Inf. Comput. 310 (2026) 105446

Contents lists available at ScienceDirect

Information and Computation

journal homepage: www.elsevier.com/locate/yinco

Descriptive complexity and weighted Turing machines ⋆

Guillermo Badia [a,*], Manfred Droste [b], Carles Noguera [c], Erik Paul [b]

[a] University of Queensland, St Lucia, Brisbane, QLD 4072, Queensland, Australia
[b] University of Leipzig, Augustusplatz 10, Leipzig, 04109, Germany
[c] University of Siena, Via Roma 56, Siena, 53100, Italy

ARTICLE INFO

Keywords:
Descriptive complexity
Weighted Turing machines
Weighted logics
Semirings

ABSTRACT

Fagin's seminal result characterizing NP in terms of existential second-order logic started the fruitful field of descriptive complexity theory. In recent years, there has been much interest in the investigation of quantitative (weighted) models of computations. In this paper, we start the study of descriptive complexity based on weighted Turing machines over arbitrary semirings. We provide machine-independent characterizations (over ordered structures) of the weighted complexity classes $\mathsf{NP}[\mathcal{S}]$, $\mathsf{L}[\mathcal{S}]$, $\mathsf{FP}[\mathcal{S}]$, $\mathsf{FPLOG}[\mathcal{S}]$, $\mathsf{FPSPACE}[\mathcal{S}]$, and $\mathsf{FPSPACE}_{poly}[\mathcal{S}]$ in terms of definability in suitable weighted logics for an arbitrary semiring $\mathcal{S}$. In particular, we state and prove weighted versions of Fagin's theorem (even for arbitrary structures, not necessarily ordered, provided that the semiring is idempotent and commutative), the Immerman–Vardi's theorem (originally for P) and the Abiteboul–Vianu–Vardi's theorem (originally for PSPACE). We also discuss a recent open problem proposed by Eiter and Kiesel.

Recently, the above mentioned weighted complexity classes have been investigated in connection to classical counting complexity classes. Furthermore, several classical counting complexity classes have been characterized in terms of particular weighted logics over the semiring $\mathbb{N}$ of natural numbers. In this work, we cover several of these classes and obtain new results for others such as NPMV, $\oplus$P, or the collection of real-valued languages realized by nondeterministic polynomial-time real-valued Turing machines. Furthermore, our results apply to classes based on many other important semirings, such as the max-plus and the min-plus semirings over the natural numbers which correspond to the classical classes $\mathsf{MaxP}[O(\log n)]$ and $\mathsf{MinP}[O(\log n)]$, respectively.

1. Introduction

Descriptive complexity is a branch of computational complexity, as well as finite model theory, where the difficulty in solving a problem by a Turing machine is characterized not by the amount of resources required (such as time, space and so on) but rather in terms of the complexity of describing the problem in some logical formalism. This field was initially started in 1974 by Ronald Fagin with the celebrated result in [2] (coined by Neil Immerman as 'Fagin's theorem') which stated that the class of NP languages coincides with the class of languages definable in existential second-order logic. Many further surprising results followed this development, particularly the Immerman–Vardi's theorem characterizing P over ordered structures using fixed-point logic [3,4] and the Abiteboul–Vianu–Vardi characterization of PSPACE in terms of partial fixed-point logic [4,5]. Today there are several textbooks that cover the fundamentals of the area as a line of research within finite model theory [6–9]. In this paper, we propose to study quantitative

⋆ This article is an expanded version of the conference publication [1], with new proofs and new results such as Theorem 1.

* Corresponding author.
E-mail address: g.badia@uq.edu.au (G. Badia).

https://doi.org/10.1016/j.ic.2026.105446
Received 20 January 2025; Received in revised form 22 March 2026; Accepted 23 March 2026
Available online 27 March 2026

versions of some of these key results in this important field in connection with weighted computation. We work over finite structures that come with a linear ordering, which is a standard restriction in descriptive complexity.

Weighted automata are nondeterministic finite automata augmented with values from a semiring as weights on the transitions [10]. These weights may model, e.g. the cost involved when executing a transition, the amount of resources or time needed for this, or the probability or reliability of its successful execution. The theory of weighted automata and weighted context-free grammars was essential for the solution of such classical automata-theoretic problems as the decidability of the equivalence of unambiguous context-free languages and regular languages [11] (in fact, the only known proofs of this involve weighted automata), the decidability of two given deterministic multitape automata [12], and the decidability of two given deterministic pushdown automata [13,14]. This led to a quick development of this field, described in the books [11,15–19]. Furthermore, weighted automata and weighted context-free grammars have been used as basic concepts in natural language processing and speech recognition, as well as in algorithms for digital image compression [20]. Weighted logic [21], with weights in an arbitrary semiring, was developed originally to obtain a weighted version of the Büchi–Elgot–Trakhtenbrot theorem, showing that a certain weighted monadic second-order logic has the same expressive power on words as weighted automata. Consequently, this weighted logic over suitable semirings like fields has similar decidability properties on words as unweighted monadic second-order logic. It is worth remarking that the classical Büchi–Elgot–Trakhtenbrot theorem is usually regarded as part of the "prehistory" of descriptive complexity [7, p. 145].

Weighted Turing machines extend the concept of weighted automata as natural quantitative counterparts of classical Turing machines. They were first introduced under the name "algebraic Turing machines" in [22,23] and they have attracted further attention in [24]. Instances of this concept include the so called "fuzzy Turing machines" [25,26]. Recently, the articles [27,28] have introduced a related notion of "semiring Turing machine" and explicitly asked for the development of descriptive complexity in this framework as an open problem, focusing specifically on Fagin's theorem in connection to weighted logic [27, p. 255]. We will address this problem at the end of Section 5.

Our contribution. The present paper develops a theory of weighted descriptive complexity and establishes quantitative versions of some celebrated classical theorems. The novel contributions of this work can be summarized in the following characterizations (for an arbitrary semiring $\mathcal{S}$):

- The weighted complexity class $\mathsf{NP}[\mathcal{S}]$ coincides with the queries definable by weighted existential second-order logic on ordered structures, with weights in $\mathcal{S}$, respectively for all structures if $\mathcal{S}$ is idempotent and commutative (Theorem 1).
- The weighted complexity class $\mathsf{FP}[\mathcal{S}]$ coincides with the queries definable by weighted inflationary fixed-point logic, with weights in $\mathcal{S}$ (Theorem 2).
- The weighted complexity class $\mathsf{FPSPACE}[\mathcal{S}]$ coincides with the queries definable by weighted partial fixed-point logic with the addition of second-order multiplicative and additive quantifiers, with weights in $\mathcal{S}$ (Theorem 3).
- The weighted complexity class $\mathsf{FPSPACE}_{poly}[\mathcal{S}]$ coincides with the queries definable by weighted partial fixed-point logic, with weights in $\mathcal{S}$ (Theorem 4).
- The weighted complexity class $\mathsf{FPLOG}[\mathcal{S}]$ coincides with the queries definable by weighted deterministic transitive closure logic (Theorem 5).
- The weighted complexity class $\mathsf{L}[\mathcal{S}]$ coincides with the queries definable by weighted first-order logic with the addition of a path operator (Theorem 6).

Related work. Despite the fact that some characterizations of counting complexity classes using Boolean logics were known [29–31], observe that the article [32] (following up on the work of [31]) already proposes the idea of using certain weighted logics (with weights in the semiring $\mathbb{N}$ of natural numbers or, in a couple of cases, $\mathbb{Z}$) to characterize well-known counting complexity classes. The authors obtain several interesting results that are also covered by our more encompassing work here (that is, they provide logical characterizations of #P, FP, FPSPACE, FPSPACE($poly$), GapP, and MaxP). There is, however, some orthogonality as they cover some classical complexity classes that we do not and, similarly, we cover some that they do not, since we do not restrict our semiring to being $\mathbb{N}$ or $\mathbb{Z}$. Moreover, the investigation in [32], by contrast to ours, focuses on the study of classical counting classes for ordered structures, while we consider both ordered and arbitrary structures (provided, in the latter case, that the semiring is idempotent and commutative; examples include e.g. the max-plus- and min-plus-semirings). In the present article, the central aim is rather to start the study of weighted complexity classes via logic, and establish results characterizing classical complexity classes are obtained as interesting byproducts of the work. In this way, we are also meeting the challenge posed in [24, p.3] of developing "quantitative descriptive complexity theory based on weighted logics […] over some fairly general class of semirings". Further work on the model theory of weighted logics includes a Feferman–Vaught result [33], but the area remains largely unexplored despite being one of the open problems suggested in [21]. Finally, an approach related to the weighted logics discussed here has recently been proposed in [34,35] motivated by problems in database theory [36] (an alternative approach to the same issue is also [37]). The idea there is that the atomic facts of a model are annotated by values from a semiring whereas in the present paper this aspect is fully classical. Moreover, in the weighted setting we do not let negation be an operation acting on semiring values other than 0 and 1, whereas in the database context it has to be an operation on arbitrary semiring values.

2. Weighted Turing machines

In order to introduce the notion of a weighted Turing machine, first we need to define the kind of algebraic structures that will provide the weights, that is, semirings.

Definition 1 (Semirings). A semiring is a tuple $\mathcal{S} = \langle S, +, \cdot, \mathbb{0}, \mathbb{1}\rangle$, with operations addition $+$ and multiplication $\cdot$ and constants $\mathbb{0}$ and $\mathbb{1}$ such that

- $\langle S, +, \mathbb{0}\rangle$ is a commutative monoid and $\langle S, \cdot, \mathbb{1}\rangle$ is a monoid,
- multiplication distributes over addition, and
- $s \cdot \mathbb{0} = \mathbb{0} \cdot s = \mathbb{0}$ for every $s \in S$.

We say that $\mathcal{S}$ is commutative if the monoid $\langle S, \cdot, \mathbb{1}\rangle$ is commutative, and we say that $\mathcal{S}$ is idempotent if the monoid $\langle S, +, \mathbb{0}\rangle$ is idempotent (that is, $s + s = s$ for each $s \in S$).

Some examples of semirings, including those that we will use in this paper, are the following:

- the *Boolean semiring* $\mathbb{B} = \langle \{0,1\}, \min, \max, 0, 1\rangle$,
- any bounded distributive lattice $\langle L, \vee, \wedge, 0, 1\rangle$,
- the semiring of natural numbers $\langle \mathbb{N}, +, \cdot, 0, 1\rangle$,
- the semiring of extended natural numbers $\langle \mathbb{N} \cup \{+\infty\}, +, \cdot, 0, 1\rangle$ where $0 \cdot (+\infty) = 0$,
- the ring of integers, $\langle \mathbb{Z}, +, \cdot, 0, 1\rangle$,
- the ring of integers modulo n, $\langle \mathbb{Z}_n, +_n, \cdot_n, \overline{0}, \overline{1}\rangle$, for each $n \in \mathbb{N}$,
- the field of rational numbers $\langle \mathbb{Q}, +, \cdot, 0, 1\rangle$,
- the *max-plus* or *arctic semiring* $\mathrm{Arct} = \langle \mathbb{R}_+ \cup \{-\infty\}, \max, +, -\infty, 0\rangle$, where $\mathbb{R}_+$ denotes the set of non-negative real numbers,
- the restriction of the arctic semiring to the natural numbers $\mathbb{N}_{\max} = \langle \mathbb{N} \cup \{-\infty\}, \max, +, -\infty, 0\rangle$,
- the *min-plus* or *tropical semiring* $\mathrm{Trop} = \langle \mathbb{R}_+ \cup \{+\infty\}, \min, +, +\infty, 0\rangle$,
- the restriction of the tropical semiring to the natural numbers $\mathbb{N}_{\min} = \langle \mathbb{N} \cup \{+\infty\}, \min, +, +\infty, 0\rangle$,
- the semiring $\mathcal{F}_* = \langle [0,1], \max, *, 0, 1\rangle$ given by a t-norm $*$ [26],
- the semiring of finite languages $2^{\Sigma^*}_{\mathrm{fin}} = \langle 2^{\Sigma^*}_{\mathrm{fin}}, \cup, \cdot, \emptyset, \{\varepsilon\}\rangle$, for an alphabet Σ,
- the semiring $\mathcal{S}_{\max} = \langle \{0,1\}^* \cup \{-\infty\}, \max, \cdot, -\infty, \varepsilon\rangle$ of binary words in which max is computed according to the *radix order* (for $x, y \in \{0,1\}^*$, $x \preceq y$ iff $|x| < |y|$ or $|x| = |y|$ and x is smaller than or equal to y in the lexicographic order) and $\max(x, -\infty) = \max(-\infty, x) = x$ for each x, $\cdot$ is the concatenation operation, and $x \cdot (-\infty) = (-\infty) \cdot x = -\infty$ for each x,
- the semiring $\mathcal{S}_{\min} = \langle \{0,1\}^* \cup \{+\infty\}, \min, \cdot, +\infty, \varepsilon\rangle$ analogous to the previous one.

Definition 2 (Weighted Turing Machines). Let $\mathcal{S}$ be a semiring and Σ an alphabet. A *weighted (or algebraic) Turing machine* over $\mathcal{S}$ and input alphabet Σ is a septuple $\mathcal{M} = \langle Q, \Gamma, \Delta, \nu, q_0, F, \square\rangle$, where

- Q is a nonempty finite set whose elements are called *states*,
- $\Gamma \supseteq \Sigma$ is an alphabet (*working alphabet*),
- $\Delta \subseteq (Q \setminus F) \times \Gamma \times Q \times \Gamma \times \{-1, 0, 1\}$ and its elements are called *transitions*,
- $\nu : \Delta \longrightarrow S$ is called a *transition weighting function*, $q_0 \in Q$ is called the *initial state*, $F \subseteq Q$ and its elements are called *accepting states*, and $\square \in \Gamma \setminus \Sigma$ is the blank symbol.

We call $\mathcal{M}$ a *Turing machine* if $\mathcal{S}$ is the Boolean semiring $\mathbb{B}$. We call $\mathcal{M}$ *deterministic* if for every pair $\langle p, a\rangle \in Q \times \Gamma$, there is at most one transition $\langle p, a, q, b, d\rangle \in \Delta$.

A *configuration* of $\mathcal{M}$ is a tuple formed by: the machine's state, the position of the machine's head, and the string in the working tape. If $e = \langle p, c, q, d, t\rangle \in \Delta$ is a transition and C_1, C_2 are configurations of $\mathcal{M}$, then we write $C_1 \longrightarrow_e C_2$ if C_1 is a configuration with state p and the head reading c, while C_2 is obtained from C_1 by changing the state to q, rewriting the originally read symbol c to d, and moving the head as prescribed by t. We write $C_1 \longrightarrow C_2$ if $C_1 \longrightarrow_e C_2$ for some $e \in \Delta$.

A *computation* of $\mathcal{M}$ is a word of the form $\gamma = C_1 e_1 C_2 e_2 C_3 \dots C_n e_n C_{n+1}$ such that $C_1, \dots, C_{n+1}$ are configurations of $\mathcal{M}$, $e_1, \dots, e_n \in \Delta$, $C_k \longrightarrow_{e_k} C_{k+1}$ for each $k \in \{1, \dots, n\}$, and C_1 is a configuration with state q_0 and the head at the leftmost non-blank cell (if there is some). The *weight* of γ is defined as $\nu(\gamma) := \nu(e_1)\nu(e_2)\dots\nu(e_n)$. γ is called an *accepting* computation if C_{n+1} has an accepting state. We say that γ is a computation on w in Σ^*, and write $\Sigma(\gamma) = w$ if C_1 is a configuration with w on the working tape. We denote the set of all computations of $\mathcal{M}$ by $C(\mathcal{M})$ and the set of all accepting computations by $A(\mathcal{M})$.

Let us stress that semiring operations are not computed on the tape and hence have no influence in the time and space used by the machine.

Convention 1. *From now on we will assume that every Turing machine $\mathcal{M}$ is* finitely terminating*, that is, the set $C_w(\mathcal{M}) = \{\gamma \in C(\mathcal{M}) \mid \Sigma(\gamma) = w\}$ is finite for each $w \in \Sigma^*$. In particular, the set $A_w(\mathcal{M}) = \{\gamma \in A(\mathcal{M}) \mid \Sigma(\gamma) = w\}$ is finite.*

By a *series* σ we mean a mapping $\sigma : \Sigma^* \longrightarrow S$ where Σ is an alphabet, Σ^* the set of all words over Σ, and $\mathcal{S}$ is a semiring. Thanks to the convention, we can introduce the following notion:

Definition 3 (Behavior of a weighted Turing machine). Let $\mathcal{M}$ be a weighted Turing machine. The *behavior* of $\mathcal{M}$ is the mapping $\|\mathcal{M}\| : \Sigma^* \longrightarrow S$ defined as

$$\|\mathcal{M}\|(w) := \sum_{\gamma \in A_w(\mathcal{M})} \nu(\gamma).$$

We say that a series $\sigma : \Sigma^* \longrightarrow S$ is recognized by a weighted Turing machine $\mathcal{M}$ if $\|\mathcal{M}\| = \sigma$.

The definition of weighted Turing machine we have used here is exactly the same as that of *algebraic Turing machines* [22, Def. 5.1] (see also [24]). Similarly, the notions of the behavior of the machine coincide. The *semiring Turing machines* of [27,28], by contrast, differ in that they impose some conditions on the allowed transitions [27, cf. Def. 12]. Given distributivity of multiplication over addition, the notion of a semiring Turing machine function in [27, Def. 13] coincides with that of the behavior we use here. Semiring Turing machines allow semiring values on the tape in somewhat of a black-box manner. Intuitively, one can transition with the weight of the value on the tape, but cannot differentiate the values on the tape or modify them. If semiring values are not allowed in the input string then the definition of weighted and semiring Turing machines are equivalent (in the sense that one can be transformed into the other without a significant change of execution time). All these definitions generalize the corresponding notions for weighted automata. (In a recent paper, we have also provided a modified version of semiring Turing machines that corrects some flaws in the original definition [38].)

3. Some weighted complexity classes

Let $\mathcal{M} = \langle Q, \Gamma, \Delta, \nu, q_0, F, \square \rangle$ be a weighted Turing machine over $\mathcal{S}$ and Σ. For $w \in \Sigma^*$, we denote by $\mathsf{TIME}(\mathcal{M}, w)$ the maximal length of a computation of $\mathcal{M}$ on w, and define, for $n \in \mathbb{N}$, $\mathsf{TIME}(\mathcal{M}, n) := \max\{\mathsf{TIME}(\mathcal{M}, w) : w \in \Sigma^*, |w| \leq n\}$.

For a function $f : \mathbb{N} \longrightarrow \mathbb{N}$, we denote by $\mathsf{SERIES}[\mathcal{S}, \Sigma](f)$ the set of all series σ such that $\sigma = \|\mathcal{M}\|$ for some weighted Turing machine $\mathcal{M}$ over $\mathcal{S}$ and Σ with $\mathsf{TIME}(\mathcal{M}, n) = O(f(n))$. Now we can define the complexity classes:

$$\mathsf{SERIES}[\mathcal{S}](f(n)) := \bigcup \{\mathsf{SERIES}[\mathcal{S}, \Sigma](f(n)) : \Sigma \text{is an alphabet}\}.$$

Definition 4. Let $\mathcal{S}$ be a semiring. We define the following weighted complexity class

$$\mathsf{NP}[\mathcal{S}] := \bigcup \{\mathsf{SERIES}[\mathcal{S}](n^k) : k \in \mathbb{N}\}.$$

$\mathsf{NP}[\mathcal{S}]$ (cf. [24, Def. 4.1]) coincides with the definition of the class $\mathcal{S} - \#\mathsf{P}$ in [22, Def. 5.2]. Furthermore, it is contained as a subclass in the similarly defined class $\mathcal{NP}[\mathcal{S}]$ from [27, Def. 14] when $\mathcal{S}$ is a commutative semiring. Below (Proposition 1), we will actually show that this containment is proper, in the sense that $\mathcal{NP}[\mathcal{S}]$ will contain some series that are not in $\mathsf{NP}[\mathcal{S}]$.

Example 1. Following [22, Prop. 5.3] and [24, Examples 4.2–4.6], we can list some prominent instances of $\mathsf{NP}[\mathcal{S}]$:

- the usual complexity class NP (i.e. the set of decision problems solvable in polynomial time by a nondeterministic Turing machine), obtained when $\mathcal{S} = \mathbb{B}$ and each transition is weighted by 1 (this is the standard way of representing a classical machine model in the weighted context),
- the counting class #P [39] (i.e. the set of functions $f : \{0,1\}^* \longrightarrow \mathbb{N}$ such that there is a polynomial time nondeterministic Turing machine with $f(x)$ many accepting computations for every binary string x), obtained when $\mathcal{S} = \langle \mathbb{N}, +, \cdot, 0, 1 \rangle$ and each transition is weighted by 1,
- the complexity class $\bigoplus \mathsf{P}$ [40] (i.e. the class of decision problems solvable by a nondeterministic Turing machine in polynomial time, where the acceptance condition is that the number of accepting computation paths is odd), obtained when $\mathcal{S} = \langle \mathbb{Z}_2, +_2, \cdot_2, \overline{0}, \overline{1} \rangle$ and each transition is weighted by 1,
- the class GapP, closure of #P under subtraction [41,42] (i.e. the counting complexity class consisting of all of the functions f such that there exists a nondeterministic polynomial-time Turing machine M where, for any input x, $f(x)$ is the number of accepting paths of M minus the number of rejecting paths of M), obtained when $\mathcal{S} = \langle \mathbb{Z}, +, \cdot, \overline{0}, \overline{1} \rangle$ and transitions are weighted by 1 and -1,
- the class $\mathsf{MOD}_q - \mathsf{P}$ (for $q \geq 2$) [43], defined similarly to #P but with respect to counting modulo q, obtained when $\mathcal{S} = \langle \mathbb{Z}_q, +_q, \cdot_q, \overline{0}, \overline{1} \rangle$ and transitions are weighted by 1.

Example 2. Some further instances of $\mathsf{NP}[\mathcal{S}]$, this time following [24, Examples 4.7–4.11], are:

- the class $\mathsf{NP}[\mathcal{F}_*]$ of all fuzzy languages realizable by fuzzy Turing machines [26] with t-norm $*$ in polynomial time, obtained when the semiring is $\mathcal{F}_* = \langle [0,1], \max, *, 0, 1 \rangle$ and the weights correspond to degrees of membership in the fuzzy language,
- the class NPMV of all multivalued functions realized by nondeterministic polynomial-time transducer machines [44], obtained when, given alphabets Σ_1 and Σ_2, the semiring is $\langle 2^{\Sigma_2^*}_{\text{fin}}, \cup, \cdot, \emptyset, \{\varepsilon\} \rangle$ and weighted Turing machines have input alphabet Σ_1,
- the class of all multiset-valued functions computed by nondeterministic polynomial time transducer machines with counting, obtained as in the previous example but using the free semiring $\langle \mathbb{N}\langle \Sigma_2^* \rangle, +, \cdot, 0, 1 \rangle$ instead,
- the class $\mathsf{MaxP} \subseteq \mathsf{OptP}$ [45] (i.e. the class of functions computable by taking the maximum of the output values over all accepting paths of an NP machine), obtained when the semiring is $\mathcal{S}_{\max}$, and the class $\mathsf{MinP} \subseteq \mathsf{OptP}$ (i.e. the class of functions computable by taking the minimum of the output values over all accepting paths of an NP machine), obtained when the semiring is $\mathcal{S}_{\min}$,
- the class $\mathsf{MaxP}[O(\log n)] \subseteq \mathsf{OptP}[O(\log n)]$ (which restricts the size of the output to $\log n$ bits), obtained when the semiring is $\mathbb{N}_{\max}$, and the class $\mathsf{MinP}[O(\log n)] \subseteq \mathsf{OptP}[O(\log n)]$, obtained when the semiring is $\mathbb{N}_{\min}$.

Let $\mathcal{M} = \langle Q, \Gamma, \Delta, \nu, q_0, F, \square \rangle$ be a weighted Turing machine over $\mathcal{S}$ and Σ. For $w \in \Sigma^*$, we denote by $\mathsf{SPACE}(\mathcal{M}, w)$ the maximal numbers of cells used in a computation of $\mathcal{M}$ on w, and define, for $n \in \mathbb{N}$, $\mathsf{SPACE}(\mathcal{M}, n) := \max\{\mathsf{SPACE}(\mathcal{M}, w) : w \in \Sigma^*, |w| \leq n\}$.

For a function $f : \mathbb{N} \longrightarrow \mathbb{N}$, we denote by $\mathsf{SERIES}^*[\mathcal{S}, \Sigma](f)$ the set of all series σ such that $\sigma = \|\mathcal{M}\|$ for some weighted Turing machine $\mathcal{M}$ over $\mathcal{S}$ and Σ with $\mathsf{SPACE}(\mathcal{M}, n) = O(f(n))$. Now we can define the complexity classes:

$$\mathsf{SERIES}^*[\mathcal{S}](f(n)) := \bigcup \{\mathsf{SERIES}^*[\mathcal{S}, \Sigma](f(n)) : \Sigma \text{is an alphabet}\}.$$

Definition 5. Let $\mathcal{S}$ be a semiring. We define the following weighted complexity class

$$\mathsf{L}[\mathcal{S}] := \mathsf{SERIES}^*[\mathcal{S}](\log n).$$

Example 3. We can list some prominent instances of $\mathsf{L}[\mathcal{S}]$:

- the usual complexity class NL (i.e. the set of decision problems solvable with log-space by a nondeterministic Turing machine), obtained when $\mathcal{S} = \mathbb{B}$ and each transition is weighted by 1, and
- the counting class #L [46] (i.e. the set of functions $f : \{0,1\}^* \longrightarrow \mathbb{N}$ such that there is a polynomial time nondeterministic log-space Turing machine with $f(x)$ many accepting computations for every binary string x), obtained when $\mathcal{S} = \langle \mathbb{N}, +, \cdot, 0, 1\rangle$ and each transition is weighted by 1.

In defining some of the complexity classes below (e.g. $\mathsf{FP}[\mathcal{S}]$, $\mathsf{FPSPACE}[\mathcal{S}]$, and $\mathsf{FPLOG}[\mathcal{S}]$), we will use the following notion from universal algebra (cf. [47]):

Definition 6 (Term algebra)**.** Consider a semiring $\mathcal{S} = \langle S, +, \cdot, \mathbb{0}, \mathbb{1}\rangle$ and a subset $X \subseteq S$. The set of terms $T(X)$ is the collection of all well-formed strings that can be constructed using the symbols in X and $+', \cdot', \mathbb{0}, \mathbb{1}$ (in particular, $\mathbb{0}, \mathbb{1} \in T(X)$), that is, the smallest set such that: (1) $X \subseteq T(X)$ and (2) $(t_1 +' t_2) \in T(X)$ and $(t_1 \cdot' t_2) \in T(X)$ for every two terms $t_1, t_2 \in T(X)$; we abuse notation and omit parentheses whenever associativity permits. The term algebra $\mathcal{T}(X)$ is the structure with universe $T(X)$ and operations $+', \cdot'$ defined in the obvious way.

Recall that classically FP is the set of function problems that can be solved by a deterministic Turing machine in polynomial time.

Definition 7. We define the complexity class $\mathsf{FP}[\mathcal{S}]$ as

$$\mathsf{FP}[\mathcal{S}] := \bigcup_{\{0,1\}\subseteq G\subseteq_{\mathrm{fin}} S\,\Sigma\text{is a finite alphabet}} \mathsf{FP}[G, \Sigma]$$

where $\mathsf{FP}[G, \Sigma]$ is the set of all series $\sigma : \Sigma^* \longrightarrow \langle G\rangle$ (where $\langle G\rangle$ is the subsemiring of $\mathcal{S}$ generated by G) such that there is a constant $k \in \mathbb{N}$ and a deterministic polynomial-time Turing machine which outputs for every word $w \in \Sigma^*$ a word of the form $\sum_{i_1=1}^{m_1} \prod_{j_1=1}^{n_1} \cdots \sum_{i_k=1}^{m_k} \prod_{j_k=1}^{n_k} s_{i_1 j_1 \cdots i_k j_k}$ in the algebra of terms $T(G)$ in $\mathcal{S}$ with value $\sigma(w)$ (i.e. the result of applying the series σ to the word w) in $\mathcal{S}$.

In Definition 7, we employ a classical deterministic Turing machine which outputs, in each transition, symbols from $G \cup \{(,), +', \cdot', \mathbb{0}, \mathbb{1}\}$ or a blank. Thus, for our outputs we could obtain arbitrarily complex expressions. Therefore, the constant k limiting the number of alternations of sums and products is a proper restriction. Hence this definition of $\mathsf{FP}[\mathcal{S}]$ differs from the one of [24]. Later on, we will model logical formulas with alternating sum and product quantifiers using Turing machines which compute functions in $\mathsf{FP}[\mathcal{S}]$, hence $k = 1$ would be insufficient to model these alternations. For the converse, in order to model these Turing machines by formulas, the number of alternations of sums and products in each such Turing machine needs to be bounded to obtain a formula with nested quantifiers.

Example 4. If $\mathcal{S} = \mathbb{B}$ is the two-element Boolean semiring, then $\mathsf{FP}[\mathbb{B}]$ is just P [24, Example 5.4]. Observe that, as required by the definition of $\mathsf{FP}[\mathcal{S}]$, the terms output by the machine in that example are already trivially of the form $\sum_{i=1}^{n} \prod_{j=1}^{m} s_{ij}$.

FP is to #P what P is to NP. Thus, considering $\mathsf{NP}[\mathcal{S}]$ as a generalization of #P (as it is done in [22]), the relationship between $\mathsf{FP}[\mathcal{S}]$ and $\mathsf{NP}[\mathcal{S}]$ is similar to that between P and NP.

Example 5. If $\mathcal{S} = \mathbb{N}$ is the natural numbers semiring, then $\mathsf{FP}[\mathbb{N}]$ is just FP [24, Example 5.5] (i.e. the set of function problems that can be solved by a deterministic Turing machine in polynomial time). As before, as required by the definition of $\mathsf{FP}[\mathcal{S}]$, observe that the terms output by the machine in that example are already of the form $\sum_{i=1}^{n} \prod_{j=1}^{m} s_{ij}$.

Definition 8. The class $\mathsf{FPLOG}[\mathcal{S}]$ is defined as $\mathsf{FP}[\mathcal{S}]$ except that we allow the machine to have logarithmic space on the length of the input rather than polynomial time.

Example 6. If $\mathcal{S} = \mathbb{B}$, then $\mathsf{FPLOG}[\mathbb{B}]$ is just DLOGSPACE.

Example 7. If $\mathcal{S} = \mathbb{N}$, then $\mathsf{FPLOG}[\mathbb{N}]$ is just FPLOG, which is defined as FP but allowing the machine to use logarithmic space on the size of the input (cf. [48]).

Definition 9. The class $\mathsf{FPSPACE}[\mathcal{S}]$ is defined as $\mathsf{FP}[\mathcal{S}]$ except that we allow the machine to have polynomial space on the length of the input rather than polynomial time.

Example 8. If $\mathcal{S} = \mathbb{B}$, then $\mathsf{FPSPACE}[\mathbb{B}]$ is just PSPACE.

Example 9. If $\mathcal{S} = \mathbb{N}$, then $\mathsf{FPSPACE}[\mathbb{N}]$ is just FPSPACE ([49]).

Definition 10. The class $\mathsf{FPSPACE}_{poly}[\mathcal{S}]$ is defined as $\mathsf{FPSPACE}[\mathcal{S}]$ except that we require the word $\sum_{i=1}^{n} \prod_{j=1}^{m} s_{ij}$ to have length bounded by a polynomial. Here, every semiring element is considered to have length 1.

Example 10. If $\mathcal{S} = \mathbb{B}$, then $\mathsf{FPSPACE}_{poly}[\mathbb{B}]$ is just PSPACE.

Example 11. If $\mathcal{S} = \mathbb{N}$, then $\mathsf{FPSPACE}_{poly}[\mathbb{N}]$ is just $\mathsf{FPSPACE}_{poly}$ ([49]).

4. Weighted logics

A *vocabulary* (or *signature*) τ is a pair $\langle \mathrm{Rel}_\tau, \mathrm{ar}\tau \rangle$ where Rel_τ is a set of relation symbols and $\mathrm{ar}\tau : \mathrm{Rel}_\tau \longrightarrow \mathbb{N}_+$ is the arity function. A *τ-structure* $\mathfrak{A}$ is a pair $\langle A, \mathcal{I}_\mathfrak{A} \rangle$ where A is a set, called the *universe of* $\mathfrak{A}$, and $\mathcal{I}_\mathfrak{A}$ is an *interpretation*, which maps every symbol $R \in \mathrm{Rel}_\tau$ to a set $R^\mathfrak{A} \subseteq A^{\mathrm{ar}\tau(R)}$. We assume that each structure is *finite*, that is, its universe is a finite set. A structure is called *ordered* if it is given for a vocabulary $\tau \cup \{<\}$ where $<$ is interpreted as a linear ordering with endpoints. By $\mathrm{Str}(\tau)_<$ we denote the class of all finite ordered τ-structures.

We provide a countable set $\mathcal{V}$ of first and second-order variables, where lower case letters like x and y denote first-order variables and capital letters like X and Y denote second-order variables. Each second-order variable X comes with an associated arity, denoted by $\mathrm{ar}(X)$. We define first-order formulas β over a signature τ and weighted first-order formulas φ over τ and a semiring $\mathcal{S}$, respectively, by the grammars

$$\beta ::= \mathtt{false} \mid R(x_1, \dots, x_n) \mid \neg\beta \mid \beta \vee \beta \mid \exists x.\beta$$
$$\varphi ::= \beta \mid s \mid \varphi \oplus \varphi \mid \varphi \otimes \varphi \mid \bigoplus x.\varphi \mid \bigotimes x.\varphi,$$

where $R \in \mathrm{Rel}_\tau$, $n = \mathrm{ar}\tau(R)$, $x, x_1, \dots, x_n \in \mathcal{V}$ are first-order variables, and $s \in S$. Likewise, we define second-order formulas β over τ and weighted second-order formulas φ over τ and $\mathcal{S}$ through

$$\beta ::= \mathtt{false} \mid R(x_1, \dots, x_n) \mid X(x_1, \dots, x_n) \mid \neg\beta \mid \beta \vee \beta \mid \exists x.\beta \mid \exists X.\beta$$
$$\varphi ::= \beta \mid s \mid \varphi \oplus \varphi \mid \varphi \otimes \varphi \mid \bigoplus x.\varphi \mid \bigotimes x.\varphi \mid \bigoplus X.\varphi \mid \bigotimes X.\varphi,$$

with $R \in \mathrm{Rel}_\tau$, $n = \mathrm{ar}\tau(R) = \mathrm{ar}(X)$, $x, x_1, \dots, x_n \in \mathcal{V}$ first-order variables, $X \in \mathcal{V}$ a second-order variable, and $s \in S$. We also allow the usual abbreviations $\wedge$, $\forall$, $\rightarrow$, $\leftrightarrow$, and $\mathtt{true}$. By $\mathrm{FO}(\tau)$ and $\mathrm{wFO}(\tau, S)$ we denote the sets of all first-order formulas over τ and all weighted first-order formulas over τ and $\mathcal{S}$, respectively, and by $\mathrm{SO}(\tau)$ and $\mathrm{wSO}(\tau, S)$ we denote the sets of all second-order formulas over τ and all weighted second-order formulas over τ and $\mathcal{S}$, respectively.

The notion of *free variables* is defined as usual, i.e., the operators $\exists, \forall, \bigoplus$, and $\bigotimes$ bind variables. We let $\mathrm{Free}(\varphi)$ be the set of all free variables of φ. A formula φ with $\mathrm{Free}(\varphi) = \emptyset$ is called a *sentence*. For a tuple $\bar{\varphi} = \langle \varphi_1, \dots, \varphi_n \rangle \in \mathrm{wSO}(\tau, S)^n$, we define $\mathrm{Free}(\bar{\varphi}) = \bigcup_{i=1}^n \mathrm{Free}(\varphi_i)$.

We define the semantics of SO and wSO as follows. Let τ be a signature, $\mathfrak{A} = \langle A, \mathcal{I}_\mathfrak{A} \rangle$ a τ-structure, and $\mathcal{V}$ a set of first and second-order variables. A $(\mathcal{V}, \mathfrak{A})$-assignment ρ is a function $\rho : \mathcal{V} \longrightarrow A \cup \bigcup_{n \geq 1} \mathcal{P}(A^n)$ such that, whenever $x \in \mathcal{V}$ is a first-order variable, we have $\rho(x) \in A$, and whenever $X \in \mathcal{V}$ is a second-order variable, we have $\rho(X) \subseteq A^{\mathrm{ar}(X)}$. Let $\mathrm{dom}(\rho)$ be the domain of ρ. For a first-order variable $x \in \mathcal{V}$ and an element $a \in A$, the *update* $\rho[x \to a]$ is defined through $\mathrm{dom}(\rho[x \to a]) = \mathrm{dom}(\rho) \cup \{x\}$, $\rho[x \to a](\mathcal{X}) = \rho(\mathcal{X})$ for all $\mathcal{X} \in \mathcal{V} \setminus \{x\}$, and $\rho[x \to a](x) = a$. For a second-order variable $X \in \mathcal{V}$ and a set $I \subseteq A^{\mathrm{ar}(X)}$, the update $\rho[X \to I]$ is defined in a similar fashion. By $\mathfrak{A}_\mathcal{V}$ we denote the set of all $(\mathcal{V}, \mathfrak{A})$-assignments.

For $\rho \in \mathfrak{A}_\mathcal{V}$ and a formula $\beta \in \mathrm{SO}(\tau)$ the relation "$\langle \mathfrak{A}, \rho \rangle$ satisfies β", denoted by $\langle \mathfrak{A}, \rho \rangle \vDash \beta$, is defined as

$$\begin{aligned}
&\langle \mathfrak{A}, \rho \rangle \vDash \mathtt{false} && \text{never holds}\\
&\langle \mathfrak{A}, \rho \rangle \vDash R(x_1, \dots, x_n) && \iff x_1, \dots, x_n \in \mathrm{dom}(\rho) \text{ and } (\rho(x_1), \dots, \rho(x_n)) \in R^\mathfrak{A}\\
&\langle \mathfrak{A}, \rho \rangle \vDash X(x_1, \dots, x_n) && \iff x_1, \dots, x_n, X \in \mathrm{dom}(\rho), \langle \rho(x_1), \dots, \rho(x_n) \rangle \in \rho(X)\\
&\langle \mathfrak{A}, \rho \rangle \vDash \neg\beta && \iff \langle \mathfrak{A}, \rho \rangle \vDash \beta \text{ does not hold}\\
&\langle \mathfrak{A}, \rho \rangle \vDash \beta_1 \vee \beta_2 && \iff \langle \mathfrak{A}, \rho \rangle \vDash \beta_1 \text{ or } \langle \mathfrak{A}, \rho \rangle \vDash \beta_2\\
&\langle \mathfrak{A}, \rho \rangle \vDash \exists x.\beta && \iff \langle \mathfrak{A}, \rho[x \to a] \rangle \vDash \beta \text{ for some } a \in A\\
&\langle \mathfrak{A}, \rho \rangle \vDash \exists X.\beta && \iff \langle \mathfrak{A}, \rho[X \to I] \rangle \vDash \beta \text{ for some } I \subseteq A.
\end{aligned}$$

Let $\varphi \in \mathrm{wSO}(\tau, S)$ and $\mathfrak{A} \in \mathrm{Str}(\tau)_<$. Let $a_1, \dots, a_k$ be an enumeration of the elements of $\mathfrak{A}$ according to the ordering that serves as the interpretation of $<$, and for each integer n, let $I_1^n, \dots, I_{l_n}^n$ be an enumeration of the subsets of A^n according to the lexicographic ordering induced by the interpretation of $<$. The *(weighted) semantics* of φ is a mapping $[\![\varphi]\!](\mathfrak{A}, \cdot) : \mathfrak{A}_\mathcal{V} \longrightarrow S$ inductively defined as

$$\begin{aligned}
[\![\beta]\!](\mathfrak{A}, \rho) &= \begin{cases} \mathbb{1} & \text{if } \langle \mathfrak{A}, \rho \rangle \vDash \beta \\ \mathbb{0} & \text{otherwise} \end{cases}\\
[\![s]\!](\mathfrak{A}, \rho) &= s\\
[\![\varphi_1 \oplus \varphi_2]\!](\mathfrak{A}, \rho) &= [\![\varphi_1]\!](\mathfrak{A}, \rho) + [\![\varphi_2]\!](\mathfrak{A}, \rho)\\
[\![\varphi_1 \otimes \varphi_2]\!](\mathfrak{A}, \rho) &= [\![\varphi_1]\!](\mathfrak{A}, \rho) \cdot [\![\varphi_2]\!](\mathfrak{A}, \rho)\\
[\![\textstyle\bigoplus x.\varphi]\!](\mathfrak{A}, \rho) &= \sum_{a \in A} [\![\varphi]\!](\mathfrak{A}, \rho[x \to a])\\
[\![\textstyle\bigotimes x.\varphi]\!](\mathfrak{A}, \rho) &= \prod_{1 \leq i \leq k} [\![\varphi]\!](\mathfrak{A}, \rho[x \to a_i])\\
[\![\textstyle\bigoplus X.\varphi]\!](\mathfrak{A}, \rho) &= \sum_{I \subseteq A^{\mathrm{ar}(X)}} [\![\varphi]\!](\mathfrak{A}, \rho[X \to I])\\
[\![\textstyle\bigotimes X.\varphi]\!](\mathfrak{A}, \rho) &= \prod_{1 \leq i \leq l_{\mathrm{ar}(X)}} [\![\varphi]\!](\mathfrak{A}, \rho[X \to I_i^{\mathrm{ar}(X)}]).
\end{aligned}$$

Thanks to the lexicografic ordering, our product quantifiers have a well-defined semantics. Note that if the semiring is commutative, in the clauses of universal quantifiers, the semantics is defined by using any order for the factors in the products.

We will usually identify a pair $\langle \mathfrak{A}, \emptyset \rangle$ (where $\emptyset$ is the empty mapping) with $\mathfrak{A}$. We will also refer to the following expansions of FO:

- Transitive closure logic (TC) is obtained by adding the following rule for building formulas: if $\varphi(\overline{x}, \overline{y})$ is a formula with variables $\overline{x} = x_1 \dots x_k$ and $\overline{y} = y_1 \dots y_k$, and $\overline{t}, \overline{s}$ are k-tuples of terms, then $[\mathbf{tc}_{\overline{x},\overline{y}} \varphi(\overline{x}, \overline{y})](\overline{t}, \overline{s})$ is also a formula, and its semantics is given as
  $\mathfrak{A} \models [\mathbf{tc}_{\overline{x},\overline{y}} \varphi(\overline{x}, \overline{y})](\overline{a}, \overline{b}) \iff$ there exist an $n \geq 1$ and $\overline{c_0}, \dots, \overline{c_n} \in A^k$ such that $\overline{c_0} = \overline{a}$, $\overline{c_n} = \overline{b}$, and $\mathfrak{A} \models \varphi(\overline{c_i}, \overline{c_{i+1}})$ for each $i \in \{0, \dots, n-1\}$.
- Deterministic transitive closure logic (DTC) is obtained by adding the following formation rule: if $\varphi(\overline{x}, \overline{y})$ is a formula with variables $\overline{x} = x_1 \dots x_k$ and $\overline{y} = y_1 \dots y_k$, and $\overline{t}, \overline{s}$ are k-tuples of terms, then $[\mathbf{dtc}_{\overline{x},\overline{y}} \varphi(\overline{x}, \overline{y})](\overline{t}, \overline{s})$ is also a formula, and its semantics is defined by the following equivalence: $[\mathbf{dtc}_{\overline{x},\overline{y}} \varphi(\overline{x}, \overline{y})](\overline{t}, \overline{s}) \equiv [\mathbf{tc}_{\overline{x},\overline{y}} \varphi(\overline{x}, \overline{y}) \wedge \forall z(\varphi(\overline{x}, \overline{z}) \to \overline{y} = \overline{z})](\overline{t}, \overline{s})$.
- Least fixed-point logic (LFP) is obtained by adding the following rules for building formulas: if $\varphi(R, \overline{x})$ is a formula of vocabulary $\tau \cup \{R\}$ with only positive occurrences of R, $\overline{x}$ is a tuple of variables, and $\overline{t}$ is a tuple of terms (both matching the arity of R), then $[\mathbf{lfp} R\overline{x}.\varphi](\overline{t})$ and $[\mathbf{gfp} R\overline{x}.\varphi](\overline{t})$ are also formulas. For their semantics, we need to define some auxiliary notions. The *update operator* $F_\varphi \colon \mathcal{P}(A^k) \longrightarrow \mathcal{P}(A^k)$ is defined by $F_\varphi(R) := \{\overline{a} \mid \langle \mathfrak{A}, R \rangle \models \varphi(R, \overline{a})\}$ for any relation R, and it is monotone because R occurs only positively in φ. A *fixed point* of F_φ is a relation R such that $F_\varphi(R) = R$. Since F_φ is monotone, it has a least and a greatest fixed point (by Knaster–Tarski Theorem). The semantics is given by: $\mathfrak{A} \models [\mathbf{lfp} R\overline{x}.\varphi](\overline{t})$ iff $\overline{t}^{\mathfrak{A}}$ is contained in the least fixed point of F_φ (analogously for $[\mathbf{gfp} R\overline{x}.\varphi](\overline{t})$ and the greatest fixed point).
- Partial fixed-point logic (PFP) is obtained by adding the following rule for building formulas: if $\varphi(R, \overline{x})$ is a formula of vocabulary $\tau \cup \{R\}$, $\overline{x}$ is a tuple of variables, and $\overline{t}$ is a tuple of terms (both matching the arity of R), then $[\mathbf{pfp} R\overline{x}.\varphi](\overline{t})$ is also a formula. For the semantics, we consider again the update operator (now not necessarily monotone) and the sequence of its finite stages: $R^0 := \emptyset$ and $R^{m+1} := F_\varphi(R^m)$. In a finite structure $\mathfrak{A}$, the sequence either reaches a fixed point or it enters a cycle of period greater than one. We define the partial fixed point of F_φ as the fixed point reached in the former case, or as the empty set in the latter case. Now, the semantics is given by: $\mathfrak{A} \models [\mathbf{pfp} R\overline{x}.\varphi](\overline{t})$ iff $\overline{t}^{\mathfrak{A}}$ is contained in the partial fixed point of F_φ.
- Inflationary fixed-point logic (IFP) is obtained by adding the following rules for building formulas: if $\varphi(R, \overline{x})$ is a formula of vocabulary $\tau \cup \{R\}$, $\overline{x}$ is a tuple of variables, and $\overline{t}$ is a tuple of terms (both matching the arity of R), then $[\mathbf{ifp} R\overline{x}.\varphi](\overline{t})$ is also a formula. For its semantics, we need to define some auxiliary notions. An operator $G \colon \mathcal{P}(B) \longrightarrow \mathcal{P}(B)$ is said to be *inflationary* if $X \subseteq G(X)$ for all $X \in \mathcal{P}(B)$. With any operator $F \colon \mathcal{P}(B) \longrightarrow \mathcal{P}(B)$ one can associate an inflationary operator G by setting $G(X) := X \cup F(X)$. Iterating G gives a fixed point that we will call the *inflationary fixed point* of F. The semantics is given by: $\mathfrak{A} \models [\mathbf{ifp} R\overline{x}.\varphi](\overline{t})$ iff $\overline{t}^{\mathfrak{A}}$ is contained in the inflationary fixed point of F_φ.

The weighted version of each of these logics is defined analogously as in the case of FO and SO by expanding the logics TC, DTC, LFP, PFP, and IFP with the same weighted constructs as given for wFO and wSO (i.e. we replace in the definition the Boolean layer, that is, the formulas denoted by β, by the formulas of the corresponding logic while excluding the second-order quantifiers in the semiring layer unless otherwise stated). By a famous result of Gurevich and Shelah [50], on finite structures, LFP coincides with IFP and thus their weighted versions, wIFP and wLFP, as we have defined them here, will also coincide in expressive power.

We recall now a standard description of cliques in graphs by second-order logic; we will use this in our subsequent examples for weighted structural properties.

Example 12. Let τ be the signature of a graph, i.e., $\mathrm{Rel}_\tau = \{\mathrm{edge}\}$ with edge binary. We call a graph $\mathfrak{G} \in \mathrm{Str}(\tau)$ undirected if its interpretation of edge is a symmetric relation on the universe of $\mathfrak{G}$. For every undirected graph $\mathfrak{G} \in \mathrm{Str}(\tau)$ and a subset I of its universe, we can check whether the nodes from I form a clique in $\mathfrak{G}$ using the SO-formula

$$\mathrm{clique}(X) := \forall x \forall y \Big(\big(Xx \wedge Xy \wedge x \neq y \big) \to \mathrm{edge}(x, y) \Big).$$

Here, the formula $x \neq y$ is an abbreviation for $\exists Y(Yy \wedge \neg(Yx))$. We have that $\langle \mathfrak{G}, [X \to I] \rangle$ satisfies $\mathrm{clique}(X)$ if and only if I is a clique in $\mathfrak{G}$.

We give next some examples of how weighted formulas can be interpreted (observe that in these structures we do not have an order relation).

Example 13. If $\mathcal{S} = \mathbb{B}$, we obtain classical logic.

Example 14. Using the arctic semiring $\mathrm{Arct} = \langle \mathbb{R}_+ \cup \{-\infty\}, \max, +, -\infty, 0 \rangle$, we can describe the size of the largest clique in a graph as follows. We reuse the signature τ of a graph and the SO-formula $\mathrm{clique}(X)$ from Example 12 and define a wSO-formula as follows.

$$\varphi := \bigoplus X. \Big(\mathrm{clique}(X) \otimes \bigotimes x. \big(0 \oplus (1 \otimes Xx) \big) \Big)$$

In this formula, 0 and 1 are the actual real numbers, not the semiring additive and multiplicative identities. Then, for every undirected graph $\mathfrak{G} \in \mathrm{Str}(\tau)$, we have that $[\![\varphi]\!](\mathfrak{G})$ is the size of the largest clique in $\mathfrak{G}$.

Example 15. Assume that $\mathcal{S} = \langle \mathbb{Q}, +, \cdot, 0, 1 \rangle$ is the field of rational numbers and that τ is the signature from the previous example. Then, for every fixed $n \in \mathbb{N}_+$, we can count the number of n-cliques of an undirected graph $\mathfrak{G} \in \mathrm{Str}(\tau)$ using the wSO-formula

$$\varphi_n := \frac{1}{n!} \otimes \bigoplus x_1 \dots \bigoplus x_n . \bigwedge_{i \neq j} \Big((x_i \neq x_j) \wedge \mathrm{edge}(x_i, x_j) \Big).$$

Here, $x_i \neq x_j$ again is an abbreviation for $\exists Y(x_j \in Y \wedge \neg(x_i \in Y))$. Because of the lack of an order relation, each n-clique is counted $n!$ times; this is taken care of by the factor $\frac{1}{n!}$.

Example 16. We consider the minimum cut of directed acyclic graphs. For this, we interpret these graphs as flow networks in the following way. Every vertex which does not have a predecessor is considered a source, every vertex without successors is considered a drain, and every edge is assumed to have a capacity of 1. Let $G = \langle V, E\rangle$ be a directed acyclic graph where V is the set of vertices and $E \subseteq V \times V$ the set of edges. A cut $\langle S, D\rangle$ of G is a partition of V, i.e., $S \cup D = V$ and $S \cap D = \emptyset$, such that all sources of G are in S, and all drains of G are in D. The minimum cut of G is the smallest number $|E \cap (S \times D)|$ such that $\langle S, D\rangle$ is a cut of G.

We can express the minimum cut of directed acyclic graphs by a weighted formula as follows. We let τ be the signature from the previous two examples and, as our semiring, we choose the tropical semiring $\mathrm{Trop} = \langle \mathbb{R}_+ \cup \{+\infty\}, \min, +, +\infty, 0\rangle$. Then, using the abbreviation

$$\mathrm{cut}(X, Y) := \forall x.\Big((Xx \leftrightarrow \neg(Yx)) \wedge (\exists y.\mathrm{edge}(y, x) \vee Xx) \wedge (\exists y.\mathrm{edge}(x, y) \vee Yx)\Big),$$

we can express the minimum cut of a directed acyclic graph $\mathfrak{G} \in \mathrm{Str}(\tau)$ using the formula

$$\varphi := \bigoplus X.\bigoplus Y.\Big(\mathrm{cut}(X, Y) \otimes \bigotimes x.\bigotimes y.(1 \oplus \neg(Xx \wedge Yy \wedge \mathrm{edge}(x, y)))\Big).$$

Example 17 (cf. [21])**.** Let $\mathcal{S} = \langle \mathbb{N}, +, \cdot, 0, 1\rangle$ be the semiring of natural numbers and let $\varphi \in \mathrm{wSO}(\tau, \mathcal{S})$ be a formula which does not contain any constants $s \in \mathbb{N}$. Then, we may understand $[\![\varphi]\!](\mathfrak{A}, \rho)$ as the number of proofs we have that $\langle \mathfrak{A}, \rho\rangle$ satisfies φ assuming that we interpret the weighted operators in the following way. For Boolean formulas, we simply consider satisfaction to give us one proof, and otherwise we have no proof. The sum $[\![\varphi_1 \oplus \varphi_2]\!]$ is the number of proofs we have that $\varphi_1 \vee \varphi_2$ is true. This says that, if we have n proofs for φ_1 and m proofs for φ_2, then we interpret this as having $n + m$ proofs for the fact that $\varphi_1 \vee \varphi_2$ is true. Likewise, we interpret the product $[\![\varphi_1 \otimes \varphi_2]\!]$ as the number of proofs we have that $\varphi_1 \wedge \varphi_2$ is true. Similar interpretations apply for the weighted quantifiers.

5. Logical characterizations of complexity classes

We are finally ready to present and prove the main results of the paper: the quantitative versions of several logical characterizations of prominent complexity classes. We may assume that every $\mathfrak{A} \in \mathrm{Str}(\tau)_<$ is encoded by a string of 0s and 1s. For example, where $\mathfrak{A} = \langle A, R_1^{\mathfrak{A}}, \ldots, R_j^{\mathfrak{A}}\rangle$ with $|A| = n$ (and we may assume in fact that $A = \{0, \ldots, n-1\}$) we might let

$$\mathrm{enc}(\mathfrak{A}) = \mathrm{enc}(R_1^{\mathfrak{A}}) \cdot \cdots \cdot \mathrm{enc}(R_j^{\mathfrak{A}})$$

where if $R_i^{\mathfrak{A}}$ is an l-ary relation, then $\mathrm{enc}(R_i^{\mathfrak{A}})$ is a string of symbols of length n^l with a 1 in its mth position if the mth tuple among the n^l one is in $R_i^{\mathfrak{A}}$ and a 0 otherwise.

Definition 11. Consider a weighted logic $\mathrm{L}[\mathcal{S}]$ (with weights in a semiring $\mathcal{S}$) and a weighted complexity class $\mathcal{C}$, which is simply a collection of series. We say that $\mathrm{L}[\mathcal{S}]$ captures $\mathcal{C}$ over ordered structures in the vocabulary $\tau = \{R_1, \ldots, R_j\}$ if:

(1) For every $\mathrm{L}[\mathcal{S}]$-formula ϕ, there exists a $P \in \mathcal{C}$ such that $P(\mathrm{enc}(\mathfrak{A})) = \|\phi\|(\mathfrak{A})$ for every finite ordered τ-structure $\mathfrak{A}$, and
(2) For every $P \in \mathcal{C}$, there exists an $\mathrm{L}[\mathcal{S}]$-formula ϕ such that $P(\mathrm{enc}(\mathfrak{A})) = \|\phi\|(\mathfrak{A})$ for every finite ordered τ-structure $\mathfrak{A}$.

The seminal Fagin's Theorem characterizes NP for ordered structures by existential second-order logic. Our goal is to present a weighted version of this result with arbitrary semirings as weighted structures. Whereas in the classical setting one obtains an equivalence between the existence of runs of a Turing machine vs. the satisfiability of an existential logical formula, in the weighted setting of general semirings we have to derive a one-to-one correspondence between the accepting runs of a Turing machine and satisfying assignments for the formulas. Moreover, due to the absence of a natural negation function in the semiring, here, beyond the classical setting, we need conjunctions and universal quantifications. We mention in passing that, for weighted finite automata over words, in [21] weighted conjunction and universal quantification turned out to be too powerful in general and had to be restricted. Surprisingly, here we do not need these restrictions, but we can show the expressive equivalence between weighted polynomial-time Turing machines and the full weighted existential second-order logic. Moreover, we do not need the commutativity of the multiplication of $\mathcal{S}$ (essential in [21]), but we can develop our characterization for arbitrary, also non-commutative, semirings $\mathcal{S}$. This is due to new constructions, in this setting, for the involved weighted Turing machines. By wESO we mean the fragment of wSO where the only second-order quantifiers (whether in the semiring layer or the Boolean layer) appear at the beginning of the formula and are additive existential.

Theorem 1 (Weighted Fagin's theorem)**.** *Let $\mathcal{S}$ be a semiring and τ a vocabulary.*

(i) The logic $\mathrm{wESO}[\mathcal{S}]$ *captures* $\mathsf{NP}[\mathcal{S}]$ *over ordered finite τ-structures.*
(ii) Assume that $\mathcal{S}$ is idempotent and commutative. Then, the logic $\mathrm{wESO}[\mathcal{S}]$ *captures* $\mathsf{NP}[\mathcal{S}]$ *over all finite structures in the vocabulary τ.*

Proof.

(*i*): In order to establish (1) from Definition 11, we construct, for every wESO-formula ϕ, an NP Turing machine $\mathcal{M}$ with $\|\phi\| = \|\mathcal{M}\|$. If $v_1, \ldots, v_m$ are the free variables of ϕ, we encode every input structure $\mathfrak{A} = \langle A, R_1^{\mathfrak{A}}, \ldots, R_j^{\mathfrak{A}}\rangle$ and free variable assignments $v_1 = a_1, \ldots, v_m = a_m$ for $\mathcal{M}$ by

$$\mathrm{enc}(\mathfrak{A}) = \mathrm{enc}(R_1^{\mathfrak{A}}) \cdot \cdots \cdot \mathrm{enc}(R_j^{\mathfrak{A}})\mathrm{enc}(a_1) \cdot \cdots \cdot \mathrm{enc}(a_m),$$

where $n = |A|$ and $\mathrm{enc}(a_i) = \mathrm{enc}(\{a_i\})$ if v_i is a first-order variable. If $\tau = \emptyset$, we can set $\mathrm{enc}(\mathfrak{A}) = \underbrace{0 \cdots 0}_{|A| \text{ times}}$ by definition, to make sure that $\mathrm{enc}(\mathfrak{A})$ is at least the same length as $|A|$. Let us assume that $A = \{0, \dots, n-1\}$. We proceed by induction on the structure of formulas and begin by showing that for every Boolean formula β without universal second-order quantifiers, there exists a deterministic polynomial time Turing machine $\mathcal{M}$ such that $\langle \mathfrak{A}, a_1, \dots, a_m \rangle \models \beta$ iff $\mathcal{M}$ accepts $\langle \mathfrak{A}, a_1, \dots, a_m \rangle$, see also [9, Proposition 6.6].

- If $\beta = R(x_1, \dots, x_l)$ or $\beta = X(x_1, \dots, x_l)$, we construct a deterministic Turing machine which deterministically checks if the bit corresponding to $\langle a_1, \dots, a_l \rangle$ is 1 in $\mathrm{enc}(R)$ or $\mathrm{enc}(X)$, respectively. More precisely, we check whether the m'th symbol of the encoding is 1 for $m = \sum_{i=1}^{l} n^{l-i} a_i$.
- If $\beta = \alpha \vee \gamma$, we let $\mathcal{M}_\alpha$ and $\mathcal{M}_\gamma$ be the deterministic polynomial time Turing machines for α and γ, respectively. We construct a Turing machine for β which (1) simulates $\mathcal{M}_\alpha$ and accepts if the simulation accepts, (2) otherwise continues to simulate $\mathcal{M}_\gamma$ and accepts if the simulation accepts and (3) otherwise rejects the input.
- If $\beta = \neg\alpha$, we let $\mathcal{M}_\alpha$ be the deterministic polynomial time Turing machine for α. We construct a Turing machine for β which simulates $\mathcal{M}_\alpha$ and accepts if the simulation rejects the input.
- If $\beta = \exists x.\alpha$, we let $\mathcal{M}_\alpha$ be the deterministic polynomial time Turing machine for α. We construct a Turing machine for β which iterates over all elements $a \in A$ and simulates $\mathcal{M}_\alpha$ on the input $\langle \mathfrak{A}, a, a_1, \dots, a_l \rangle$ and accepts once a simulation accepts. If no simulation accepts, the input is rejected. In total, we simulate $\mathcal{M}_\alpha$ at most $|A|$ times.

All of these constructions produce deterministic polynomial time Turing machines. Let $\mathcal{M}_\beta$ be the Turing machine constructed for β. We obtain a weighted Turing machine $\mathcal{M}$ from $\mathcal{M}_\beta$ with $\|\beta\| = \|\mathcal{M}\|$ by defining the weight of every transition by 1. Note that, as $\mathcal{M}_\beta$ is deterministic, there is exactly one run with weight 1 in $\mathcal{M}$ for every input accepted by $\mathcal{M}_\beta$.

We continue with the weighted formulas ϕ. The case $\phi = \beta$ is already covered. For $\phi = s$, we construct a Turing machine which accepts in a single transition with weight s to a final state.

- For $\phi = \psi \oplus \zeta$, we let $\mathcal{M}_\psi$ and $\mathcal{M}_\zeta$ be the weighted nondeterministic Turing machines for ψ and ζ, respectively. We construct a Turing machine $\mathcal{M}$ for ϕ which nondeterministically simulates either $\mathcal{M}_\psi$ or $\mathcal{M}_\zeta$ on the input. The simulations are started by a single transition of weight 1 from the new initial state into the initial state of either $\mathcal{M}_\psi$ or $\mathcal{M}_\zeta$. Thus, every run of either $\mathcal{M}_\psi$ or $\mathcal{M}_\zeta$ on the input is simulated by exactly one run of $\mathcal{M}$.
- For $\phi = \psi \otimes \zeta$, we let $\mathcal{M}_\psi$ and $\mathcal{M}_\zeta$ be the weighted nondeterministic Turing machines for ψ and ζ, respectively. We construct a Turing machine $\mathcal{M}$ for ϕ which first simulates $\mathcal{M}_\psi$ and then $\mathcal{M}_\zeta$ on the input. All transitions outside of the simulations, e.g., preparing the input for a simulation and clearing the tape for the second simulation, have weight 1 and are deterministic. Thus, every combination of a run r_ψ of $\mathcal{M}_\psi$ and a run r_ζ of $\mathcal{M}_\zeta$ on the input is simulated by exactly one run of $\mathcal{M}$ and the weight of this run is the product of the weights of r_ψ and r_ζ. As multiplication distributes over addition, $\mathcal{M}$ recognizes ϕ.
- If $\phi = \bigoplus x.\psi$, we let $\mathcal{M}_\psi$ be the weighted nondeterministic Turing machine for ψ. We construct a Turing machine $\mathcal{M}$ for ϕ which nondeterministically guesses an element $a \in A$ and simulates $\mathcal{M}_\psi$ on the input $\langle \mathfrak{A}, a, a_1, \dots, a_l \rangle$. We ensure that, for every $a \in A$, there is exactly one run of $\mathcal{M}$ which guesses a and that all transitions which prepare the simulation have weight 1. Thus, for every choice of $a \in A$, every run of $\mathcal{M}_\psi$ on $\langle \mathfrak{A}, a, a_1, \dots, a_l \rangle$ is simulated by exactly one run of $\mathcal{M}$.
- If $\phi = \bigotimes x.\psi$, we let $\mathcal{M}_\psi$ be the weighted nondeterministic Turing machine for ψ. We construct a Turing machine for ϕ which iterates over all elements $a \in A$ and simulates $\mathcal{M}_\psi$ on the input $\langle \mathfrak{A}, a, a_1, \dots, a_l \rangle$. All transitions outside of the simulations, e.g., preparing the input for a simulation and clearing the tape for the next simulation, have weight 1 and are deterministic. Thus, every combination of runs $r_0, \dots, r_{n-1}$ of $\mathcal{M}_\psi$ on $\langle \mathfrak{A}, 0, a_1, \dots, a_l \rangle, \dots, \langle \mathfrak{A}, n-1, a_1, \dots, a_l \rangle$, respectively, is simulated by exactly one run of $\mathcal{M}$ and the weight of this run is the product of the weights of $r_0, \dots, r_{n-1}$. As multiplication distributes over addition, $\mathcal{M}$ recognizes ϕ.
- If $\phi = \bigoplus X.\psi$, we proceed like in the case of $\bigoplus x.\psi$ but guess a relation R of appropriate arity for X instead of an element $a \in A$.

Now, for (2) from Definition 11, suppose that P is recognizable by a nondeterministic weighted Turing machine in polynomial time.

Let $\mathcal{M} = \langle Q, \Gamma, \Delta, \nu, q_0, F, \square \rangle$ be the machine such that $\|\mathcal{M}\| = P$ in time n^k where n is the length of the input (the encoding of a structure from $\mathrm{Str}(\tau)_<$) and k is bigger than the maximum of the arities of the relational symbols in τ. We will construct a wESO-formula ϕ such that $\|\phi\| = P$. Let $Q = \{q_0, \dots, q_{m-1}\}$. In what follows we assume that Γ in our Turing machine is $\{0, 1, \square\}$ (it is not difficult to generalize the proof to an arbitrary alphabet by adding more corresponding predicates in the encoding below).

The first task is to build a Boolean second-order formula

$$\psi(T_0, T_1, T_2, H_{q_0}, \dots, H_{q_{m-1}}),$$

without second-order quantifiers but where $T_0, T_1, T_2, H_{q_0}, \dots, H_{q_{m-1}}$ are fresh second-order variables, such that there is a one-to-one correspondence between the accepting computation paths for the input $\mathrm{enc}(\mathfrak{A})$ and the expansions of the model $\mathfrak{A}$ that satisfy $\psi(T_0, T_1, T_2, H_{q_0}, \dots, H_{q_{m-1}})$, i.e. it takes value 1.

Recall that $\mathfrak{A}$ with $|A| = n$ is linearly ordered by $<$. We will represent the n^k time and space parameters as the elements of the set A^k, i.e. k-tuples from A. From $<$, we can define in first-order logic an associated successor relation $\mathtt{Succ}$, as well as the bottom $\bot$ and top elements $\top$. With this at hand, if $\overline{x}, \overline{y}$ are k-tuples of variables, we can define a successor relation on the elements of A^k by the formula

$$\overline{x} = \overline{y} + 1 := \bigwedge_{i<k} \Big(\bigwedge_{j<i} (x_j = \top \wedge y_j = \bot) \wedge \mathtt{Succ}(x_i, y_i) \wedge \bigwedge_{j>i} x_j = y_j \Big).$$

Next, let us spell out the meaning of the predicates $T_0, T_1, T_2, H_{q_0}, \dots, H_{q_{m-1}}$:

(1) $T_i(\overline{p}, \overline{t})(i = 0, 1)$, where $\overline{p}, \overline{t}$ are k-tuples of first-order variables, is meant to represent that at time $\overline{t}$ the position $\overline{p}$ of the tape contains the symbol i. T_2 does the same but for the blank symbol.
(2) $H_{q_i}(\overline{p}, \overline{t})(0 \leq i \leq m-1)$ represents that at time $\overline{t}$ the machine is in state q_i with its head in position $\overline{p}$.

The idea is that we will use the predicates T_i's and H_q's to describe an accepting computation of $\mathcal{M}$ started with input enc($\mathfrak{A}$). For simplicity, we would like to assume that all computations on a structure of size n have length n^k, i.e., that there are no shorter computations. In order to do this, we modify the Turing machine by adding transitions which "do nothing" from all accepting states. Note that our definition of Turing machines specifically forbids transitions from accepting states. The resulting Turing machine has the same behavior as our given Turing machine when only taking into account computations of exactly length n^k, which is what the formula we construct will do. More precisely, we add the transitions $\{\langle q, a, q, a, 0\rangle \mid q \in F, a \in \Gamma\}$ and assign weight 1 to all of them.[1]

Given k-tuples of variables $\overline{x} = x_1, \dots, x_k$ and $\overline{y} = y_1, \dots, y_k$, we write $\overline{x} \neq \overline{y}$ as an abbreviation for $\bigvee_{1 \leq i \leq k} x_i \neq y_i$. We let $\psi(T_0, T_1, T_2, H_{q_0}, \dots, H_{q_{m-1}})$ be the conjunction of the following:

- $\forall \overline{p} \forall \overline{t}(T_0(\overline{p}, \overline{t}) \leftrightarrow \neg T_1(\overline{p}, \overline{t}))$
  "In every configuration no cell of the tape contains more than one symbol from the alphabet Γ."
- $\forall \overline{t} \exists! \overline{p}(\bigvee_{q \in Q} H_q(\overline{p}, \overline{t})) \wedge \forall \overline{p} \forall \overline{t}(\bigvee_{q,q' \in Q q \neq q'}(\neg H_q(\overline{p}, \overline{t}) \vee \neg H_{q'}(\overline{p}, \overline{t})))$
  "At any time the machine $\mathcal{M}$ is in exactly one state."
- $\exists \overline{t} \exists \overline{p}(\bigvee_{q \in F} H_q(\overline{p}, \overline{t}))$
  "Eventually the machine $\mathcal{M}$ enters an accepting state."
- $\bigvee_{\langle p,a,q,b,D\rangle \in \Delta, D \in \{-1,0,1\}} \theta_{\langle p,a,q,b,D\rangle}$, where
  $\theta_{\langle p,a,q,b,-1\rangle} := \forall \overline{t} \forall \overline{p}((H_p(\overline{p}, \overline{t}) \wedge T_a(\overline{p}, \overline{t})) \rightarrow (H_q(\overline{p} - 1, \overline{t} + 1) \wedge T_b(\overline{p}, \overline{t} + 1)) \wedge \forall \overline{p}'(\overline{p} \neq \overline{p}' \rightarrow (\bigwedge_{i=0,1,2}(T_i(\overline{p}', \overline{t} + 1) \leftrightarrow T_i(\overline{p}', \overline{t})))))$
  $\theta_{\langle p,a,q,b,1\rangle} := \forall \overline{t} \forall \overline{p}((H_p(\overline{p}, \overline{t}) \wedge T_a(\overline{p}, \overline{t})) \rightarrow (H_q(\overline{p} + 1, \overline{t} + 1) \wedge T_b(\overline{p}, \overline{t} + 1)) \wedge \forall \overline{p}'(\overline{p} \neq \overline{p}' \rightarrow (\bigwedge_{i=0,1,2}(T_i(\overline{p}', \overline{t} + 1) \leftrightarrow T_i(\overline{p}', \overline{t})))))$
  $\theta_{\langle p,a,q,b,0\rangle} := \forall \overline{t} \forall \overline{p}((H_p(\overline{p}, \overline{t}) \wedge T_a(\overline{p}, \overline{t})) \rightarrow (H_q(\overline{p}, \overline{t} + 1) \wedge T_b(\overline{p}, \overline{t} + 1)) \wedge \forall \overline{p}'(\overline{p} \neq \overline{p}' \rightarrow (\bigwedge_{i=0,1,2}(T_i(\overline{p}', \overline{t} + 1) \leftrightarrow T_i(\overline{p}', \overline{t})))))$
  "The configurations respect the transitions in Δ."
- $H_{q_0}(\underbrace{\bot \cdot \dots \cdot \bot}_{k-\text{times}}, \underbrace{\bot \cdot \dots \cdot \bot}_{k-\text{times}}) \wedge \bigwedge_{R_i \in \tau} \forall x_1, \dots, x_{r_i}((R_i(x_1, \dots, x_{r_i}) \rightarrow T_1(\underbrace{\bot \cdot \dots \cdot \bot}_{k-r_i-\text{times}} x_1 \dots x_{r_i}, \underbrace{\bot \cdot \dots \cdot \bot}_{k-\text{times}})) \wedge \wedge(\neg R_i(x_1, \dots, x_{r_i}) \rightarrow T_0(\underbrace{\bot \cdot \dots \cdot \bot}_{k-r_i-\text{times}} x_1 \dots x_{r_i}, \underbrace{\bot \cdot \dots \cdot \bot}_{k-\text{times}})) \wedge \wedge \forall x_1, \dots, x_k((x_1 \neq \bot \vee \dots \vee x_{k-r_i} \neq \bot) \rightarrow T_2(x_1 \cdots x_k, \underbrace{\bot \cdot \dots \cdot \bot}_{k-\text{times}}))$
  "At the initial time the tape contains enc($\mathfrak{A}$) and it is in the initial state q_0."
  Here r_i is the arity of the relation symbol R_i.

Finally, to get a wESO-formula ϕ such that $\|\phi\| = P$, we must first consider χ: $\psi(T_0, T_1, T_2, H_{q_0}, \dots, H_{q_{m-1}}) \otimes \otimes \bigotimes \overline{t}(\overline{t} = (\top, \dots, \top)) \oplus \bigoplus_{\langle p,a,q,b,D\rangle \in \Delta D \in \{-1,0,1\}} \mathrm{wt}\langle p, a, q, b, D\rangle \otimes \beta_{\langle p,a,q,b,D\rangle}(\overline{t})$ where $\beta_{\langle p,a,q,b,1\rangle}(\overline{t}) := \exists p_1 \dots \exists p_k \exists q_1 \dots \exists q_k \exists s_1 \dots \exists s_k \Big((\overline{p} = \overline{q} + 1) \wedge (\overline{s} = \overline{t} + 1) \wedge H_p(\overline{p}, \overline{t}) \wedge T_a(\overline{p}, \overline{t}) \wedge H_q(\overline{q}, \overline{s}) \wedge T_b(\overline{p}, \overline{s})\Big)$,

$\beta_{\langle p,a,q,b,-1\rangle}(\overline{t}) := \exists p_1 \dots \exists p_k \exists q_1 \dots \exists q_k \exists s_1 \dots \exists s_k \Big((\overline{p} = \overline{q} - 1) \wedge (\overline{s} = \overline{t} + 1) \wedge H_p(\overline{p}, \overline{t}) \wedge T_a(\overline{p}, \overline{t}) \wedge H_q(\overline{q}, \overline{s}) \wedge T_b(\overline{p}, \overline{s})\Big)$,

$\beta_{\langle p,a,q,b,0\rangle}(\overline{t}) := \exists p_1 \dots \exists p_k \exists q_1 \dots \exists q_k \exists s_1 \dots \exists s_k \Big((\overline{p} = \overline{q}) \wedge (\overline{s} = \overline{t} + 1) \wedge H_p(\overline{p}, \overline{t}) \wedge T_a(\overline{p}, \overline{t}) \wedge H_q(\overline{q}, \overline{s}) \wedge T_b(\overline{p}, \overline{s})\Big)$,

which intuitively tells us that $\langle p, a, q, b, D\rangle$ is a transition made by some configuration in the computation in question. The point of χ is that when we get for some interpretations of $T_0, T_1, T_2, H_{q_0}, \dots, H_{q_{m-1}}$ an accepting computation $C_1 e_1 C_2 e_2 C_3 \dots C_n e_n C_{n+1}$, we want χ to give us its weight, $\nu(e_1)\nu(e_2) \dots \nu(e_n)$ as value. The order of the tuples $\langle t_1, \dots, t_k\rangle$ in the universal quantification in χ reflects their enumeration in the lexicographic order (indeed, our quantifiers respect the order of the structure and the evaluation of the quantifiers starts with the innermost and ends with the outermost quantifier). With all this in mind, ϕ is $\bigoplus T_0 \bigoplus T_1 \bigoplus T_2 \bigoplus H_{q_0} \dots \bigoplus H_{q_{m-1}} \chi$.

(*ii*): For this part, we reason similarly as before. In the proof of (1) we must observe that the semantics of multiplicative quantifiers can be now defined independently of the order thanks to the commutativity of the multiplication. In the proof of (2), we simply consider a Boolean formula $\theta(L)$ (which takes as only possible values 0 or 1) that expresses that the binary relation L is a suitable ordering, we take the formula

$$\bigoplus L \bigoplus T_0 \bigoplus T_1 \bigoplus T_2 \bigoplus H_{q_0} \dots \bigoplus H_{q_{m-1}}(\theta(L) \otimes \chi),$$

and we replace every formula $x \leq y$ by $L(x, y)$. Note that in (*i*), if multiplication is commutative, the value of ϕ is independent of any given order. Thus, idempotency of $\oplus$ guarantees that the value of

$$\bigoplus T_0 \bigoplus T_1 \bigoplus T_2 \bigoplus H_{q_0} \dots \bigoplus H_{q_{m-1}}(\theta(L) \otimes \chi)$$

is going to be that of

$$\bigoplus T_0 \bigoplus T_1 \bigoplus T_2 \bigoplus H_{q_0} \dots \bigoplus H_{q_{m-1}} \chi$$

when L is indeed an ordering. □

[1] Technically, we do not add these transitions to the machine but rather to the transition relation we use in our encoding.

Remark 1. Notice that, in part (1) of the proof of Theorem 1, the constructed weighted Turing machine uses as weights for the transitions, besides 0 and 1, the same weights that occur in the given formula ϕ. Analogously, in part (2), the constructed formula uses only the weights that appeared in the transitions of the given Turing machine. Moreover, the two constructions are effective for all semirings.

From Theorem 1 and Examples 1 and 2, we immediately obtain the following corollary:

Corollary 1. *For ordered structures in a finite vocabulary τ, we have that:*

(1) wESO[$\mathbb{B}$] *captures* NP *(originally proved in [2]).*
(2) wESO[$\mathbb{N}$] *captures* #P *(originally proved in [32] and [31]).*
(3) wESO[$\mathbb{Z}$] *captures* GapP *(originally proved in [32]).*
(4) wESO[$S_{\max}$] *and* wESO[$S_{\min}$] *respectively capture* MaxP *and* MinP *(originally proved in [32]).*
(5) wESO[$\mathbb{Z}_2$] *captures* $\bigoplus$P.
(6) wESO[$\mathbb{Z}_q$] *captures* MOD_q − P.
(7) wESO[$\mathbb{N}_{\max}$] *and* wESO[$\mathbb{N}_{\min}$] *respectively capture* MaxP[$O(\log n)$] *and* MinP[$O(\log n)$].
(8) wESO[$\mathcal{F}_*$] *captures the class of all fuzzy languages realizable by fuzzy Turing machines with t-norm $*$ in polynomial time.*
(9) wESO[$2^{\Sigma^*}_{\text{fin}}$] *captures* NPMV.
(10) wESO[$\mathbb{N}\langle\Sigma_2^*\rangle$] *captures the class of all multiset-valued functions computed by nondeterministic polynomial-time transducer machines with counting.*

Remark 2. It is worth observing that the proofs of (2)-(4) in [32] (Prop. 4.2, Cor. 4.8, and Thm. 4.10) are (as expected) different from ours. Our argument works in all those cases but neither of the three arguments given in [32] works for our more general setting. For example, the proof of (2) in [32] (Prop. 4.2) uses a specific characterization of #P from [31] that is not meant for arbitrary counting classes.

Our next application of the weighted Fagin's theorem consists in providing a natural computational problem complete for NP[S] for certain semirings S. Given a semiring S, alphabets Σ_1, Σ_2, and series $\sigma_1 : \Sigma_1^* \longrightarrow S$ and $\sigma_2 : \Sigma_2^* \longrightarrow S$, we say that σ_1 is *polynomially many-one reducible* to σ_2 ($\sigma_1 \leq_m \sigma_2$, in symbols) if there is an $f : \Sigma_1^* \longrightarrow \Sigma_2^*$ computable deterministically in polynomial time such that $\sigma_2(f(w)) = \sigma_1(w)$ for each $w \in \Sigma_1^*$. A series $\sigma : \Sigma^* \longrightarrow S$ is said to be NP[S]-*hard* if $\sigma' \leq_m \sigma$ for all σ' in NP[S]. If, moreover, σ belongs to NP[S], then it is called NP[S]-*complete*.

Fix an infinite set X. The language of the weighted propositional logic over a finitely generated semiring S is built from X as propositional variables, elements of S as truth-constants, and logical connectives $\wedge, \vee, \neg$ (where negation is only applied to propositional variables). Let Fmla[S] be the set of all formulas.

A *truth assignment* is a mapping $V : X \longrightarrow \{0,1\}$ extended to $\overline{V}$ for all formulas in the following way:

1. For each propositional variable $x \in X$, let $\overline{V}(x) := V(x)$ and $\overline{V}(\neg x) := 1$ iff $V(x) = 0$. Moreover, let $\overline{V}(a) := a$ for each $a \in S$.
2. $\overline{V}(\varphi \vee \psi) := \overline{V}(\varphi) + \overline{V}(\psi)$ and $\overline{V}(\varphi \wedge \psi) := \overline{V}(\varphi) \cdot \overline{V}(\psi)$.

For each formula $\varphi \in$ Fmla[S], let X_φ be the set of propositional variables that occur in φ. Clearly, $\overline{V}(\varphi)$ depends only on the values of V on X_φ. The 'problem' SAT[S] is the series $\sigma :$ Fmla[S] $\longrightarrow S$ defined as follows: $\mathrm{SAT}[S](\varphi) = \sum_{V \in \{0,1\}^{X_\varphi}} \overline{V}(\varphi)$.

The following corollary of our weighted version of Fagin's theorem has also appeared as [24, Thm. 6.3] with a direct proof. Our proof generalizes the reasoning for the Boolean case in [7].

Corollary 2 (Weighted Cook–Levin's theorem). *Let S be a finitely generated semiring. Then,* SAT[S] *is* NP[S]*-complete.*

Proof. By Theorem 1, we know that SAT[S] is in NP[S], so all that is left to show is that any series $\sigma : \Sigma^* \longrightarrow S$ (where Σ is an alphabet) recognizable in NP[S] is polynomially many-one reducible to the series SAT[S]. First, observe that the set of words Σ^* can be regarded as a set Struct$_<$[τ] of ordered finite structures for a vocabulary τ (namely, the vocabulary that has a unary predicate for each symbol of the alphabet). Thus, by the weighted Fagin's theorem, we have a wESO-formula ϕ such that $\|\phi\| = \sigma$. Our goal consists in finding a weighted propositional formula ϕ' such that $\|\phi'\| = \|\phi\|$.

We may assume that $\phi = \bigoplus P_1, \ldots P_n \psi$ as described in Theorem 1. Next, we polynomially associate any structure $\mathfrak{A} \in \mathrm{Struct}_<[\tau]$ with a propositional formula $\psi_\mathfrak{A} \in$ Fmla[S] such that $\mathrm{SAT}[S](\psi_\mathfrak{A}) = \|\phi\|(\mathfrak{A}) = \sigma(\mathfrak{A})$. Start by considering a propositional vocabulary $\{P_i^{\overline{a}} \mid i = 1, \ldots, n, \overline{a} \in A^{\mathrm{ar}(P_i)}\} \cup \{Q^{a=b}, Q^{a<b} \mid a, b \in A\}$ and suppose that we have first-order constants $\{a \mid a \in A\}$. Replace every quantifier $\exists x \theta(x)$ by the formula $\bigvee_{a \in A} \theta(x/a)$. Then, replace every quantifier of the form $\forall x \theta(x)$ by the formula $\bigwedge_{a \in A} \theta(x/a)$. Then, replace every formula of the form $a < b, a = b$ or $R(\overline{a})$ ($R \in \tau$) by its corresponding truth-value in $\mathfrak{A}$ (i.e. 0 or 1). Finally, replace every formula of the form $P_i(\overline{a})$ by the propositional variable $P_i^{\overline{a}}$. Notice that the second-order quantifiers are unproblematic because they basically represent the propositional satisfaction problem. The resulting propositional formula $\psi_\mathfrak{A}$ is such that

$$\mathrm{SAT}[S](\psi_\mathfrak{A}) = \sum_{V \in \{0,1\}^{X_{\psi_\mathfrak{A}}}} \overline{V}(\psi_\mathfrak{A}) = \sum_{\substack{I_i \subseteq A^{\mathrm{ar}(P_i)} \\ i=1,\ldots,n}} \|\psi\|(\mathfrak{A}, I_1, \ldots, I_1) = \|\phi\|(\mathfrak{A}).$$

□

Now it is natural to wonder what happens with other well-known descriptive complexity results. In the remainder of this section, we will tackle a few more of these. We start with the Immerman–Vardi's theorem, a result that first appeared for the Boolean case in the papers [3,4]. Our own approach is inspired by [32, Thm. 4.4] where a version of the result for the counting complexity class FP is provided using a weighted logic with the semiring $\mathbb{N}$. We must observe, however, that our proof is a generalization of that in [32] that works for all semirings and not only for $\mathbb{N}$.

We need some notation that we will use in the next few proofs. For a Boolean formula β and possibly weighted φ, we define the abbreviation $\beta \rhd \varphi = (\beta \otimes \varphi) \oplus (\neg\beta \otimes \mathbb{1})$, i.e.,

$$[\![\beta \rhd \varphi]\!](\mathfrak{A}, \rho) = \begin{cases} [\![\varphi]\!](\mathfrak{A}, \rho) & \text{if } \langle \mathfrak{A}, \rho \rangle \vDash \beta \\ \mathbb{1} & \text{otherwise.} \end{cases}$$

Theorem 2 (Weighted Immerman–Vardi's theorem)**.** *The logic* wLFP[S] *(with weights in a semiring S) captures the class* FP[S] *over ordered structures in the vocabulary τ.*

Proof. To show (1) from Definition 11, first note that every LFP-formula β can be evaluated in polynomial time and hence a polynomial time Turing machine can output 1 or 0 depending on whether β is satisfied or not. Also, for a semiring element s, the Turing machine outputting the term s for every input runs in constant and hence polynomial time.

Furthermore, FP[S] is closed under polynomial sums as we may compute a term of polynomially many summands, each of which is computable in polynomial time, in polynomial time. Similarly, FP[S] is closed under polynomial products. All this follows, by simply writing the outputs one after the other which is allowed by the sum/product alternations.

To show (2), suppose that $\sigma \in \mathsf{FP}[S]$, that is, $\sigma : \Sigma^* \longrightarrow \langle G \rangle$ for a finite $G \subseteq S$ and a finite alphabet Σ, and there exists a polynomial-time deterministic Turing machine, which given a word $\mathrm{enc}(\mathfrak{A})$, outputs a word $w_{\mathrm{enc}(\mathfrak{A})}$ in the algebra of terms $\mathcal{T}(G)$ such that $w_{\mathrm{enc}(\mathfrak{A})}$ evaluates to $\sigma(\mathrm{enc}(\mathfrak{A}))$. Then for some $l \in \mathbb{N}$, we have $|w_{\mathrm{enc}(\mathfrak{A})}| \leq |A|^l$ for all structures $\mathfrak{A}$, where A is the universe of $\mathfrak{A}$. Like in the proof of Theorem 1, we encode numbers in $\{0, \ldots, |A|^l - 1\}$ using tuples from A^l.

For each $s_p \in G = \{s_1, \ldots, s_\ell\}$, consider the language

$$\mathcal{L}_p = \{\langle \mathfrak{A}, \overline{a}_1, \overline{b}_1, \ldots, \overline{a}_k, \overline{b}_k \rangle \mid w_{\mathrm{enc}(\mathfrak{A})} = \sum_{i_1=1}^{m_1} \prod_{j_1=1}^{n_1} \cdots \sum_{i_k=1}^{m_k} \prod_{j_k=1}^{n_k} s_{i_1 j_1 \cdots i_k j_k} \text{ and } s_{\overline{a}_1 \overline{b}_1 \cdots \overline{a}_k \overline{b}_k} = s_p\}.$$

Note that $m_1, n_1, \ldots, m_k, n_k$ are all bounded by A^l. Then, $\mathcal{L}_p$ is recognizable in polynomial time, so by [3,4], there is an IFL-formula $\phi_p(\overline{x}_1, \overline{y}_1, \ldots, \overline{x}_k, \overline{y}_k)$ such that $\mathfrak{A} \vDash \phi_p(\overline{a}_1, \overline{b}_1, \ldots, \overline{a}_k, \overline{b}_k)$ iff $\langle \mathfrak{A}, \overline{a}_1, \overline{b}_1, \ldots, \overline{a}_k, \overline{b}_k \rangle \in \mathcal{L}_p$.

Now we take the wIFL-formula

$$\psi := \bigoplus \overline{x}_1 \bigotimes \overline{y}_1 \cdots \bigoplus \overline{x}_k \bigotimes \overline{y}_k \bigotimes_{p=1}^{\ell} (\phi_p(\overline{x}_1, \overline{y}_1, \ldots, \overline{x}_k, \overline{y}_k) \rhd s_p).$$

We have then that $\|\psi\|(\mathfrak{A}) = \sum_{i_1=1}^{m_1} \prod_{j_1=1}^{n_1} \cdots \sum_{i_k=1}^{m_k} \prod_{j_k=1}^{n_k} s_{i_1 j_1 \cdots i_k j_k} = \sigma(\mathrm{enc}(\mathfrak{A}))$. □

This theorem formally contains the following particular cases (observe, however, that we used (1) in the proof).

Corollary 3. *For ordered structures in a finite vocabulary τ, we have that:*

(1) wLFP[$\mathbb{B}$] *captures* P *(originally proved in [3,4]).*
(2) wLFP[$\mathbb{N}$] *captures* FP *(originally proved in [32]).*

Remark 3. Observe that using second-order Horn logic (which is known to capture P [51]) instead of least fixed-point logic, would not work for us, as in the weighted version one can encode a #P-complete problem (namely #HORNSAT). This was already noted in [32].

In the next result, wPFP[S] + $\{\prod X, \sum X\}$ will denote the logic that is obtained from wPFP[S] by the addition of the second-order quantitative quantifiers $\prod X$ and $\sum X$. Clearly, when $S = \mathbb{B}$, this is the same as second-order logic with partial fixed points. The Boolean counterpart of Theorem 3, namely that second-order logic extended with partial fixed points characterizes PSPACE is folklore, but a proof can be found in [52, Thm. 4]. The classical argument also uses the result for partial fixed-point logic in [4,5] stating that the logic characterizes PSPACE over ordered structures.

Theorem 3. *The logic* wPFP[S] + $\{\prod X, \sum X\}$ *(with weights in a semiring S) captures* FPSPACE[S] *over ordered structures in the vocabulary τ.*

Proof. To show (1) from Definition 11, we proceed like in Theorem 2. First, every PFP-formula β can be evaluated in PSPACE, so we may compute its characteristic function in PSPACE as well. Also, constant functions can be computed in PSPACE. Finally, FPSPACE[S] is closed under exponential sums since an exponential counter can be stored in polynomial space. Similarly, FPSPACE[S] is closed under exponential products.

To show (2), suppose that $\sigma \in \mathsf{FPSPACE}[S]$, that is, $\sigma : \Sigma^* \longrightarrow \langle G \rangle$ for a finite $G \subseteq S$ and a finite alphabet σ, and there exists a polynomial-space deterministic Turing machine which given a word $\mathrm{enc}(\mathfrak{A})$ outputs a word $w_{\mathrm{enc}(\mathfrak{A})}$ in the algebra of terms $\mathcal{T}(G)$ such that $w_{\mathrm{enc}(\mathfrak{A})}$ evaluates to $\sigma(\mathrm{enc}(\mathfrak{A}))$. Then, for some $l \in \mathbb{N}$, we have $|w_{\mathrm{enc}(\mathfrak{A})}| \leq 2^{|A|^l}$ for all structures $\mathfrak{A}$, where A is the universe of $\mathfrak{A}$. We encode numbers in $\{0, \ldots, 2^{|A|^l} - 1\}$ using subsets of A^l as follows.

Let $\pi : 2^{A^l} \longrightarrow \mathbb{N}$ be a function that lets $\pi(B)$ be the number of relations in 2^{A^l} that are smaller than B according to the following induced linear order on relations of arity l: $X <^* Y$ iff there is a $u \in Y \setminus X$ such that if $v > u$, $v \in Y$ iff $u \in X$. For each $s_p \in G = \{s_1, \ldots, s_\ell\}$, consider the language

$$\mathcal{L}_p = \{\langle \mathfrak{A}, B_1, C_1, \ldots, B_k, C_k\rangle \mid w_{\mathrm{enc}(\mathfrak{A})} = \sum_{i_1=1}^{m_1}\prod_{j_1=1}^{n_1}\cdots\sum_{i_k=1}^{m_k}\prod_{j_k=1}^{n_k} s_{i_1 j_1 \cdots i_k j_k},$$

$$B_1, C_1, \ldots, B_k, C_k \subseteq A^l \text{and} s_{\pi(B_1)\pi(C_1)\cdots\pi(B_k)\pi(C_k)} = s_p\}.$$

Notice that $m_1, n_1, \ldots, m_k, n_k$ are all bounded by 2^{A^l}. Then, $\mathcal{L}_p$ is recognizable in PSPACE as $s_{\pi(B_1)\pi(C_1)\cdots\pi(B_k)\pi(C_k)}$ can be computed in PSPACE and compared to s_p, so by [4,5], there is a PFP-formula $\phi_p(X_1, Y_1, \ldots, X_k, Y_k)$ such that $\langle \mathfrak{A}, B_1, C_1, \ldots, B_k, C_k\rangle \vDash \phi_p(X_1, Y_1, \ldots, X_k, Y_k)$ iff $\langle \mathfrak{A}, B_1, C_1, \ldots, B_k, C_k\rangle \in \mathcal{L}_p$.

Now we take the wPFP$[S] + \{\prod X, \sum X\}$-formula $\psi := \bigoplus X_1 \bigotimes Y_1 \cdots \bigoplus X_k \bigotimes Y_k \bigotimes_{p=1}^{\ell} (\phi_p(X_1, Y_1, \ldots, X_k, Y_k) \rhd s_p)$. We have then that $\|\psi\|(\mathfrak{A})$ is exactly $\sum_{i_1=1}^{m_1}\prod_{j_1=1}^{n_1}\cdots\sum_{i_k=1}^{m_k}\prod_{j_k=1}^{n_k} s_{i_1 j_1 \cdots i_k j_k} = \sigma(\mathrm{enc}(\mathfrak{A}))$. □

Again, Theorem 3 formally contains some important particular cases (observe, as before, that we used (1) in the proof of the theorem).

Corollary 4. *For ordered structures in a finite vocabulary τ, we have that:*

(1) wPFP$[\mathbb{B}] + \{\prod X, \sum X\}$ *captures* PSPACE *(folklore, cf. [52]).*
(2) wPFP$[\mathbb{N}] + \{\prod X, \sum X\}$ *captures* FPSPACE *(originally proved in [32]).*

Theorem 4. *The logic* wPFP$[S]$ *(with weights in a semiring S) captures* FPSPACE$_{poly}[S]$ *over ordered structures in the vocabulary τ.*

Proof. To show (1) from Definition 11, we need to prove that FPSPACE$_{poly}[S]$ is closed under the relevant semiring operations. We proceed as in the first half of Theorem 2.

To show (2), suppose that $\sigma \in$ FPSPACE$_{poly}[S]$, that is, $\sigma : \Sigma^* \longrightarrow \langle G\rangle$ for a finite $G \subseteq S$ and a finite alphabet Σ, and there exists a polynomial-space deterministic Turing machine with polynomial size output which given a word $\mathrm{enc}(\mathfrak{A})$ outputs a word $w_{\mathrm{enc}(\mathfrak{A})}$ in the algebra of terms $\mathcal{T}(G)$ such that $w_{\mathrm{enc}(\mathfrak{A})}$ evaluates to $\sigma(\mathrm{enc}(\mathfrak{A}))$. Then, similarly to the proof of Theorem 2, there exists some $l \in \mathbb{N}$ with $|w_{\mathrm{enc}(\mathfrak{A})}| \leq |A|^l$ for all structures $\mathfrak{A}$, where A is the universe of $\mathfrak{A}$.

For each $s_p \in G = \{s_1, \ldots, s_\ell\}$, consider the language

$$\mathcal{L}_p = \{\langle \mathfrak{A}, \overline{a}_1, \overline{b}_1, \ldots, \overline{a}_k, \overline{b}_k\rangle \mid w_{\mathrm{enc}(\mathfrak{A})} = \sum_{i_1=1}^{m_1}\prod_{j_1=1}^{n_1}\cdots\sum_{i_k=1}^{m_k}\prod_{j_k=1}^{n_k} s_{i_1 j_1 \cdots i_k j_k} \text{ and } s_{\overline{a}_1\overline{b}_1\cdots\overline{a}_k\overline{b}_k} = s_p\}.$$

$\mathcal{L}_p$ is recognizable in PSPACE. Therefore, by [4,5], we know that there is a PFP-formula $\phi_p(\overline{x}_1, \overline{y}_1, \ldots, \overline{x}_k, \overline{y}_k)$ such that:

$$\mathfrak{A} \vDash \phi_p(\overline{a}_1, \overline{b}_1, \ldots, \overline{a}_k, \overline{b}_k) \text{ iff } \langle \mathfrak{A}, \overline{a}_1, \overline{b}_1, \ldots, \overline{a}_k, \overline{b}_k\rangle \in \mathcal{L}_p.$$

Now we take the wPFP-formula

$$\psi := \bigoplus \overline{x}_1 \bigotimes \overline{y}_1 \cdots \bigoplus \overline{x}_k \bigotimes \overline{y}_k \bigotimes_{p=1}^{\ell} (\phi_p(\overline{x}_1, \overline{y}_1, \ldots, \overline{x}_k, \overline{y}_k) \rhd s_p).$$

We have then that $\|\psi\|(\mathfrak{A})$ is exactly $\sum_{i_1=1}^{m_1}\prod_{j_1=1}^{n_1}\cdots\sum_{i_k=1}^{m_k}\prod_{j_k=1}^{n_k} s_{i_1 j_1 \cdots i_k j_k} = \sigma(\mathrm{enc}(\mathfrak{A}))$. □

The following are important formal consequences of Theorem 4 (and again we used (1) in the proof).

Corollary 5. *For ordered structures in a finite vocabulary τ, we have that:*

(1) wPFP$[\mathbb{B}]$ *captures* PSPACE *(originally proved in [4,5]).*
(2) wPFP$[\mathbb{N}]$ *captures* FPSPACE$_{poly}$ *(originally proved in [32]).*

Theorem 5. *The logic* wDTC$[S]$ *(with weights in a semiring S) captures* FPLOG$[S]$ *over ordered structures in the vocabulary τ.*

Proof. To show (1) from Definition 11, again we need to prove that FPLOG$[S]$ is closed under the relevant semiring operations. We proceed as in the first half of Theorem 2 and note that FPLOG$[S]$ is closed under polynomial sums and products since polynomial counters can be stored in logarithmic space.

To show (2), suppose that $\sigma \in$ FPLOG$[S]$, that is, $\sigma : \Sigma^* \longrightarrow \langle G\rangle$ for a finite $G \subseteq S$ and a finite alphabet Σ, and there exists a logarithmic-space deterministic Turing machine such that given a word $\mathrm{enc}(\mathfrak{A})$ outputs a word $w_{\mathrm{enc}(\mathfrak{A})}$ in the algebra of terms $\mathcal{T}(G)$ such that $w_{\mathrm{enc}(\mathfrak{A})}$ evaluates to $\sigma(\mathrm{enc}(\mathfrak{A}))$. Again, since the output size of a logarithmic-space Turing machine is at most polynomial, there exists some $l \in \mathbb{N}$ with $|w_{\mathrm{enc}(\mathfrak{A})}| \leq |A|^l$ for all structures $\mathfrak{A}$, where A is the universe of $\mathfrak{A}$.

For each $s_p \in G = \{s_1, \ldots, s_\ell\}$, consider the language

$$\mathcal{L}_p = \{\langle \mathfrak{A}, \overline{a}_1, \overline{b}_1, \ldots, \overline{a}_k, \overline{b}_k\rangle \mid w_{\mathrm{enc}(\mathfrak{A})} = \sum_{i_1=1}^{m_1}\prod_{j_1=1}^{n_1}\cdots\sum_{i_k=1}^{m_k}\prod_{j_k=1}^{n_k} s_{i_1 j_1 \cdots i_k j_k} \text{ and } s_{\overline{a}_1\overline{b}_1\cdots\overline{a}_k\overline{b}_k} = s_p\}.$$

$\mathcal{L}_p$ is recognizable in DLOGSPACE, so by [3], we know that there is a DTC-formula $\phi_p(\overline{x}_1, \overline{y}_1, \dots, \overline{x}_k, \overline{y}_k)$ such that we have $\mathfrak{A} \vDash \phi_p(\overline{a}_1, \overline{b}_1, \dots, \overline{a}_k, \overline{b}_k)$ iff $\langle \mathfrak{A}, \overline{a}_1, \overline{b}_1, \dots, \overline{a}_k, \overline{b}_k \rangle \in \mathcal{L}_p$.

Now we take the wTC-formula

$$\psi := \bigoplus \overline{x}_1 \bigotimes \overline{y}_1 \cdots \bigoplus \overline{x}_k \bigotimes \overline{y}_k \bigotimes_{p=1}^{\ell} (\phi_p(\overline{x}_1, \overline{y}_1, \dots, \overline{x}_k, \overline{y}_k) \rhd s_p).$$

We have then that $\|\psi\|(\mathfrak{A})$ is exactly $\sum_{i_1=1}^{m_1} \prod_{j_1=1}^{n_1} \cdots \sum_{i_k=1}^{m_k} \prod_{j_k=1}^{n_k} s_{i_1 j_1 \cdots i_k j_k} = \sigma(\mathrm{enc}(\mathfrak{A}))$. □

Once more, we have the following important formal particular cases of Theorem 5 (and again we used (1) in the proof).

Corollary 6. *For ordered structures in a finite vocabulary τ, we have that:*

(1) wDTC[$\mathbb{B}$] *captures* DLOGSPACE *(originally proved in [3]).*
(2) wDTC[$\mathbb{N}$] *captures* FPLOG.

Remark 4. There appears to be a pattern behind the proofs of the preceding theorems. However, it is not obvious whether the theorems and their proofs can be fit into a common framework as the appropriate weighted quantifiers for the logical characterization are specific to the complexity class. For instance, FP[S] requires polynomial sums and products and is also closed under them, FPSPACE[S] on the other hand requires exponential sums and products. Other complexity classes may require additional restrictions to the quantifiers and closure properties under sums or products of a certain size may require arguments specific to the class.

We obtain the logic wFO[S] + {**path**} that we will use in Theorem 6 below by adding to wFO[S] the logical operation **path** with the following semantics (and the obvious syntax):

$$[\![\mathbf{path}\psi(\overline{x}, \overline{y})]\!](\mathfrak{A}, \rho) = \sum_{\substack{\overline{x}=\overline{x}_0, \dots, \overline{x}_\ell = \overline{y} \\ \ell \leq |A^k|}} \prod_{i=0}^{\ell-1} [\![\psi(\overline{x}_i, \overline{x}_{i+1})]\!](\mathfrak{A}, \rho),$$

where $\overline{x}$ and $\overline{y}$ are k-tuples of variables. Notice that the logic with **path** added is not wDTC since this new logical expression uses the semiring operations for its semantics, while the deterministic transitive closure operator is exclusively Boolean.

Theorem 6. *The logic* wFO[S] + {**path**} *(with weights in a semiring S) captures* L[S] *over ordered structures in the vocabulary τ.*

Proof. In order to establish (1) from Definition 11, we proceed like in the proof of Theorem 1 and construct, for every wFO[S] + {**path**}-formula ϕ, an L[S] Turing machine $\mathcal{M}$ with $\|\phi\| = \|\mathcal{M}\|$. We use the same encoding as in the proof of Theorem 1 and again assume that $A = \{0, \dots, n-1\}$. For every Boolean formula β, there exists a deterministic LOGSPACE Turing machine $\mathcal{M}'$ such that $\langle \mathfrak{A}, a_1, \dots, a_m \rangle \vDash \beta$ iff $\mathcal{M}'$ accepts $\langle \mathfrak{A}, a_1, \dots, a_m \rangle$, see also [53, Thm. 7.4.1]. We obtain $\mathcal{M}$ by weighing every transition in $\mathcal{M}'$ by $\mathbb{1}$. For the construction of $\mathcal{M}'$, the construction steps from the proof of Theorem 1 apply also here. For $\beta = R(x_1, \dots, x_l)$ and $\beta = X(x_1, \dots, x_l)$, it is only necessary to store a binary counter in order to find the correct position of the input encoding. For $\beta = \alpha \vee \gamma$, the space for the consecutive computations can be reused and is thus only the maximum of the space needed to check α and γ, respectively. The case where $\beta = \neg\alpha$ is clear and for $\beta = \exists x.\alpha$, storing x requires only $\log_2(n)$ bits and the space to compute α can be reused and is thus logarithmic by induction.

The induction steps for $\phi \in \{s, \psi \oplus \zeta, \psi \otimes \zeta, \bigoplus x.\psi, \bigotimes x.\psi\}$ can also be handled like in the proof of Theorem 1 by reusing space and since storing a variable is possible in logarithmic space. Finally, we consider the case $\phi = \mathbf{path}\psi(\overline{x}, \overline{y})$. By induction, there exists an L[S] Turing machine $\mathcal{M}'$ for ψ. We construct $\mathcal{M}$ as follows. First, $\mathcal{M}$ writes $\overline{x}$ and an arbitrary tuple $\overline{x}'$ on the tape; this uses $2\ell \log_2(n)$ cells, where ℓ is the length of the tuple $\overline{x}$. In addition, $\mathcal{M}$ initializes a counter with $k \log_2(n)$ bits to zero; that is, the counter may count up to $|A^k|$. Then $\mathcal{M}$ simulates $\mathcal{M}'$ with the two tuples on the tape. If $\overline{x}' = \overline{y}$, then $\mathcal{M}$ may go to a final state with weight $\mathbb{1}$ after the simulation. If the counter is not maxed out, $\mathcal{M}$ can increase the counter by one, replace the first tuple with the second tuple, guess a new second tuple, clear the tape contents of the previous simulation of $\mathcal{M}'$, and simulate $\mathcal{M}'$ again. The weights of all transitions used to initialize $\mathcal{M}$ and to restart the simulation of $\mathcal{M}'$ are given weight $\mathbb{1}$. In conclusion, $\mathcal{M}$ computes the sum of the weights of all paths of length at most $|A^k|$ from $\overline{x}$ to $\overline{y}$ in the weighted graph induced by ψ.

Now, for (2) from Definition 11, let $\mathcal{M}$ be an L[S] Turing machine. Ignoring the weights of $\mathcal{M}$, we may consider $\mathcal{M}$ as a LOGSPACE Turing machine. Then, by [53, Lemma 7.3.7][2] the configurations of $\mathcal{M}$ can be encoded by fixed length tuples $\overline{a}$ over A and there exists for every transition δ a formula $\mathrm{next}_\delta(\overline{x}, \overline{y})$ such that $\langle \mathfrak{A}, \overline{a}, \overline{b} \rangle \vDash \mathrm{next}_\delta(\overline{x}, \overline{y})$ iff $\overline{a}$ and $\overline{b}$ are valid configurations and $\overline{b}$ is the result of applying the transition δ to the configuration $\overline{a}$. Moreover, there exist FO formulas $\mathrm{initial}(\overline{x})$ and $\mathrm{accepting}(\overline{y})$ such that $\langle \mathfrak{A}, \overline{a} \rangle \vDash \mathrm{initial}(\overline{x})$ iff $\overline{a}$ is an initial configuration of M and $\langle \mathfrak{A}, \overline{b} \rangle \vDash \mathrm{accepting}(\overline{y})$ iff $\overline{b}$ is an accepting configuration of M. Then, for the formula

$$\phi = \bigoplus \overline{x}. \bigoplus \overline{y}.\mathrm{initial}(\overline{x}) \otimes \mathrm{accepting}(\overline{y}) \otimes \mathbf{path} \bigoplus_{\delta \in \Delta} \mathrm{wt}(\delta) \otimes \mathrm{next}_\delta(\overline{x}, \overline{y}),$$

we have $\|\phi\| = \|\mathcal{M}\|$. □

[2] Observe that the formula used in [53, Lemma 7.3.7(b)] to express β_δ uses a transitive closure operator but this is not necessary in our case. The reason is that, in contrast to us, [53] takes a machine M is to be f-space-bounded if for all strings u accepted by M, *there is an accepting run* which uses $f(|u|)$ cells before stopping [53, p.126].

Let us list again some particular cases of Theorem 6 (in contrast to the previous cases, these were not used in the proof, but only the mentioned results from [53]).

Corollary 7. *For ordered structures in a finite vocabulary τ, we have that:*

(1) Positive transitive closure logic (the restriction of transitive closure logic to formulas where the transitive closure operator never appears within a negation) captures NL *(originally proved in [54]).*
(2) wFO[$\mathbb{N}$] + {**path**} *captures* #L *(originally proved in [32]).*

To end the present section, we address the general and interesting open problem suggested in [27], regarding a Fagin theorem that characterizes the class $\mathcal{NP}[S]$ from [27, Def. 14]. We begin by observing that for the machine model in [27, Def. 12], Fagin's theorem will fail if the logic considered is wESO. This is essentially due to the fact that semiring Turing machines allow for arbitrary semiring values on the tape and can transition with these values. However, such a large set of transitions, is only actually needed when there are infinitely many semiring values in the input words.

Proposition 1. *Let S be a commutative semiring. There is a series $P \in NP[S]$ such that for no $\varphi \in$ wESO, $||\varphi|| = P$.*

Proof. By Theorem 1 (weighted Fagin's theorem), it suffices to find $P \in \mathcal{NP}[S]$ such that $P \notin \mathsf{NP}[S]$. Let S be any commutative non finitely generated semiring (e.g. the field of rational numbers) and $\mathcal{M} = \langle S, \emptyset, \{\iota, \lambda\}, \{\sqcup\}, \iota, \sqcup, \delta\rangle$ a semiring Turing machine, where $\delta = \{\langle \iota, s, \lambda, s, 1, s\rangle \mid s \in S\}$. Then, the behavior of $\mathcal{M}$ cannot be modeled by any weighted Turing machine, as the set of weights assigned to inputs by a weighted Turing machine are always contained in some finitely generated subsemiring of S. □

Therefore, one might reasonably further ask what kind of logic would capture the class $\mathcal{NP}[S]$. Observe that an obvious challenge here is that in the proof of Fagin's theorem at some point we need to encode in the logic by means of a sentence involving a long (but finite) disjunction what the legal transitions of our machine are.

By contrast to the above situation, we might ask a more restricted question in case we are trying to capture $\mathcal{NP}[S]$ over the class of all finite ordered structures. Recall that we are considering finite structures to be given via their binary encodings, and thus the relevant series in $\mathcal{NP}[S]$ are those that take as input merely binary strings. These series are not computed by semiring Turing machines (SRTMs) that involve infinitely many transitions because the input words do not involve semiring values. So let us consider now the modification of [27, Def. 12] that only allows semiring Turing machines to come with a finite set of transitions. In this case, we will easily see that their machine model coincides with ours.

Proposition 2. *Let S be a commutative semiring and allow only finitely many transitions in a semiring Turing machine. Then, $\mathcal{NP}[S] =$ NP[S], i.e. the NP class in the sense of [27] coincides with the NP class in our sense.*

Proof. The inclusion $\mathsf{NP}[S] \subseteq \mathcal{NP}[S]$ is not too difficult to see. For every weighted Turing machine $\mathcal{M} = \langle Q, \Gamma, \Delta, \nu, q_0, F, \square\rangle$ over a commutative semiring S, there exists an SRTM $\mathcal{M}' = \langle S, S', Q, \Gamma, q_0, \square, \delta'\rangle$ with the same behavior as $\mathcal{M}$. For this, choose S' as the set of all values assigned by ν and $\delta' = \{\langle p, a, q, b, d, s\rangle \mid \langle p, a, q, b, d\rangle \in \Delta, \nu(p, a, q, b, d) = s, s \in S\}$. Note that formally, SRTMs always have to move left or right, but introducing transitions which simulate this behavior using a right and a left move are a simple exercise. The restrictions imposed on SRTMs are clearly satisfied, as S' can neither read nor write semiring values, all transition weights are possible as S' contains all of the finitely many transitions weights, and $\mathcal{M}'$ cannot distinguish between semiring values as it it cannot even read them.

We continue with the inclusion $\mathcal{NP}[S] \subseteq \mathsf{NP}[S]$. Suppose that $P \in \mathcal{NP}[S]$, i.e. $P : \Sigma^* \longrightarrow R$ is a series such that there is an SRTM $\mathcal{M} = \langle S, S', Q, \Gamma, q_0, \square, \delta'\rangle$ that computes P in polynomial time. This mean that for any $x \in \Sigma^*$, the value of $\mathcal{M}$ on the configuration $\langle \iota, x, 0\rangle$ (where ι is the initial state and 0 the position of the head) is $P(x)$. We define a weighted Turing machine $\mathcal{M}' = \langle Q, \Gamma, \Delta, \nu, q_0, F, \square\rangle$ by setting $\Delta = \{\langle p, a, q, b, d\rangle \mid$ there is an $s \in S$ s.t. $\langle p, a, q, b, d, s\rangle \in \delta'\}$ and $\nu(p, a, q, b, d) = \sum_{\langle p,a,q,b,d,s\rangle \in \delta'} s$ (this is finite since there are only finitely many transitions in δ'). Observe that in the definition of a SRTM the same transition can be done with different weights, which is why in our weighted version we need to define this sum. Using distributivity of the semiring, then the function computed by $\mathcal{M}$ (i.e. a series) coincides with the behaviour of $\mathcal{M}'$. □

6. Conclusions and further work

In this paper, we have established a few central results in weighted descriptive complexity, providing quantitative versions of Fagin's theorem and the Immerman–Vardi theorem, among other logical characterizations of complexity classes. In future research, we plan to extend our weighted Fagin's theorem to the even larger class of valuation monoids containing all semirings and supporting average calculations by the theory developed in [55] for weighted finite automata over words and weighted EMSO logic. With Thomas Eiter and Rafael Kiesel, we also have a Fagin theorem [38] for the class $\mathcal{NP}[S]$ from [27]. We have some preliminary results in this direction that we plan to submit in due time.

CRediT authorship contribution statement

Guillermo Badia: Investigation; **Manfred Droste:** Investigation; **Carles Noguera:** Investigation; **Erik Paul:** Investigation.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests:

Guillermo Badia reports financial support was provided by Australian Research Council. If there are other authors, they declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgement

We thank the editor and the referees for their careful reading and remarks that helped us improving the article. Badia was partly supported by the Australian Research Council grant DE220100544. Badia and Noguera were also supported by the European Union's Marie Sklodowska–Curie grant no. 101,007,627 (MOSAIC project) and GNSAGA, gruppo nazionale di ricerca INdAM.

References

[1] G. Badia, M. Droste, C. Noguera, E. Paul, Logical characterizations of weighted complexity classes, 49th International Symposium on Mathematical Foundations of Computer Science (MFCS 2024 306 (2024) 1–14. Leibniz International Proceedings in Informatics (LIPIcs).
[2] R. Fagin, Generalized first-order spectra and polynomial-time recognizable sets, Complexity Comput. 7 (1974) 43–73.
[3] N. Immerman, Relational queries computable in polynomial time, Inf. Contr. 68 (1–3) (1986) 86–104.
[4] M. Vardi, The complexity of relational query languages, in: STOC 1982 Proceedings of the Fourteenth Annual ACM Symposium on Theory of Computing, 1982, pp. 137–146. Extended Abstract.
[5] S. Abiteboul, V. Vianu, Fixpoint extensions of first-order logic and datalog-like languages, in: Proceedings of Fourth Annual Symposium on Logic in Computer Science, IEEE Computing Society Press, 1989, pp. 71–79.
[6] J. Flum, H.D. Ebbinghaus, Finite Model Theory, Perspectives in Mathematical Logic, Springer, 1995.
[7] E. Grädel, P. Kolaitis, L. Libkin, M. Marx, J. Spencer, M. Vardi, Y. Venema, S. Weinstein, Finite Model Theory and Its Applications, Springer-Verlag, 2007.
[8] N. Immerman, Descriptive Complexity, Graduate Texts in Computer Science, Springer, 1999.
[9] L. Libkin, Elements of finite model theory, Texts in Theoretical Computer Science Springer, 2004.
[10] M.-P. Schützenberger, On the definition of a family of automata, Inf. Contr. 4 (2) (1961) 245–270.
[11] A. Salomaa, M. Soittola, Automata-theoretic aspects of formal power series, in: Monographs in Computer Science, Springer, 1978.
[12] T. Harju, J. Karhumäki, The equivalence problem of multitape finite automata, Theor. Comput. Sci. 78 (1991) 347–355.
[13] V.Y. Meitus, Decidability of the equivalence problem for deterministic pushdown automata, Cybern. Syst. Anal. 5 (1992) 20–45.
[14] G. Senizergues, The equivalence problem for deterministic pushdown automata is decidable, Proc. Int. Colloq. Autom. Lang. Program. ICALP 1997 1256 (1997) 671–681.
[15] J. Berstel, D. Perrin, C. Reutenauer, Codes and Automata, Cambridge University Press, 2009.
[16] Handbook of weighted automata, in: M. Droste, W. Kuich, H. Vogler (Eds.), Monographs in Theoretical Computer Science, Springer-Verlag, Berlin, Heidelberg, 2009.
[17] S. Eilenberg, Automata, Languages, and Machines, Academic Press,New York and London, 1974.
[18] W. Kuich, A. Salomaa, Semirings, Automata, Languages, Monographs in Theoretical Computer Science, Springer Verlag, 1985.
[19] J. Sakarovitch, Elements of Automata Theory, Cambridge University Press, 2009.
[20] J. Albert, J. Kari, Digital image compression, in: M. Droste, W. Kuich, H. Vogler (Eds.), Handbook of Weighted Automata, Berlin, Heidelberg, Springer-Verlag, 2009, pp. 453–479.
[21] M. Droste, P. Gastin, Weighted automata and weighted logics, Theor. Comput. Sci. 380 (2007) 69–86.
[22] C. Damm, M. Holzer, P. Mckenzie, The complexity of tensor calculus, Comput. Complexity 11 (2002) 54–89.
[23] C. Damm, M. Holzer, P. Mckenzie, The complexity of tensor calculus, in: Proceedings 15th Annual Conference on Computational Complexity Conference, IEEE Computing Society Press, 2000, pp. 70–86.
[24] P. Kostolányi, Weighted automata and logics meet computational complexity, Inf. Comput. 301 (2024) 105213.
[25] B.C. Bedregal, S. Figueira, On the computing power of fuzzy Turing machines, Fuzzy Sets Syst. 159 (9) (2008) 1072–1083.
[26] J. Wiedermann, Characterizing the super-Turing computing power and efficiency of classical fuzzy Turing machines, Theor. Comput. Sci. 317 (1–3) (2004) 61–69.
[27] T. Eiter, R. Kiesel, Semiring reasoning frameworks in AI and their computational complexity, J. Artif. Intell. Res. 77 (2023) 207–293.
[28] T. Eiter, R. Kiesel, On the complexity of sum-of-products problems over semirings, in: AAAI Conference on Artificial Intelligence, AAAI-21, 2021, pp. 6304–6311.
[29] A. Durand, A. Haak, J. Kontinen, H. Vollmer, Descriptive complexity of #PLXP functions: a new perspective, J. Comput. Syst. Sci. 116 (2021) 40–54.
[30] J. Kontinen, A logical characterization of the counting hierarchy, ACM Trans. Comput. Logic 10 (1) (2009) 1–21.
[31] S. Saluja, K. V. Subrahmanyam, M.N. Thakur, Descriptive complexity of #PLXP functions, J. Comput. Syst. Sci. 50 (3) (1995) 493–505.
[32] M. Arenas, M.M. noz, C. Riveros, Descriptive complexity for counting complexity classes, Logical Methods Comput. Sci. 16 (1) (2020) 42.
[33] M. Droste, E. Paul, A Feferman-Vaught decomposition theorem for weighted, MSO logic 43rd Int. Symp. Math. Found. Comput. Sci. (MFCS 76 (2018) 1–15.
[34] E. Grädel, V. Tannen, Semiring Provenance for First-Order Model Checking, 2017. arXiv:1712.01980.
[35] E. Grädel, V. Tannen, Provenance analysis for logic and games, Moscow J. Comb. Number Theory 9; 3, 2020 203–228. https://arxiv.org/abs/1907.08470. https://doi.org/10.2140/moscow.2020.9.203
[36] T.J. Green, G. Karvounarakis, V. Tannen, Provenance semirings, in: Proceedings of the TwentySixth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, 2007, pp. 31–40.
[37] G. Badia, P.G. Kolaitis, C. Noguera, Codd's theorem for databases over semirings, Proc. ACM Manag. Data 3 (2025) 26. https://doi.org/10.1145/3767713
[38] G. Badia, M. Droste, T. Eiter, R. Kiesel, C. Noguera, E. Paul, Fagin's Theorem for Semiring Turing Machines, 2025. arXiv:2507.18375.
[39] L. Valiant, The complexity of enumeration and reliability problems, SIAM J. Comput. 8 (3) (1979) 410–421.
[40] C.H. Papadimitriou, S. Zachos, Two remarks on the power of counting, in: 6th GI Conference in Theoretical Computer Science, 1983, pp. 269–275.
[41] S.A. Fenner, L.J. Fortnow, S.A. Kurtz, Gap-definable counting classes, J. Comput. Syst. Sci. 48 (1) (1994) 116–148.
[42] S. Gupta, Closure properties and witness reduction, J. Comput. Syst. Sci. 50 (3) (1995) 412–432.
[43] J.-Y. Cai, L.A. Hemachandra, Lecture Notes in Computer Science, Proceedings of the 6th Symposium on Theoretical Aspects of Computer Science, 349 Springer-Verlag, 1989 229–240.
[44] R.V. Book, T.J. Long, A.L. Selman, Qualitative relativizations of complexity classes, J. Comput. Syst. Sci. 30 (3) (1985) 395–413.
[45] M.W. Krentel, The complexity of optimization problems, J. Comput. Syst. Sci. 36 (3) (1988) 490–509.
[46] C. Alvarez, B. Jenner, A very hard log-space counting class, Theor. Comput. Sci. 107 (1) (1993) 3–30.
[47] C. Bergman, Universal Algebra: Fundamentals and Selected Topics, Chapman and Hall/CRC, 2011.
[48] C. Glasser, Space-efficient informational redundancy, J. Comput. Syst. Sci. 76 (8) (2010) 792–811.

[49] R.E. Ladner, Polynomial space counting problems, SIAM J. Comput. 18 (6) (1989) 1087–1097.
[50] Y. Gurevich, S. Shelah, Fixed-point extensions of first-order logic, Ann. Pure Appl. Logic 32 (1986) 265–280.
[51] E. Grädel, Capturing complexity classes by fragments of second-order logic, Theor. Comput. Sci. 101 (1) (1992) 35–57.
[52] D. Richerby, Logical Characterization of PSPACE, J.Marcinkowski, A.Tarlecki(Eds.) Computer Science Logic. CSL 2004 Lecture Notes in Computer Science, 3210 Springer, 2004 370–384.
[53] J. Flum, H.-D. Ebbinghaus, Finite Model Theory Springer, 1999.
[54] N. Immerman, Languages which capture complexity classes (preliminary report), in: Proceedings of STOC'83, 1983, pp. 347–354.
[55] P. Gastin, B. Monmege, A unifying survey on weighted logics and weighted automata - core weighted logic: minimal and versatile specification of quantitative properties, Soft Comput. 22 (4) (2018) 1047–1065.