\mathcal{R}(\mathcal{I}).$$

- Case $\alpha = \alpha_1 * \alpha_2$:
  The algorithm returns one of the results of the execution paths of $\textsc{Eval}_\mathcal{R}(\alpha_1, \mathcal{I})$ multiplied by one of the results of the execution paths of $\textsc{Eval}_\mathcal{R}(\alpha_2, \mathcal{I})$. By the induction hypothesis, we know that the sum of all the execution paths of $\textsc{Eval}_\mathcal{R}(\alpha_i, \mathcal{I})$ is equal to $[\![\alpha_i]\!]_\mathcal{R}(\mathcal{I})$. For $\textsc{Eval}_\mathcal{R}(\alpha, \mathcal{I})$, we have one nondeterministic execution path for every combination of nondeterministic execution paths of $\textsc{Eval}_\mathcal{R}(\alpha_i, \mathcal{I}), i = 1, 2$ since the nondeterministic choices are made independently. Therefore, if the results of the execution paths of $\textsc{Eval}_\mathcal{R}(\alpha_i, \mathcal{I})$ are $k_1^{(i)}, \ldots, k_{n_i}^{(i)}$ we see that the sum of all the execution paths of $\textsc{Eval}_\mathcal{R}(\alpha, \mathcal{I})$ is
  $$\bigoplus_{j_1=1}^{n_1} \bigoplus_{j_2=1}^{n_2} k_{j_1}^{(1)} \otimes k_{j_2}^{(1)}$$
  By using distributivity, we obtain that this is equal to
  $$\bigoplus_{j_1=1}^{n_1} \left(k_{j_1}^{(1)} \otimes \bigoplus_{j_2=1}^{n_2} k_{j_2}^{(1)} \right) = \bigoplus_{j_1=1}^{n_1} k_{j_1}^{(1)} \otimes \bigoplus_{j_2=1}^{n_2} k_{j_2}^{(1)} \overset{IH}{=} [\![\alpha_1]\!]_\mathcal{R}(\mathcal{I}) \otimes [\![\alpha_2]\!]_\mathcal{R}(\mathcal{I}) = [\![\alpha_1 * \alpha_2]\!]_\mathcal{R}(\mathcal{I}).$$

- Case $\alpha = \Sigma v\ \alpha$:
  Works analogously to case $\alpha = \alpha_1 + \alpha_2$.

□

B.2 Complexity of SumProd($\mathcal{R}$), AMC, SCSP, ΣFO-Eval($\mathcal{R}$), mrg(F), Datalog Semiring Provenance

↪ **Theorems 20, 23 to 25 and 29**

For the Karp-reducibility between SAT($\mathcal{R}$), SumProd($\mathcal{R}$), AMC, SCSP, ΣFO-Eval($\mathcal{R}$), mrg(F), we do not prove interreducibility between two problems at a time but prove it by using the strategy visualized in Figure 3.

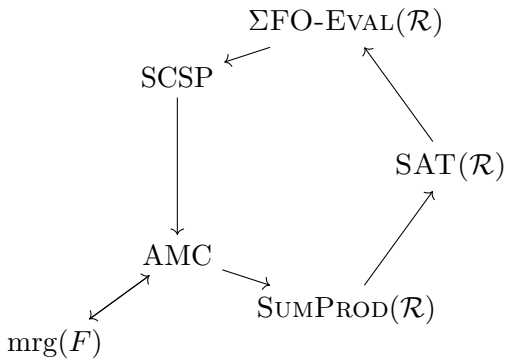

Figure 3: Karp s-reductions that are proven to show Theorems 20, 23 to 25 and 29. $P \to Q$ means that a Karp s-reduction from P to Q is given.

Theorem B.2. *For every commutative semiring $\mathcal{R}$ it holds that each of the following problems can be reduced to one another using Karp s-reductions:*

- SAT($\mathcal{R}$)*,*
- ΣFO-Eval($\mathcal{R}$)*,*
- *Computing blevel(P) of an SCSP P over $\mathcal{R}$,*
- SumProd*($\mathcal{R}$),*
- *AMC over $\mathcal{R}$, and*
- *Computing mrg(F) for a semiring-labeled CNF F over $\mathcal{R}$.*

Proof. Karp s-reducibility is transitive, therefore it suffices to prove

- SAT($\mathcal{R}$) is Karp s-reducible to ΣFO-Eval($\mathcal{R}$)
- ΣFO-Eval($\mathcal{R}$) is Karp s-reducible to computing *blevel*(P) of an SCSP P over $\mathcal{R}$
- Computing *blevel*(P) of an SCSP P over $\mathcal{R}$ is Karp s-reducible to SumProd($\mathcal{R}$)
- SumProd($\mathcal{R}$) is Karp s-reducible to AMC over $\mathcal{R}$
- AMC over $\mathcal{R}$ is Karp s-reducible to SAT($\mathcal{R}$)
- AMC over $\mathcal{R}$ is Karp s-reducible to mrg(F) over $\mathcal{R}$
- mrg(F) over $\mathcal{R}$ is Karp s-reducible to AMC over $\mathcal{R}$

□

This gives us that as desired

Corollary B.3. *Theorems 20, 23 to 25 and 29 hold.*

Proof. We proved in Theorem 18 that SAT($\mathcal{R}$) is NP($\mathcal{R}$)-complete. It follows from Theorem B.2 and Lemma 17 that also all other problems are NP($\mathcal{R}$)-complete. Additionally, Karp s-reducibility is shown in Theorem B.2. □

We proceed to prove the Karp s-reducibilities in the order specified in the above Theorem.

Lemma B.4. SAT($\mathcal{R}$) *is Karp s-reducible to* ΣFO-Eval($\mathcal{R}$)

Proof. Let α be a ΣBF over $\mathcal{R}$ with variable $v_1, \ldots, v_n$. We choose σ as the triple $\langle\{\bot, \top\}, \{t(.)\}, \{x_{v_1}, \ldots, x_{v_n}\}\rangle$ and $\mathcal{I} = \{t(\top), \neg t(\bot)\}$. Then we replace every propositional variable v in α by $t(x_v)$, which symbolizes that v has truth value $\top$, and every quantifier Σv with the corresponding first order quantifier Σx_v. For the resulting ΣFO formula β, it is easy to see that

$$[\![\alpha]\!]_{\mathcal{R}}(\emptyset) = [\![\beta]\!]_{\mathcal{R}}(\mathcal{I})$$

since the semantics are analogously defined. □

Lemma B.5. ΣFO-Eval($\mathcal{R}$) *is Karp s-reducible to computing blevel*(P) *of an SCSP* P *over* $\mathcal{R}$

This lemma is the one that contains the most difficult steps, since it requires reducing arbitrary combinations of sums and products to a sum of product.

We first prove some general structural assumptions that we can make about ΣFO formulas:

Lemma B.6. *For every ΣFO weighted formula α over a commutative semiring $\mathcal{R}$, domain $\mathcal{D}$ and interpretation $\mathcal{I}$ we can construct in polynomial time in $\alpha, \mathcal{D}, \mathcal{I}$ a ΣFO weighted formula $\beta = \Sigma x_1 \ldots \Sigma x_n \gamma$, where γ is quantifier free, over $\mathcal{R}$ and an interpretation $\mathcal{I}'$ such that*

$$[\![\alpha]\!]^{\sigma}_{\mathcal{R}}(\mathcal{I}) = [\![\beta]\!]^{\sigma}_{\mathcal{R}}(\mathcal{I}').$$

Proof. Let α be a ΣFO weighted formula over semiring $\mathcal{R}$, $\mathcal{D}$ be a domain and $\mathcal{I}$ be an interpretation. We show that we can always push out the sum quantifiers.
Let $\alpha = (\Sigma x \alpha_1) * \alpha_2$. W.l.o.g. we can assume that α_2 does not contain x. Then

$$\begin{aligned}
[\![\alpha]\!]^{\sigma}_{\mathcal{R}}(\mathcal{I}) &= [\![(\Sigma x \alpha_1) * \alpha_2]\!]^{\sigma}_{\mathcal{R}}(\mathcal{I}) \\
&= [\![\Sigma x \alpha_1]\!]^{\sigma}_{\mathcal{R}}(\mathcal{I}) \otimes [\![\alpha_2]\!]^{\sigma}_{\mathcal{R}}(\mathcal{I}) \\
&= \left(\bigoplus_{d \in \mathcal{D}} [\![\alpha_1\{x \mapsto d\}]\!]^{\sigma}_{\mathcal{R}}(\mathcal{I})\right) \otimes [\![\alpha_2]\!]^{\sigma}_{\mathcal{R}}(\mathcal{I})
\end{aligned}$$

Since addition distributes over multiplication and α_2 does not contain x, we get

$$\begin{aligned}
[\![\alpha]\!]^{\sigma}_{\mathcal{R}}(\mathcal{I}) &= \bigoplus_{d \in \mathcal{D}} \left([\![\alpha_1\{x \mapsto d\}]\!]^{\sigma}_{\mathcal{R}}(\mathcal{I}) \otimes [\![\alpha_2\{x \mapsto d\}]\!]^{\sigma}_{\mathcal{R}}(\mathcal{I})\right. \\
&= \bigoplus_{d \in \mathcal{D}} [\![(\alpha_1 * \alpha_2)\{x \mapsto d\}]\!]^{\sigma}_{\mathcal{R}}(\mathcal{I}) \\
&= [\![\Sigma x(\alpha_1 * \alpha_2)]\!]^{\sigma}_{\mathcal{R}}(\mathcal{I})
\end{aligned}$$

So we can choose $\beta = \Sigma x(\alpha_1 * \alpha_2)$ and $\mathcal{I}' = \mathcal{I}$.
Let $\alpha = \alpha_1 * (\Sigma x \alpha_2)$. This case works analogously to the previous one.

Let $\alpha = (\Sigma x \alpha_1) + \alpha_2$. We choose some $d \in D$, a new predicate symbol p_{d^*} with arity one and define $\mathcal{I}' = \mathcal{I} \cup \{p_{d^*}(d^*)\}$. Then $\beta = \Sigma x(\alpha_1 + (\alpha_2 * p_{d^*}(x)))$ has the following semantics w.r.t $\mathcal{I}'$:

$$\begin{aligned}
[\![\beta]\!]^\sigma_\mathcal{R}(\mathcal{I}') &= [\![\Sigma x(\alpha_1 + (\alpha_2 * p_{d^*}(x)))]\!]^\sigma_\mathcal{R}(\mathcal{I}') \\
&= \bigoplus_{d \in D} [\![(\alpha_1 + (\alpha_2 * p_{d^*}(x)))\{x \mapsto d\}]\!]^\sigma_\mathcal{R}(\mathcal{I}') \\
&= \bigoplus_{d \in D} [\![\alpha_1\{x \mapsto d\}]\!]^\sigma_\mathcal{R}(\mathcal{I}') \oplus [\![(\alpha_2 * p_{d^*}(x))\{x \mapsto d\}]\!]^\sigma_\mathcal{R}(\mathcal{I}') \\
&= \bigoplus_{d \in D} [\![\alpha_1\{x \mapsto d\}]\!]^\sigma_\mathcal{R}(\mathcal{I}') \oplus \begin{cases} [\![\alpha_2\{x \mapsto d\}]\!]^\sigma_\mathcal{R}(\mathcal{I}') & d = d^* \\ e_\oplus & \text{otherwise.} \end{cases}
\end{aligned}$$

Since p_{d^*} is a new predicate that does not occur in α, we can remove it from $\mathcal{I}'$

$$\begin{aligned}
[\![\beta]\!]^\sigma_\mathcal{R}(\mathcal{I}') &= \bigoplus_{d \in D} [\![\alpha_1\{x \mapsto d\}]\!]^\sigma_\mathcal{R}(\mathcal{I}) \oplus \begin{cases} [\![\alpha_2]\!]^\sigma_\mathcal{R}(\mathcal{I}) & d = d^* \\ e_\oplus & \text{otherwise.} \end{cases} \\
&= \left(\bigoplus_{d \in D} [\![\alpha_1\{x \mapsto d\}]\!]^\sigma_\mathcal{R}(\mathcal{I})\right) \oplus [\![\alpha_2]\!]^\sigma_\mathcal{R}(\mathcal{I}) \\
&= [\![\Sigma x \alpha_1]\!]^\sigma_\mathcal{R}(\mathcal{I}) \oplus [\![\alpha_2]\!]^\sigma_\mathcal{R}(\mathcal{I}) \\
&= [\![(\Sigma x \alpha_1) + \alpha_2]\!]^\sigma_\mathcal{R}(\mathcal{I})
\end{aligned}$$

So we can choose $\beta = \Sigma x(\alpha_1 + (\alpha_2 * p_{d^*}(x)))$, $\mathcal{I}' = \mathcal{I} \cup \{p_{d^*}(d^*)\}$.
The case $\alpha = \alpha_1 + (\Sigma x \alpha_2)$ works analogously.
This shows that we can always push the Σ quantifier to the top in polynomially many steps in the number of occurrences of $*$ and $+$ in α. Since we only need to add one predicate p_{d^*} to handle all the disjunctions, the size of the interpretation only increases polynomially. Last but not least, we only add $*p_{d^*}(x)$ to the formula at most once for every combination of Σ and $+$ in α, the size of β is also polynomial in α. □

Lemma B.7. *For every ΣFO weighted formula α over a commutative semiring $\mathcal{R}$, finite domain $\mathcal{D}$, and interpretation $\mathcal{I}$, then we can construct in polynomial time a ΣFO weighted formula β over $\mathcal{R}$ such that in every subformula $\Sigma x \gamma$ the variable x occurs in γ and*

$$[\![\alpha]\!]^\sigma_\mathcal{R}(\mathcal{I}) = [\![\beta]\!]^\sigma_\mathcal{R}(\mathcal{I}).$$

Proof. Let α be a ΣFO weighted formula over semiring $\mathcal{R}$, $\mathcal{D}$ be a domain, and $\mathcal{I}$ be an interpretation. We iteratively replace every subformula $\Sigma x \gamma$ where γ does not contain x of α by a formula with the same semantics w.r.t. $\mathcal{D}$ and $\mathcal{I}$.
So let $\Sigma x \gamma$ be a weighted formula where γ does not contain x. It has the semantics

$$\begin{aligned}
[\![\Sigma x \gamma]\!]^\sigma_\mathcal{R}(\mathcal{I}) &= \bigoplus_{d \in \mathcal{D}} [\![\gamma\{x \mapsto d\}]\!]^\sigma_\mathcal{R}(\mathcal{I}) \\
&= \bigoplus_{d \in \mathcal{D}} (e_\otimes \otimes [\![\gamma]\!]^\sigma_\mathcal{R}(\mathcal{I})) \\
&= \left(\bigoplus_{d \in \mathcal{D}} e_\otimes\right) \otimes [\![\gamma]\!]^\sigma_\mathcal{R}(\mathcal{I}) \\
&= \left(\bigoplus_{d \in \mathcal{D}} [\![e_\otimes]\!]^\sigma_\mathcal{R}(\mathcal{I})\right) \otimes [\![\gamma]\!]^\sigma_\mathcal{R}(\mathcal{I}) \\
&= [\![e_\otimes + \ldots + e_\otimes]\!]^\sigma_\mathcal{R}(\mathcal{I}) \otimes [\![\gamma]\!]^\sigma_\mathcal{R}(\mathcal{I})
\end{aligned}$$

$$= [\![(e_\otimes + \ldots + e_\otimes) * \gamma]\!]^\sigma_{\mathcal{R}}(\mathcal{I}).$$

So we can replace $\Sigma x \gamma$ by $(e_\otimes + \ldots + e_\otimes) * \gamma$ without changing the semantics of the formula. This is further possible in polynomial time and the resulting formula only increases in size polynomially in $|\mathcal{D}|$. This stays true, when we iteratively perform the replacement for all such quantifiers Σx. □

We need one more auxiliary lemma:

Lemma B.8. *For each clause $a_1 \vee a_2 \vee a_3$, where a_i is a literal over the variables $v_s, s \in S$ we can construct a constraint $c_{a_1 \vee a_2 \vee a_3} = \langle def, con \rangle$ s.t. $def(d_1, d_2, d_3) = e_\otimes$ if d_1, d_2, d_3 is a satisfying assignment of $a_1 \vee a_2 \vee a_3$.*

Proof. We can do this as follows.

$$def(d_1, d_2, d_3) = \begin{cases} & d_1, d_2, d_3 \in \{\bot, \top\}, \\ e_\otimes & \exists i : (a_i = v_s \wedge d_i = \top) \\ & \vee (a_i = \neg v_s \wedge d_i = \bot) \\ e_\oplus & \text{otherwise.} \end{cases} \tag{3}$$

con is simply the set of variables that occur in $a_1 \vee a_2 \vee a_3$. □

Proof of Lemma B.5. So let α be a ΣFO weighted formula over $\mathcal{R}$ and domain $\mathcal{D}$ that is to be evaluated using interpretation $\mathcal{I}$.

We use Lemmas B.6 and B.7 to ensure that α the input formula is of the form

$$\Sigma x_1 \ldots \Sigma x_n \beta$$

for some quantifier free β, that contains every variable $x_i, i = 1, \ldots, n$. We define $P = \langle C, \{s, x_1, \ldots, x_n\} \rangle$ inductively on the structure of β s.t.

$$(\Pi C) \Downarrow_{\{x_1, \ldots, x_n\}} = \langle def, \{x_1, \ldots, x_n\} \rangle,$$

and for all $(d_1, \ldots, d_n) \in \mathcal{D}^n$ it holds that

$$def(d_1, \ldots, d_n) = [\![\beta\{x_i \mapsto d_i\}]\!]^\sigma_{\mathcal{R}}(\mathcal{I})$$

whereas for all $(d_1, \ldots, d_n) \in D^n \setminus \mathcal{D}^n$ (remember that $\mathcal{D}$ is the domain of the ΣFO formula and D is the domain of the SCSP) it holds that

$$def'(d_1, \ldots, d_n) = e_\oplus.$$

Then

$$\begin{aligned} blevel(P) &= (\Pi C) \Downarrow_\emptyset \\ &= (\Pi C) \Downarrow_{\{x_1, \ldots, x_n\}} \Downarrow_\emptyset \\ &= \langle def, \{x_1, \ldots, x_n\} \rangle \Downarrow_\emptyset \\ &= \bigoplus_{(d_1, \ldots, d_n) \in D^n} def'(d_1, \ldots, d_n) \\ &= \bigoplus_{(d_1, \ldots, d_n) \in \mathcal{D}^n} def'(d_1, \ldots, d_n) \end{aligned}$$

$$= \bigoplus_{(d_1,\ldots,d_n)\in\mathcal{D}^n} [\![\beta\{x_i \mapsto d_i\}]\!]^{\sigma}_{\mathcal{R}}(\mathcal{I})$$
$$= [\![\Sigma x_1 \ldots \Sigma x_n \beta]\!]^{\sigma}_{\mathcal{R}}(\mathcal{I})$$
$$= [\![\alpha]\!]^{\sigma}_{\mathcal{R}}(\mathcal{I})$$

So let $\mathcal{I}$ be an interpretation and $\mathcal{D}$ a domain. We first define the domain D of the constraint system as $D := \mathcal{D} \cup \{\top, \bot\}$

Definition B.9 (Subformulas)**.** *Let α a quantifier free weighted formula. Then $\mathcal{S}(\alpha)$ the set of subformulas of α indexed by position $p \in \{0,1\} \times \{.0,.1\}^*$ is $\mathcal{S}^0(\alpha)$, where $\mathcal{S}^i(\alpha)$ for $i = 0, 1$ is defined using induction on the structure of the formula as follows:*

- *Case $\alpha \in \{p(\vec{x}), \neg p(\vec{x}), k\}$: Then $\mathcal{S}^i(\alpha) = \{(i, \alpha)\}$ for $i = 0, 1$.*
- *Case $\alpha = \alpha_1 + \alpha_2$ or $\alpha = \alpha_1 * \alpha_2$: Then $\mathcal{S}^i(\alpha) = \{(i, \alpha)\} \cup \{(i.r, \beta) \mid (r, \beta) \in \mathcal{S}^0(\alpha_1)\} \cup \{(i.r, \beta) \mid (r, \beta) \in \mathcal{S}^1(\alpha_2)\}$ for $i = 0, 1$.*

We assert (in the following referred to as (*)) that for values $\vec{d} \in D^k$, for which we do not fix the value $def(\vec{d})$, it is zero, i.e., $def(\vec{d}) = e_{\oplus}$. We define P using $\mathcal{S}(\beta)$. Intuitively, the strategy is as follows: For each indexed subformula of (s, γ) we add a variable v_s that determines whether the subformula should be included in the computation. Then for atomic subformulas we add a constraint that has the value of the atomic formula if it is included. For complex subformula $\alpha_1 * \alpha_2$, resp. $\alpha_1 + \alpha_2$ we add constraints that ensure that if the subformula should be included, then also α_1 and α_2, resp. α_1 xor α_2 have to be included.

Formally, we add the following constraints:

- For each $(s, k) \in \mathcal{S}(\beta)$ s.t. $k \in R$:
  Add the constraint that evaluates to k when the variable v_s has value $\top$ and evaluates to $e_{\otimes}$ otherwise. We add the constraint set
  $$C_{(s,k)} = \{c_{(s,k)}\}$$
  $$c_{(s,k)} = \langle\{\top \mapsto k, \bot \mapsto e_{\otimes}\}, \{v_s\}\rangle$$
- For each $(s, p(\vec{x})) \in \mathcal{S}(\beta)$:
  We can see $p(\vec{x})$ as a constraint on the variables in $\vec{x}$ that evaluates to $e_{\otimes}$ when $p(\sigma(\vec{x})) \in \mathcal{I}$ for the assignment σ of the variables $\mathrm{var}(\vec{x})$ in $\vec{x}$ and $e_{\oplus}$ otherwise. This however only happens when the variable v_s has value $\top$, otherwise the value is $e_{\otimes}$. As a consequence we add the constraint set
  $$C_{(s,p(\vec{x}))} = \{c_{(s,p(\vec{x}))}\}$$
  $$\begin{aligned} c_{(s,p(\vec{x}))} = \langle &\{(\top, d_1, \ldots, d_k) \mapsto e_{\otimes} \mid p(\vec{x})\{x_i \mapsto d_i\} \in \mathcal{I}, d_i \in \mathcal{D}\} \\ &\cup \{(\top, d_1, \ldots, d_k) \mapsto e_{\oplus} \mid p(\vec{x})\{x_i \mapsto d_i\} \notin \mathcal{I}, d_i \in \mathcal{D}\} \\ &\cup \{(\bot, d_1, \ldots, d_k) \mapsto e_{\otimes} \mid d_i \in \mathcal{D}\}, \\ &\{v_s\} \cup \mathrm{var}(\vec{x})\rangle \end{aligned}$$
- For each $(s, \neg p(\vec{x})) \in \mathcal{S}(\beta)$:
  Analogously with $e_{\otimes}$ and $e_{\oplus}$ swapped. We add the constraint set
  $$C_{(s,\neg p(\vec{x}))} = \{c_{(s,\neg p(\vec{x}))}\}$$

$$\begin{aligned} c_{(s,\neg p(\vec{x}))} = \langle &\{(\top, d_1, \ldots, d_k) \mapsto e_\oplus \mid p(\vec{x})\{x_i \mapsto d_i\} \in \mathcal{I}, d_i \in \mathcal{D}\} \\ &\cup \{(\top, d_1, \ldots, d_k) \mapsto e_\otimes \mid p(\vec{x})\{x_i \mapsto d_i\} \notin \mathcal{I}, d_i \in \mathcal{D}\} \\ &\cup \{(\bot, d_1, \ldots, d_k) \mapsto e_\otimes \mid d_i \in \mathcal{D}\}, \\ &\{v_s\} \cup \mathrm{var}(\vec{x})\rangle \end{aligned}$$

- For each $(s, \alpha) \in \mathcal{S}(\beta)$, $\alpha = \alpha_1 * \alpha_2$:
  Then we know that $(s.0, \alpha_1), (s.1, \alpha_2) \in \mathcal{S}(\beta)$. We require that for $v_s, v_{s.0}, v_{s.1}$ the following relation holds:

$$\begin{aligned} v_s &\to (v_{s.0} \wedge v_{s.1}) \\ \neg v_s &\to (\neg v_{s.0} \wedge \neg v_{s.1}) \end{aligned}$$

  We can rewrite them in 3CNF as follows

$$\begin{aligned} &(\neg v_s \vee v_{s.0}) \wedge (\neg v_s \vee v_{s.1}) \\ &(v_s \vee \neg v_{s.0}) \wedge (v_s \vee \neg v_{s.1}). \end{aligned}$$

  Then using Lemma B.8 (where $a_2 = a_3$ is possible) we add the constraint set

$$C_{(s,\alpha)} = \{c_{\neg v_s \vee v_{s.0}}, c_{\neg v_s \vee v_{s.1}}, c_{v_s \vee \neg v_{s.0}}, c_{v_s \vee \neg v_{s.1}}\}.$$

- For each $(s, \alpha) \in \mathcal{S}(\beta)$, $\alpha = \alpha_1 + \alpha_2$:
  Then we know that $(s.0, \alpha_1), (s.1, \alpha_2) \in \mathcal{S}(\beta)$. We require that for $v_s, v_{s.0}, v_{s.1}$ the following relation holds:

$$\begin{aligned} v_s &\to (v_{s.0} \wedge \neg v_{s.1} \vee \neg v_{s.0} \wedge v_{s.1}) \\ \neg v_s &\to (\neg v_{s.0} \wedge \neg v_{s.1}) \end{aligned}$$

  We can rewrite them in 3CNF as follows

$$\begin{aligned} &(\neg v_s \vee v_{s.0} \vee v_{s.1}) \wedge (\neg v_s \vee \neg v_{s.0} \vee \neg v_{s.1}) \\ &(v_s \vee \neg v_{s.0}) \wedge (v_s \vee \neg v_{s.1}). \end{aligned}$$

  Then using Lemma B.8 we add the constraint set

$$C_{(s,\alpha)} = \{c_{\neg v_s \vee v_{s.0} \vee v_{s.1}}, c_{\neg v_s \vee \neg v_{s.0} \vee \neg v_{s.1}}, c_{v_s \vee \neg v_{s.0}}, c_{v_s \vee \neg v_{s.1}}\}.$$

- We further need to assert that s_0, i.e. the variable specifying whether the root node will be included in the computation, is true: We add it again using Lemma B.8 (with $a_1 = a_2 = a_3 = v_0$)

$$c_{v_0}.$$

So overall the constraint problem P is given as

$$\langle \bigcup_{(s,\alpha) \in \mathcal{S}(\beta)} C_{(s,\alpha)} \cup \{c_{v_0}\}, \{v_s \mid (s, \alpha) \in \mathcal{S}(\beta)\} \cup \{x_1, \ldots, x_n\}\rangle.$$

This is feasible to construct in polynomial time in the size of β.

We prove the correctness of the approach by structural induction on β. We use the induction invariant that for

$$(\Pi \bigcup_{(s,\alpha)\in\mathcal{S}(\beta)} C_{(s,\alpha)}) \Downarrow_{\{v_0,x_1,\ldots,x_n\}} = \langle def, \{v_0, x_1, \ldots, x_n\}\rangle,$$

and for all $d_1, \ldots, d_n \in \mathcal{D}$

$$def(d_0, \ldots, d_n) = \begin{cases} [\![\beta\{x_i \mapsto d_i\}]\!]^{\sigma}_{\mathcal{R}}(\mathcal{I}) & d_0 = \top \\ e_{\otimes} & d_0 = \bot \\ e_{\oplus} & \text{otherwise.} \end{cases}$$

holds.

- $\beta = k$, $k \in R$:
  Then $P = \langle\langle\{\top \mapsto k\} \cup \{\bot \mapsto e_{\otimes}\}, \{v_0\}\rangle, \{v_0\}\rangle$. We know

  $$(\Pi C) \Downarrow_{v_0} = \langle\langle\{\top \mapsto k, \bot \mapsto e_{\otimes}\}, \{v_0\}\rangle.$$

  Then due to Assertion (*)

  $$def(d_0) = \begin{cases} k & d = \top \\ e_{\otimes} & d = \bot \\ e_{\oplus} & \text{otherwise.} \end{cases} = \begin{cases} [\![\beta]\!]^{\sigma}_{\mathcal{R}}(\mathcal{I}) & d_0 = \top \\ e_{\otimes} & d_0 = \bot \\ e_{\oplus} & \text{otherwise.} \end{cases}$$

- $\beta = p(\vec{x})$:
  Then

  $$\begin{aligned} P = \langle\{\langle\{&(\top, d_1, \ldots, d_k) \mapsto e_{\otimes} \mid p(\vec{x})\{x_i \mapsto d_i\} \in \mathcal{I}, d_i \in \mathcal{D}\} \\ &\cup \{(\top, d_1, \ldots, d_k) \mapsto e_{\oplus} \mid p(\vec{x})\{x_i \mapsto d_i\} \notin \mathcal{I}, d_i \in \mathcal{D}\} \\ &\cup \{(\bot, d_1, \ldots, d_k) \mapsto e_{\otimes} \mid d_i \in \mathcal{D}\}, \\ &\{v_0\} \cup \mathrm{var}(\vec{x})\rangle\} \cup \{c_{v_0}\}, \\ &\{v_0\} \cup \mathrm{var}(\vec{x})\rangle. \end{aligned}$$

  Let

  $$(\Pi C) \Downarrow_{\{v_0\}\cup\mathrm{var}(\vec{x})} = \langle def, \{v_0, x_1, \ldots, x_n\}\rangle,$$

  and $d_1, \ldots, d_n \in \mathcal{D}$. Then

  $$def(d_0, d_1, \ldots, d_n) = \begin{cases} e_{\otimes} & d_0 = \top, p(\vec{x})\{x_i \mapsto d_i\} \in \mathcal{I} \\ e_{\oplus} & d_0 = \top, p(\vec{x})\{x_i \mapsto d_i\} \notin \mathcal{I} \\ e_{\otimes} & d_0 = \bot, \\ e_{\oplus} & \text{otherwise.} \end{cases} = \begin{cases} [\![\beta]\!]^{\sigma}_{\mathcal{R}}(\mathcal{I}) & d_0 = \top \\ e_{\otimes} & d_0 = \bot \\ e_{\oplus} & \text{otherwise.} \end{cases}$$

- $\beta = \neg p(\vec{x})$:
  Then

$$\begin{aligned} P = \langle \{ \langle \{ &(\top, d_1, \ldots, d_k) \mapsto e_\otimes \mid p(\vec{x})\{x_i \mapsto d_i\} \notin \mathcal{I}, d_i \in \mathcal{D}\} \\ &\cup \{(\top, d_1, \ldots, d_k) \mapsto e_\oplus \mid p(\vec{x})\{x_i \mapsto d_i\} \in \mathcal{I}, d_i \in \mathcal{D}\} \\ &\cup \{(\bot, d_1, \ldots, d_k) \mapsto e_\otimes \mid d_i \in \mathcal{D}\}, \\ &\{v_0\} \cup \mathrm{var}(\vec{x})\rangle\} \cup \{c_{v_0}\}, \\ &\{v_0\} \cup \mathrm{var}(\vec{x})\rangle. \end{aligned}$$

  Let

$$(\Pi C) \Downarrow_{\{v_0\} \cup \mathrm{var}(\vec{x})} = \langle def, \{v_0, x_1, \ldots, x_n\}\rangle,$$

  and $d_1, \ldots, d_n \in \mathcal{D}$. Then

$$\begin{aligned} def(d_0, d_1, \ldots, d_n) &= \bigoplus_{d \in D} \begin{cases} e_\otimes & d_0 = \top, p(\vec{x})\{x_i \mapsto d_i\} \notin \mathcal{I} \\ e_\oplus & d_0 = \top, p(\vec{x})\{x_i \mapsto d_i\} \in \mathcal{I} \\ e_\otimes & d_0 = \bot, \\ e_\oplus & \text{otherwise.} \end{cases} \\ &= \begin{cases} [\![\beta]\!]^\sigma_{\mathcal{R}}(\mathcal{I}) & d_0 = \top \\ e_\otimes & d_0 = \bot \\ e_\oplus & \text{otherwise.} \end{cases} \end{aligned}$$

- $\beta = \beta_1 * \beta_2$:
  Then

$$\begin{aligned} &\langle \bigcup_{(s,\alpha) \in \mathcal{S}(\beta)} C_{(s,\alpha)}, \{v_s \mid (s,\alpha) \in \mathcal{S}(\beta)\} \cup \{x_1, \ldots, x_n\}\rangle \\ = &\langle \bigcup_{(s,\alpha) \in \mathcal{S}^0(\beta_1)} C_{(0.s,\alpha)} \cup \bigcup_{(s,\alpha) \in \mathcal{S}^1(\beta_2)} C_{(1.s,\alpha)} \cup C_{(0,\beta)}, \\ &\{v_{0.s} \mid (s,\alpha) \in \mathcal{S}^0(\beta_1)\} \cup \{v_{1.s} \mid (s,\alpha) \in \mathcal{S}^1(\beta_2)\} \cup \{v_0\} \cup \mathrm{var}(\beta_1) \cup \mathrm{var}(\beta_2)\rangle \end{aligned}$$

  Now if we let the constraint problems defined for β_1, β_2 be $P_i = \langle C_i, con_i\rangle$, we can see that (modulo appropriate renaming of the variables of the form v_s)

$$\begin{aligned} &\langle \bigcup_{(s,\alpha) \in \mathcal{S}(\beta)} C_{(s,\alpha)}, \{v_s \mid (s,\alpha) \in \mathcal{S}(\alpha)\} \cup \{x_1, \ldots, x_n\}\rangle \\ = &\langle C_1 \cup C_2 \cup C_{(0,\beta)}, \\ &con_1 \cup con_2 \cup \{v_0\}\rangle \end{aligned}$$

  Using the induction hypothesis we get that

$$(\Pi C_i) \Downarrow_{\{v_{0.i}, \mathrm{var}(\beta_i)\}} = \langle def_i, \{v_{0.i}\} \cup \mathrm{var}(\beta_i)\rangle,$$

  and for $d_1, \ldots, d_n \in \mathcal{D}$

$$def_i(d_0, \ldots, d_{n_i}) = \begin{cases} [\![\beta_i\{x_i \mapsto d_i\}]\!]^\sigma_{\mathcal{R}}(\mathcal{I}) & d_0 = \top \\ e_\otimes & d_0 = \bot \\ e_\oplus & \text{otherwise.} \end{cases}$$

Then for

$$(\Pi C) \Downarrow_{\{v_0,\mathrm{var}(\beta)\}} = \langle def, \{v_0\} \cup \mathrm{var}(\beta)\rangle,$$

we see that

$$
\begin{aligned}
& def(d_0,\ldots,d_n)\\
=& \bigoplus_{\vec{d}\in D^{con_1\cup con_2}} \bigotimes_{\langle def_c,con_c\rangle\in C_1\cup C_2\cup C_{(0,\beta)}} def_c(\vec{d}\downarrow^{con}_{con_c})\\
=& \bigoplus_{\vec{d}\in D^{con_1\cup con_2}} \bigotimes_{\langle def_c,con_c\rangle\in C_1\cup C_2} def_c(\vec{d}\downarrow^{con}_{con_c}) \otimes \begin{cases} & d_0, d_{v_{0.0}}, d_{v_{0.1}} \in \{\bot,\top\},\\ e_\otimes & d_0 \to (d_{v_{0.0}} \wedge d_{v_{0.1}})\\ & \wedge\neg d_0 \to (\neg d_{v_{0.0}} \wedge \neg d_{v_{0.1}})\\ e_\oplus & \text{otherwise.}\end{cases}\\
=& \bigoplus_{\vec{d}\in D^{con_1\cup con_2}} \begin{cases} \bigotimes_{\langle def_c,con_c\rangle\in C_1\cup C_2} def_c(\vec{d}\downarrow^{con}_{con_c}) & d_0 = d_{v_{0.0}} = d_{v_{0.1}} = \top,\\ \bigotimes_{\langle def_c,con_c\rangle\in C_1\cup C_2} def_c(\vec{d}\downarrow^{con}_{con_c}) & d_0 = d_{v_{0.0}} = d_{v_{0.1}} = \bot,\\ e_\oplus & \text{otherwise.}\end{cases}\\
=& \bigoplus_{\vec{d}\in D^{con_1\cup con_2}} \begin{cases} \bigotimes_{\langle def_c,con_c\rangle\in C_1} def_c(\vec{d}\downarrow^{con}_{con_c}) & \\ \otimes\bigotimes_{\langle def_c,con_c\rangle\in C_2} def_c(\vec{d}\downarrow^{con}_{con_c}) & d_0 = d_{v_{0.0}} = d_{v_{0.1}} = \top,\\ \bigotimes_{\langle def_c,con_c\rangle\in C_1} def_c(\vec{d}\downarrow^{con}_{con_c}) & \\ \otimes\bigotimes_{\langle def_c,con_c\rangle\in C_2} def_c(\vec{d}\downarrow^{con}_{con_c}) & d_0 = d_{v_{0.0}} = d_{v_{0.1}} = \bot,\\ e_\oplus & \text{otherwise.}\end{cases}\\
=& \bigoplus_{\vec{d}_1\in D^{con_1}} \bigoplus_{\vec{d}_2\in D^{con_2}} \begin{cases} \bigotimes_{\langle def_c,con_c\rangle\in C_1} def_c(\vec{d}_1\downarrow^{con}_{con_c}) & d_0 = d_{v_{0.0}}\\ & \quad\| \\ \otimes\bigotimes_{\langle def_c,con_c\rangle\in C_2} def_c(\vec{d}_2\downarrow^{con}_{con_c}) & d_{v_{0.1}} = \top,\\ \bigotimes_{\langle def_c,con_c\rangle\in C_1} def_c(\vec{d}_1\downarrow^{con}_{con_c}) & d_0 = d_{v_{0.0}}\\ & \quad\| \\ \otimes\bigotimes_{\langle def_c,con_c\rangle\in C_2} def_c(\vec{d}_2\downarrow^{con}_{con_c}) & d_{v_{0.1}} = \bot,\\ e_\oplus & \text{otherwise.}\end{cases}\\
=& \bigoplus_{d_{v_{0.0}},d_{v_{0.1}}\in D} \begin{cases} \bigoplus_{\vec{d}_1\in D^{con_1}\setminus\{v_{0.0}\}} \bigotimes_{\langle def_c,con_c\rangle\in C_1} def_c(\vec{d}_1\downarrow^{con}_{con_c}) & d_0 = d_{v_{0.0}}\\ & \quad\| \\ \otimes \bigoplus_{\vec{d}_2\in D^{con_2}\setminus\{v_{0.1}\}} \bigotimes_{\langle def_c,con_c\rangle\in C_2} def_c(\vec{d}_2\downarrow^{con}_{con_c}) & d_{v_{0.1}} = \top,\\ \bigoplus_{\vec{d}_1\in D^{con_1}\setminus\{v_{0.0}\}} \bigotimes_{\langle def_c,con_c\rangle\in C_1} def_c(\vec{d}_1\downarrow^{con}_{con_c}) & d_0 = d_{v_{0.0}}\\ & \quad\| \\ \otimes \bigoplus_{\vec{d}_2\in D^{con_2}\setminus\{v_{0.1}\}} \bigotimes_{\langle def_c,con_c\rangle\in C_2} def_c(\vec{d}_2\downarrow^{con}_{con_c}) & d_{v_{0.1}} = \bot,\\ e_\oplus & \text{otherwise.}\end{cases}\\
=& \begin{cases} def_1(\top,d_1,\ldots,d_{n_1}) * def_2(\top,d_1,\ldots,d_{n_2}) & d_0 = \top,\\ def_1(\bot,d_1,\ldots,d_{n_1}) * def_2(\bot,d_1,\ldots,d_{n_2}) & d_0 = \bot,\\ e_\oplus & \text{otherwise.}\end{cases}\\
=& \begin{cases} [\![\beta_1\{x_i \mapsto d_i\}]\!]^\sigma_{\mathcal{R}}(\mathcal{I}) \otimes [\![\beta_2\{x_i \mapsto d_i\}]\!]^\sigma_{\mathcal{R}}(\mathcal{I}) & d_0 = \top,\\ e_\otimes & d_0 = \bot,\\ e_\oplus & \text{otherwise.}\end{cases}
\end{aligned}
$$

$$= \begin{cases} [\![(\beta_1 * \beta_2)\{x_i \mapsto d_i\}]\!]^{\sigma}_{\mathcal{R}}(\mathcal{I}) & d_0 = \top, \\ e_{\otimes} & d_0 = \bot, \\ e_{\oplus} & \text{otherwise.} \end{cases}$$

- $\beta = \beta_1 + \beta_2$:
  Then

$$\begin{aligned} &\langle \bigcup_{(s,\alpha)\in\mathcal{S}(\beta)} C_{(s,\alpha)}, \{v_s \mid (s,\alpha) \in \mathcal{S}(\beta)\} \cup \{x_1, \ldots, x_n\}\rangle \\ =\langle &\bigcup_{(s,\alpha)\in\mathcal{S}^0(\beta_1)} C_{(0.s,\alpha)} \cup \bigcup_{(s,\alpha)\in\mathcal{S}^1(\beta_2)} C_{(1.s,\alpha)} \cup C_{(0,\beta)}, \\ &\{v_{0.s} \mid (s,\alpha) \in \mathcal{S}^0(\beta_1)\} \cup \{v_{1.s} \mid (s,\alpha) \in \mathcal{S}^1(\beta_2)\} \cup \{v_0\} \cup \mathrm{var}(\beta_1) \cup \mathrm{var}(\beta_2)\rangle \end{aligned}$$

  Now if we let the constraint problems defined for β_1, β_2 be $P_i = \langle C_i, con_i\rangle$, we can see that (modulo appropriate renaming of the variables of the form v_s)

$$\begin{aligned} &\langle \bigcup_{(s,\alpha)\in\mathcal{S}(\beta)} C_{(s,\alpha)}, \{v_s \mid (s,\alpha) \in \mathcal{S}(\alpha)\} \cup \{x_1, \ldots, x_n\}\rangle \\ =&\langle C_1 \cup C_2 \cup C_{(0,\beta)}, \\ &con_1 \cup con_2 \cup \{v_0\}\rangle \end{aligned}$$

  Using the induction hypothesis we get that

$$(\Pi C_i) \Downarrow_{\{v_{0.i}, \mathrm{var}(\beta_i)\}} = \langle def_i, \{v_{0.i}\} \cup \mathrm{var}(\beta_i)\rangle,$$

  and for $d_1, \ldots, d_n \in \mathcal{D}$

$$def_i(d_0, \ldots, d_{n_i}) = \begin{cases} [\![\beta_i\{x_i \mapsto d_i\}]\!]^{\sigma}_{\mathcal{R}}(\mathcal{I}) & d_0 = \top \\ e_{\otimes} & d_0 = \bot \\ e_{\oplus} & \text{otherwise.} \end{cases}$$

  Then

$$\begin{aligned} &def(d_0, \ldots, d_n) \\ =&\bigoplus_{\vec{d}\in D^{con_1\cup con_2}} \bigotimes_{\langle def_c, con_c\rangle \in C_1\cup C_2\cup C_{(0,\beta)}} def_c(\vec{d}\downarrow^{con}_{con_c}) \\ =&\bigoplus_{\vec{d}\in D^{con_1\cup con_2}} \bigotimes_{\langle def_c, con_c\rangle \in C_1\cup C_2} def_c(\vec{d}\downarrow^{con}_{con_c}) \otimes \begin{cases} & d_0, d_{v_{0.0}}, d_{v_{0.1}} \in \{\bot, \top\}, \\ e_{\otimes} & d_0 \to (d_{v_{0.0}} \text{ xor } d_{v_{0.1}}) \\ & \wedge \neg d_0 \to (\neg d_{v_{0.0}} \wedge \neg d_{v_{0.1}}) \\ e_{\oplus} & \text{otherwise.} \end{cases} \\ =&\bigoplus_{\vec{d}\in D^{con_1\cup con_2}} \begin{cases} \bigotimes_{\langle def_c, con_c\rangle \in C_1\cup C_2} def_c(\vec{d}\downarrow^{con}_{con_c}) & d_0 = d_{v_{0.1}} = \top, d_{v_{0.0}} = \bot, \\ \bigotimes_{\langle def_c, con_c\rangle \in C_1\cup C_2} def_c(\vec{d}\downarrow^{con}_{con_c}) & d_0 = d_{v_{0.0}} = \top, d_{v_{0.1}} = \bot, \\ \bigotimes_{\langle def_c, con_c\rangle \in C_1\cup C_2} def_c(\vec{d}\downarrow^{con}_{con_c}) & d_0 = d_{v_{0.0}} = d_{v_{0.1}} = \bot, \\ e_{\oplus} & \text{otherwise.} \end{cases} \end{aligned}$$

$$= \bigoplus_{\vec{d_1}\in D^{con_1}} \bigoplus_{\vec{d_2}\in D^{con_2}} \begin{cases} \bigotimes_{\langle def_c, con_c\rangle\in C_1} def_c(\vec{d_1}\downarrow^{con}_{con_c}) & d_0 = d_{v_{0.1}} = \top \\ \otimes\bigotimes_{\langle def_c, con_c\rangle\in C_2} def_c(\vec{d_2}\downarrow^{con}_{con_c}) & \text{and } d_{v_{0.0}} = \bot, \\ \bigotimes_{\langle def_c, con_c\rangle\in C_1} def_c(\vec{d_1}\downarrow^{con}_{con_c}) & d_0 = d_{v_{0.0}} = \top \\ \otimes\bigotimes_{\langle def_c, con_c\rangle\in C_2} def_c(\vec{d_2}\downarrow^{con}_{con_c}) & \text{and } d_{v_{0.1}} = \bot, \\ \bigotimes_{\langle def_c, con_c\rangle\in C_1} def_c(\vec{d_1}\downarrow^{con}_{con_c}) & d_0 = d_{v_{0.0}} \\ & \quad \| \\ \otimes\bigotimes_{\langle def_c, con_c\rangle\in C_2} def_c(\vec{d_2}\downarrow^{con}_{con_c}) & d_{v_{0.1}} = \bot, \\ e_\oplus & \text{otherwise.} \end{cases}$$

$$= \bigoplus_{d_{v_{0.0}}, d_{v_{0.1}}\in D} \begin{cases} \bigoplus_{\vec{d_1}\in D^{con_1}\setminus\{v_{0.0}\}} \bigotimes_{\langle def_c, con_c\rangle\in C_1} def_c(\vec{d_1}\downarrow^{con}_{con_c}) & d_0 = d_{v_{0.1}} = \top \\ \otimes \bigoplus_{\vec{d_2}\in D^{con_2}\setminus\{v_{0.1}\}} \bigotimes_{\langle def_c, con_c\rangle\in C_2} def_c(\vec{d_2}\downarrow^{con}_{con_c}) & \text{and } d_{v_{0.0}} = \bot, \\ \bigoplus_{\vec{d_1}\in D^{con_1}\setminus\{v_{0.0}\}} \bigotimes_{\langle def_c, con_c\rangle\in C_1} def_c(\vec{d_1}\downarrow^{con}_{con_c}) & d_0 = d_{v_{0.0}} = \top \\ \otimes \bigoplus_{\vec{d_2}\in D^{con_2}\setminus\{v_{0.1}\}} \bigotimes_{\langle def_c, con_c\rangle\in C_2} def_c(\vec{d_2}\downarrow^{con}_{con_c}) & \text{and } d_{v_{0.1}} = \bot, \\ \bigoplus_{\vec{d_1}\in D^{con_1}\setminus\{v_{0.0}\}} \bigotimes_{\langle def_c, con_c\rangle\in C_1} def_c(\vec{d_1}\downarrow^{con}_{con_c}) & d_0 = d_{v_{0.0}} \\ & \quad \| \\ \otimes \bigoplus_{\vec{d_2}\in D^{con_2}\setminus\{v_{0.1}\}} \bigotimes_{\langle def_c, con_c\rangle\in C_2} def_c(\vec{d_2}\downarrow^{con}_{con_c}) & d_{v_{0.1}} = \bot, \\ e_\oplus & \text{otherwise.} \end{cases}$$

$$= \begin{cases} def_1(\top, d_1, \ldots, d_{n_1}) \otimes def_2(\bot, d_1, \ldots, d_{n_2}) & \\ \oplus def_1(\bot, d_1, \ldots, d_{n_1}) \otimes def_2(\top, d_1, \ldots, d_{n_2}) & d_0 = \top, \\ def_1(\bot, d_1, \ldots, d_{n_1}) \otimes def_2(\bot, d_1, \ldots, d_{n_2}) & d_0 = \bot, \\ e_\oplus & \text{otherwise.} \end{cases}$$

$$= \begin{cases} [\![\beta_1\{x_i \mapsto d_i\}]\!]^\sigma_{\mathcal{R}}(\mathcal{I}) \otimes e_\otimes \oplus e_\otimes \otimes [\![\beta_2\{x_i \mapsto d_i\}]\!]^\sigma_{\mathcal{R}}(\mathcal{I}) & d_0 = \top, \\ e_\otimes & d_0 = \bot, \\ e_\oplus & \text{otherwise.} \end{cases}$$

$$= \begin{cases} [\![(\beta_1 + \beta_2)\{x_i \mapsto d_i\}]\!]^\sigma_{\mathcal{R}}(\mathcal{I}) & d_0 = \top, \\ e_\otimes & d_0 = \bot, \\ e_\oplus & \text{otherwise.} \end{cases}$$

Now that we know that the induction invariant holds, i.e., for

$$(\Pi \bigcup_{(s,\alpha)\in\mathcal{S}(\beta)} C_{(s,\alpha)}) \Downarrow_{\{v_0, x_1, \ldots, x_n\}} = \langle def, \{v_0, x_1, \ldots, x_n\}\rangle,$$

and $d_1, \ldots, d_n \in \mathcal{D}$

$$def(d_0, \ldots, d_n) = \begin{cases} [\![\beta\{x_i \mapsto d_i\}]\!]^\sigma_{\mathcal{R}}(\mathcal{I}) & d_0 = \top \\ e_\otimes & d_0 = \bot \\ e_\oplus & \text{otherwise.} \end{cases}$$

it remains to show that for

$$(\Pi \bigcup_{(s,\alpha)\in\mathcal{S}(\beta)} C_{(s,\alpha)} \cup \{c_{v_0}\}) \Downarrow_{\{x_1, \ldots, x_n\}} = \langle def', \{x_1, \ldots, x_n\}\rangle,$$

and for all $d_1, \ldots, d_n \in \mathcal{D}$

$$def'(d_1, \ldots, d_n) = [\![\beta\{x_i \mapsto d_i\}]\!]^\sigma_{\mathcal{R}}(\mathcal{I}).$$

This is because

$$\begin{aligned}
def'(d_1,\dots,d_n) &= \bigoplus_{d_0\in D} def(d_0,\dots,d_n)\otimes\bigotimes_{\langle def_c,con_c\rangle\in\{c_{v_0}\}} def_c(\vec{d}\downarrow^{con}_{con_c}) \\
&= \bigoplus_{d_0\in D} \begin{cases} [\![\beta\{x_i\mapsto d_i\}]\!]^{\sigma}_{\mathcal{R}}(\mathcal{I}) & d_0=\top \\ e_{\otimes} & d_0=\bot \\ e_{\oplus} & \text{otherwise.}\end{cases} \otimes \begin{cases} e_{\otimes} & d_0=\top, \\ e_{\oplus} & \text{otherwise.}\end{cases} \\
&= \bigoplus_{d_0\in D} \begin{cases} [\![\beta\{x_i\mapsto d_i\}]\!]^{\sigma}_{\mathcal{R}}(\mathcal{I}) & d_0=\top \\ e_{\oplus} & \text{otherwise.}\end{cases} \\
&= [\![\beta\{x_i\mapsto d_i\}]\!]^{\sigma}_{\mathcal{R}}(\mathcal{I}).
\end{aligned}$$

□

Lemma B.10. *Computing $blevel(P)$ of an SCSP P over $\mathcal{R}$ is Karp s-reducible to the problem* SumProd*$(\mathcal{R})$.*

Proof. Let $P=\langle C, con\rangle$ be some SCSP over $\langle\mathcal{R}, D, V\rangle$. Recall that

$$\begin{aligned}
blevel(P) &= (\Pi_{\langle def,con'\rangle\in C}\ \langle def, con'\rangle)\Downarrow_{\emptyset} \\
&= \bigoplus_{\vec{d}\in D^{con}}\bigotimes_{\langle def,con'\rangle\in C} def(\vec{d}\downarrow^{con}_{con'})
\end{aligned}$$

We therefore choose the SumProd$(\mathcal{R})$-instance with variables $X_1,\dots,X_n$, where $n=|con|$ and functions $\{def \mid \langle def, con\rangle\in C\}$. Then the value of this SumProd$(\mathcal{R})$-instance is equal to $blevel(P)$. □

Lemma B.11. SumProd*$(\mathcal{R})$ is Karp s-reducible to AMC over $\mathcal{R}$*

Proof. Assume we are given a SumProd$(\mathcal{R})$-instance over variables $X_1,\dots,X_n$, functions $f_1,\dots,f_m$ with inputs $\vec{Y_1},\dots,\vec{Y_m}$ and domain $\mathcal{D}$. We construct the AMC-instance as follows.

We add the variables

- $v_{i,d}$ for each $i=1,\dots,n$ and $d\in D$, where $v_{i,j}$ is true if X_i takes values d
- $v_{j,\vec{d}}$ for each $j=1,\dots,m$ and $d\in D^{Y_j}$, where $v_{j,\vec{d}}$ is true if the input to the function f_j is equal to $\vec{d}$

We define α by setting

- $\alpha(v_{i,d})=\alpha(\neg v_{i,d})=e_{\otimes}$
- $\alpha(v_{j,\vec{d}})=f_j(\vec{d})$ and $\alpha(\neg v_{j,\vec{d}})=e_{\otimes}$

We define T as the propositional theory containing the following formulas

- $\neg v_{i,d}\vee\neg v_{i,d'}$ for each $i=1,\dots,n$ and $d\neq d'\in D$
- $\bigvee_{d\in D} v_{i,d}$ for each $i=1,\dots,n$
- $v_{j,\vec{d}}\leftrightarrow\bigwedge_{X_i\in\vec{Y_j}} v_{i,\vec{d}_{X_i}}$ for each $j=1,\dots,m$ and $\vec{d}\in D^{\vec{Y_j}}$

Then every satisfying interpretation $\mathcal{I}$ of T corresponds to one assignment of the variables X_i to domain values d_i. Furthermore, for each such $\mathcal{I}$ the variables $v_{j,\vec{d}}$ tell us which input the function f_j gets.

Since we sum up $\bigotimes_{v\in\mathcal{I}}\alpha(v)$ for each satisfying interpretation $\mathcal{I}$ and for each such $\mathcal{I}$ it holds that $\bigotimes_{j=1}^{m}\alpha(v_{j,\vec{d}}) = \bigotimes_{j=1}^{m} f_j(\vec{d})$, we see that $A(T)$ over $\mathcal{R}$ is exactly the value of the SumProd($\mathcal{R}$)-instance. □

Lemma B.12. *AMC over $\mathcal{R}$ is Karp s-reducible to* SAT($\mathcal{R}$).

Proof. Let the AMC-instance be over variables $v_1,\dots,v_n$, theory T and weight function α.

W.l.o.g. we can assume that the theory T consists of a single propositional formula ϕ (otherwise we simply take their conjunction). We apply the Tseitin transformation (Tseitin, 1983) to ϕ and obtain a 3CNF ψ with additional variables $x_1,\dots,x_m$ s.t. each satisfying assignment of ϕ is uniquely extendable to a satisfying assignment of ψ. Furthermore, each satisfying assignment of ψ is a satisfying assignment of ϕ when restricted to the variables $v_1,\dots,v_n$.

Recall that by Lemma 35, for each clause $l_1\vee l_2\vee l_3$ it holds that

$$\begin{aligned}\mathcal{I}\models l_1\vee l_2\vee l_3 &\iff [\![l_1 + \neg l_1 * l_2 + \neg l_1 * \neg l_2 * l_3]\!]_{\mathcal{R}}(\mathcal{I}) = e_{\otimes}\\ \mathcal{I}\not\models l_1\vee l_2\vee l_3 &\iff [\![l_1 + \neg l_1 * l_2 + \neg l_1 * \neg l_2 * l_3]\!]_{\mathcal{R}}(\mathcal{I}) = e_{\oplus}.\end{aligned}$$

So we can proceed as follows: in ψ we replace every clause $l_1\vee l_2\vee l_3$ by $l_1 + \neg l_1 * l_2 + \neg l_1 * \neg l_2 * l_3$ and subsequently every $\wedge$ by $*$. We call the resulting formula β and see that

$$\begin{aligned}\mathcal{I}\models\psi &\iff [\![\beta]\!]_{\mathcal{R}}(\mathcal{I}) = e_{\otimes}\\ \mathcal{I}\not\models\psi &\iff [\![\beta]\!]_{\mathcal{R}}(\mathcal{I}) = e_{\oplus}\end{aligned}$$

Therefore, the ΣBF γ defined as

$$\Sigma v_1\dots\Sigma v_n\Sigma x_1\dots\Sigma x_m\beta * \Pi_{i=1}^{n}(v_i * \alpha(v_i) + \neg v_i * \alpha(\neg v_i))$$

fulfills

$$A(T) = [\![\gamma]\!]_{\mathcal{R}}(\emptyset).$$

□

Lemma B.13. *AMC over $\mathcal{R}$ is Karp s-reducible to mrg(F) over $\mathcal{R}$ and vice versa.*

Proof. We first reduce AMC over $\mathcal{R}$ to mrg(F) over $\mathcal{R}$. Assume we are given an AMC-instance (T,α) over a commutative semiring $\mathcal{R}$ and variables $\mathcal{V}$. W.l.o.g., we can assume that T is a CNF $C_1\wedge\dots\wedge C_n$. If this was not the case, we could apply the Tseitin-transformation and extend α to the added variables by letting $\alpha(l) = e_{\otimes}$ for each of their literals l. We construct a semiring-weighted CNF F as

$$\begin{aligned}F &= F_T\cup F_+\cup F_-, \text{ where}\\ F_T &= \{(C_1,e_{\oplus}),\dots,(C_n,e_{\oplus})\},\\ F_+ &= \{(\neg v,\alpha(v)) \mid v\in\mathcal{V}\},\end{aligned}$$

$$F_- = \{(v, \alpha(\neg v)) \mid v \in \mathcal{V}\}.$$

Then

$$\begin{aligned}
mrg(F) &= \bigoplus_{\mathcal{I} \in \mathrm{Int}(F)} \phi_F(\mathcal{I}) \\
&= \bigoplus_{\mathcal{I} \models T} \phi_{F_+ \cup F_-}(\mathcal{I}) \\
&= \bigoplus_{\mathcal{I} \models T} \bigotimes_{v \in V} \left\{ \begin{array}{cc} \alpha(v) & \mathcal{I} \not\models \neg v \\ e_\otimes & \text{otherwise.} \end{array} \right\} \otimes \bigotimes_{v \in V} \left\{ \begin{array}{cc} \alpha(\neg v) & \mathcal{I} \not\models v \\ e_\otimes & \text{otherwise.} \end{array} \right\} \\
&= \bigoplus_{\mathcal{I} \models T} \bigotimes_{v \in V} \left\{ \begin{array}{cc} \alpha(v) & \mathcal{I} \models v \\ e_\otimes & \text{otherwise.} \end{array} \right\} \otimes \bigotimes_{v \in V} \left\{ \begin{array}{cc} \alpha(\neg v) & \mathcal{I} \models \neg v \\ e_\otimes & \text{otherwise.} \end{array} \right\} \\
&= \bigoplus_{\mathcal{I} \models T} \bigotimes_{v \in \mathcal{I}} \alpha(v) \otimes \bigotimes_{v \notin \mathcal{I}} \alpha(\neg v) \\
&= A(T).
\end{aligned}$$

We see that the result of the marginalization problem of F over $\mathcal{R}$ is equal to $A(T)$, the algebraic model count of the instance (T, α). Since we can construct F in polynomial time from (T, α), this shows that the AMC over $\mathcal{R}$ is Karp s-reducible to $\mathrm{mrg}(F)$ over $\mathcal{R}$.

Next, we reduce $\mathrm{mrg}(F)$ over $\mathcal{R}$ to AMC over $\mathcal{R}$. Assume we are given a semiring-labeled CNF $F = \{(C_1, w_1), \ldots, (C_n, w_n)\}$. Consider the following AMC instance with theory T and labeling function α such that

$$\begin{aligned}
T &= \bigwedge_{i=1}^{n} C_i \leftrightarrow c_i \\
\alpha(l) &= \left\{ \begin{array}{ll} w_i & \text{if } l = \neg c_i, \\ e_\otimes & \text{otherwise.} \end{array} \right. ,
\end{aligned}$$

where $c_1, \ldots, c_n$ are distinct variables that do not occur in F. Then

$$\begin{aligned}
A(T) &= \bigoplus_{\mathcal{I} \models T} \bigotimes_{v \in \mathcal{I}} \alpha(v) \otimes \bigotimes_{\neg v \in \mathcal{I}} \alpha(\neg v) \\
&= \bigoplus_{\mathcal{I} \models T} \bigotimes_{\neg c_i \in \mathcal{I}} \alpha(\neg c_i) \\
&= \bigoplus_{\mathcal{I}} \bigotimes_{i, \text{ s.t. } \mathcal{I} \not\models C_i} w_i \\
&= mrg(F).
\end{aligned}$$

Since we can construct (T, α) in polynomial time from F, this shows that the $\mathrm{mrg}(F)$ over $\mathcal{R}$ is Karp s-reducible to AMC over $\mathcal{R}$. □

Appendix C. Relation to Classical Complexity Classes

C.1 FPSpace(poly)-membership of SAT($\mathcal{R}$) for Efficiently Encoded Semirings ↪ Proposition 32

Proposition 32 (FPSpace(poly) Upper-Bound)**.** *If $\tau(\mathcal{R})$ is an efficiently encoded commutative semiring, then* $\mathrm{SAT}(\tau(\mathcal{R}))$ *is in* FPSpace(poly).

Proof. We have seen that $\mathrm{SAT}(\tau(\mathcal{R}))$ is Karp s-reducible to SumProd($\tau(\mathcal{R})$), therefore it is sufficient to prove FPSpace(poly)-membership for SumProd($\tau(\mathcal{R})$).

Assume we are given a SumProd($\tau(\mathcal{R})$)-instance and a domain $\mathcal{D}$ as a set of variables $X_1, \ldots, X_n$ and functions $f_1, \ldots, f_m$ with input vectors $\vec{Y_1}, \ldots, \vec{Y_m}$ consisting of variables from $X_1, \ldots, X_n$. The value of this instance is then given by

$$a = \bigoplus_{x_1,\ldots,x_n \in \mathcal{D}} \bigotimes_{i=1}^{m} f_i(\vec{y_i}).$$

We can bound $\|a\|_\tau$ by using that $\tau(\mathcal{R})$ is efficiently encoded.

$$\begin{aligned}
\|a\|_\tau &= \|\bigoplus_{x_1,\ldots,x_n \in \mathcal{D}} \bigotimes_{i=1}^{m} f_i(\vec{y_i})\|_\tau \\
&\leq p(\log_2 |\mathcal{D}|^n) \max_{x_1,\ldots,x_n \in \mathcal{D}} \|\bigotimes_{i=1}^{m} f_i(\vec{y_i})\|_\tau \\
&\leq p(n \log_2 |\mathcal{D}|) \max_{x_1,\ldots,x_n \in \mathcal{D}} p(m) \max_{i=1,\ldots,m} \|f_i(\vec{y_i})\|_\tau \\
&\leq p(n \log_2 |\mathcal{D}|) p(m) \max_{x_1,\ldots,x_n \in \mathcal{D}} \max_{i=1,\ldots,m} \|f_i(\vec{y_i})\|_\tau
\end{aligned}$$

Therefore, the size of the result a and consequently also all intermediate results are bounded polynomially in the size of the input. We see that we at least have enough space to store all the results and intermediate results. Furthermore, we know that multiplication and addition are in FP and therefore also in FPSpace(poly). Last but not least, iterating over all the assignments of values $d_i \in \mathcal{D}$ to variables X_i is also possible in polynomial space. □

C.2 NP, #P, GapP, OptP-completeness of SAT($\mathbb{B}$), SAT($\mathbb{N}$), SAT($\mathbb{Z}$), SAT($\mathcal{R}_{\max}$)

↪ **Theorem 33**

Theorem 33. *For* $(\mathcal{R}, \mathcal{C}) = (\mathbb{B}, \mathrm{NP})$, $(\mathbb{N}, \#\mathrm{P})$, $(\mathbb{Z}, \mathrm{GapP})$, $(\mathcal{R}_{\max}, \mathrm{OptP})$ *and the binary representation* bin *of the integers,* SAT(bin($\mathcal{R}$)) *is* $\mathcal{C}$*-complete w.r.t. Karp reductions.*

Proof. Membership is not hard to see:

- SAT(bin($\mathbb{B}$)) is SAT;
- SAT(bin($\mathbb{N}$)) can be solved in #P by simulating Algorithm 1, where instead of returning k we generate k accepting paths;
- SAT(bin($\mathbb{Z}$)) can be solved in GapP by simulating Algorithm 1, where instead of returning k we generate k accepting and zero rejecting paths if $k \geq 0$ and zero accepting and $|k|$ rejecting paths, otherwise;
- SAT(bin($\mathcal{R}_{\max}$)) can be solved in OptP by simulating Algorithm 1, exactly as it is.

.

For $\mathcal{C}$-hardness, we consider the following reductions

- SAT(bin($\mathbb{B}$)) is SAT;
- We can reduce #SAT to SAT(bin($\mathbb{N}$)) as follows.

  Let ϕ be the propositional formula, whose satisfying assignments we want to count. We apply the Tseitin transformation (Tseitin, 1983) to ϕ and obtain a 3CNF ψ with

additional variables $x_1, \ldots, x_m$ s.t. each satisfying assignment of ϕ is uniquely extendable to a satisfying assignment of ψ. Furthermore, for each satisfying assignment of ψ is a satisfying assignment of ϕ when restricted to the original variables $v_1, \ldots, v_n$.

Recall that by Lemma 35, for each clause $l_1 \vee l_2 \vee l_3$ it holds that

$$\mathcal{I} \models l_1 \vee l_2 \vee l_3 \iff [\![l_1 + \neg l_1 * l_2 + \neg l_1 * \neg l_2 * l_3]\!]_{\mathcal{R}}(\mathcal{I}) = e_{\otimes}$$
$$\mathcal{I} \not\models l_1 \vee l_2 \vee l_3 \iff [\![l_1 + \neg l_1 * l_2 + \neg l_1 * \neg l_2 * l_3]\!]_{\mathcal{R}}(\mathcal{I}) = e_{\oplus}.$$

So we can proceed as follows: in ψ we replace every clause $l_1 \vee l_2 \vee l_3$ by $l_1 + \neg l_1 * l_2 + \neg l_1 * \neg l_2 * l_3$ and subsequently every $\wedge$ by $*$. The resulting formula β fulfills

$$\mathcal{I} \models \psi \iff [\![\beta]\!]_{\mathcal{R}}(\mathcal{I}) = e_{\otimes}$$
$$\mathcal{I} \not\models \psi \iff [\![\beta]\!]_{\mathcal{R}}(\mathcal{I}) = e_{\oplus}$$

Therefore, the semantics of the ΣBF defined as

$$\Sigma v_1 \ldots \Sigma v_n \Sigma x_1 \ldots \Sigma x_m \beta$$

is exactly the number of satisfying assignments of ϕ.

- Recall that for an $n \times n$ matrix A with entries a_{ij},the permanent is given by

$$\sum_{\sigma \in S_n} \prod_{i=1}^{n} a_{i\sigma(i)},$$

where S_n is the set of permutations of the numbers $1, \ldots, n$. We reduce computing the permanent of a given integer matrix A, which is well-known to be #P-complete (Valiant, 1979), to SAT(bin($\mathbb{Z}$)) as follows.

We use variables v_{ij}, which are true when we include the value a_{ij} in the current product. We construct a weighted QBF α s.t. $[\![\alpha]\!]_{\mathbb{Z}}(\mathcal{I}) = e_{\otimes}$ if the variables $v_{ij} \in \mathcal{I}$ correspond to a permutation and $e_{\oplus}$ otherwise.

We define α as the product of the following weighted formulas

- $\neg v_{ij} + v_{ij} * \neg v_{ij'}$ for each $i = 1, \ldots, n$ and $j \neq j' = 1, \ldots, n$ (include at most one element per row)
- $v_{i1} + \ldots + v_{in}$ for each $i = 1, \ldots, n$ (include at least one element per column)
- $\neg v_{ij} + v_{ij} * \neg v_{i'j}$ for each $i \neq i' = 1, \ldots, n$ and $j = 1, \ldots, n$ (include at most one element per column)

Then the semantics of the ΣBF

$$\Sigma a_{11} \ldots \Sigma a_{nn} \alpha * \Pi_{i,j=1}^{n} (v_{ij} * a_{ij} + \neg v_{ij})$$

is exactly the permanent of A.

- We can reduce LexMaxSAT to SAT(bin($\mathcal{R}_{\max}$)) as follows. Let ϕ be the propositional formula whose maximum satisfying assignments we want to know. We again as in the proof of #P-hardness for SAT(bin($\mathbb{N}$)) obtain the formula β s.t.

$$\mathcal{I} \models \psi \iff [\![\beta]\!]_{\mathcal{R}}(\mathcal{I}) = e_{\otimes}$$
$$\mathcal{I} \not\models \psi \iff [\![\beta]\!]_{\mathcal{R}}(\mathcal{I}) = e_{\oplus}$$

  Then the semantics of the ΣBF

$$\Sigma v_1 \dots \Sigma v_n \Sigma x_1 \dots \Sigma x_m \beta * \Pi_{i=1}^{n}(v_i * 2^{n-i} + \neg v_i)$$

  is the bitstring representing the maximum satisfying assignment of ϕ respectively $-\infty$ if there is none.

□

C.3 Results for Classes of Semirings

C.3.1 Reducibility via Epimorphisms
↪ Theorem 40

Theorem 40. *Let $\tau_i(\mathcal{R}_i), i = 1, 2$ be two encoded commutative semirings, such that*

1. *there exists a polynomial time computable epimorphism $f : \tau_1(R_1) \to \tau_2(R_2)$, and*
2. *for each $\tau_2(r_2) \in \tau(R_2)$ one can compute in polynomial time $\tau_1(r_1)$ s.t. $f(\tau_1(r_1)) = \tau_2(r_2)$ from $\tau_2(r_2)$.*

Then SAT($\tau_2(\mathcal{R}_2)$) *is counting-reducible to* SAT($\tau_1(\mathcal{R}_1)$).

Proof. Let the assumptions of the theorem be given. Furthermore, let α be a SAT($\tau_2(\mathcal{R})$)-instance. We can compute $[\![\alpha]\!]_{\tau_2(\mathcal{R})}(\emptyset)$ as follows.

First we replace every occurrence of a value $\tau_2(r_2)$ with a value $\tau_1(r_1)$ s.t. $f(\tau_1(r_1)) = \tau_2(r_2)$. This is possible in polynomial time. The resulting ΣBF β is a SAT($\tau_1(\mathcal{R}_1)$)-instance. We now compute $[\![\beta]\!]_{\tau_1(\mathcal{R}_1)}(\emptyset)$. Then we only need to apply f to the result and have the solution $[\![\alpha]\!]_{\tau_2(\mathcal{R})}(\emptyset)$. This is possible in polynomial time with one oracle call to SAT($\tau_1(\mathcal{R}_1)$).

Since we only need to solve one SAT($\tau_1(\mathcal{R}_1)$)-instance and do not require any information about the original instance to obtain the final solution from the solution of the SAT($\tau_1(\mathcal{R}_1)$)-instance we have a counting-reduction. □

Lemma 41 (Hardest Semirings)**.** *There exists an encoding τ^* for*

1. $\mathbb{N}_{\leq o}[(x_i)_\infty]$;
2. $\mathbb{Z}_p[(x_i)_\infty]$;
3. $\mathbb{N}_{\leq o} \times \mathbb{Z}_p[(x_i)_\infty]$;
4. $\mathbb{N}[(x_i)_\infty]$

such that for any commutative efficiently encoded commutative semiring $\tau(\mathcal{R})$ that is in addition

1. *periodic with periodicity* 1*;*
2. *periodic with periodicity $p \geq 2$ and offset* 0*;*
3. *periodic with periodicity $p \geq 2$ and offset $o > 0$;*
4. *not periodic*

it holds that $\mathrm{SAT}(\tau(\mathcal{R}))$ *is counting reducible to*

1. $\mathrm{SAT}(\tau^*(\mathbb{N}_{\leq o}[(x_i)_\infty]))$*;*
2. $\mathrm{SAT}(\tau^*(\mathbb{Z}_p[(x_i)_\infty]))$*;*
3. $\mathrm{SAT}(\tau^*(\mathbb{N}_{\leq o} \times \mathbb{Z}_p[(x_i)_\infty]))$*;*
4. $\mathrm{SAT}(\tau^*(\mathbb{N}[(x_i)_\infty]))$*, respectively.*

Proof. As the encoding function τ^* we take the function that encodes a monomial by representing the coefficient and index of each variable in binary and each exponent in unary. Furthermore, a polynomial is encoded as the list of the encodings of its monomials with non-zero coefficients. Then we can define an epimorphism f by letting

$$f\left(\sum_{\vec{i}\in\mathbb{N}^*} a_{\vec{i}} x_0^{i_0} \cdot \dots \cdot x_{|\vec{i}|}^{i_{|\vec{i}|}}\right) := \bigoplus_{\vec{i}\in\mathbb{N}^*, i_j\neq 0 \Rightarrow \mathrm{bin}(i_j)\in\tau(R)} a_{\vec{i}} \cdot \left(\mathrm{bin}(0)^{i_0} \otimes \dots \otimes \mathrm{bin}(|\vec{i}|)^{i_{|\vec{i}|}}\right).$$

For this definition to make sense, recall that τ maps to $\{0,1\}^*$ and therefore $\tau(R) \subseteq \{0,1\}^*$. Since we can assume w.l.o.g. that for all values $r \in R$ it holds that $\tau(r) \in \{0,1\}^*$ has a 1 as the first letter, it follows that for some $i_j \in \mathbb{N}$ it holds that $\mathrm{bin}(i_j) \in \tau(R)$. Therefore, the right-hand side of the definition is always a value in $\tau(R)$.

The idea of this definition is that we identify x_{i_j} with the value that $\mathrm{bin}(i_j)$ represents in $\tau(\mathcal{R})$. Then we can perform any calculation over $\tau^*(\mathcal{S}[(x_i)_\infty])$ with $\mathcal{S} = \mathbb{N}_{\leq o}, \mathbb{Z}_p, \mathbb{N}_{\leq o} \times \mathbb{Z}_p, \mathbb{N}$, depending on the periodicity and offset of $\mathcal{R}$, by using the variables as placeholders for the actual values. Finally, we can obtain the actual value over $\tau(R)$ using f. Note, that the sum only considers exponent vectors $\vec{i} \in \mathbb{N}^*$ where any exponent i_j of x_j is equal to zero, when $\mathrm{bin}(i_j)$ is not in $\tau(R)$ to ensure that f is well defined.

The check whether $\mathrm{bin}(i_j)$ is in $\tau(R)$ may be expensive. Therefore, f is not necessarily polynomial time computable over the whole semiring $\tau^*(\mathcal{S}[(x_i)_\infty])$. We consider instead $\tau^*(\mathcal{S}')$, where $\mathcal{S}' = \mathcal{S}[\{x_i \mid \mathrm{bin}(i) \in \tau(R)\}]$. Then, evaluating f over $\tau^*(\mathcal{S}')$ takes polynomial time in the size of the encoding of the input since $\tau(\mathcal{R})$ is efficiently encoded.

It is easy to see that the second condition of Theorem 40 is satisfied, since we can map $\mathrm{bin}(n) \in \tau(R)$ to $\tau^*(x_n)$, which is obviously contained in $\tau^*(\mathcal{S}')$.

Thus, we can apply Theorem 40 to show that $\mathrm{SAT}(\tau(\mathcal{R})))$ is counting reducible to $\mathrm{SAT}(\tau^*(\mathcal{S}'))$. Since $\tau^*(S')$ is a subset of $\tau^*(\mathcal{S}[(x_i)_\infty]$ it follows that $\mathrm{SAT}(\tau(\mathcal{S}')))$ is counting reducible to $\mathrm{SAT}(\tau^*(\mathcal{S}[(x_i)_\infty]))$. Since counting reducibility is transitive, we are done. □

C.3.2 Impossibility Results for Polynomial Semirings
↪ Theorems 42, 43, 45 and 46

Theorem 42. *Let $\mathcal{R} = \mathbb{N}[(x_i)_\infty]$ (resp. $\mathcal{R} = \mathbb{B}[(x_i)_\infty]$). If there is an encoding function τ for $\mathcal{R}$ s.t.*

1) *$\|[\![\alpha]\!]_\mathcal{R}(\emptyset)\|_\tau$ is polynomial in the size of α,*
2) *we can extract the binary representation of the coefficient $n \in \mathbb{N}$ (resp. $b \in \mathbb{B}$) of $x_{i_1}^{j_1}...x_{i_n}^{j_n}$ from $\tau(r)$ in time polynomial in $\|r\|_\tau$, and*
3) *$\|x_i\|_\tau$ is polynomial in i,*

then $\#\mathrm{P} \subseteq \mathrm{FP/poly}$ (resp. $\mathrm{NP} \subseteq \mathrm{P/poly}$).

For the proof we use the following theorem due to Cadoli et al. (1996).

Theorem C.14. *Let Π be an NP-complete problem, and let $\Pi = \bigcup_{n\in\mathbb{N}} \Pi_n$, where $\Pi_n = \{\pi \in \Pi \mid |\pi| = n\}$. Moreover, let $[P, F, V]$ be a problem we want to compile, divided into fixed part F and varying part V. Suppose that there exists a polynomial p such that, for each $n > 0$, there exists an $f_n \in F$ with the following properties:*

1. *$|f_n| < p(n)$;*
2. *for all $\pi \in \Pi_n$, there exists a $v_\pi \in V$ such that:*
   (a) *v_π can be computed from π in polynomial time;*
   (b) *$\langle f_n, v_\pi\rangle$ is a "yes" instance of P iff π is a "yes" instance of Π.*

With the above hypothesis, if $[P, F, V]$ is compilable, then NP $\subseteq$ P/poly.

The proof can be found in (Cadoli et al., 1996) and can be extended to a proof for an analogous statement resulting in $\#\mathrm{P} \subseteq \mathrm{FP/poly}$.

Proof of Theorem 42. We use Theorem C.14 with the following $[P, F, V]$: P, the problem we consider, is checking for a monomial $V = x_{i_1}^{j_1}...x_{i_n}^{j_n}$ whether the corresponding coefficient in $[\![\alpha]\!]_{\mathbb{N}[(x_i)_\infty]}(\emptyset)$ is unequal to zero. The fixed part F is given by the ΣBF α over the semiring $\tau(\mathbb{N}[(x_i)_\infty])$. We assume the encoding τ of the polynomials is one that satisfies the precondition of Theorem 42.

We want to show that $[P, F, V]$ can be used to solve a #P-complete problem Π as in the precondition of Theorem C.14. As Π we choose #3SAT. We know that when for $\pi \in \Pi$ it holds that $|\pi| \leq n$ then π can contain at most n different variables. W.l.o.g. we can assume that the variables that are used are $a_1, \ldots, a_n$. Let $C_1, \ldots, C_k$ be all the three literal clauses constructible from $a_1, \ldots, a_n$; clearly k is polynomial in n.

We choose $f_n = \alpha_n$.

$$\alpha_n = \Sigma c_1 \ldots \Sigma c_k \Sigma a_1 \ldots \Sigma a_n \Pi_{i=1}^k (C_i' * c_i * \tau(x_i) + \neg c_i)$$

where $C_i = l_1 \vee l_2 \vee l_3$ is replaced by $C_i' = l_1 + \neg l_1 * l_2 + \neg l_1 * \neg l_2 * l_3$. Since k is polynomial in n and $\|x_i\|_\tau$ is polynomial in i, we know that α_n and therefore f_n is of polynomial size in n.

Given $\pi \in \Pi$ with $|\pi_n|$ we know that all the clauses are three literal clauses using only the variables $a_1, \ldots, a_n$. W.l.o.g. we can assume that every variable is used at least once, otherwise we can simply add $a_i \vee a_i \vee a_i$ for each variable that is not used without changing the number of satisfying assignments. Therefore, the set $C(\pi)$ of clauses in π is a subset of the clauses $C_1, \ldots, C_k$, i.e., $C(\pi) = \{C_{i_1}, \ldots, C_{i_m}\}$ where $1 \leq i_j < i_{j+1} \leq k$ for $j = 1, \ldots, m-1$. Then the coefficient of $x_{i_1} \cdot \cdots \cdot x_{i_m}$ in $[\![\alpha_n]\!]_{\mathbb{N}[(x_i)_\infty]}(\emptyset)$, which is

$$[\![\Sigma a_1 \ldots \Sigma a_n \Pi_{j=1}^m C'_{i_j}]\!]_{\mathbb{N}[(x_i)_\infty]}(\emptyset),$$

is equal to the number of satisfying assignments of π. Since we can extract the binary representation of the coefficient in polynomial time from the encoded value $\tau([\![\alpha_n]\!]_{\mathbb{N}[(x_i)_\infty]}(\emptyset))$, this would imply that $\#\mathrm{P} \subseteq \mathrm{FP/poly}$.

For $\mathbb{B}[(x_i)_\infty]$ we use the same setting, except that we use 3SAT and then the coefficient of $x_{i_1} \cdot \cdots \cdot x_{i_m}$ is equal to whether the 3SAT-instance is satisfiable. □

Theorem 43. *Let $\mathcal{R} = \mathbb{N}[x]$ (resp. $\mathcal{R} = \mathbb{B}[x]$). If there is an encoding function τ for $\mathcal{R}$ s.t.*

1) *$\|[\![\alpha]\!]_{\mathcal{R}}(\emptyset)\|_\tau$ is polynomial in the size of α,*
2) *we can extract the binary representation of the coefficient $n \in \mathbb{N}$ (resp. $b \in \mathbb{B}$) of x^i from $\tau(r)$ in time polynomial in $\|r\|_\tau$, and*
3) *$\|x^i\|_\tau$ is polynomial in $\log_2(i)$,*

then $\#\mathrm{P} \subseteq \mathrm{FP/poly}$ (resp. $\mathrm{NP} \subseteq \mathrm{P/poly}$).

Proof. We proceed as in the proof of Theorem 42, this time however we use $f_n = \alpha_n$ where

$$\begin{aligned} \alpha_n =& \Sigma c_1 \ldots \Sigma c_k \Sigma a_1 \ldots \Sigma a_n \Pi_{i=1}^k C'_i * c_i * \tau(x^{2^i}) + \neg c_i, \\ v_n =& x^{2^{i_1} + \cdots + 2^{i_m}}. \end{aligned}$$

□

Theorem 44. *If $\#\mathrm{P} \subseteq \mathrm{FP/poly}$ (resp. $\mathrm{NP} \subseteq \mathrm{P/poly}$), then there exist encodings τ_∞ and τ_1 for $\mathbb{N}[(x_i)_\infty]$ and $\mathbb{N}[x]$ (resp. $\mathbb{B}[(x_i)_\infty]$ and $\mathbb{B}[x]$) such that the preconditions 1) - 3) of Theorems 42 and 43 are satisfied.*

Proof (sketch). For simplicity, we only consider the case where $\#\mathrm{P} \subseteq \mathrm{FP/poly}$ and argue for the existence of an encoding τ_∞ for $\mathbb{N}[(x_i)_\infty]$ that satisfies the preconditions of Theorem 42. The proofs for the other cases use similar ideas.

The idea is as follows: First, we choose a basic encoding τ of $\mathbb{N}[(x_i)_\infty]$, where (i) coefficients are encoded in binary, (ii) exponents are encoded in binary, (iii) variables indices are encoded in unary, and (iv) monomials are encoded as tuples consisting of their coefficient plus a list of the variables together with their exponents. Finally, (v) polynomials are encoded as lists of their monomials with non-zero coefficients.

Clearly, this encoding does not satisfy condition 1). However, importantly it does satisfy condition 3). Before we continue, recall that given a $\mathrm{SAT}(\tau(\mathbb{N}[(x_i)_\infty]))$-instance α and a monomial $\tau(x_{i_1}^{j_1} \ldots x_{i_n}^{j_n})$, it is possible to compute the coefficient of the monomial in $[\![\alpha]\!]_{\tau(\mathbb{N}[(x_i)_\infty])}(\emptyset)$ in $\#\mathrm{P}$. Assuming that $\#\mathrm{P} \subseteq \mathrm{FP/poly}$, it is also possible in $\mathrm{FP/poly}$. Then,

let A be an advice oracle that returns the polynomial size advice string $A(n)$ that can be used to solve any such coefficient query of input size n. Now we consider the representation of a polynomial p in $\mathbb{N}[(x_i)_\infty]$ as

$$(\beta, A(|(\beta, maxmon(\beta))|), A(|(\beta, maxmon(\beta))| - 1), \ldots, A(0)),$$

where

1. β is a $\mathrm{SAT}(\tau(\mathbb{N}[(x_i)_\infty]))$-instance such that $[\![\beta]\!]_{\tau(\mathbb{N}[(x_i)_\infty])}(\emptyset) = \tau(p)$,
2. $maxmon(\beta)$ is the monomial of maximum size encoding that has a non-zero coefficient in $[\![\beta]\!]_{\mathbb{N}[(x_i)_\infty]}(\emptyset)$, and
3. $|(\beta, maxmon(\beta))|$ denotes the size of the string that encodes the pair.

The size of the representation is polynomial in $\|\beta\|_\tau$, since

- $maxmon(\beta)$ is polynomial in $\|\beta\|_\tau$
- $|A(n)|$ is polynomial in n and, therefore,
- $|A(n)|$ is polynomial in $\|\beta\|_\tau$ for $n \leq |(\beta, maxmon(\beta))|$ and, therefore, also
- all $|(\beta, maxmon(\beta))|$ advice strings $A(i)$ concatenated are polynomial in $\|\beta\|_\tau$.

Consequently, there is a representation of the solution $[\![\alpha]\!]_{\tau(\mathbb{N}[(x_i)_\infty])}(\emptyset)$ of any instance α of $\mathrm{SAT}(\tau(\mathbb{N}[(x_i)_\infty]))$ whose size is polynomial in the size of α. Namely we can simply take

$$(\alpha, A(|(\alpha, maxmon(\alpha))|), A(|(\alpha, maxmon(\alpha))| - 1), \ldots, A(0)).$$

Furthermore, given this representation of $[\![\alpha]\!]_{\tau(\mathbb{N}[(x_i)_\infty])}(\emptyset)$, we can obtain the binary representation of any coefficient of any monomial in polynomial time in the size of the representation, due to the presence of the advice strings.

Last but not least, for x_i, the representation

$$(\tau(x_i), A(|(\tau(x_i), \tau(x_i))|), \ldots, A(0))$$

is polynomial in i, since $\|x_i\|_\tau$ is polynomial in i. Thus, we have a representation of the polynomials that satisfies all three conditions. However, we need an encoding and therefore cannot allow multiple representations of the same value but we must choose exactly one. This is easily fixed though, by choosing that representation of p that has the shortest ΣBF formula α, breaking ties using the lexicographical ordering. We can verify that this only makes representations smaller and use this encoding as τ_∞. □

Theorem 45. *Let $\mathcal{R} \neq \mathbb{T}$ be a commutative semiring. If there is an encoding function τ for $\mathcal{R}[(x_i)_\infty]$ s.t.*

1) $\|[\![\alpha]\!]_{\mathcal{R}[(x_i)_\infty]}(\emptyset)\|_\tau$ is polynomial in the size of α,

2) we can extract the encoding $\tau(r') \in \tau(R)$ of the coefficient of $x_{i_1}^{j_1} \ldots x_{i_n}^{j_n}$ from $\tau(r)$ in time polynomial in $\|r\|_\tau$, and

3) $\|x_i\|_\tau$ *is polynomial in* i*,*

then either NP $\subseteq$ P/poly *or* $\text{Mod}_p\text{P} \subseteq$ P/poly *for some* $p \in \mathbb{N}$.

Theorem 46. *Let* $\mathcal{R} \neq \mathbb{T}$ *be a commutative semiring. If there is an encoding function* τ *for* $\mathcal{R}[x]$ *s.t.*

1) $\|[\![\alpha]\!]_{\mathcal{R}[x]}(\emptyset)\|_\tau$ *is polynomial in the size of* α*,*

2) we can extract the encoding $\tau(r') \in \tau(R)$ *of the coefficient of* x^i *from* $\tau(r)$ *in time polynomial in* $\|r\|_\tau$*, and*

3) $\|x^i\|_\tau$ *is polynomial in* $\log_2(i)$*,*

then either NP $\subseteq$ P/poly *or* $\text{Mod}_p\text{P} \subseteq$ P/poly *for some* $p \in \mathbb{N}$.

Proof of Theorem 45 and 46. As in the proof of Theorem 34, we use that for any non-trivial commutative semiring $\tau(\mathcal{R})$ it holds that either

1. $k \cdot \tau(e_\otimes) = \tau(e_\oplus)$ implies $k = 0$ or
2. $\tau(\langle e_\otimes \rangle) \equiv \mathbb{Z}_p$ for some $p \in \mathbb{N}$.

In the first case we can derive NP $\subseteq$ P/poly, in the second $\text{Mod}_p\text{P} \subseteq$ P/poly from the preconditions of the theorems.

To prove this, we can reuse the ΣBFs from the proofs of Theorem 42 and Theorem 43 for Theorem 45 and 46, respectively. The size restrictions are given as before. Apart from that, we only need to use that Mod_p3CNF is Mod_pP-complete.

Then, we can see as before that we can read off the solution of any (Mod_p)3CNF-instance of size at most n as a coefficient of $[\![\alpha_n]\!]_{\tau(\mathcal{R}[(x_i)_\infty)}(\emptyset)$. Here, we note again, that as in the proof of Theorem 34 we only need to recognize a fixed set of values - which is possible in constant time - to derive the result of the (Mod_p)3CNF-instance from the coefficient. □

Lemma 47. *If* $\text{Mod}_p\text{P} \subseteq$ P/poly *for* $p \in \mathbb{N}, p > 1$*, then* NP $\subseteq$ P/poly.

Proof. Valiant and Vazirani (1986) showed that NP $\subseteq \text{RP}^{\text{Unique-SAT}}$. Here, Unique-SAT is the problem of determining whether a propositional formula ϕ has a satisfying assignment, given the additional knowledge that ϕ has at most one satisfying assignment. Furthermore, recall that RP is the class of languages L for which there exists a polynomial time non-deterministic Turing machine M such that if $x \in L$ then at least half of the computation paths of $M(x)$ accept and if $x \notin L$ then none of the computation paths accept. Intuitively, the definition of RP is more restrictive than that of BPP because if a computation path of the RP-machine M that recognizes L accepts on x, we *know* that $x \in L$, whereas for a BPP-machine, this means that $x \in L$ is likely but not necessarily the case. Indeed, RP $\subseteq$ BPP, since if L is in RP due to NTM M, then we can prove it is also in BPP by executing M twice independently on a given input x.

Using these insights, it follows that NP $\subseteq \text{RP}^{\text{Unique-SAT}} \subseteq \text{BPP}^{\text{Unique-SAT}}$. Furthermore, clearly Unique-SAT is in Mod_pP for any $p \in \mathbb{N}, p > 1$. Thus, we can replace the Unique-SAT-oracle with a Mod_pP-oracle and obtain NP $\subseteq \text{BPP}^{\text{Mod}_p\text{P}}$.

It remains to show that BPP $^{\text{MOD}_p\text{P}} \subseteq$ P/poly follows from $\text{MOD}_p\text{P} \subseteq$ P/poly. We already know that BPP $\subseteq$ P/poly (Bennett & Gill, 1981) with an argument based on machine simulation. If $\text{MOD}_p\text{P} \subseteq$ P/poly holds, we then obtain BPP $^{\text{MOD}_p\text{P}} \subseteq \text{P}^{\text{P/poly}}$/poly. To prove the result, it thus remains to show that $\text{P}^{\text{P/poly}}$/poly $\subseteq$ P/poly.

Consider a language L that is in $\text{P}^{\text{P/poly}}$/poly and solved by a polynomial time machines M with advice oracle A_M, where M uses a further oracle that is evaluated by a polynomial time machine O that uses and advice oracle A_O. Given that $\text{P}^{\text{P}} = \text{P}$, we can combine M and O into a single polynomial time machine M_C and only need to make sure that we can supply it with the advice from both A_M and A_O. The latter is, however, easily achieved. First note that since M takes polynomial time the queries posed to O are all of polynomial size $p(n)$ in the size n of the input to M. Thus, given input x of size n, we can use the following polynomial size advice for M_C:

$$(A_M(n), A_O(p(n)), A_O(p(n)-1), \ldots, A_O(0)).$$

This is still polynomial in n and we have both the necessary advice for the part of M and of O. □

C.3.3 Possibility Results for $\mathcal{R}[(x_i)_k]$
↪ Theorem 48

Theorem 48. *Let $\tau(\mathcal{R})$ be a commutative semiring that is efficiently encoded. Then $\text{SAT}(\tau(\mathcal{R}[(x_i)_k]))$ is $\text{FP}_{\parallel}^{\text{NP}(\tau(\mathcal{R}))}$-complete for metric reductions, if we extend τ to $\mathcal{R}[(x_i)_k]$ by representing polynomials as lists of monomials with exponents in unary and coefficients encoded by τ.*

Proof. Let α be some ΣBF over $\tau(\mathcal{R}[(x_i)_k])$. We can bound the number of monomials in $[\![\alpha]\!]_{\tau(\mathcal{R}[(x_i)_k])}(\emptyset)$ by the monomials that occur in α. Let n be the maximum exponent e_i occurring in the monomials $x_1^{e_1} \ldots x_k^{e_k}$ in α that have a nonzero coefficient. Furthermore, let m be the number of monomials in α with nonzero coefficients. Then we know that for all monomials $x_1^{e_1} \ldots x_k^{e_k}$ in $[\![\alpha]\!]_{\tau(\mathcal{R}[(x_i)_k])}(\emptyset)$ with nonzero coefficients it holds that $e_i \leq nm$ since every product can only have m factors, where each factor has at most exponent n. Therefore, the number of monomials with nonzero coefficients in $[\![\alpha]\!]_{\tau(\mathcal{R}[(x_i)_k])}(\emptyset)$ is bounded by $(nm)^k$. Since k is a constant and n and m are polynomial in the size of the input since the exponents are encoded in unary, we know that there are at most polynomially many monomials in $[\![\alpha]\!]_{\tau(\mathcal{R}[(x_i)_k])}(\emptyset)$ with nonzero coefficients. Finding each of their coefficients is possible with a call to a $\text{NP}(\tau(\mathcal{R}))$-oracle that is independent of the other calls. This shows that $\text{SAT}(\tau(\mathcal{R}[(x_i)_k]))$ is in $\text{FP}_{\parallel}^{\text{NP}(\tau(\mathcal{R}))}$.

For $\text{FP}_{\parallel}^{\text{NP}(\tau(\mathcal{R}))}$-hardness, it is sufficient to show that we can obtain the answers to polynomially many $\text{NP}(\tau(\mathcal{R}))$-queries by a metric reduction to solving one $\text{SAT}(\tau(\mathcal{R}[(x_i)_k]))$-instance. Since $\text{SAT}(\tau(\mathcal{R}))$ is $\text{NP}(\tau(\mathcal{R}))$-complete, we may assume that we have polynomially many $\text{SAT}(\tau(\mathcal{R}))$-instances $\alpha_1, \ldots, \alpha_n$ and moreover that each α_i in prefix normal form with the same quantifier prefix $\Sigma v_1 \ldots \Sigma v_m$. We then construct the following $\text{SAT}(\tau(\mathcal{R}[(x_i)_k]))$-instance β:

$$\Sigma v_1 \ldots \Sigma v_m \, \alpha_1 * \tau(x_1) + \ldots + \alpha_n * \tau(x_1^n).$$

The coefficient of x_1^i is then the solution of α_i. As β is constructible in time polynomial in the size of $\alpha_1, \ldots, \alpha_n$, we have the desired metic reduction, which opmpletes the proof. □

C.3.4 Possibility Results for Commutative Finitely Generated Semirings ↪ Theorem 51

Theorem 51. *Let $\tau(\mathcal{R})$ be an efficiently encoded commutative semiring that is generated by $\{r_1, \ldots, r_k\}$. Suppose every $r \in R$ is of the form $r = \bigoplus_{i=1}^{n} a_i \cdot \bigotimes_{j=1}^{k} r_j^{e_{i,j}}$ for some $a_i, e_{i,j} \in \mathbb{N}$ such that*

- *$\max\{e_{i,j}, \log_2(a_i), n\}$ is polynomial in $\|r\|_\tau$, and*
- *we can obtain $a_i, e_{i,j}$ from $\tau(r)$ in polynomial time.*

If $\tau(\mathcal{R})$ is

1. *periodic with periodicity 1;*
2. *periodic with periodicity $p \geq 2$ and offset 0;*
3. *periodic with periodicity $p \geq 2$ and offset $o > 0$;*
4. *not periodic;*

then $\mathrm{SAT}(\tau(\mathcal{R}))$ is in

1. $\mathrm{FP}_{\|}^{\mathrm{NP}}$;
2. $\mathrm{FP}_{\|}^{\mathrm{Mod}_p\mathrm{P}}$;
3. $\mathrm{FP}_{\|}^{\mathrm{Mod}_p\mathrm{P} \cup \mathrm{NP}}$;
4. $\mathrm{FP}_{\|}^{\#\mathrm{P}}$;

respectively.

Proof. Let α a ΣBF over $\tau(\mathcal{R})$. We can transform α into a ΣBF β over $\mathrm{bin}(\mathcal{S}[(x_i)_k])$, where the binary encoding of the natural numbers bin is extended to the polynomials by representing polynomials as lists of monomials with exponents in unary and coefficients encoded by bin and $\mathcal{S}$ is

1. $\mathbb{N}_{\leq o}$,
2. $\mathbb{Z}_p$,
3. $\mathbb{Z}_p \times \mathbb{N}_{\leq o}$,
4. $\mathbb{N}$

if $\tau(\mathcal{R})$ is, respectively,

1. periodic with periodicity 1 and offset o

2. periodic with periodicity $p \geq 2$ and offset 0
3. periodic with periodicity $p \geq 2$ and offset $o > 0$
4. not periodic.

The transformation works by replacing every value $\tau(r) \in \tau(R)$ that occurs in α by

$$\mathrm{bin}\left(\sum_{i=1}^{n} a_i \prod_{j=i}^{m_i} x_j^{e_{i,j}}\right)$$

when

$$\tau(r) = \bigoplus_{i=1}^{n} a_i \bigotimes_{j=i}^{m_i} \tau(r_j)^{e_{i,j}}.$$

Recall that when $k = 0$, i.e., $\tau(\mathcal{R})$ is already generated by the empty set, these equations collapse to

$$\mathrm{bin}\left(\sum_{i=1}^{n} a_i \cdot 1\right)$$

and

$$\tau(r) = \bigoplus_{i=1}^{n} a_i \cdot e_{\otimes}$$

since we take the neutral element of multiplication as the value of an empty product.

During the above transformation, we only need to check whether a_i is in $\mathcal{S}$ and replace a_i by an equivalent value if not. Namely, if $\mathcal{S} = \mathbb{N}_{\leq o}$, we use the minimum of a_i and o, if $\mathcal{S} = \mathbb{Z}_p$, we use the unique integer n in $\{0, \ldots, p-1\}$ such that $a_i \equiv n \mod p$, if $\mathcal{S} = \mathbb{Z}_p \times \mathbb{N}_{\leq o}$, we use (n, m), where n is the unique integer in $\{0, \ldots, p-1\}$ such that $a_i \equiv n \mod p$ and m is the minimum of a_i and o, otherwise, $\mathcal{S} = \mathbb{N}$ in which case we know that $a_i \in \mathbb{N}$. We can obtain the above expression in polynomial time. Hence, the size of β is also polynomial in the input.

Theorem 48 tells us, that $\mathrm{SAT}(\mathrm{bin}(\mathcal{S}[(x_i)_k]))$ satisfies the desired complexity assertion. Furthermore, given $[\![\beta]\!]_{\mathrm{bin}(\mathcal{S}[(x_i)_k])}(\emptyset)$ we can compute $[\![\alpha]\!]_{\tau(\mathcal{R})}(\emptyset)$ in polynomial time. This can be seen as follows. Let

$$[\![\beta]\!]_{\mathcal{S}[(x_i)_k]}(\emptyset) = \sum_{i_1,\ldots,i_k=1}^{n} a_{\bar{i}} \prod_{j=1}^{k} x_j^{e_{\bar{i},j}}.$$

Then we know that

$$[\![\alpha]\!]_{\mathcal{R}}(\emptyset) = \bigoplus_{i_1,\ldots,i_k=1}^{n} \bigoplus_{v=1}^{a_{\bar{i}}} \bigotimes_{j=1}^{k} r_j^{e_{\bar{i},j}}.$$

Since the size of the representation of $[\![\beta]\!]_{\mathrm{bin}(\mathcal{S}[(x_i)_k])}(\emptyset)$ is polynomial, also the number of monomials in $[\![\beta]\!]_{\mathrm{bin}(\mathcal{S}[(x_i)_k])}(\emptyset)$ is polynomial. Therefore, we can calculate the values of all the "monomials" $\tau(\bigotimes_{j=1}^{k} r_j^{e_{\bar{i},j}})$ in polynomial time.

To compute

$$\tau(\bigoplus_{v=1}^{a_{\bar{i}}} \bigotimes_{j=1}^{k} r_j^{e_{\bar{i},j}})$$

first consider the binary representation $c_{\lceil \log_2(a_{\bar{i}}) \rceil} \ldots c_0 \in \{0,1\}^*$ of $a_{\bar{i}}$. We know that

$$\tau(\bigoplus_{v=1}^{a_{\bar{i}}} \bigotimes_{j=1}^{k} r_j^{e_{\bar{i},j}}) = \bigoplus_{t=0}^{\lceil \log_2(a_{\bar{i}}) \rceil} c_t \cdot \tau(\bigoplus_{v=1}^{2^t} \bigotimes_{j=1}^{k} r_j^{e_{\bar{i},j}}). \tag{4}$$

Thus, we can proceed by first iteratively computing

$$\tau(\bigoplus_{v=1}^{2^t}\bigotimes_{j=1}^{k} r_j^{e_{\bar{i},j}}) = \tau(\bigoplus_{v=1}^{2^{t-1}}\bigotimes_{j=1}^{k} r_j^{e_{\bar{i},j}}) \oplus \tau(\bigoplus_{v=1}^{2^{t-1}}\bigotimes_{j=1}^{k} r_j^{e_{\bar{i},j}})$$

for $t = 1, \dots, \lceil \log_2(a_{\bar{i}}) \rceil$. This is possible in $\lceil \log_2(a_{\bar{i}}) \rceil$ steps, where each of them takes polynomial time and leads to a result that is polynomial in size, since $\tau(\mathcal{R})$ is efficiently encoded.

After having computed these values, we can plug them into Equation (4) and evaluate it. This is again possible in polynomial time, and leads to a result that is polynomial in size, since $\tau(\mathcal{R})$ is efficiently encoded. For the same reason, summing up all the values $\tau(\bigoplus_{v=1}^{a_{\bar{i}}}\bigotimes_{j=1}^{k} r_j^{e_{\bar{i},j}})$ over $\bar{i}$ is also possible in polynomial time and leads to a polynomial-size result.

From metric-reducibility to $\mathrm{SAT}(\mathcal{S}[(x_i)_k])$ and Theorem 48, it follows that $\mathrm{SAT}(\tau(\mathcal{R}))$ is in $\mathrm{FP}_{\|}^{\mathrm{FP}^{\mathcal{C}}}$. The latter class is equal to $\mathrm{FP}_{\|}^{\mathcal{C}}$, which complete the proof. □

C.4 Derived Results
↪ Theorem 52

Theorem 52. *Let $\mathbb{S} = \mathbb{N}, \mathbb{Z}, \mathbb{Q}$. For $\mathbb{S}^n$, $n \in \mathbb{N}$, the semiring $\mathbb{S}$ over multiple dimensions, we have that $\mathrm{SAT}(\mathrm{bin}(\mathbb{S})^n)$ is $\mathrm{FP}_{\|}^{\#\mathrm{P}}$-complete with respect to metric reductions, where $\mathrm{bin}(\mathbb{Q})$ represents $r \in \mathbb{Q}$ as pair $(\mathrm{bin}(p), \mathrm{bin}(q))$ such that $p/q = r$, $p \in \mathbb{Z}$, $q \in \mathbb{N}$ and the greatest common divisor of $|p|$ and q is 1.*

Proof. For $\mathbb{S} = \mathbb{N}$ and $\mathbb{S} = \mathbb{Z}$ we see that $\mathbb{S}^n$ is finitely generated by $\{e_1, \dots, e_n\}$ and $\{e_1, \dots, e_n, (-1, \dots, -1)\}$, respectively, where e_i is the vector whose j^{th} entry is 0 if $j \neq i$ and 1 if $i = j$. Furthermore, every number $m = (m_1, \dots, m_n) \in \mathbb{S}$ can be represented as a sum

$$m = \sum_{i=1}^{n}\sum_{j=1}^{|m_i|} \begin{cases} e_i & \text{if } m_i \geq 0, \\ (-1, \dots, -1)e_i & \text{if } m_i < 0. \end{cases}$$

such that $\log_2(m_i)$ is polynomial in $\|m\|_{\mathrm{bin}}$. Therefore, the preconditions of Theorem 51 are satisfied and we obtain that $\mathrm{SAT}(\mathrm{bin}(\mathbb{S})^n)$ is in $\mathrm{FP}_{\|}^{\#\mathrm{P}}$.

For $\mathbb{S} = \mathbb{Q}$, this strategy does not apply completely analogously since $\mathbb{Q}$ is not finitely generated. However, we can apply a very similar strategy as follows.

Consider a $\mathrm{SAT}(\mathrm{bin}(\mathbb{Q})^n)$-instance α. Let $\mathrm{bin}(p_1/q_1), \dots, \mathrm{bin}(p_m/q_m) \in \mathrm{bin}(\mathbb{Q})$ be the entries of the weight vectors in α. Furthermore, let l be the least common multiple of $q_1, \dots, q_m$. Then we can replace these values by polynomials from $\mathrm{bin}(\mathbb{N}[(x_i)_2])$ by using

$$\mathrm{poly}(p_i, q_i) = \begin{cases} \mathrm{bin}(p_i l / q_i x_2) & \text{if } p_i \geq 0, \\ \mathrm{bin}(|p_i| l / q_i x_1 x_2) & \text{if } p_i < 0. \end{cases}$$

for $\mathrm{bin}(p_i/q_i)$. Note that when we replace x_1 by -1 and x_2 by $1/l$, we get the original value again. The same still holds after addition and multiplication, that is, $f : \mathrm{bin}(\mathbb{N}[(x_i)_2]) \to \mathrm{bin}(\mathbb{Q}), p(x_1, x_2) \mapsto p(-1, l)$ is a homomorphism. In fact, when we restrict the codomain of f to $\mathrm{bin}(\langle 1/l, -1 \rangle)$, it is even an epimorphism, which is moreover computable in polynomial time.

Thus, we can metrically reduce $\mathrm{SAT}(\mathrm{bin}(\mathbb{Q})^n)$ to $\mathrm{SAT}(\mathrm{bin}(\mathbb{N}[(x_i)_2])^n)$. However, the semiring $\mathrm{bin}(\mathbb{N}[(x_i)_2])^n$ is finitely generated, by $\{e_i, e_i x_1, e_i x_2 \mid i = 1, \ldots, n\}$ and satisfies the preconditions of Theorem 51. Thus, $\mathrm{SAT}(\mathrm{bin}(\mathbb{N}[(x_i)_2)^n)$ and consequently also $\mathrm{SAT}(\mathrm{bin}(\mathbb{Q})^n)$ is in $\mathrm{FP}_{\|}^{\#\mathrm{P}}$.

Regarding hardness, note that already $\mathrm{SAT}(\mathrm{bin}(\mathbb{N}))$ is $\mathrm{FP}_{\|}^{\#\mathrm{P}}$-hard with respect to metric reductions: Given n $\mathrm{SAT}(\mathrm{bin}(\mathbb{N}))$-instances $I_1, \ldots, I_n$ we can compute the value of the $\mathrm{SAT}(\mathrm{bin}(\mathbb{N}))$-instance $\alpha_n = I_1 * \mathrm{bin}(2^k) + \ldots + I_n * \mathrm{bin}(2^{k*n})$, where k is such that $2^k > [\![I_i]\!]_{\mathrm{bin}(\mathbb{N})}(\emptyset)$ for all i. Such a k can be found and is polynomial in the size of the instances. Due to the choice of k we can understand the result of α_n as a list of the results of I_i in binary representation. From the $\mathrm{FP}_{\|}^{\#\mathrm{P}}$-hardness it immediately follows, that $\mathrm{SAT}(\mathrm{bin}(\mathbb{S})^n)$ is even $\mathrm{FP}_{\|}^{\#\mathrm{P}}$-complete with respect to metric reductions for $\mathbb{S} = \mathbb{N}, \mathbb{Z}, \mathbb{Q}$. □

References

Abiteboul, S., & Vianu, V. (1991). Generic computation and its complexity. In Koutsougeras, C., & Vitter, J. S. (Eds.), *Proc. 23rd Annual ACM Symposium on Theory of Computing, May 5-8, 1991, New Orleans, Louisiana, USA*, pp. 209–219. ACM.

Abiteboul, S., & Vianu, V. (1995). Computing with first-order logic. *J. Comput. Syst. Sci.*, *50*(2), 309–335.

Allender, E., & Wagner, K. W. (1993). Counting hierarchies: Polynomial time and constant depth circuits. In Rozenberg, G., & Salomaa, A. (Eds.), *Current Trends in Theoretical Computer Science - Essays and Tutorials*, Vol. 40 of *World Scientific Series in Computer Science*, pp. 469–483. World Scientific.

Arenas, M., Muñoz, M., & Riveros, C. (2020). Descriptive complexity for counting complexity classes. *Log. Methods Comput. Sci.*, *16*(1).

Bacchus, F., Dalmao, S., & Pitassi, T. (2009). Solving #sat and bayesian inference with backtracking search. *J. Artif. Intell. Res.*, *34*, 391–442.

Baral, C., Gelfond, M., & Rushton, J. N. (2009). Probabilistic reasoning with answer sets. *Theory Pract. Log. Program.*, *9*(1), 57–144.

Beigel, R., & Gill, J. (1992). Counting classes: Thresholds, parity, mods, and fewness. *Theor. Comput. Sci.*, *103*(1), 3–23.

Belle, V., & De Raedt, L. (2020). Semiring programming: A semantic framework for generalized sum product problems. *Int. J. Approx. Reason.*, *126*, 181–201.

Bennett, C. H., & Gill, J. (1981). Relative to a random oracle A, P^{A} != NP^{A} != co-NP^{A} with probability 1. *SIAM Journal on Computing*, *10*(1), 96–113.

Bistarelli, S., Montanari, U., Rossi, F., Schiex, T., Verfaillie, G., & Fargier, H. (1999). Semiring-based csps and valued csps: Frameworks, properties, and comparison. *Constraints An Int. J.*, *4*(3), 199–240.

Blum, L. (1998). *Complexity and real computation*. Springer.

Blum, L., Shub, M., & Smale, S. (1989). On a theory of computation and complexity over the real numbers: NP- completeness, recursive functions and universal machines. *Bulletin (New Series) of the American Mathematical Society*, *21*(1), 1 – 46.

Brailsford, S. C., Potts, C. N., & Smith, B. M. (1999). Constraint satisfaction problems: Algorithms and applications. *Eur. J. Oper. Res.*, *119*(3), 557–581.

Brewka, G., Delgrande, J. P., Romero, J., & Schaub, T. (2015). asprin: Customizing answer set preferences without a headache. In Bonet, B., & Koenig, S. (Eds.), *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, January 25-30, 2015, Austin, Texas, USA*, pp. 1467–1474. AAAI Press.

Büchi, J. R. (1960). Weak second-order arithmetic and finite automata. *Mathematical Logic Quarterly*, *6*(1-6), 66–92.

Cadoli, M., Donini, F. M., & Schaerf, M. (1996). Is intractability of nonmonotonic reasoning a real drawback?. *Artif. Intell.*, *88*(1-2), 215–251.

Cook, S. A. (1971). The complexity of theorem-proving procedures. In Harrison, M. A., Banerji, R. B., & Ullman, J. D. (Eds.), *Proc. 3rd Annual ACM Symposium on Theory of Computing, May 3-5, 1971, Shaker Heights, Ohio, USA*, pp. 151–158. ACM.

Cui, Y. (2002). *Lineage tracing in data warehouses*. Ph.D. thesis, Stanford University.

Dantsin, E., Eiter, T., Gottlob, G., & Voronkov, A. (2001). Complexity and expressive power of logic programming. *ACM Comput. Surv.*, *33*(3), 374–425.

Darwiche, A., & Marquis, P. (2002). A knowledge compilation map. *J. Artif. Intell. Res.*, *17*, 229–264.

de Campos, C. P., Stamoulis, G., & Weyland, D. (2020). A structured view on weighted counting with relations to counting, quantum computation and applications. *Inf. Comput.*, *275*, 104627.

De Raedt, L., Kimmig, A., & Toivonen, H. (2007). Problog: A probabilistic prolog and its application in link discovery. In Veloso, M. M. (Ed.), *IJCAI 2007, Proc. 20th International Joint Conference on Artificial Intelligence, Hyderabad, India, January 6-12, 2007*, pp. 2462–2467.

Droste, M., & Gastin, P. (2007). Weighted automata and weighted logics. *Theor. Comput. Sci.*, *380*(1-2), 69–86.

Dudek, J. M., Phan, V., & Vardi, M. Y. (2020). ADDMC: weighted model counting with algebraic decision diagrams. In *Proc. Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020 New York, NY, USA, February 7-12, 2020*, pp. 1468–1476. AAAI Press.

Eisner, J. (2002). Parameter estimation for probabilistic finite-state transducers. In *Proc. 40th Annual Meeting of the Association for Computational Linguistics, July 6-12, 2002, Philadelphia, PA, USA*, pp. 1–8. ACL.

Eisner, J., Goldlust, E., & Smith, N. A. (2005). Compiling comp ling: Weighted dynamic programming and the dyna language. In Proc. Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing, *HLT/EMNLP*

2005, 6-8 October 2005, Vancouver, British Columbia, Canada, pp. 281–290. The Association for Computational Linguistics.

Eiter, T., Hecher, M., & Kiesel, R. (2021). Treewidth-aware cycle breaking for algebraic answer set counting. In Bienvenu, M., Lakemeyer, G., & Erdem, E. (Eds.), *Proc. 18th International Conference on Principles of Knowledge Representation and Reasoning, KR 2021, Online event, November 3-12, 2021*, pp. 269–279.

Eiter, T., & Kiesel, R. (2020). ASP($\mathcal{AC}$): Answer set programming with algebraic constraints. *Theory Pract. Log. Program.*, *20*(6), 895–910.

Eiter, T., & Kiesel, R. (2021). On the complexity of sum-of-products problems over semirings. In *Proc. Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Virtual Event, February 2-9, 2021*, pp. 6304–6311. AAAI Press.

Erdem, E., Gelfond, M., & Leone, N. (2016). Applications of answer set programming. *AI Mag.*, *37*(3), 53–68.

Fagin, R. (1974). Generalized first-order spectra and polynomial-time recognizable sets. *Complexity of computation*, *7*, 43–73.

Fenner, S. A., Fortnow, L., & Kurtz, S. A. (1994). Gap-definable counting classes. *J. Comput. Syst. Sci.*, *48*(1), 116–148.

Friesen, A. L., & Domingos, P. M. (2016). The sum-product theorem: A foundation for learning tractable models. In Balcan, M., & Weinberger, K. Q. (Eds.), *Proc. 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016*, Vol. 48 of *JMLR Workshop and Conference Proceedings*, pp. 1909–1918. JMLR.org.

Ganian, R., Kim, E. J., Slivovsky, F., & Szeider, S. (2022). Sum-of-products with default values: Algorithms and complexity results. *J. Artif. Intell. Res.*, *73*, 535–552.

Gary, M. R., & Johnson, D. S. (1979). Computers and intractability: A guide to the theory of np-completeness..

Gill, J. (1977). Computational complexity of probabilistic turing machines. *SIAM J. Comput.*, *6*(4), 675–695.

Goodman, J. (1999). Semiring parsing. *Comput. Linguistics*, *25*(4), 573–605.

Grädel, E., & Meer, K. (1995). Descriptive complexity theory over the real numbers. In Leighton, F. T., & Borodin, A. (Eds.), *Proc. Twenty-Seventh Annual ACM Symposium on Theory of Computing, 29 May-1 June 1995, Las Vegas, Nevada, USA*, pp. 315–324. ACM.

Green, T. J. (2011). Containment of conjunctive queries on annotated relations. *Theory Comput. Syst.*, *49*(2), 429–459.

Green, T. J., Karvounarakis, G., & Tannen, V. (2007). Provenance semirings. In Libkin, L. (Ed.), *Proc. Twenty-Sixth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, June 11-13, 2007, Beijing, China*, pp. 31–40. ACM.

Green, T. J., & Tannen, V. (2017). The semiring framework for database provenance. In Sallinger, E., Van den Bussche, J., & Geerts, F. (Eds.), *Proc. 36th ACM SIGMOD-*

SIGACT-SIGAI Symposium on Principles of Database Systems, PODS 2017, Chicago, IL, USA, May 14-19, 2017, pp. 93–99. ACM.

Heckerman, D. (2008). A tutorial on learning with bayesian networks. In Holmes, D. E., & Jain, L. C. (Eds.), *Innovations in Bayesian Networks: Theory and Applications*, Vol. 156 of *Studies in Computational Intelligence*, pp. 33–82. Springer.

Hertrampf, U. (1990). Relations among mod-classes. *Theor. Comput. Sci.*, *74*(3), 325–328.

Impagliazzo, R., & Paturi, R. (2001). On the complexity of k-sat. *J. Comput. Syst. Sci.*, *62*(2), 367–375.

Jenner, B., & Torán, J. (1995). Computing functions with parallel queries to NP. *Theor. Comput. Sci.*, *141*(1&2), 175–193.

Karp, R., & Lipton, R. J. (1982). Turing machines that take advice. In *L'Enseign. Math.*, Vol. 28.

Karp, R. M., & Lipton, R. J. (1980). Some connections between nonuniform and uniform complexity classes. In Miller, R. E., Ginsburg, S., Burkhard, W. A., & Lipton, R. J. (Eds.), *Proc. 12th Annual ACM Symposium on Theory of Computing, April 28-30, 1980, Los Angeles, California, USA*, pp. 302–309. ACM.

Kautz, H. A., & Selman, B. (1996). Pushing the envelope: Planning, propositional logic and stochastic search. In Clancey, W. J., & Weld, D. S. (Eds.), *Proc. Thirteenth National Conference on Artificial Intelligence, AAAI 96, Portland, Oregon, USA, August 4-8, 1996, Volume 2*, pp. 1194–1201. AAAI Press / The MIT Press.

Khamis, M. A., Ngo, H. Q., & Rudra, A. (2016). FAQ: questions asked frequently. In Milo, T., & Tan, W. (Eds.), *Proc. 35th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems, PODS 2016, San Francisco, CA, USA, June 26 - July 01, 2016*, pp. 13–28. ACM.

Khardon, R., & Roth, D. (1997). Learning to reason. *J. ACM*, *44*(5), 697–725.

Kiesel, R., Totis, P., & Kimmig, A. (2022). Efficient knowledge compilation beyond weighted model counting. *CoRR*, *abs/2205.07496*.

Kimmig, A., Van den Broeck, G., & De Raedt, L. (2011). An algebraic prolog for reasoning about possible worlds. In Burgard, W., & Roth, D. (Eds.), *Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2011, San Francisco, California, USA, August 7-11, 2011*. AAAI Press.

Kimmig, A., Van den Broeck, G., & De Raedt, L. (2017). Algebraic model counting. *J. Appl. Log.*, *22*, 46–62.

Kohlas, J., & Wilson, N. (2008). Semiring induced valuation algebras: Exact and approximate local computation algorithms. *Artif. Intell.*, *172*(11), 1360–1399.

Krentel, M. W. (1988). The complexity of optimization problems. *J. Comput. Syst. Sci.*, *36*(3), 490–509.

Ladner, R. E. (1989). Polynomial space counting problems. *SIAM J. Comput.*, *18*(6), 1087–1097.

Larrosa, J., Oliveras, A., & Rodríguez-Carbonell, E. (2010). Semiring-induced propositional logic: Definition and basic algorithms. In Clarke, E. M., & Voronkov, A. (Eds.), *Logic for Programming, Artificial Intelligence, and Reasoning - 16th International Conference, LPAR-16, Dakar, Senegal, April 25-May 1, 2010, Revised Selected Papers*, Vol. 6355 of *Lecture Notes in Computer Science*, pp. 332–347. Springer.

Lee, J., & Yang, Z. (2017). Lpmln, weak constraints, and p-log. In Singh, S., & Markovitch, S. (Eds.), *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA*, pp. 1170–1177. AAAI Press.

Li, C. M., & Manyà, F. (2021). Maxsat, hard and soft constraints. In Biere, A., Heule, M., van Maaren, H., & Walsh, T. (Eds.), *Handbook of Satisfiability - Second Edition*, Vol. 336 of *Frontiers in Artificial Intelligence and Applications*, pp. 903–927. IOS Press.

Mandrali, E., & Rahonis, G. (2015). Weighted first-order logics over semirings. *Acta Cybern.*, *22*(2), 435–483.

Manhaeve, R., Dumancic, S., Kimmig, A., Demeester, T., & De Raedt, L. (2019). Deepproblog: Neural probabilistic logic programming. In Beuls, K., et al. (Eds.), *Proc. 31st Benelux Conference on Artificial Intelligence (BNAIC 2019), Brussels, Belgium, November 6-8, 2019*, Vol. 2491 of *CEUR Workshop Proceedings*. CEUR-WS.org.

Meer, K. (2000). Counting problems over the reals. *Theor. Comput. Sci.*, *242*(1-2), 41–58.

Mengel, S. (2021). *Counting, Knowledge Compilation and Applications*.

Niedermayer, I. S. P. D. (2008). An introduction to bayesian networks and their contemporary applications. In Holmes, D. E., & Jain, L. C. (Eds.), *Innovations in Bayesian Networks: Theory and Applications*, Vol. 156 of *Studies in Computational Intelligence*, pp. 117–130. Springer.

Oztok, U., Choi, A., & Darwiche, A. (2016). Solving pp^{pp}-complete problems using knowledge compilation. In Baral, C., Delgrande, J. P., & Wolter, F. (Eds.), *Principles of Knowledge Representation and Reasoning: Proc. Fifteenth International Conference, KR 2016, Cape Town, South Africa, April 25-29, 2016*, pp. 94–103. AAAI Press.

Pearl, J. (1985). Bayesian networks : a model of self-activated memory for evidential reasoning. In *Proc. 7th Conference of the Cognitive Science Society, 1985*, pp. 329–334.

Pearl, J. (2014). Graphical models for probabilistic and causal reasoning. In Gonzalez, T. F., Diaz-Herrera, J., & Tucker, A. (Eds.), *Computing Handbook, Third Edition: Computer Science and Software Engineering*, pp. 44: 1–24. CRC Press.

Ruttkay, Z. (1994). Fuzzy constraint satisfaction. In *Proceedings of 1994 IEEE 3rd International Fuzzy Systems Conference*, Vol. 2, pp. 1263–1268.

Sato, T., & Kameya, Y. (1997). PRISM: A language for symbolic-statistical modeling. In *Proc. Fifteenth International Joint Conference on Artificial Intelligence, IJCAI 97, Nagoya, Japan, August 23-29, 1997, 2 Volumes*, pp. 1330–1339. Morgan Kaufmann.

Senellart, P., Jachiet, L., Maniu, S., & Ramusat, Y. (2018). Provsql: Provenance and probability management in postgresql. *Proc. VLDB Endow.*, *11*(12), 2034–2037.

Sharma, A., & Singh, S. K. (2016). One way functions–conjecture, status, applications and future research scope. *International Journal of Computer Applications*, *153*(8).

Skryagin, A., Stammer, W., Ochs, D., Dhami, D. S., & Kersting, K. (2021). SLASH: embracing probabilistic circuits into neural answer set programming. *CoRR*, *abs/2110.03395*.

Stearns, R. E., & III, H. B. H. (1996). An algebraic model for combinatorial problems. *SIAM J. Comput.*, *25*(2), 448–476.

Toda, S. (1989). On the computational power of PP and +p. In *30th Annual Symposium on Foundations of Computer Science, Research Triangle Park, North Carolina, USA, 30 October - 1 November 1989*, pp. 514–519. IEEE Computer Society.

Tseitin, G. S. (1983). *On the Complexity of Derivation in Propositional Calculus*, p. 466–483. Springer Berlin Heidelberg, Berlin, Heidelberg.

Valiant, L. G. (1979). The complexity of enumeration and reliability problems. *SIAM J. Comput.*, *8*(3), 410–421.

Valiant, L. G., & Vazirani, V. V. (1986). NP is as easy as detecting unique solutions. *Theor. Comput. Sci.*, *47*(3), 85–93.

Van den Berg, B., Van Bremen, T., Derkinderen, V., Kimmig, A., Schrijvers, T., & De Raedt, L. (2021). From probabilistic netkat to problog: New algorithms for inference and learning in probabilistic networks. In *International Conference on Probabilistic Programming, Location: Online*.

Van den Broeck, G., Meert, W., & Darwiche, A. (2014). Skolemization for weighted first-order model counting. In Baral, C., De Giacomo, G., & Eiter, T. (Eds.), *Principles of Knowledge Representation and Reasoning: Proc. Fourteenth International Conference, KR 2014, Vienna, Austria, July 20-24, 2014*. AAAI Press.

Van den Broeck, G., Thon, I., Van Otterlo, M., & De Raedt, L. (2010). Dtproblog: A decision-theoretic probabilistic prolog. In Fox, M., & Poole, D. (Eds.), *Proc. Twenty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2010, Atlanta, Georgia, USA, July 11-15, 2010*. AAAI Press.