HAL open science

New Perspectives on Semiring Applications to Dynamic Programming

Ambroise Baril, Miguel Couceiro, Victor Lagerkvist

To cite this version:

Ambroise Baril, Miguel Couceiro, Victor Lagerkvist. New Perspectives on Semiring Applications to Dynamic Programming. 2024. hal-04434071

HAL Id: hal-04434071

https://hal.science/hal-04434071v1

Preprint submitted on 2 Feb 2024

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Distributed under a Creative Commons Attribution - NonCommercial - NoDerivatives 4.0 International License

Highlights

New Perspectives on Semiring Applications to Dynamic Programming

Ambroise Baril[a], Miguel Couceiro[a], Victor Lagerkvist[b]

- We unify many algorithmic extension under the semiring formalism.
- We identify new relevant extensions that are part of this formalism.
- We construct two efficient algorithms for semiring extensions of two well-known computational problems.

New Perspectives on Semiring Applications to Dynamic Programming

Ambroise Baril[a], Miguel Couceiro[a], Victor Lagerkvist[b]

[a] *Université de Lorraine, CNRS, LORIA, Nancy, F-54000 Nancy, France*
[b] *Linköping University, Linköping, Sweden*

Abstract

Semiring algebras have been shown to provide a suitable language to formalize many noteworthy combinatorial problems. For instance, the Shortest-Path problem can be seen as a special case of the Algebraic-Path problem when applied to the tropical semiring. The application of semirings typically makes it possible to solve extended problems without increasing the computational complexity. In this article we further exploit the idea of using semiring algebras to address and tackle several extensions of classical computational problems by dynamic programming.

We consider a general approach which allows us to define a semiring extension of *any* problem with a reasonable notion of a certificate (e.g., an NP problem). This allows us to consider cost variants of these combinatorial problems, as well as their counting extensions where the goal is to determine how many solutions a given problem admits. The approach makes no particular assumptions (such as idempotence) on the semiring structure. We also propose a new associative algebraic operation on semirings, called Δ-product, which enables our dynamic programming algorithms to count the number of solutions of minimal costs. We illustrate the advantages of our framework on two well-known but computationally very different NP-hard problems, namely, Connected-Dominating-Set problems and finite-domain Constraint Satisfaction Problems (Csps). In particular, we prove fixed parameter tractability (FPT) with respect to clique-width and tree-width of the input. This also allows us to count solutions of minimal cost that, to the best of our knowledge, cannot be done by any algorithm in the literature.

Keywords:
Semiring, Dynamic Programming, Fixed Parameter Tractability, Constraint Satisfaction Problems, Connected Dominating Set

1. Introduction

In this article we investigate *semiring extensions* of computational problems. First, we take an algebraic viewpoint and define a novel semiring operation that increases the range of semiring algorithms. Second, we apply our semiring

framework algorithmically and construct fixed-parameter tractable algorithms for semiring extensions of Connected-Dominating-Set and the Constraint Satisfaction Problem (Csp).

1.1. Background

A *semiring* is an algebra $\mathcal{A} = (A, +, \times, 0_A, 1_A)$ where $(A, +, 0_A)$ is a commutative monoid, $(A, \times, 1_A)$ is a monoid, $\times$ distributes over $+$, and where 0_A is absorbing with respect to $\times$. Well-known examples include the Boolean semiring $(\{\bot, \top\}, \vee, \wedge, \bot, \top)$, the integer semiring $(\mathbb{N}, +, \times, 0, 1)$, and the tropical semiring $(\mathbb{R} \cup \{\infty\}, \min, +, \infty, 0)$. Semirings have proven to be versatile in extending various types of algorithms. A classic example is the Floyd-Warshall algorithm used for computing the minimal path between any two nodes in a weighted graph [15, 33]. Hence, in this problem we are interested in computing the minimum value of all sums of weights corresponding to paths in the given graph, and from the semiring perspective we are then simply evaluating an expression with respect to the tropical semiring. Extending the basic Floyd-Warshall algorithm to an arbitrary semiring is relatively easy and increases the expressive strength tremendously (see, e.g., Lehmann [26] for a general treatment). For example, if we use the Boolean semiring instead of the tropical semiring then the Floyd-Warshall algorithm can be used to compute the transitive closure of a given graph. Generally, when a problem is generalized by a semiring then the resulting problem is called a *semiring extension*. This problem has attracted significant attention and is in its most general form known as the *algebraic path* problem, whose roots can be traced back to Kleene's algorithm for converting regular expressions to finite automata [24]. For a comprehensive discussion of the literature, see e.g. Mohri [28]. However, semiring extensions are by no means limited to graph problems. Notable formalisms include the *semiring constraint satisfaction* problem (Scsp) by Bistarelli et al.[4] which greatly generalizes various forms of fuzzy reasoning where constraints are allowed to be soft (an arbitrary semiring value) rather than crisp (true or false). This problem is in turn subsumed by the *sum-of-product CSP* problem which is currently of central importance in artificial intelligence [1, 13].

In this article we are interested in semiring extensions and take a very general approach and define a semiring extension over *any* computational problem in NP. Thus, for any problem $\mathbf{\Pi}$ in NP and any reasonable notion of a solution space (e.g., the set of certificates) and any semiring we consider the problem of computing a *semiring expression* of the set of solutions of a given instance of $\mathbf{\Pi}$ (see Section 3 for a definition of a semiring expression). This strictly generalizes the problem of *counting* the number of solutions (typically denoted $\#\mathbf{\Pi}$) and the problem of finding a solution of minimal cost (Cost-$\mathbf{\Pi}$). Let us consider two examples. First, the well-known problem of determining whether an input graph G admits a homomorphism to the template graph H is known as the H-Coloring problem. Here, a natural certificate of a yes-instance is simply the homomorphism itself. Then $\#H$-Coloring is the problem of counting the number of homomorphisms, while in the Cost-H-Coloring each assignment of a variable to a value is associated with a weight, and the goal is to

find the homomorphism which minimizes the total sum of weights. Notice that the decision problem H-Coloring asks for an element of the Boolean semiring $\mathbf{2} = \{\bot, \top\}$ ($\top$ if a homomorphism exists and $\bot$ otherwise), and that the counting extension $\#H$-Coloring expects an answer in the natural semiring $(\mathbb{N}, +, \times, 0, 1)$. The cost version Cost-$H$-Coloring can be modelized using the tropical semiring $\mathbb{R}_{\min} = (\overline{\mathbb{R}}, \min, +, \infty, 0)$. Moreover, the well studied List-k-Coloring problem, where each vertex is restricted to a list of "allowed colors" [22, 31], can also be seen as an extension of k-Coloring, even if it also lies in the boolean semiring. The semiring extension of H-Coloring then generalizes all of these problems under a single umbrella, potentially allowing one to benefit from a single algorithm. For a second example, the Dominating-Set problem is the problem of deciding whether a given graph contains a subset of k vertices such that every other vertex has at least one neighboor in the subset. In the Connected-Dominating-Set problem, we, as the name suggests, additionally require that the subset of vertices is connected. This problem is important in, for example, network communication where the connected dominating set is viewed as network backbone that the other nodes can communicate via (for more applications see e.g. the book [10]). A natural certificate is then simply the subset of vertices, and the semiring extension of (Connected-)Dominating-Set then makes it possible to count the number of (connected) dominating sets as well as finding a (connected) dominating set of minimal cost.

Solving semiring extensions is thus highly desirable from a practical perspective since one effectively gets many algorithms for the price of one and can reuse the algorithm in different applications simply by choosing new semirings. From a complexity perspective many complexity classes have been introduced in order to study semiring extensions of NP problems (corresponding to the Boolean semiring $\mathbb{B}$ since they are decision problems). For example, the class $\#\mathsf{P}$ [32] for the counting extension (lying in the natural semiring $\mathbb{N}$), the class MOD_p [21] for the counting problems modulo an integer $p \geq 2$ (in the semiring $\mathbb{Z}_p$), and the class OptP for optimization problem [25] (often computed in the tropical semiring $\mathbb{R}_{\min}$). More generally, Eiter & Kiesel [13] define a semiring extension $\mathsf{NP}(\mathcal{R})$ of NP for any commutative semiring $\mathcal{R}$ and a notion of a semiring Turing machine. This makes it possible to prove a unifying meta-theorem which identifies the corresponding variant of Sat as a complete problem for each of the aforementioned classes. Despite this, there has been comparably little research done on *combining* semiring extensions. For example, consider problems consisting in counting the number of solutions of minimal cost (described by the class $\#\cdot\mathsf{OptP}$ [20]). Thus, intuitively, one wishes to combine the natural semiring $\mathbb{N}$ (counting) and the tropical semiring $\mathbb{R}_{\min}$ (weighted) into a new semiring with the hope of efficiently solving the associated problem of counting solutions of minimal cost. This raises an intriguing question: can counting solutions of minimal cost (or similar variations) be addressed using the same algorithmic techniques as counting problems or minimal cost problems alone? And can we accomplish this while staying in the familiar semiring framework? Or do these combinations yield fundamentally different problems?

1.2. Our contribution

We begin (in Section 3) by introducing our semiring framework. As remarked, we adopt a very general approach and define a semiring extension for *every* problem in NP. This drastically increases the reach of our semiring framework and makes it possible to study both the aforementioned finite domain Csp problem and the (Connected-)Dominating-Set even though these problems have a very different flair. Furthermore, to obtain as general results as possible we avoid the (otherwise commonly assumed) assumption that the semiring in question is idempotent. We remark that idempotence is typically assumed in the algebraic path problem as well as in the Scsp formalism by Bistarelli [4] via so-called *c-semirings*. Semiring approaches that requires the milder, but arguably less natural, assumptions such as the *k-closure*, was considered by Mohri [28] in the context of the Algebraic-Path problem. Idempotency is typically assumed since it in an algorithmic context makes it easier to combine (via union) two partial (but not necessarily disjoint) sets of solutions into a larger one. For instance, if we want to solve a problem of the form Cost-$\mathbf{\Pi}$, we can find the minimal cost in the union of two sets by computing the minimal costs in both sets, and keep the minimum. This approach is correct because of the idempotence of the underlying tropical semiring $\mathbb{R}_{\min}$. However, this approach would fail to solve the $\#\mathbf{\Pi}$ problem, as the cardinality of the union can only be expressed as the sum of the cardinality of both sets if the sets are disjoint. The approach fails because of the non-idempotence of the natural semiring $\mathbb{N}$. The basic objects in our approach is then that one is given a computational problem with a reasonable notion of a solution and is tasked with constructing a semiring expression representing the set of solutions of a problem instance. The interest is that individual components in the semiring expression can be interpreted as dynamic programming operations, e.g., combining two solutions to subproblems to a solution to a larger problem. Given this link one might suspect that polynomial sized semiring expressions cannot always be computed efficiently, and we indeed provide examples (in Section 3.6) where this is not possible (unless $\mathsf{P} = \mathsf{NP}$). Since this is a purely combinatorial statement it is an interesting open question whether this could be proved unconditionally.

In Section 4 we then completely resolve the problem of combining different semiring extensions. Our main technical tool is the Δ-*product* which, given two semirings $\mathcal{D}$ and $\mathcal{A}$ effectively produces a new semiring $\mathcal{D}\Delta\mathcal{A}$ which combines the relevant properties. More specifically, we assume that $\mathcal{A}$ is a commutative semiring and $\mathcal{D}$ a totally ordered, idempotent, commutative dioid, which allows it to represent a set of weights. The algorithmic applications of this new semiring correspond exactly to applying the semiring $\mathcal{A}$ to optimal solutions with respect to the weights in $\mathcal{D}$. In particular, by choosing $\mathcal{A} = \mathbb{N}$ and $\mathcal{D} = \mathbb{R}_{\min}$, it enables us to count the solutions of minimal cost. Our construction therefore makes the problems in the class $\#\cdot\mathsf{OptP}$ [20] fall into the category of semiring extensions. In particular, the results of completeness of the problems #Min-Card-Sat and #Min-Weight-Sat [20] now become particular improved cases of the meta-theorem over the completeness of any semiring variant of Sat [13].

Moreover, since the Δ-products implies that $\#\cdot\mathsf{OptP}$ is subsumed by the semiring formalism, any general positive result applicable to all semirings are now applicable specifically to these problems.

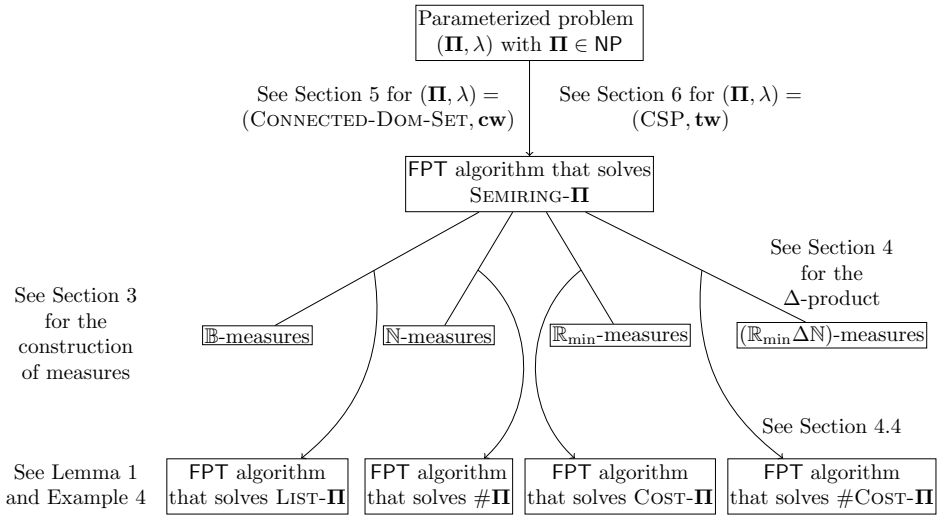

Figure 1: General overview of our contributions.

We showcase our framework on two distinct problems. First, in Section 5 we target the Connected-dominating-set problem under the lens of the *clique-width* of the input graph. Clique-width is, together with *tree-width*, the most well-known graph parameter, and measures how many basic operations that are required to construct the graph. The class of graphs (with labelled vertices) with clique-width at most $k \geq 1$ is defined as the smallest class of graphs that contains the one vertex graphs $\bullet_i$ with 1 vertex labelled by $i \in [k]$, and that is stable by the following operations for $(i, j) \in [k]^2$ with $i \neq j$: (i) disjoint union of graphs, (ii) relabelling every vertex of label i to label j, and (iii) constructing edges between every vertex labelled by i and every vertex labelled by j. Note that the class of cographs (which contains cliques) is exactly that of graphs with clique-width at most 2. For graph parameters such as clique-width the task is usually to prove *fixed parameter tractability* (FPT), i.e., an algorithm with a running time bounded by $f(k) \cdot ||I||^{O(1)}$ where (1) $f \colon \mathbb{N} \to \mathbb{N}$ is a computable function, (2) k is the parameter associated with the instance I, and (3) $||I||$ is the number of bits required to represent I. For classes of instances where the parameter k is believed to be reasonably small an FPT algorithm is thus essentially as good as a polynomial time algorithm. It is known that the Connected-dominating-set problem can be solved by a FPT (with respect to the clique-width of the input graph) one-sided error Monte-Carlo algorithm [17], as it can be solved in $O^*(5^{\mathbf{cw}})$ time, knowing that an exact algorithm solving this problem can not run in $O^*((5-\varepsilon)^{\mathbf{cw}})$ time for any $\varepsilon > 0$, unless the SETH

fails. The SETH [7] is a well-known conjecture within complexity theory which states that for all $\varepsilon > 0$, there exists $k \geq 3$ such that the k-SAT problem can not be solved in time $O^*((2-\varepsilon)^n)$ (where n is the number of variables).

Second, in Section 6 we turn our attention to the finite-domain *constraint satisfaction problem*, i.e., the problem of verifying whether a set of finite-domain constraints admits at least one satisfying assignment. For this problem we consider the tree-width of the *Gaifman graph* of an input instance and produce an FPT-algorithm for SEMIRING-CSP. The algorithm is based on the algorithm by Ganian et al. [16] for #CSP but is strictly more general since it can also solve the associated cost problem as well as counting solutions of minimal cost (via the Δ-product). We additionally remark that for every finite domain D our $O^*(|D|^k)$ (where k is the tree-width of the Gaifman graph) algorithm is optimal (under the SETH) since it as a special case includes the H-COLORING problem which is known to not admit an improved algorithm under the SETH via Okrasa & Rzążewski [29]. In Section 3.7 we also notice that our approach can be extended to solve the more general SUM-PRODUCT-CSP problem.

1.3. Organization of the article

In Figure 1 we summarize the role of each contribution to our novel semiring formalism. The upper half of the Figure 1 (that refers to Sections 5 and Section 6) deal with algorithmic results: we explain how to solve the SEMIRING extension of CONNECTED-DOMINATING-SET and CSP in FPT time (with respect to respectively the clique-width and the tree-width). Whereas, the lower half of Figure 1 refers to the algebraic results of Section 3 and Section 4: we construct algebraic tools that enables us to construct an algorithm that solves the counting, cost and list extensions of any NP problem, provided that an algorithm for the SEMIRING version is given.

2. Preliminaries

2.1. Sets and functions

For any integer n, we let $[n] := \{1, \ldots, n\}$. Given a set V, we denote by 2^V the set of subsets of V.

For two finite sets S and T let T^S be the set of functions from S to T. Note that its cardinal is $|T^S| = |T|^{|S|}$ and that, in particular, the cardinal of $T^\emptyset$ is $|T|^0 = 1$ since it contains the only function with codomain T and empty domain. For all $s \in S$ and $t \in T$, we denote by

$$(s \mapsto t) : \begin{array}{ccc} \{s\} & \mapsto & T \\ s & \mapsto & t \end{array}$$

the function with domain $\{s\}$ and codomain T, and that maps s to t. Given $S' \subseteq S$ and $f \in T^S$, *the restriction of f to S'* is the function $f|_{S'} \colon S' \to T$ defined by $f|_{S'}(s) = f(s)$, for every $s \in S'$.

For a set V and $S \subseteq V$, if there is no ambiguity on the set V, we denote by

$$\mathbf{1}_S : \begin{array}{ccc} V & \to & \{0,1\} \\ u & \mapsto & \begin{cases} 1 & \text{if } u \in S \\ 0 & \text{if } u \notin S \end{cases} \end{array}$$

the *indicator function* of S. Notice that the mapping of any $S \in 2^V$ to $\mathbf{1}_S$ is an isomorphism between 2^V and $\{0,1\}^V$.

2.2. Permutations

Let $n \geq 0$. We write $[n]$ for the set $\{1, \ldots, n\}$ and $\mathfrak{S}_n$ for the set of *permutations* of $[n]$. A permutation $\sigma \in \mathfrak{S}_n$ is denoted by $\sigma = \begin{pmatrix} 1 & \ldots & n \\ \sigma(1) & \ldots & \sigma(n) \end{pmatrix}$.

For $k \in \{2, \ldots, n\}$ and k pairwise distinct elements $a_1, a_2 \ldots, a_k$ of $[n]$, $(a_1 a_2 \ldots a_k)$ denotes the element of $\mathfrak{S}_n$ that maps a_1 to a_2, a_2 to a_3, $\ldots$, and a_k to a_1 (the other elements of $[n]$ are sent to themselves). A permutation $\sigma \in \mathfrak{S}_n$ is said to be a *k-cycle* if it is of the form $(a_1 a_2 \ldots a_k)$. We write $\mathfrak{C}_n$ for the set of n-cycles of $\mathfrak{S}_n$.

2.3. Graph and trees

By a *graph* we mean a tuple $G = (V_G, E_G)$ where V_G a finite set called the set of *vertices* of G, and E_G is an irreflexive and symmetrical relation over V_G called the set of *edges* of G. The set of vertices and edges of a graph G will always be denoted V_G and E_G. Given $S \subseteq V_G$, we let $G[S] := (S, E_G \cap 2^S)$ be the graph induced by S on G.

A *tree* is a connected graph $T = (N_T, E_T)$ with no cycle, N_T is said to be the set of *nodes* of T. A *rooted tree* is a pair (T, r) where $T = (N_T, E_T)$ is a tree and $r \in N_T$ is said to be the *root* of (T, r). Abusing notation, we will say that r is the root of T. The *ancestors* of a node $N \in N_T$ in a rooted tree (T, r) are the nodes on the unique simple path from r to N. A *descendant* of a node $N \in N_T$ is a node $N' \in N_T$ such that N is an ancestor of N'. The *children* of a node $N \in N_T$ are the descendant of N that are also neighbors of N. A *leaf* of a rooted tree is a node that has no children.

2.4. The dominating set problem

The Dominating-Set problem is a widely known and important NP-complete problem over graphs, which is not subsumed by the Csp formalism.

Given a graph G and $u \in V_G$, the *closed neighborhood* of u in G is the set $N_G[u] = \{v \in V_G \mid \{u, v\} \in E_G\} \cup \{u\}$.

Definition 1. A *dominating set* of a graph G is a subset $S \subseteq V_G$ such that $N_G[S] = V_G$.

The goal of the Dominating-Set problem is then to determine the smallest size of a dominating set of an input graph. We present here the optimization problem associated with the NP-problem.

Dominating-Set:
Input: A graph G.
Output: The smallest $k \geq 0$ such that there exists $S \subseteq V_G$ where $|S| \leq k$ and $N_G[S] = V_G$.

Recall that a semiring expression can only represent a set of functions. It is therefore natural to represent a subset $S \subseteq V_G$ by its indicator function $\mathbf{1}_S$.

In this section, we focus on the significatively more difficult variant where we in addition require that the dominating set is connected. This problem is important in, for example, network communication [10] and is essentially an entire research field in itself.

Connected-Dominating-Set:
Input: A graph G.
Output: The smallest $k \geq 0$ such that there exists $S \subseteq V_G$ where $|S| \leq k$, $N_G[S] = V_G$ and $G[S]$ is connected.

Such a set S (satisfying $N_G[S] = V_G$ and that $G[S]$ is connected) is called a *connected dominating-set*.

2.5. The constraint satisfaction problem

A finite set of relations Γ over a finite domain D is called a *constraint language*. We define the *constraint satisfaction problem* (Csp) over a constraint language Γ as follows.

Csp(Γ):
Input: $I = (V, C)$ where V is a set of variables and C a set of constraints $R(x_1, \ldots, x_m) \in C$ where $R \in \Gamma$ is an m-ary relation and $x_1, \ldots, x_m \in V$.
Question: Does there exist a function $f : V \mapsto D$ which satisfies all constraints, i.e., $(f(x_1), \ldots, f(x_m)) \in R$ for every $R(x_1, \ldots, x_m) \in C$?

The function $f\colon V \to D$ is sometimes said to be a *model*, or a *solution*. A function $f'\colon V' \to D$ with $V' \subseteq V$ which satisfies every constraint $R(x_1, \ldots, x_m) \in C$ where $x_1, \ldots, x_m \in V'$ is said to be a *partial solution*.

Remark 1. *The finite domain D of a* Csp *instance is the* codomain *of the valuations $f : V \mapsto D$ (their domain is the set of variables V).*

Let us also remark that the well-known problem of deciding whether an input graph G admits a homomorphism to a template graph H (the H-Coloring problem) can be seen as a particular case of Csp, by simply letting $\Gamma = \{E_H\}$. It is also easy to see that the Boolean satisfiability problem for instances in conjunctive normal form can be seen as a special case of Csps with Boolean domain.

H-Coloring:
Input: A graph G.
Question: Does there exist a function $f : V_G \to V_H$ which satisfies for all $\{u, v\} \in E_G$ that $\{f(u), f(v)\} \in E_H$?

For example, if we let K_k be the k-vertex clique, K_k-Coloring is an alternative formulation of the well-known computational problem of deciding whether a graph with n vertices can be colored with k distinct colors (k-Coloring).

The *Gaifman graph* [30] of an instance $\mathcal{I} = (V, C)$ of Csp is the graph $G = (V, E)$, with $\{x, y\} \in E$ if there exists a constraint $c \in C$ that involves both x and y (for all pair $\{x, y\} \subseteq V$). Note that in the case of H-Coloring, the Gaifman graph of any input graph G is G itself.

2.6. Computational complexity

Given two infinite sequences $(u_n)_{n\in\mathbb{N}}$ and $(v_n)_{n\in\mathbb{N}}$ in $\mathbb{R}^{\mathbb{N}}$, we write $u_n = O^*(v_n)$ if there exists a polynomial p such that $u_n = O(v_n \times p(n))$.

The class NP is the class of computational decision problems $\mathbf{\Pi}$ for which there exists a deterministic Turing machine M running in polynomial time and a polynomial p, such that for all $n \geq 0$ and $x \in \{0, 1\}^n$:

$$x \in \mathbf{\Pi} \iff \exists y \in \{0, 1\}^{p(n)},\ M(x, y) \text{ accepts},$$

in which case such an $y \in \{0, 1\}^{p(n)}$ is called a *certificate* for the instance x. Here, we stick to the fixed alphabet $\{0, 1\}$ to simplify the presentation, but, naturally, any other alphabet involving at least two symbols could be used instead.

Take a computational problem $\mathbf{\Pi}$ in NP. A certificate $y = y_1 \dots y_{p(n)} \in \{0, 1\}^{p(n)}$ of an instance of size $n \geq 0$ of $\mathbf{\Pi}$ (where p is polynomial) can be represented bijectively by the function

$$\tilde{y} : \begin{array}{ccc} [p(n)] & \to & \{0, 1\} \\ i & \mapsto & y_i \end{array}.$$

However, most of the time it is more natural to give a higher level representations of "solutions" (i.e., certificates) as functions that have other domain and codomain than $[p(n)]$ and $\{0, 1\}$. For instance, a "solution" of k-Coloring (with $k \geq 1$) on an input graph G will be seen as a function from V_G to $[k]$. Generally, we assume that the set of "solutions" of our problem $\mathbf{\Pi}$ on a given instance is a subset of T^S, for some sets finite sets S and T having polynomial size in the size of the input (which can safely be assumed since $\mathbf{\Pi}$ is in NP). In this context we refer to S and T as the *domain* and *codomain* of the solutions.

The class $\#\mathsf{P}$ is the class of the counting versions of the NP problems. Formally, the class $\#\mathsf{P}$ is described by the problems of the form:

Input: $x \in \{0,1\}^n$ with $n \geq 0$.
Output: How many $y \in \{0,1\}^{p(n)}$ are there such that $M(x,y)$ accepts ?

where M is a deterministic Turing machine running in polynomial time and p a polynomial.

Let $\mathbf{\Pi}$ be a computational problem, and $\mathcal{I}$ the set of instances of $\mathbf{\Pi}$. A *parameter* of $\mathbf{\Pi}$ is a function $\lambda : \mathcal{I} \to \mathbb{N}$ computable in polynomial time[1]. The computational problem $\mathbf{\Pi}$ is said to be *fixed-parameter tractable* (FPT) when parameterized by λ if there exists a computable function f and an algorithm that solves $\mathbf{\Pi}$ in time:

$$f(\lambda(x)) \times \|x\|^{O(1)}$$

on any instance $x \in \mathcal{I}$ (where $\|x\|$ is the size of x).

3. Semirings and Measures

In this section we introduce *semiring extensions* of computational problems and our method based on *measures*.

3.1. Computational problems extensions

Let $\mathbf{\Pi}$ be a problem in NP, and p a polynomial such that for every $n \geq 0$, the size of the certificates of instances of size n is bounded by $p(n)$. Note that $\mathbf{\Pi}$ can be reformulated as:

$\underline{\mathbf{\Pi}:}$
Input: An instance x.
Output: Does x have a certificate?

For example, it is easy to see that Csp is in NP for every finite-domain constraint language Γ since a satisfying assignment can be used as a certificate. It is common to consider some extensions of $\mathbf{\Pi}$ consisting in asking a more general question than determining whether the set of certificates (often simply called solutions) is empty or not. For example, let us consider the following extended problems.

$\underline{\#\mathbf{\Pi}:}$
Input: An instance x of $\mathbf{\Pi}$.
Output: How many solutions does x have?

[1] In the case of treewidth and cliquewidth, it is common to assume that the tree decomposition and cliquewidth expression are given.

The problem $\#\mathbf{\Pi}$ is refered as the *counting version* of the decision problem $\mathbf{\Pi}$. Notice that if the decision problem $\mathbf{\Pi}$ is in NP, then its counting version $\#\mathbf{\Pi}$ is in the class $\#\mathsf{P}$.

<u>List-$\mathbf{\Pi}$:</u>
Input: An instance x and for all $i \in [p(n)]$, $L_i \subseteq \{0, 1\}$.
Output: Does x have a solution $y_1, \ldots, y_{p(n)}$ such that for all $i \in [p(n)]$, $y_i \in L_i$?

In particular, the List-k-Coloring problem defined thereby is the well studied extension of k-Coloring where each vertex can be restricted to a list of "allowed colors" [22, 31].

<u>Cost-$\mathbf{\Pi}$:</u>
Input: An instance x, and for all $i \in [p(n)]$, $(C_i^0, C_i^1) \in \mathbb{R}^2$.
Output: What is the minimal $\sum_{i=1}^{p(n)} C_i^{y_i}$ for a solution $y_1, \ldots, y_{p(n)}$ of x ?

The goal of this section is to show how all of these extensions of $\mathbf{\Pi}$ are subsumed by one particular extension:

<u>Semiring-$\mathbf{\Pi}$:</u>
Input: An instance x.
Output: A semiring expression of the set of solutions of x.

See Definition 6 and 7 for the formal definition of a semiring expression of a set. Intuitively, the point of a semiring expression is to give a compact description of the set of solutions that factors the computations necessary to compute the counting, list and cost extensions. Note also that the Semiring-$\mathbf{\Pi}$ problem depends on the set of "solutions" considered for $\mathbf{\Pi}$. However, this set will most of the time be clear from the context (and since we assume that $\mathbf{\Pi}$ is in NP a set of certificates of polynomial size can always be chosen).

Note that the Semiring-$\mathbf{\Pi}$ problems do not belong to complexity classes of the form $\mathsf{NP}(\mathcal{R})$ (for any semiring $\mathcal{R}$) [13] since the output (a semiring expression) does not belong to any specific semiring.

3.2. Semirings and Dioids

Many algorithms that are able to solve counting versions of a problem can often be adapted to solve a cost version. This adaptation uses the fact that both algorithms implicitly rely on a common algebraic structure, the *semiring*.

Definition 2 (Semiring). A *semiring* is a structure $\mathcal{A} = (A, +, \times, 0_A, 1_A)$ such that $(A, +, 0_A)$ is a commutative monoid, $(A, \times, 1_A)$ is a monoid, $\times$ is distributive over $+$, 0_A is absorbing for $\times$, and $0_A \neq 1_A$. The semiring $\mathcal{A}$ is said to be *commutative* if $\times$ is commutative.

Fact 1. *Observe that a* ring *is a semiring* $\mathcal{A} = (A, +, \times, 0_A, 1_A)$ *where* $(A, +_A, 0_A)$ *is a group.*

Throughout this paper, we implicitly assume that the operations of the semiring can be performed in constant time. For more precise computational considerations we refer the reader to the model of semiring Turing machines [13].

Example 1. Typical examples of semirings subsume:

- the *Boolean semiring* $\mathbb{B} = (\mathbf{2}, \vee, \wedge, \bot, \top)$, where $\mathbf{2} := \{\bot, \top\}$. Here, $\bot$ and $\top$ are constants (to be interpreted as true and false, respectively), and $\vee$ and $\wedge$ denote the disjunction and conjunction.
- the *integer semiring* $(\mathbb{N}, +, \times, 0, 1)$, where $+$ and $\times$ respectively denote the usual addition and multiplication over the natural numbers $\mathbb{N}$, with their respective neutral elements 0 and 1, and
- the *tropical semiring* $\overline{\mathbb{R}}_{min} := (\overline{\mathbb{R}}, \min, +, \infty, 0)$ where $\overline{\mathbb{R}} = \mathbb{R} \cup \{\infty\}$ is the set of real numbers together with the neutral element ∞ for min. We also have the *dual tropical semiring* $\overline{\mathbb{R}}_{max} := (\overline{\mathbb{R}}, \max, +, -\infty, 0)$, with the neutral element $-\infty$ for max.

Throughout this paper, we will further assume that $\mathcal{A} = (A, +, \times, 0_A, 1_A)$ is *commutative*, that is, $x \times y = y \times x$, for every $x, y \in A$.

Remark 2. *Tropical semirings have been used in the context of finding the minimal cost of a path: the cost of the path is the product of the costs of its edges, and the best cost amongst two distinct paths is given by the sum. Here,* $+$ *is interpreted as the product of the tropical semiring, whereas* min *is interpreted as the sum.*

The properties entailed by the interpretations in Remark 2 are captured by notion of "dioid".

Definition 3 (Dioid). A *dioid* is a semiring $(D, \min, +_D, \infty_D, 0_D)$, where $\leq_D$ is the binary relation given by

$$\forall (d_1, d_2) \in D^2, d_1 \leq_D d_2 \iff \exists c \in D, \min(d_2, c) = d_1,$$

that orders D. Such a dioid D is said to be *idempotent* if $\forall d \in D, \min(d, d) = d$, and it is said to be *totally ordered* if the order $\leq_D$ is total, *i.e.*, $\forall d_1, d_2 \in D$, $d_1 \leq_D d_2$ or $d_2 \leq_D d_1$.

Example 2. The structures of Example 1 give rise to natural examples of dioids.

- The Boolean semiring $\mathbb{B} = (\{\bot, \top\}, \vee, \wedge, \bot, \top)$ is endowed with a totally ordered idempotent dioid structure with the ordering $\top \leq_{\mathbb{B}} \bot$. This ordering is rooted in Remark 2 where the elements of the dioid are thought of as weights to minimize.

- The natural semiring $(\mathbb{N}, +, \times, 0, 1)$ ordered by the usual $\geq$ on $\mathbb{N}$ is a totally ordered dioid, which is not idempotent.
- The tropical semiring $\overline{\mathbb{R}}_{min} = (\overline{\mathbb{R}}, \min, +, \infty, 0)$ is also a totally ordered idempotent dioid, its order coincides with the usual order $\leq$ on $\overline{\mathbb{R}}$.
- If we let ω the first infinite ordinal number [23], the structure $(\Omega, \min, +, \infty, 0)$ with $\Omega := \{a + b \times \omega \mid (a, b) \in \mathbb{N}^2\} \cup \{\infty\}$ (where the symbol ∞ is neutral for min and absorbing for $+$) is a totally ordered idempotent dioid. More generally, for any set Ω of ordinal numbers, where a neutral symbol ∞ for min has been added, $(\Omega, \min, +, \infty, 0)$ is a totally ordered idempotent dioid.

From now on, $\mathcal{D} = (D, \min, +, \infty_D, 0_D)$ will denote a totally ordered idempotent dioid. We remark that the assumption of total ordering will most of the time not be necessary in our results. However, it seems natural to require that the costs are totally ordered.

Let us now verify that the sum "min" of the dioid effectively returns the minimum of its two arguments relatively to the order $\leq_D$.

Property 1. *For a totally ordered idempotent dioid $\mathcal{D} = (D, \min, +, \infty_D, 0_D)$, the following properties hold.*

1. *For every $(d_1, d_2) \in D^2$, if $d_1 \leq_D d_2$, then $\min(d_1, d_2) = d_1$.*
2. *For every $(d_1, d_2, d_3) \in D^3$, if $d_1 \leq_D d_2$ and $d_1 \leq d_3$, then $d_1 \leq \min(d_2, d_3)$.*

Proof. (1). If $d_1 \leq_D d_2$, then there exists $d_3 \in D$ such that $d_1 = \min(d_2, d_3)$. By commutativity and associativity of min, it follows that

$$\min(d_1, d_2) = \min(\min(d_2, d_3), d_2) = \min(\min(d_2, d_2), d_3),$$

and, by idempotency of min, $\min(d_1, d_2) = \min(d_2, d_3) = d_1$.

(2). If $d_1 \leq_D d_2$ and $d_1 \leq d_3$, then $\min(d_1, d_2) = \min(d_1, d_3) = d_1$. Hence,

$$\min(\min(d_2, d_3), d_1) = \min(\min(d_1, d_2), d_3) = \min(d_1, d_3) = d_1.$$

Thus $d_1 \leq \min(d_2, d_3)$. □

3.3. Uplus and Join

In dynamic programming, one generally designs algorithms that inductively compute over incomplete sets of solutions (where some solutions are missing), or over sets of solutions that only cover a subset of the whole domain (meaning that some variables are not assigned any value). In the former case, these partial sets of solutions $\mathcal{F}_1$ and $\mathcal{F}_2$ are then combined together to form a larger set of solutions via union $\cup$ or disjoint union $\uplus$. In the latter case, two (or more) partial solutions f_1 and f_2 over disjoint subsets of the whole domain are combined to obtain a solution denoted by $(f_1 \bowtie f_2)$ over a larger domain, an operation that is isomorphic to Cartesian product. The idea is to establish a correspondence

between the semiring operations $+$ and $\times$ and the two operations for combining (partial) solutions.

To simplify the presentation, we let S and T be two finite sets. Recall from Section 3.1 that we aim to associate functions $S \to T$ with solutions to a given a computational problem. Under this interpretation, two disjoint subsets S_1 and S_2 of S correspond to domains of partial solutions, which can be combined into a solution with a larger domain.

Definition 4 (Disjoint union $\uplus$). Let $\mathcal{F}_1 \subseteq T^S$ and $\mathcal{F}_2 \subseteq T^S$, with $\mathcal{F}_1 \cap \mathcal{F}_2 = \emptyset$. The *disjoint union* $\mathcal{F}_1 \uplus \mathcal{F}_2$ is defined by

$$\mathcal{F}_1 \uplus \mathcal{F}_2 = \{f \in T^S \mid f \in \mathcal{F}_1 \text{ or } f \in \mathcal{F}_2\}.$$

Notice that $\emptyset$ is neutral for $\uplus$: for every $\mathcal{F} \subseteq T^S$. Indeed, $\mathcal{F} \uplus \emptyset$ is well defined and $\mathcal{F} \uplus \emptyset = \mathcal{F}$. We can observe the correspondence between $\emptyset$ and $\uplus$, and $0_{\mathcal{A}}$ and $+$ of a semiring $\mathcal{A}$. This observation will be further exploited in Section 3.5.

We proceed by describing the operator corresponding to the product $\times$ in the underlying semiring. Here, the idea is to combine partial functions f_1 and f_2 over two disjoint domains S_1 and S_2 into a function $f_1 \bowtie f_2$ with the greater domain $S_1 \uplus S_2$. The condition that S_1 and S_2 are disjoint is indeed necessary to ensure that $f_1 \bowtie f_2$ is well-defined.

Definition 5 (Join $\bowtie$). Given $f_1 \in T^{S_1}$ and $f_2 \in T^{S_2}$, we define the *join* $f_1 \bowtie f_2 \in T^{S_1 \uplus S_2}$ of f_1 and f_2 by

$$(f_1 \bowtie f_2) : s \mapsto \begin{cases} f_1(s) & \text{if } s \in S_1 \\ f_2(s) & \text{if } s \in S_2 \end{cases}$$

Then, given $\mathcal{F}_1 \subseteq T^{S_1}$ and $\mathcal{F}_2 \subseteq T^{S_2}$, the set $\mathcal{F}_1 \bowtie \mathcal{F}_2$ is defined as

$$\mathcal{F}_1 \bowtie \mathcal{F}_2 = \{f_1 \bowtie f_2 \mid (f_1, f_2) \in \mathcal{F}_1 \times \mathcal{F}_2\}$$

Note that $T^{\emptyset}$ is neutral for $\bowtie$: for every $\mathcal{F} \subseteq T^S$: $\mathcal{F} \bowtie T^{\emptyset}$ is well defined and $\mathcal{F} \bowtie T^{\emptyset} = \mathcal{F}$. Also, $\emptyset$ is absorbing for $\bowtie$: for every $\mathcal{F} \subseteq T^S$: $\mathcal{F} \bowtie \emptyset$ is well defined and $\mathcal{F} \bowtie \emptyset = \emptyset$. Thus, to relate to semirings, if $\bowtie$ corresponds to the product of the semiring, it is expected that $\emptyset$ corresponds to the zero, and that $T^{\emptyset}$ correspond to the unit. This remark justifies the links between $T^{\emptyset}$ and the unit element of the semiring developed in Section 3.5.

This operator, although typically not explicitly named, is central in dynamic programming. For example, "divide-and-conquer algorithms" works by solving smaller sub-problems and then "combining" the solutions of the small problems to a solution to the larger problem. The way small solutions are combined is often implicit with the aforementioned $\bowtie$ operator. However, it should be noted that this operator is typically not given an explicit name.

Example 3. Recall that a *cograph* G is by definition either

- the graph K_1 (ie. G is a single vertex),
- the disjoint union $G = G_1 \oplus G_2 := (V_{G_1} \uplus V_{G_2}, E_{G_1} \uplus E_{G_2})$ of two cographs G_1 and G_2 with disjoint set of vertices,
- the *joint union* $G = G_1 \otimes G_2 = (V_{G_1} \uplus V_{G_2}, E_{G_1} \uplus E_{G_2} \uplus \{\{u_1, u_2\} \mid (u_1, u_2) \in V_{G_1} \times V_{G_2}\})$ of two cographs G_1 and G_2 with disjoint sets of vertices.

As an example of an implicit use of the $\bowtie$ operator, note that deciding if a cograph G on n vertices is k-colorable (with $k \geq 1$) can be done in $O(kn)$ time by dynamic programming, by the following observations.

- K_1 is k-colorable.
- $G_1 \oplus G_2$ is k-colorable if and only if G_1 and G_2 are k-colorable.
- $G_1 \otimes G_2$ is k-colorable if and only if there exists $k' \in [k-1]$ such that G_1 is k'-colorable and G_2 is $(k-k')$-colorable.

Indeed, the second item relies implicitely on the fact that if G_1 and G_2 are both k-colorable by the k-colorings f_1 and f_2, then so is $G_1 \oplus G_2$, with the k-coloring $f_1 \bowtie f_2$. An equivalent remark also holds for the third item. Note that in order to efficiently compute a k-coloring of the cograph with this method, the $\bowtie$ operator would be used implicitly.

3.4. Semiring Expressions

We now turn to the problem of building a semiring expression representing the set of solutions of a problem instance.

Definition 6 (Syntax of an expression). A *semiring expression* E over the sets (S, T) is an element of the grammar:

$$E : \emptyset \mid T^{\emptyset} \mid (s \mapsto t) \mid E \uplus E \mid E \bowtie E$$

with $(s, t) \in S \times T$.

The *size* of an expression E denoted $\|E\|$ is its number of leaves (viewing E as a binary tree, where the leaves are labelled by $\emptyset$, $T^{\emptyset}$ and the $(s \mapsto t)$ for $(s, t) \in S \times T$, and where the internal nodes are labelled by either $\uplus$ or $\bowtie$).

The *domain* of an expression E (denoted $dom(E)$) is the set of all $s \in S$ such that E has a leaf of the form $(s \mapsto t)$ with $t \in T$.

Definition 7 (Semantic of an expression). Given a *semiring expression* E over the sets (S, T), its *semantic* $[E]$ is either a special symbol FAIL, or a subset of $T^{dom(E)}$ defined inductively as:

- $[\emptyset] = \emptyset$,

- $[T^{\emptyset}] = T^{\emptyset}$,
- $[s \mapsto t] = \{(s \mapsto t)\}$ for all $(s,t) \in S \times T$,
- $[E_1 \uplus E_2] = \begin{cases} \text{FAIL} & \text{if } [E_1] = \text{FAIL or } [E_2] = \text{FAIL} \\ [E_1] & \text{else, if } [E_2] = \emptyset \\ [E_2] & \text{else, if } [E_1] = \emptyset \\ \text{FAIL} & \text{else, if } dom(E_1) \neq dom(E_2) \\ \text{FAIL} & \text{else, if } [E_1] \cap [E_2] \neq \emptyset \\ [E_1] \uplus [E_2] & \text{else,} \end{cases}$
- $[E_1 \bowtie E_2] = \begin{cases} \text{FAIL} & \text{if } [E_1] = \text{FAIL or } [E_2] = \text{FAIL} \\ \text{FAIL} & \text{else, if } dom(E_1) \cap dom(E_2) \neq \emptyset \\ [E_1] \bowtie [E_2] & \text{else.} \end{cases}$

We say that a semiring expression E is *correct* if $[E] \neq$ FAIL. A *semiring expression of a set* $\mathcal{F} \subseteq T^S$ is a correct semiring expression E such that $\mathcal{F} = [E]$. Alternatively, we say that E *encodes* or *represents* $\mathcal{F}$ if $[E] = \mathcal{F}$.

3.5. Measures and Matrix Representations

Using dynamic programming, it is sometimes possible to compute a semiring expression of the set of solutions of an instance of a computational problem $\mathbf{\Pi}$. Then, in order to take advantage of this semiring expression to solve computational variants of $\mathbf{\Pi}$ (such as $\#\mathbf{\Pi}$ or Cost-$\mathbf{\Pi}$) we introduce the concept of a *measure*, which takes its values in a semiring while mapping the fundamental operations $\uplus$ and $\bowtie$ of the semiring expressions to the sum and product of the semiring.

Definition 8 (Measure). An $\mathcal{A}$-*measure* over the finite sets (S,T) is a function μ such that for all $S' \subseteq S$ and $\mathcal{F} \subseteq T^{S'}$, μ maps $\mathcal{F}$ to $\mu(\mathcal{F}) \in A$, respecting:

1. **zero axiom:** $\mu(\emptyset) = 0$,
2. **unit axiom:** $\mu(T^{\emptyset}) = 1$,
3. **additivity:** for $\mathcal{F}_1, \mathcal{F}_2 \subseteq T^{S'}$ disjoint: $\mu(\mathcal{F}_1 \uplus \mathcal{F}_2) = \mu(\mathcal{F}_1) + \mu(\mathcal{F}_2)$,
4. **elementary multiplicativity:** for all $f_1 \in T^{S_1}$ and $f_2 \in T^{S_2}$ (with S_1 and S_2 disjoint): $\mu(\{f_1 \bowtie f_2\}) = \mu(\{f_1\}) \times \mu(\{f_2\})$.

Once we know an expression of the set SOL of solutions of an instance of a computational problem, one deduces easily for any $\mathcal{A}$-measure μ how to compute $\mu(SOL)$ as a sum and product of the $\mu(\{s \mapsto t\})$ with $(s,t) \in S \times T$ by transforming the $\uplus$ into $+$, and the $\bowtie$ into $\times$.

Let us remark that the distributivity of $\times$ over $+$ in the semiring ensures that the elementary multiplicativity axiom of measures can be extended to true multiplicativity (over whole sets).

Property 2 (multiplicativity). *Let μ be a $\mathcal{A}$-measure. Let $\mathcal{F}_1 \subseteq T^{S_1}$ and $\mathcal{F}_2 \subseteq T^{S_2}$. Then $\mu(\mathcal{F}_1 \bowtie \mathcal{F}_2) = \mu(\mathcal{F}_1) \times \mu(\mathcal{F}_2)$.*

Proof. The proof follows from the following sequence of identities that are not difficult to verify:

$$\begin{aligned}
\mu(\mathcal{F}_1 \bowtie \mathcal{F}_2) &= \mu(\{f_1 \bowtie f_2 \mid (f_1, f_2) \in \mathcal{F}_1 \times \mathcal{F}_2\}) \quad \text{(By definition of } \mathcal{F}_1 \bowtie \mathcal{F}_2) \\
&= \mu(\biguplus_{(f_1,f_2)\in\mathcal{F}_1\times\mathcal{F}_2} \{f_1 \bowtie f_2\}) \\
&= \sum_{(f_1,f_2)\in\mathcal{F}_1\times\mathcal{F}_2} \mu(\{f_1 \bowtie f_2\}) \quad \text{(By additivity of } \mu) \\
&= \sum_{(f_1,f_2)\in\mathcal{F}_1\times\mathcal{F}_2} \mu(\{f_1\}) \times \mu(\{f_2\}) \quad \text{(By elementary multiplicativity of } \mu) \\
&= (\sum_{f_1\in\mathcal{F}_1} \mu(\{f_1\})) \times (\sum_{f_2\in\mathcal{F}_2} \mu(\{f_2\})) \quad \text{(By distributivity of } \times \text{ over } +) \\
&= \mu(\biguplus_{f_1\in\mathcal{F}_1} \{f_1\}) \times \mu(\biguplus_{f_2\in\mathcal{F}_2} \{f_2\}) \quad \text{(By additivity of } \mu) \\
&= \mu(\mathcal{F}_1) \times \mu(\mathcal{F}_2). \quad \square
\end{aligned}$$

We summarize the conclusion of Property 2 by saying that *measures are multiplicative*. Given a set $\mathcal{F} \subseteq T^S$, and a measure μ, we then realize that $\mu(\mathcal{F})$ only depends[2] on the values of $\mu(\{s \mapsto t\})$, for $(s,t) \in S \times T$. Indeed, remarking

$$\mathcal{F} = \biguplus_{f\in\mathcal{F}} \{f\} = \biguplus_{f\in\mathcal{F}} \bowtie_{s\in S} \{s \mapsto f(s)\},$$

it then follows from the properties of measures that

$$\mu(\mathcal{F}) = \sum_{f\in\mathcal{F}} \prod_{s\in S} \mu(\{s \mapsto f(s)\}).$$

Now that we have identified the minimal information to describe a measure, we can easily represent this as a matrix.

Definition 9 (Matrix of a measure). The *matrix of the measure* μ is the matrix $M_\mu = (\mu(\{s \mapsto t\}))_{(s,t)\in S\times T} \in A^{S\times T}$.

Reciprocally, we can associate a measure μ_M to every matrix $M \in A^{S\times T}$ by $\forall S' \subseteq S, \forall \mathcal{F} \subseteq T^{S'}$, $\mu_M(\mathcal{F}) := \sum_{f\in\mathcal{F}} \prod_{s\in S'} M[s, f(s)]$. It is indeed easy to verify that the function μ_M defined thereby is a measure, and that for all measure ν, $\mu_{M_\nu} = \nu$ and that for all matrix N, $M_{\mu_N} = N$.

The practical relevance of semiring expressions and measures is summarized in Lemma 1. Here, the link between semiring expressions and measures are ensured by the additivity and multiplicativity axioms of measures.

Lemma 1. *Given a correct semiring expression E over the finite sets S and T, and an A-measure μ represented by its matrix $M_\mu = (M_\mu[s,t])_{s,t} \in A^{S\times T}$, $\mu([E])$ can be computed in $O(\|E\|)$ time.*

[2]Similarly to how a linear application depend only on its image on a linear basis.

Proof. We inductively compute the result by following the structure of a semiring expression.

- If $E = \emptyset$ we can return 0 by the zero axiom.
- If $E = T^{\emptyset}$ we can return 1 by the unit axiom.
- If $E = (s \mapsto t)$ with $(s,t) \in S \times T$, we can return $M_\mu[s,t]$ by definition of the matrix M_μ of μ.
- If $E = E_1 \uplus E_2$, inductively compute $\mu([E_1])$ and $\mu([E_2])$. We can then return $\mu([E_1]) + \mu([E_2])$ by additivity.
- If $E = E_1 \bowtie E_2$, inductively compute $\mu([E_1])$ and $\mu([E_2])$. We can then return $\mu([E_1]) \times \mu([E_2])$ by multiplicativity (Property 2).

The algorithm performs as many operations as there are nodes in the semiring expression E seen as a binary tree. Recall that any binary tree with $m \geq 1$ leaves has $2m - 1$ nodes in total: the number of nodes of a semiring expression is linear in its size. Hence, we compute $\mu([E])$ in $O(\|E\|)$ time. □

The main interest of Lemma 1 lies in the fact that many usual computational extension of NP problems can be formulated as computing the image of the set of solutions by a semiring-measure.

Example 4. Given a computational problem $\mathbf{\Pi}$ of NP, the problems $\mathbf{\Pi}$, $\#\mathbf{\Pi}$, List-$\mathbf{\Pi}$ and Cost-$\mathbf{\Pi}$ all consists in computing the image by a measure of the set of solutions SOL of $\mathbf{\Pi}$. For all of these extensions, let us now consider the semiring and the matrix of the corresponding measure.

- Solving $\mathbf{\Pi}$ means computing $\mathbf{1}_{\neq\emptyset}(\text{SOL})$, with $\mathbf{1}_{\neq\emptyset} : \mathcal{F} \mapsto \begin{cases} \top & \text{if } \mathcal{F} \neq \emptyset \\ \bot & \text{if } \mathcal{F} = \emptyset \end{cases}$

  $\mathbf{1}_{\neq\emptyset}$ is a $\mathbb{B}$-measure, and its matrix only has $\top$ coefficients.
- Solving $\#\mathbf{\Pi}$ means computing $\#(\text{SOL})$, with $\# : \mathcal{F} \mapsto |\mathcal{F}|$.

  $\#$ is a $\mathbb{N}$-measure, and its matrix only has 1 as coefficients.
- Solving List-$\mathbf{\Pi}$ means computing $L(\text{SOL})$ with L the $\mathbb{B}$-measure given by: for all $(s,t) \in S \times T$ $M_L[s,t] = \top$ if sending s to t is allowed, and $M_L[s,t] = \bot$ if it is forbidden.
- Solving Cost-$\mathbf{\Pi}$ means computing $C(\text{SOL})$ with C the $\overline{\mathbb{R}}_{\min}$-measure of the cost matrix $M_C[s,t] \in \overline{\mathbb{R}}$ is the cost of mapping s to t.

3.6. Computing semiring expressions

In order to take advantage of Lemma 1, it is useful to be able to compute a semiring expressions of small size representing the set of solutions of the considered problem. Note that small semiring expression can encode large sets. For instance, for $n \geq 0$, the set $[n]^{[n]}$ of all functions from $[n]$ to $[n]$ has the semiring expression of size n^2:

$$E_{[n]^{[n]}} = \bigtimes_{i=1}^{n} (\biguplus_{j=1}^{n} (i \mapsto j)).$$

However, the semiring expression does not seem relevant to efficiently encode the set $\mathfrak{S}_n$ of all permutations over $[n]$. Intuitively, the $E_1 \bowtie E_2$ operations does not allow one to chose the image in the domain of E_2 dependently from the images of the elements chose in E_1. Indeed, as we will now prove, such a semiring expression is unlikely to be computable in polynomial time.

Theorem 1. *Unless* P=NP*, a semiring expression of* $\mathfrak{S}_n$ *can not be computed in time* $poly(n)$*.*

Proof. Recall the definition of the Permanent problem:

Permanent:
Input: A matrix $M \in \{0,1\}^{n \times n}$ with $n \geq 0$.
Output: The permanent of M, ie. $perm(M) := \sum_{\sigma \in \mathfrak{S}_n} \prod_{i=1}^{n} m_{i,\sigma(i)}$.

The Permanent problem is well-known to be #P-complete due to Valiant's theorem [32]. Therefore, if it is solvable in $poly(n)$ time, then P=NP.

Assume that there exists an algorithm that takes an integer n as input and returns a semiring expression $E_{\mathfrak{S}_n}$ of $\mathfrak{S}_n$. This expression has size $poly(n)$ since it has been computed in $poly(n)$ time.

Let $M \in \{0,1\}^{n \times n}$ with $n \geq 0$ be an instance of Permanent. Then, by Lemma 1 the $\mathbb{N}$-measure μ_M of matrix M applied to the set $\mathfrak{S}_n$ can be computed in $poly(n)$ time, i.e., $\mu_M(\mathfrak{S}_n) = \sum_{\sigma \in \mathfrak{S}_n} \prod_{i=1}^{n} m_{i,\sigma(i)} = perm(M)$ can be computed in $poly(n)$ time. This proves that P=NP. □

By Theorem 1, we know that a semiring expression of $\mathfrak{S}_n$ is unlikely to be computable in $poly(n)$ time. We view it as an interesting open question to prove this unconditionally and pose the following conjecture.

Conjecture 1. $\mathfrak{S}_n$ *does not have a semiring expression of* $poly(n)$ *size.*

Note that a semiring expression $E_{\mathfrak{S}_n}$ of $\mathfrak{S}_n$ can be defined recursively by $E_{\mathfrak{S}_{n+1}} = \biguplus_{m \in [n+1]} ((n+1 \mapsto m) \bowtie E_{\mathfrak{S}_n}[m \leftarrow n+1])$, with $E_{\mathfrak{S}_n}[m \leftarrow n+1]$ being a copy of $E_{\mathfrak{S}_n}$ where every leaf of the form $(i \mapsto m)$ have been replaced by a leaf $(i \mapsto n+1)$. However, this would result in a formula with $n!$ leaves. The issue is that the replacement of leaves must be operated dependently of the

image we have chosen for $n+1$, which rules out the possibility of an efficient factorization of the expression.

A similar result can be proven for the set $\mathfrak{C}_n$ of n-cycles of $\mathfrak{S}_n$.

Theorem 2. *Unless* P=NP*, a semiring expression of* $\mathfrak{C}_n$ *can not be computed in* $poly(n)$ *time.*

Proof. Recall the definition of the Travelling-Salesman problem.

Travelling-Salesman:
Input: A graph G on $n \geq 0$ vertices, and a weight function $w : E_G \mapsto \mathbb{R}_+$.
Output: The minimal weight of a Hamiltonian cycle in G.

The Travelling-Salesman problem is known to be NP-complete. Therefore, if it is solvable in time $poly(n)$, then P=NP. Assume that there exists an algorithm that takes as an input an integer n, and returns a semiring expression $E_{\mathfrak{C}_n}$ of $\mathfrak{C}_n$. This expression has $poly(n)$ size since it has been computed in $poly(n)$ time.

Let (G, w) an instance of Travelling-Salesman. Without loss of generality, assume that $V_G = [n]$ with with $n := |V_G|$. Consider the matrix $W = (w_{i,j}) \in \overline{\mathbb{R}}^{[n]\times[n]}$, with for all $(i,j) \in [n]^2$,

$$w_{i,j} = \begin{cases} w(\{i,j\}) & \text{if } \{i,j\} \in E_G \\ \infty & \text{if } \{i,j\} \notin E_G \end{cases}$$

Then, by Lemma 1, the $\overline{\mathbb{R}}_{min}$-measure μ_W (of the tropical semiring $(\overline{\mathbb{R}}, \min, +, \infty, 0)$) of the matrix W applied to the set $\mathfrak{C}_n$ can be computed in $poly(n)$ time, i.e., $\mu_W(\mathfrak{C}_n) = \min_{c \in \mathfrak{C}_n} \sum_{i=1}^{n} w_{i,c(i)}$ can be computed in $poly(n)$ time. This solves Travelling-Salesman in polynomial time, implying that P=NP. □

Again, Theorem 2 indicates that a semiring expression of $\mathfrak{C}_n$ of size $poly(n)$ is unlikely to exist, and we pose the following explicit conjecture.

Conjecture 2. *The set* $\mathfrak{C}_n$ *of* n*-cycles of* $\mathfrak{S}_n$ *does not have a semiring expression of size* $poly(n)$*.*

One can also remark that Conjecture 2 implies that a semiring expression of size $poly(n)$ does not exists for the set of *all cycles* of $\mathfrak{S}_n$. Indeed, if such an expression exists, we can use it to obtain a semiring expression of size $poly(n)$ of $\mathfrak{C}_n$ by replacing every leaf of the form $(i \mapsto i)$ by $\emptyset$, contradicting Conjecture 2.

3.7. Differences with Sum-Product-Csp

At first, the concept of a measure may seem very reminiscent of the Sum-Product-Csp framework [12, 14]. Here, we are given a semiring $\mathcal{A}$, a domain D and a set of constraints Γ where each constraint is a polynomial time computable

function $C : D^{ar(C)} \mapsto A$, where $ar(C) \geq 0$ is the *arity* of C. The problem Sum-Product-Csp(Γ) is then defined as follows.

Sum-Product-Csp(Γ) :

Input: A set of variable V, and a finite set of constraints $\{(C_i, x_i^1, \ldots, x_i^{ar(C)}) \mid i \in [m]\}$ with $m \geq 0$, and $\forall i \in [m], C_i \in \Gamma$.

Output: The value $\sum_{f \in D^V} \prod_{i=1}^{m} C_i(f(x_i^1), \ldots, f(x_i^{ar(C)})) \in A$.

Note that the constraints defined in the context of Sum-Product-Csp subsumes the less general contraints defined for Csp, as a relation C over D of arity $ar(C)$ can be seen as a function $C : D^{ar(C)} \mapsto \{\bot, \top\}$ taking its value in the Boolean semiring. With this identification, Csp can be seen as a the particular case of Sum-Product-Csp, where the only semiring considered is the Boolean semiring $\mathbb{B}$.

However, the general Sum-Product-Csp is not subsumed by the Semiring-Csp problem. The principal difference is that we in the Sum-Product-Csp problem use the semiring to to relax the constraints, that can now take arbitrary values in a semiring instead of only $\top$ or $\bot$ in the Boolean semiring. In comparison, in our Semiring approach we can only work with a combination of arbitrary arity "hard constraints" (either true or false), and only unary "relaxed constraints" (taking values in the semiring). However, recall that our approach applies to every NP problem and not only Csp, since it is defined as long as the considered problem has a "set of solutions", for instance the set of certificate of the NP problem. For example, the Connected-Dominating-Set problem does not benefit from the Sum-Product-Csp formalism, but the Semiring-Connected-Dominating-Set problem is perfectly well defined, and subsumes the counting, list and cost version of Connected-Dominating-Set.

Moreover, even if we restrict the study to Csps, the Sum-Product-Csp approach is often too general to benefit from unified algorithms, and we frequently need to distinguish cases depending on the nature of the semiring [12]. In contrast, any upper bound on the complexity of Semiring-Csp is automatically derived as an upper bound of Csp, #Csp, List-Csp, and Cost-Csp.

4. Δ-product

We have seen that semiring extensions generalizes many natural extensions of computational problems such as counting, list and cost. We now explain how to *combine* some of these algorithmic extensions together by defining a novel operation over semirings called the Δ*-product*. As a concrete application, we will see that Lemma 1 enables to "count the number of solutions of minimal cost". Therefore, solving the semiring version of a NP problem is sufficient to count its solution of minimal cost. To the best of our knowledge, the algebraic constructions we present are completely novel, and no general semiring based algorithms for counting solutions of minimal cost exist in the literature. This new tool enables to create a semiring that makes the class #·OptP [20] fall

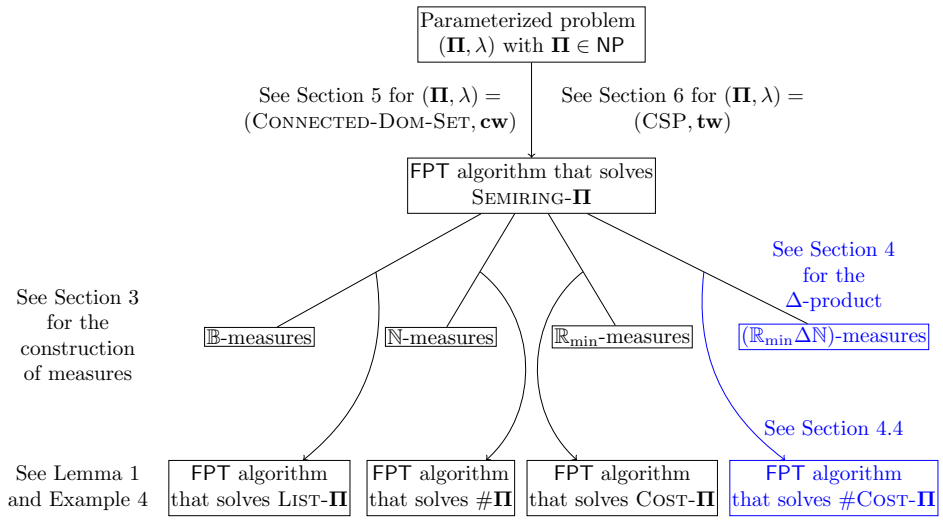

Figure 2: A general overview of our contributions where the contribution in this section is colored in blue.

into the semiring formalism, similarily as the classes #P [32] corresponds to the natural semiring $(\mathbb{N}, +, \times, 0, 1)$ for instance, via the $\mathbb{N}$-measure $\# : \mathcal{F} \mapsto |\mathcal{F}|$.

The contribution of this section to the semiring formalism presented in this paper is summarized in Figure 2.

For the rest of this section, we let $\mathcal{D} = (D, \min, +_D, \infty_D, 0_D)$ be a totally-ordered idempotent commutative dioid and $\mathcal{A} = (A, +_A, \times, 0_A, 1_A)$ be a commutative semiring.

4.1. Regularity

We begin by introducing the notion of multiplicative regularity. Let w a $\mathcal{D}$-measure over some finite sets (S, T), and $S' \subseteq S$. For $\mathcal{F} \subseteq T^{S'}$, we say that a function $f \in \mathcal{F}$ is of *minimal weight* in $\mathcal{F}$ (with respect to w) if $w(\{f\}) = w(\mathcal{F})$. This definition is motivated by the fact that $w(\mathcal{F}) = \min_{f \in \mathcal{F}} w(\{f\})$ (since $\mathcal{F}$ is a $\mathcal{D}$-measure). We also denote by $\operatorname{argmin}_w(\mathcal{F})$ the sets of elements of minimal weight of $\mathcal{F}$ (with respect to w).

Given a semiring expression E of $\mathcal{F}$, in order to count the number of solutions of minimal weight, we in particular need to handle the case where E_1 is of the form $E_1 \bowtie E_2$ in order to perform a structural induction. Ideally, it would be desirable to have $\operatorname{argmin}_w([E]) = \operatorname{argmin}_w([E_1]) \bowtie \operatorname{argmin}_w([E_2])$. The issue is that this equality is not always true, an instance where it fails is is illustrated in Example 5.

Example 5. Let S_1 and S_2 be two disjoint subsets of S, $f_1 \in T^{S_1}$, $f_2 \in T^{S_2}$ and $f_2' \in T^{S_2}$, with $f_2 \neq f_2'$. Let also $\mathcal{F}_1 := \{f_1\}$ and $\mathcal{F}_2 := \{f_2, f_2'\}$. Finally,

let w be a $\overline{\mathbb{R}}_{min}$-measure (i.e., of the tropical semiring) with $w(\{f_1\}) = \infty$, $w(\{f_2\}) = 7$ and $w(\{f_2'\}) = 31$.

- We have $\text{argmin}_w(\mathcal{F}_1 \bowtie \mathcal{F}_2) = \{f_1 \bowtie f_2, f_1 \bowtie f_2'\}$, as both functions $f_1 \bowtie f_2$ and $f_1 \bowtie f_2'$ have weight $\infty = \infty + 7 = \infty + 31$ by multiplicativity of w (Property 2). Recall that $+$ is the *product* of the tropical semiring.
- However, $\text{argmin}_w(\mathcal{F}_2) = \{f_2\}$, since $w(f_2) = 7 < 31 = w(f_2')$. Thus $\text{argmin}_w(\mathcal{F}_1) \bowtie \text{argmin}_w(\mathcal{F}_2) = \{f_1 \bowtie f_2\}$.

We see that in this case, $\text{argmin}_w(\mathcal{F}_1 \bowtie \mathcal{F}_2) \neq \text{argmin}_w(\mathcal{F}_1) \bowtie \text{argmin}_w(\mathcal{F}_2)$.

The issue in Example 5 comes from the property of ∞ that $\infty + 7 = \infty + 31$, even though $7 < 31$. In order to guarantee that $\text{argmin}_w([E_1 \bowtie E_2]) = \text{argmin}_w([E_1]) \bowtie \text{argmin}_w([E_2])$, we need to separately handle the elements that behaves similarly to ∞ in Example 5. In Definition 10 we introduce an algebraic property that guarantees that this undesirable behaviour never occurs.

Definition 10 (Multiplicative Regularity). A *multiplicatively regular element* of $\mathcal{A}$ is an element $a \in A \setminus \{0\}$ such that

$$\forall (b, c) \in A^2, a \times b = a \times c \implies b = c.$$

We denote by $reg(\mathcal{A})$ the set of multiplicatively regular elements of the semiring $\mathcal{A}$, and by $\overline{reg}(\mathcal{A}) := A \setminus reg(\mathcal{A})$ the set of non multiplicatively regular elements of A.

We have the following link between multiplicatively regular elements and products.

Property 3. *Let $(a, b) \in A^2$. Then $a \times b \in reg(\mathcal{A}) \iff (a, b) \in reg(\mathcal{A})^2$.*

Proof. We prove the two cases separately.

- Assume that $(a, b) \in reg(\mathcal{A})^2$. Let $(c, d) \in A^2$ with $(a \times b) \times c = (a \times b) \times d$. Then $a \times (b \times c) = a \times (b \times d)$. Since a is regular, $b \times c = b \times d$, and since b is regular, $c = d$, which proves that $(a \times b)$ is regular.
- Assume that $a \times b \in reg(\mathcal{A})$. Let $(c, d) \in A^2$ such that $a \times c = a \times d$. Then, $a \times c \times b = a \times d \times b$ ie. $(a \times b) \times c = (a \times b) \times d$, and by regularity of $a \times b$, $c = d$, which proves that a is regular. Symmetrically, b is also regular, which concludes the proof.

□

We now establish that the product $+$ of the dioid $\mathcal{D}$ behaves as expected with respect to the order of the dioid. We also obtain stronger properties under a hypothesis of multiplicative regularity.

Remark now that we study the multiplicative regularity in the dioid $\mathcal{D}$ where $+$ is the *product* and not the sum.

Property 4. *Let $\leq_D$ be the order associated to the dioid $\mathcal{D}$.*

- *For all $(a, b, c, d) \in D^4$, $a \leq_D b$ and $c \leq_D d \implies a + c \leq_D b + d$.*
- *If, in addition, c or d is a multiplicatively regular element of D where $a <_D b$, then $a + c <_D b + d$.*

Proof. We prove each case in turn.

- It is sufficient to prove that $a + c \leq_D b + c$ since this result can be used to derive $c + b \leq_D d + b$ (i.e, $b + c \leq_D b + d$ by commutativity), and $a + c \leq_D b + d$ follows from the transitivity of $\leq_D$. Thus, let us prove that $a + c \leq_D b + c$. Since $a \leq_D b$, there exists $b' \in D$ such that $a = \min(b, b')$. By distributivity of $+$ over inf (axiom of the semiring $(D, \min, +, \infty, 0)$), we have $a+c = \min(b+c, b'+c)$, from where we can conclude $a+c \leq_D b+c$.
- Assume that c is a multiplicatively regular element of D and $a <_D b$. Since $b + c \leq_D b + d$ (first item), it is sufficient to prove that $a + c <_D b + c$. By the first item, we already have $a + c \leq_D b + c$. Assume by contradiction that $a + c = b + c$. Then, since c is multiplicatively regular, $a = b$ which contradicts the hypothesis $a <_D b$. The same reasoning holds if d is a multiplicatively regular element of D. □

The result in the second item of Property 4 guarantees that the unwanted behaviour described in Example 5 does not occur for regular elements.

4.2. Δ-product of semirings

In this section, we construct a new type of semiring, obtained by combining together a dioid $\mathcal{D}$ and a semiring $\mathcal{A}$, through an operation that we call a Δ-*product*. The algorithmic applications of this new semiring will typically be to "count the solutions of minimal weight" of an NP problem.

In order to improve the reading experience the proofs of the theorems of this section have been moved to Appendix A.

We first define which elements of $D \times A$ will belong to the new semiring. Any regular element d of D can be associated with any $a \in A$, but the non regular elements can only be associated to 0_A. The reason is that when the minimal weight of a set $\mathcal{F} = \mathcal{F}_1 \bowtie \mathcal{F}_2$ is not regular, we can not guarantee that $\text{argmin}_w(\mathcal{F}) = \text{argmin}_w(\mathcal{F}_1) \bowtie \text{argmin}_w(\mathcal{F}_2)$. We thus give up on the operations in A and instead output 0_A by default.

Definition 11. For $d \in D$ and $a \in A$, denote by

$$d\Delta a = \left\{ \begin{array}{ll} (d, a) & \text{if } d \in reg(\mathcal{D}) \\ (d, 0_A) & \text{if } d \notin reg(\mathcal{D}) \end{array} \right\}.$$

We are now ready to build a semiring that will be able to handle the combination of the $\mathcal{D}$-measure w and the $\mathcal{A}$-measure μ that is intended to be applied to the $\text{argmin}_w(\mathcal{F})$.

Theorem 3. *The structure* $\mathcal{D}\Delta\mathcal{A} = (D\Delta A, \oplus, \otimes, (\infty_D, 0_A), (0_D, 1_A))$ *is a commutative semiring with:*

- $D\Delta A := \{d\Delta a \mid (d,a) \in D \times A\} = \{(d,a) \in D \times A \mid d \in reg(\mathcal{D})$ *or* $a = 0_A\}$.
- $\oplus : \begin{array}{ccc} (D\Delta A)^2 & \to & D\Delta A \\ ((d_1,a_1),(d_2,a_2)) & \mapsto & (\min(d_1,d_2), \left\{ \begin{array}{cc} a_1 & \text{if } d_1 <_D d_2 \\ a_2 & \text{if } d_2 <_D d_1 \\ a_1 +_A a_2 & \text{if } d_1 = d_2 \end{array} \right\}) \end{array}$

  and
- $\otimes : \begin{array}{ccc} (D\Delta A)^2 & \to & D\Delta A \\ ((d_1,a_1),(d_2,a_2)) & \mapsto & ((d_1 +_D d_2),(a_1 \times_A a_2)) \end{array}$.

Note that we can reformulate the definition of $\oplus$ as:

$$\oplus : \begin{array}{ccc} (D\Delta A)^2 & \to & D\Delta A \\ ((d_1,a_1),(d_2,a_2)) & \mapsto & \left\{ \begin{array}{cc} (d_1,a_1) & \text{if } d_1 <_D d_2 \\ (d_2,a_2) & \text{if } d_2 <_D d_1 \\ (d_1, a_1 +_A a_2) & \text{if } d_1 = d_2 \end{array} \right\} \end{array}$$

by Property 1. We also observe that $\otimes$ is the Cartesian product of the two products $+_D$ and $\times_A$ of the semirings D and A.

Intuitively, $\oplus$ is defined in order to mimic how we would count the minimum, and number of minimal elements of a disjoint union of two sets $\mathcal{F}_1$ and $\mathcal{F}_2$, given their respective minimum d_1, d_2 and the number of minimum elements a_1, a_2. In order to study the associativity and commutativity of this operation, we let $\mathcal{D}_1 = (D_1, \min_1, +_1, \infty_1, 0_1)$ and $\mathcal{D}_2 = (D_2, \min_2, +_2, \infty_2, 0_2)$ be two commutative totally-ordered idempotent dioids.

It is now natural to study the algebraic properties of this newly defined Δ-product. We prove in the following that the Δ-product is associative.

Theorem 4. *The semiring* $\mathcal{D}_1\Delta\mathcal{D}_2$ *is also a commutative totally ordered idempotent dioid, and the associated order* $\leq_{\mathcal{D}_1\Delta\mathcal{D}_2}$ *is the lexicographical order* $\leq_{\mathcal{D}_1 lex \mathcal{D}_2}$.

Studying the associativity of the Δ-product requires to compare $(\mathcal{D}_1\Delta\mathcal{D}_2)\Delta\mathcal{A}$ with $\mathcal{D}_1\Delta(\mathcal{D}_2\Delta\mathcal{A})$. Even though the definition of $\mathcal{D}_1\Delta(\mathcal{D}_2\Delta\mathcal{A})$ does not raise any issue, considering $(\mathcal{D}_1\Delta\mathcal{D}_2)\Delta\mathcal{A}$ requires that we justify that $(\mathcal{D}_1\Delta\mathcal{D}_2)$ is a commutative totally ordered idempotent dioid, which is ensured by Theorem 4.

Theorem 5. *The* Δ*-product of semirings is associative. More precisely, up to identifying* $(D_1 \times D_2) \times A$ *with* $D_1 \times (D_2 \times A)$, $(\mathcal{D}_1\Delta\mathcal{D}_2)\Delta\mathcal{A} = \mathcal{D}_1(\Delta\mathcal{D}_2\Delta\mathcal{A})$.

This is essentially due to the fact that the lexicographic order is associative. Note that the Δ-product of semirings is not commutative, even up to isomorphism, since the lexicographic order depends on the ordering of its coordinates.

Example 6. A surprising application of Δ-product is to build the dioids of the Kleene algebras of *multi-valued logic* [3]. The trilean dioid $(\mathbf{3}, \vee, \wedge, \bot, \top)$ (with $\mathbf{3} = \{\bot, \top, ?\}$) defined by the table:

$\vee$	$\top$	$?$	$\bot$
$\top$	$\top$	$\top$	$\top$
$?$	$\top$	$?$	$?$
$\bot$	$\top$	$?$	$\bot$

$\wedge$	$\top$	$?$	$\bot$
$\top$	$\top$	$?$	$\bot$
$?$	$?$	$?$	$\bot$
$\bot$	$\bot$	$\bot$	$\bot$

is isomorphic to the dioid $\mathbb{B}\Delta\mathbb{B}$, through the isomorphism Φ defined by $\Phi(\bot,\bot)=\bot, \Phi(\top,\top)=\top, \Phi(\top,\bot)=?$. Recall that $(\bot,\top)\notin \mathbf{2}\Delta\mathbf{2}$, since $\bot$ is not regular.

More generally, for all $n\geq 2$ the dioid of the Kleene-algebra of the n-valued logic $([n],\max,\min,1,n)$ is isomorphic to $\underbrace{\mathbb{B}\Delta\mathbb{B}\Delta\ldots\Delta\mathbb{B}}_{n-1 \text{ occurrences of } \mathbb{B}}$, through the isomorphism:

$$\Phi : \begin{array}{ccc} \mathbf{2}\Delta\mathbf{2}\Delta\ldots\Delta\mathbf{2} & \mapsto & [n] \\ \underbrace{(\top,\ldots,\top,\bot,\ldots,\bot)}_{i \text{ occurrences of } \top} & \mapsto & i+1 \end{array}$$

However, the algorithmic applications of the dioids of n-valued logic are limited by the fact that their only multiplicatively regular element is $(\top,\ldots,\top)$. Therefore studying the measures of the dioid $[n]\Delta\mathcal{A}$ with $\mathcal{A}$ will not be especially relevant, since $[n]\Delta A = (\{n\}\times A)\uplus\{(i,0_A)\mid i\in[n-1]\}$.

4.3. Δ-product of measures

The main purpose of Δ-product of semirings is their ability to shelter Δ-product of measures. More precisely, if w is a $\mathcal{D}$-measure (seen as a weight function) and μ is a $\mathcal{A}$-measure (for instance, the $\mathbb{N}$-measure of cardinality $\# : \mathcal{F}\mapsto|\mathcal{F}|$), $(w\Delta\mu)(\mathcal{F})$ is the image by μ of the set of minimal functions (with respect to w) of $\mathcal{F}$.

Theorem 6. *The function $w\Delta\mu$ that associates to every $\mathcal{F}\subseteq T^{S'}$ with $S'\subseteq S$:*

$$\begin{array}{rcl} (w\Delta\mu)(\mathcal{F}) & := & \left\{ \begin{array}{cl} (w(\mathcal{F}),\mu(\mathrm{argmin}_w(\mathcal{F}))) & \textit{if } w(\mathcal{F})\in reg(D) \\ (w(\mathcal{F}),0_A) & \textit{if } w(\mathcal{F})\notin reg(D) \end{array} \right. \\ & = & (w(\mathcal{F}))\Delta(\mu(\mathrm{argmin}_w(\mathcal{F}))) \end{array}$$

with $\mathrm{argmin}_w(\mathcal{F}) := \{f\in\mathcal{F}, w(\{f\})=w(\mathcal{F})\}$ *is a* $(\mathcal{D}\Delta\mathcal{A})$*-measure.*

The main interest of the Δ-compositions of semiring and measures is then that it makes new extensions of NP computational problems fall under the various semiring formalism. This is true for the Semiring-$\mathbf{\Pi}$ problems (for $\mathbf{\Pi}\in$ NP) introduced in Section 3 as well as for the related problems in the Sum-Product family (introduced in Section 3.7).

4.4. Computational extensions

Let $\mathbf{\Pi}$ be a problem in NP, and let x be an instance of $\mathbf{\Pi}$. As usual we assume, without loss of generality, that the set of certificates of x are functions from a set S to a set T (where S and T have polynomial size with respect to x).

By Lemma 1, for any computable function g, if Semiring-$\mathbf{\Pi}$ is solvable in time $O(g(n))$, then so is the #Cost-$\mathbf{\Pi}$ problem defined as:

#Cost-$\mathbf{\Pi}$:

Input: An instance x of $\mathbf{\Pi}$, and a cost matrix $C \in (\overline{\mathbb{R}})^{S\times T}$.

Output: How many solutions f of x of minimal cost $\sum_{s\in S} C[s, f(s)]$ are there, and what is this minimal cost ?

Solving #Cost-Π is indeed equivalent to computing the image of to the set of solutions SOL of the instance by the $(\overline{\mathbb{R}}_{min}\Delta\mathbb{N})$-measure $(C\Delta\#)$ in the case where there exists at least one solution of cost $\neq \infty$. If all solutions have infinite cost, $(C\Delta\#)(SOL)$ will output $(\infty, 0)$, and then all solutions have minimal cost: we can count them by computing $\#(SOL)$.

Note that in practice, to avoid having to compute both $(C\Delta\#)(SOL)$ and $\#(SOL)$ by using the algorithm described in Lemma 1 twice, it is possible to use the cartesian product

$$(C\Delta\#) \times \# : \mathcal{F} \to (C\Delta\#)(\mathcal{F}) \times \#(\mathcal{F})$$

of the measures $(C\Delta\#)$ and $\#$. The measure $(C\Delta\#) \times \#$ is indeed a measure of the cartesian product of semirings $(\overline{\mathbb{R}}_{min}\Delta\mathbb{N}) \times \mathbb{N}$.

In particular we observe that the well-known SAT problem of counting solutions of minimal cardinality from Hermann & Pichler [20] is a particular case of #Cost-SAT.

#Min-Card-Sat:

Input: An instance $\varphi(X)$ of Sat (i.e., a quantifier-free formula) on a set of variables X.

Output: The number of models $\sigma : X \mapsto \{\bot, \top\}$ of φ of minimal cardinality (i.e., where $|\sigma^{-1}(\{\top\})|$ is minimal).

Similarly, the problem

#Min-Lex-Sat:

Input: An instance $\varphi(X)$ of Sat (ie. a quantifyier free formula) on a set of variable X, and $(x_1, \ldots, x_\ell) \in X^\ell$ (with $\ell \geq 0$).

Output: The number of models $\sigma : X \mapsto \{\bot, \top\}$ of φ where $(\sigma(x_1), \ldots, \sigma(x_\ell))$ is lexicographically minimal with respect to the order $\bot \leq \top$.

from Hermann & Pichler [19] and even the more general problem

#Min-Weight-Sat:

Input: An instance $\varphi(X)$ of SAT (i.e., a quantifier-free formula) on a set of variables X, and a weight function $w : X \mapsto \mathbb{N}$.

Output: The number of models $\sigma : X \to \{\bot, \top\}$ of φ of minimal weight, i.e., where

$$\sum_{x \in \sigma^{-1}(\{\top\})} w(x) \quad \text{is minimal.}$$

from Hermann & Pichler [20] are also particular cases of #COST-SAT. Indeed, if we for all $w : X \mapsto \mathbb{N}$ define the matrix $W \in (\overline{\mathbb{R}})^{X \times \{\bot, \top\}}$ by $W[x, \top] = w(x)$ and $W[x, \bot] = 0$, the #MIN-WEIGHT-SAT problem consists in applying the $(\mathbb{R}_{min}\Delta\mathbb{N})$-measure $\mu_W \Delta\#$ to the set of solution of an instance of SAT. Also, the #MIN-CARD-SAT problem is the particular case where w is constantly 1, and the #MIN-LEX-SAT problem is the particular case where

$$w : \begin{array}{ccc} X & \to & \mathbb{N} \\ y & \mapsto & \begin{cases} 2^{\ell - i} & \text{if } y = x_i \text{ with } i \in [\ell] \\ 0 & \text{if } y \notin \{x_1, \ldots, x_\ell\} \end{cases} \end{array}$$

5. Semiring-Connected-Dominating-Set and Cliquewidth

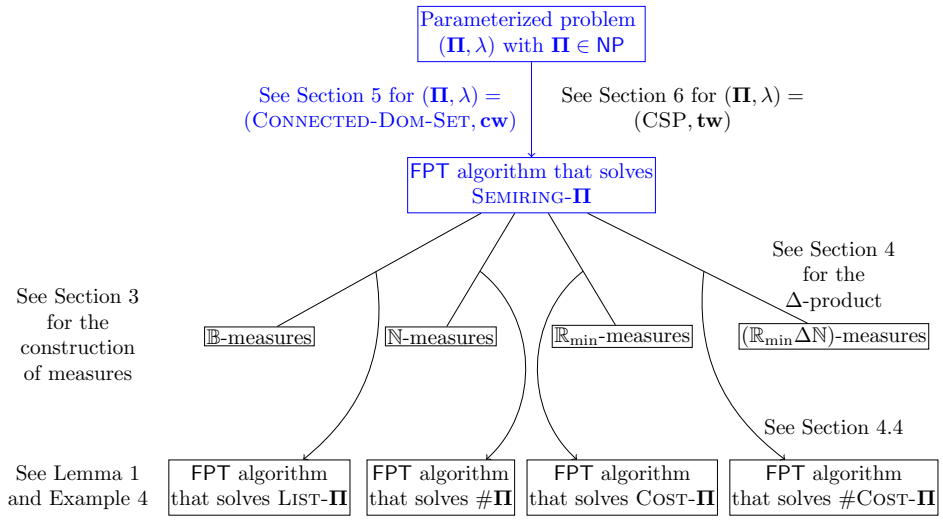

Figure 3: A general overview of our contributions where the contribution in this section is colored in blue.

Recall that the framework is built upon the existence of an algorithm for SEMIRING-$\mathbf{\Pi}$ (with $\mathbf{\Pi} \in \mathsf{NP}$), which via Lemma 1 can then be exploited to solve the usual semiring extensions #$\mathbf{\Pi}$, LIST-$\mathbf{\Pi}$, COST-$\mathbf{\Pi}$. The Δ-product

presented in Section 4 also makes it possible to subsume the problems of the form #Cost-$\mathbf{\Pi}$ in the semiring formalism. Thus, an algorithm for Semiring-$\mathbf{\Pi}$ is highly desirable since it can be used to solve many different (combinations of) problem extensions. To exemplify that this is indeed feasible we consider two well-known problems: Connected-Dominating-Set, and Csp. For both these problems we construct FPT algorithms that solve the semiring extended problems, and, as a corollary, may derive upper bounds on all of the aforementioned problem extensions. In particular, we are able to solve both #Cost-Connected-Dominating-Set and #Cost-CSP by Section 4. The contribution of this section to the semiring formalism presented in this paper is summarized in Figure 3.

Both of these algorithms (in Section 5 and 6, respectively) follow a similar idea since they are both based on bounded width properties of graphs. We utilize a "method of construction of the graph", such as a k-expression in Section 5, or a tree decomposition in Section 6. We will explain in Section 6.1 how tree decompositions can be seen as a method of construction of a graph.

Remark 3. *Our algorithms only require that we "compress" each candidate solution by considering its so-called* trace*. The definition of the trace depends strongly on the problem and the parameter considered, but the trace must always verify these three properties:*

1. *there must be only be an* FPT *number of possible traces,*
2. *it is possible to decide if a candidate function is a solution knowing only its trace, and*
3. *the new trace of a candidate solution after performing a step of the "method of construction of the graph", depends only on its trace before this step.*

Then, our algorithm maintains the invariant that for every possible trace τ, we know a semiring expression E_τ of the set of all candidates that have the trace τ. Note that if τ and τ' are two different traces, E_τ and $E_{\tau'}$ represent disjoint sets, and thus, the semiring expression $E_\tau \uplus E_{\tau'}$ is always correct. By property 3., this invariant can be preserved through every step of the construction of the graph, and is therefore eventually true for the whole graph. Then, by property 2., this enables us to recover a semiring expression of the set of solutions via disjoint union, thereby answering the Semiring extension of our problem. Due to property 1., all of these operations are done in FPT time.

Also, a variant of this paradigm can be to only encode the set of *solutions* with a given trace, instead of all *candidate solutions*, in which case property 2. is not necessary, but property 3. must apply to *sets of solutions* instead of *candidates of solutions*. This is what we will do in Section 6 to solve Csp.

5.1. Clique-width

In this section, we recall the definition of the *clique-width* of a graph which, together with tree-width, is the most well-known and extensively studied graph

parameter. The general idea behind clique-width is that graphs with low clique-width can be decomposed into simpler structures, making it amenable to efficient algorithmic solutions for problems that are otherwise intractable [8, 9].

For $k \geq 1$, let $[k] = \{1, \ldots, k\}$. A *k-labelled graph* G is a tuple (V_G, E_G, l_G), where (V_G, E_G) is a graph and $l_G : V_G \to [k]$. For $i \in [k]$ and a k-labelled graph G, denote by $V_G^i = l_G^{-1}(\{i\})$ the set of vertices of G of label i. A *k-expression* φ of a k-labelled graph G, denoted $[\varphi] = G$, is an expression defined inductively [9] using:

1. **Single vertex:** $\bullet_i$ with $i \in [k]$: $[\bullet_i]$ is a k-labelled graph with one vertex labelled by i (we sometimes write $\bullet_i(u)$ to state that the vertex is named u),
2. **Disjoint Union:** $\varphi_1 \oplus \varphi_2$: $[\varphi_1 \oplus \varphi_2]$ is the disjoint union of the graphs $[\varphi_1]$ and $[\varphi_2]$.
3. **Relabelling:** $\rho_{i \to j}(\varphi)$ with $(i, j) \in [k]^2$ and $i \neq j$: $[\rho_{i \to j}(\varphi)]$ is the same graph as $[\varphi]$, but where all vertices of G with label i now have label j,
4. **Edge Creation:** $\eta_{i,j}(\varphi)$ with $(i, j) \in [k]^2$ and $i \neq j$: $[\eta_{i,j}(\varphi)]$ is the same graph as $[\varphi]$, but where all pairs of the form $\{u, v\}$ with $\{l_G(u), l_G(v)\} = \{i, j\}$ is now an edge, and

A graph G has a k-expression φ if there exists $l : V_G \mapsto [k]$ such that $[\varphi] = (V_G, E_G, l)$. The *clique-width* of a graph G (denoted by $\mathbf{cw}(G)$) is the minimum $k \geq 1$ such that G has a k-expression.

Moreover, if $S \subseteq V_G$ is a subset of vertices of a k-labelled graph G, we denote by $l_G(S) := \{l_G(u) \mid u \in S\} \in 2^{[k]}$ the set of labels of vertices in S.

5.2. FPT Algorithm

We propose an algorithm that solves Semiring-Connected-Dominating-Set that is FPT when parameterized by the clique-width of the input graph. Thus, we manage to solve all common problem extensions (minimal cost, list, and counting) as well as problem extensions via the Δ-product from Section 4. Let us remark that the basic problem Connected-Dominating-Set is known to be solvable in $O^*(2^{(\omega+4)k})$ time [2], where $\omega < 2.37188$ is the matrix multiplication exponent [11], and, additionally, in $O^*(5^k)$ time via a randomized (Monte-Carlo) algorithm [17]. However, no FPT algorithm is known for the counting extension (which we vastly generalize). More formally, we study the problem:

<u>Semiring-Connected-Dominating-Set:</u>
Input: A graph G.
Output: A semiring expression of the set of indicator fonctions of connected dominating-sets of G.

and provide an FPT algorithm parameterized by the clique-width of the input graph.

$+$	0	1	$\overline{2}$
0	0	1	$\overline{2}$
1	1	$\overline{2}$	$\overline{2}$
$\overline{2}$	$\overline{2}$	$\overline{2}$	$\overline{2}$

Table 1: Addition in $\overline{\mathbf{3}}$.

Theorem 7. *The problem* Semiring-Connected-Dominating-Set *is* FPT *when parameterized by the cliquewidth of the input graph (assuming that a* $\mathbf{cw}(G)$*-expression of the input graph* G *is given).*

Proof. We give an algorithm that solves Semiring-Connected-Dominating-Set in time $O^*((3^{2^{cw(G)}} \times 2^{cw(G)})^2) = O^*(3^{2^{cw(G)+1}} \times 4^{cw(G)})$. To do so, take $k \geq 1$, a k-expression φ of a graph G, and $S \subseteq V_G$. The k-expression will play the role of the "method of construction of the graph" evoked in Remark 3.

In order to define the "trace" of Remark 3, we will use three different labels, called 0, 1 and $\overline{2}$. They should be interpreted as respectively "zero", "one" and "at least two". We let $\overline{\mathbf{3}} := \{0, 1, \overline{2}\}$, and we define the operation $+$ in $\overline{\mathbf{3}}$ in Table 1 agreeing with this interpretation of 0, 1 and $\overline{2}$.

We define the following operations for all k-expression φ:

- The *signature* of S in φ, denoted by $\sigma_\varphi(S) \in \overline{\mathbf{3}}^{2^{[k]}}$ is the function

$$\sigma_\varphi(S) : \begin{array}{ccc} 2^{[k]} & \to & \overline{\mathbf{3}} \\ C & \mapsto & \begin{cases} 0 & \text{if there exists no } t \in [q] \text{ with } C = l_G(C_t) \\ 1 & \text{if there exists is a unique } t \in [q] \text{ with } C = l_G(C_t) \\ \overline{2} & \text{if there exists several } t \in [q] \text{ with } C = l_G(C_t) \end{cases} \end{array},$$

  where $S = C_1 \uplus \cdots \uplus C_q$ with $q \geq 0$ is the partition of S into connected components of $G[S]$. The label $\overline{2}$ should be interpreted as "at least 2".

  The quantity $(\sigma_\varphi(S))(C) \in \overline{\mathbf{3}}$ with $C \in 2^{[k]}$ gives the answer to the question: "How many connected components $C_t \subseteq S$ of $G[S]$ are there such that the set of labels (i.e. the labels in $[k]$ of the k-expression φ) that appear in C_t is exactly C ?". The value of $(\sigma_\varphi(S))(C)$ is either: 0 if the answer is "0 connected components", 1 if the answer is "exactly 1 connected component", or $\overline{2}$ if the answer is "at least 2 different connected components". Note that $G[S]$ is connected if and only if $\|\sigma_\varphi(S)\| = 1$, with

$$\|\sigma_\varphi(S)\| = \sum_{C \in 2^{[k]}} (\sigma_\varphi(S))(C)$$

  being the number of connected components of $G[S]$ (in $\overline{\mathbf{3}}$).

- The *domination* of S in φ, denoted by $dom_\varphi(S) \in 2^{[k]}$ is the set $dom_\varphi(S) = \{d \in [k] \mid V_G^d \subseteq N_G[S]\}$. Note that S is a dominating set of G if and only if $dom_\varphi(S) = [k]$.

- Finally, the *trace* of S in φ is $Tr_\varphi(S) = (\sigma_\varphi(S), dom_\varphi(S)) \in \overline{\mathbf{3}}^{2^{[k]}} \times 2^{[k]}$.

Also, for all $(\sigma, D) \in \overline{\mathbf{3}}^{2^{[k]}} \times 2^{[k]}$, $Tr_\varphi^{-1}(\sigma, D) \subseteq V_G$ denotes the inverse image of $\{(\sigma, D)\}$ by the function Tr_φ, *i.e.,*

$$Tr_\varphi^{-1}(\sigma, D) := \{S \subseteq V_G \mid Tr_\varphi(S) = (\sigma, D)\}.$$

Note that σ_φ is considered here as a function (that outputs a function):

$$\sigma_\varphi : \begin{array}{ccc} 2^{V_G} & \to & \overline{\mathbf{3}}^{2^{[k]}} \\ S & \mapsto & \sigma_\varphi(S) \end{array}$$

from which we consider the inverse images.

The idea of the algorithm is then to compute the values of $Tr_\varphi^{-1}(\sigma, D)$ for all $(\sigma, D) \in \overline{\mathbf{3}}^{2^{[k]}} \times 2^{[k]}$, by induction over the structure of φ. What will be especially useful will be that:

1. $\{Tr_\varphi^{-1}(\sigma, D), (\sigma, D) \in \overline{\mathbf{3}}^{2^{[k]}} \times 2^{[k]}\}$ forms a partition of V_G, which makes it possible to use $\uplus$. This is a partition into a FPT number of subsets $(3^{2^k} \times 2^k)$ with respects to $k = \mathbf{cw}(G)$,
2. the set of connected dominating set of G is exactly

$$\biguplus_{\|\sigma\|=1} Tr_\varphi^{-1}(\sigma, [k]), \quad \text{with } \|\sigma\| = \sum_{C \in 2^{[k]}} \sigma(C), \text{ and}$$

3. for $S \subseteq V_G$, the value of $Tr_\varphi(S)$ depends only on the $Tr_{\varphi'}(S)$ with φ being of the form $\varphi = \eta_{i,j}(\varphi')$ or $\varphi = \rho_{i \to j}(\varphi')$. A similar remark holds for $\varphi = \varphi_1 \oplus \varphi_2$. Also, $Tr_{\bullet_i(u)}^{-1}(\sigma, D)$ (for $(\sigma, D) \in \overline{\mathbf{3}}^{2^{[k]}} \times 2^{[k]}$) are easy to compute.

We now focus on justifying this third remark. Since the semiring expressions are made to express sets of functions and not subsets of vertices, we will give a semiring expression $E_\varphi(\sigma, D)$ of the sets of indicator functions of the subsets of vertices in $Tr_\varphi^{-1}(\sigma, D)$.

To compute the values of the $Tr_\varphi^{-1}(\sigma, D)$ for $(\sigma, D) \in \overline{\mathbf{3}}^{2^{[k]}} \times 2^{[k]}$ (by induction over the structure of φ), note the following:

- **Single vertex:** For $i \in [k]$, $Tr_{\bullet_i(u)}(\{u\}) = (\sigma_{\bullet_i(u)}(\{u\}), [k])$ and $Tr_{\bullet_i(u)}(\emptyset) = (\sigma_{\bullet_i(u)}(\emptyset), [k] \setminus \{i\})$, with:

$$\sigma_{\bullet_i(u)}(\{u\}) : \begin{array}{ccc} 2^{[k]} & \to & \overline{\mathbf{3}} \\ C & \mapsto & \begin{cases} 1 & \text{if } C = \{i\} \\ 0 & \text{otherwise} \end{cases} \end{array}$$

and

$$\sigma_{\bullet_i(u)}(\emptyset) : \begin{array}{ccc} 2^{[k]} & \to & \overline{\mathbf{3}} \\ C & \mapsto & 0 \end{array}$$

Since $\{u\}$ and $\emptyset$ are the only two subsets of $\{u\}$ ($\{u\}$ is the set of vertices of the graph expressed by the k-expression $\bullet_i(u)$), we have handled all cases. In other words, we have proven for all $(\sigma, D) \in \overline{\mathbf{3}}^{2^{[k]}} \times 2^{[k]}$:

$$Tr^{-1}_{\bullet_i(u)}(\sigma, D) = \begin{cases} \{\{u\}\} & \text{if } (\sigma, D) = (\sigma_{\bullet_i(u)}(\{u\}), [k]) \\ \{\emptyset\} & \text{if } (\sigma, D) = (\sigma_{\bullet_i(u)}(\emptyset), [k] \setminus \{i\}) \\ \emptyset & \text{otherwise.} \end{cases}$$

Note that the indicator function of the set $\{u\}$ (respectively $\emptyset$) over the domain $\{u\}$ is $(u \mapsto 1)$ (respectively $(u \mapsto 0)$). Thus, a semiring expression of the set of indicator functions of the subsets of $Tr^{-1}_{\bullet_i(u)}(\sigma, D)$ would be:

$$E_{\bullet_i(u)}(\sigma, D) := \begin{cases} (u \mapsto 1) & \text{if } (\sigma, D) = (\sigma_{\bullet_i(u)}(\{u\}), [k]) \\ (u \mapsto 0) & \text{if } (\sigma, D) = (\sigma_{\bullet_i(u)}(\emptyset), [k] \setminus \{i\}) \\ \emptyset & \text{otherwise.} \end{cases}$$

- **Disjoint union:** Let φ be a k-expression of the form $\varphi = \varphi_1 \oplus \varphi_2$. Let $G = [\varphi]$, $G_1 = [\varphi_1]$ and $G_2 = [\varphi_2]$. Then, V_{G_1} and V_{G_2} are disjoint. Let $S \subseteq V_G$, decompose $S = S_1 \uplus S_2$, with $S_1 \subseteq V_{G_1}$ and $S_2 \subseteq V_{G_2}$ or equivalently, $\mathbf{1}_S = \mathbf{1}_{S_1} \bowtie \mathbf{1}_{S_2}$ (it is here implicit that the domains of $\mathbf{1}_S$, $\mathbf{1}_{S_1}$ and $\mathbf{1}_{S_2}$ are respectively V_G, V_{G_1} and V_{G_2}). Then, denoting $Tr_{\varphi_1}(S_1) = (\sigma_1, D_1)$ and $Tr_{\varphi_2}(S_2) = (\sigma_2, D_2)$, we have

$$Tr_\varphi(S) = (\sigma_1 + \sigma_2, D_1 \cap D_2)$$

with

$$\sigma_1 + \sigma_2 : \begin{array}{ccc} 2^{[k]} & \to & \overline{\mathbf{3}} \\ C & \mapsto & \sigma_1(C) + \sigma_2(C). \end{array}$$

Indeed, the connected components of G are exactly the connected components of G_1 and the connected components of G_2. Also, a label is dominated in G if and only if it is dominated both in G_1 and in G_2. This proves that:

$$Tr^{-1}_\varphi(\sigma, D) = \biguplus_{\substack{\sigma_1 + \sigma_2 = \sigma \\ D_1 \cap D_2 = D}} \{S_1 \uplus S_2 \mid (S_1, S_2) \in Tr^{-1}_{\varphi_1}(\sigma_1, D_1) \times Tr^{-1}_{\varphi_2}(\sigma_2, D_2)\}.$$

for all $(\sigma, D) \in \overline{\mathbf{3}}^{2^{[k]}} \times 2^{[k]}$.

It follows a similar relation for the indicator functions:

$$E_\varphi(\sigma, D) := \biguplus_{\substack{\sigma_1 + \sigma_2 = \sigma \\ D_1 \cap D_2 = D}} E_{\varphi_1}(\sigma_1, D_1) \bowtie E_{\varphi_2}(\sigma_2, D_2).$$

- **Relabelling:** Let φ be a k-expression of the form $\varphi = \rho_{i\to j}(\varphi')$, $G = [\varphi]$ and $G' = [\varphi']$. Let $S \subseteq V_G$ and $(\sigma', D') = Tr_{\varphi'}(S)$. Note that the connected components of $G[S]$ are exactly the connected components of $G'[S]$. Moreover, if the set of labels of a connected component C_t of $G'[S]$ is exactly $C' \in 2^{[k]}$, then the set of labels of C_t in $G[S]$ will be exactly $\rho_{i\to j}(C')$ with:

$$\rho_{i\to j}(C') = \begin{cases} C' & \text{if } i \notin C' \\ (C' \cup \{j\}) \setminus \{i\} & \text{if } i \in C'. \end{cases}$$

It follows that the signature $\sigma_\varphi(S)$ is exactly $\sigma_\varphi(S) = \rho_{i\to j}(\sigma')$ with

$$(i)\ \rho_{i\to j}(\sigma') : \begin{array}{ccc} 2^{[k]} & \mapsto & \overline{\mathbf{3}} \\ C & \mapsto & \displaystyle\sum_{C' \in 2^{[k]}, \rho_{i\to j}(C') = C} \sigma'(C') \end{array}$$

with the convention that an empty sum equals 0.

Also, as the domination of S in G' is D', the domination of S in G is exactly $dom_\varphi(S) = \rho_{i\to j}(D')$ with

$$(ii)\ \rho_{i\to j}(D') = \begin{cases} D' & \text{if } \{i, j\} \subseteq D' \\ D' \cup \{i\} \setminus \{j\} & \text{otherwise.} \end{cases}$$

This proves that:

$$Tr_\varphi^{-1}(\sigma, D) = \biguplus_{(\rho_{i\to j}(\sigma'), \rho_{i\to j}(D')) = (\sigma, D)} Tr_{\varphi'}^{-1}(\sigma', D')$$

for all $(\sigma, D) \in \overline{\mathbf{3}}^{2^{[k]}} \times 2^{[k]}$. The same relation holds for the indicator function, leading to the semiring expression:

$$E_\varphi(\sigma, D) = \biguplus_{(\rho_{i\to j}(\sigma'), \rho_{i\to j}(D')) = (\sigma, D)} E_{\varphi'}(\sigma', D').$$

- **Edge creation:** Let φ be a k-expression of the form $\varphi = \eta_{i,j}(\varphi')$, $G = [\varphi]$ and $G' = [\varphi']$. Let $S \subseteq V_G$ and $(\sigma', D') = Tr_{\varphi'}(S)$.

  First, recall that all the vertices have the same label in G and in G', ie. $l_G = l_{G'}$. Note that if the connected components of $G'[S]$ are $C_1, \ldots, C_q$ with $q \geq 1$, and if the connected components of $G'[S]$ that contains either

a i-vertex or a j-vertex are $C_{\iota+1}, \dots, C_q$ with $\iota \in [q]$, then the connected components of $G[S]$ are:

$$\begin{cases} C_1, \dots, C_q & \text{if } \{i,j\} \not\subseteq l_G(S) \\ C_1, \dots, C_\iota \text{ and } \bigcup_{t=\iota+1}^{q} C_t & \text{otherwise.} \end{cases}$$

Let $C_0^{i,j}(\sigma') := \bigcup_{\substack{C \in 2^{[k]}, \sigma'(C) \neq 0 \\ \{i,j\} \cap C \neq \emptyset}} C \in 2^{[k]}$. Notice that by definition of the signature

$$l_G(S) = \bigcup_{C \in 2^{[k]}, \sigma'(C) \neq 0} C \in 2^{[k]}$$

By what precedes, we have:

$$\{i,j\} \cap C_0^{i,j}(\sigma') = \{i,j\} \cap l_G(S)$$

Thus if $C_0^{i,j}(\sigma')$ does not contain both a i and j, it means that S does not contain either a i-vertex or a j-vertex. Then, since no edge is created in $G'[S]$ after performing $\eta_{i,j}$, we have $G[S] = G'[S]$, and the signature is thus the same in both graphs.

Otherwise, all connected components that contain either a i-vertex or a j-vertex are merged into a greater connected component C_0, and we have $l_G(C_0) = C_0^{i,j}(\sigma')$. The other connected components of $G'[S]$ (that contains neither i nor j) are unchanged.

It follows that the signature of $\sigma_\varphi(S)$ is exactly $\eta_{i,j}(\sigma')$ with:

$$\eta_{i,j}(\sigma') := \sigma' \text{ if } \{i,j\} \cap C_0^{i,j}(\sigma') \neq \{i,j\},$$

otherwise:

$$\eta_{i,j}(\sigma') : \begin{array}{ccc} 2^{[k]} & \to & \overline{\mathbf{3}} \\ C & \mapsto & \begin{cases} \sigma'(C) & \text{if } \{i,j\} \cap C = \emptyset \\ 1 & \text{if } C = C_0^{i,j}(\sigma') \\ 0 & \text{otherwise.} \end{cases} \end{array}$$

Also, the domination of S in G' is exactly $D_{i,j}(\sigma', D')$ with:

$$D_{i,j}(\sigma', D') := \begin{cases} D' & \text{if } \{i,j\} \cap C_0^{i,j}(\sigma') = \emptyset \\ D' \cup \{i\} & \text{if } \{i,j\} \cap C_0^{i,j}(\sigma') = \{j\} \\ D' \cup \{j\} & \text{if } \{i,j\} \cap C_0^{i,j}(\sigma') = \{i\} \\ D' \cup \{i,j\} & \text{if } \{i,j\} \cap C_0^{i,j}(\sigma') = \{i,j\} \end{cases}$$

Indeed, assume that S contains a i-vertex. Since, in G, every i-vertex shares an edge with every j-vertex (because of $\eta_{i,j}$), then every j-vertex is dominated by S in G. A symmetric argument applies if S contains a j-vertex.

Finally, letting:

$$\eta_{i,j}(\sigma', D') := (\eta_{i,j}(\sigma'), D_{i,j}(\sigma', D')) \in \overline{\mathbf{3}}^{2^{[k]}} \times 2^{[k]},$$

we have:

$$Tr_{\varphi}^{-1}(\sigma, D) = \biguplus_{\eta_{i,j}(\sigma', D')=(\sigma, D)} Tr_{\varphi'}^{-1}(\sigma', D').$$

for all $(\sigma, D) \in \overline{\mathbf{3}}^{2^{[k]}} \times 2^{[k]}$. The same relation is true for the indicator function, leading to the semiring expression:

$$E_{\varphi}(\sigma, D) = \biguplus_{\eta_{i,j}(\sigma', D')=(\sigma, D)} E_{\varphi'}(\sigma', D').$$

This proves (by induction on the structure of a k-expression φ) that during the execution of Algorithm 1, at the end of each call of **CONNECTED-DOM-SET**(φ), the variable E_{φ} contains a semiring expression of the set $Tr_{\varphi}^{-1}(\sigma, D)$ for all $(\sigma, D) \in \overline{\mathbf{3}}^{2^{[k]}} \times 2^{[k]}$ by induction on the structure of φ.

As Algorithm 1 outputs

$$\biguplus_{\sigma \in \overline{\mathbf{3}}^{2^{[k]}}, \|\sigma\|=1} E_{\varphi_G}[\sigma, [k]] \quad (\text{with } \|\sigma\| = \sum_{\sigma \in \overline{\mathbf{3}}^{2^{[k]}}} \sigma(C)),$$

it thus outputs a semiring expression of the set of connected dominating set of G. Indeed, for all $S \subseteq V_G$, letting $Tr_{\varphi_G}(S) = (\sigma, D)$, S is a dominating-set if and only if $D = [k]$, and $G[S]$ is connected if and only if $\|\sigma\| = 1$.

Moreover, Algorithm 1 runs in $(3^{2^k} \times 2^k)^2 = 3^{2^{k+1}} \times 4^k$ time, from the case $\varphi = \varphi_1 \oplus \varphi_2$, □

It would be interesting to investigate whether one could even achieve a single exponential running time close to the $O((2^{\omega+4})^k)$ time algorithm for Connected-Dominating-Set [2] (with $\omega < 2.37188$ the exponent of the optimal time complexity of matrix multiplication [11]).

By performing a simpler variant of this idea it is also possible to derive an algorithm solving Semiring-Dominating-Set in time $O^*(16^k)$, assuming a k-expression of the input graph is given. The main difference with Algorithm 1 is that we instead define the signature of $S \subseteq V_G$ as simply $l_G(S) \in 2^{[k]}$ (it is not necessary to distinguish the connected components of $G[S]$). Thus the signature takes its values in $2^{[k]}$ instead of $\overline{\mathbf{3}}^{2^{[k]}}$, justifying that the complexity

Data: A graph G, a k-expression φ_G of G.
Result: A semiring expression of the set of connected dominating sets of G.

Run **CONNECTED-DOM-SET**(φ_G) and return $\biguplus_{\sigma \in \overline{\mathbf{3}}^{2^{[k]}}, \|\sigma\|=1} E_{\varphi_G}[\sigma, [k]]$, with **CONNECTED-DOM-SET**(φ) being defined for all k-expression φ as:

CONNECTED-DOM-SET(φ):
for $(\sigma, D) \in \overline{\mathbf{3}}^{2^{[k]}} \times 2^{[k]}$ **do**
  $E_\varphi[\sigma, D] := \emptyset$
end
if $\varphi = \bullet_i(u)$*:* **then**
  $E_\varphi[\sigma_{\bullet_i(u)}(\emptyset), [k] \setminus \{i\}] \leftarrow (u \mapsto 0)$
  $E_\varphi[\sigma_{\bullet_i(u)}(\{u\}), [k]] \leftarrow (u \mapsto 1)$
end
if $\varphi = \rho_{i \to j}(\varphi')$*:* **then**
  Run **CONNECTED-DOM-SET**(φ')
  for $(\sigma, D) \in \overline{\mathbf{3}}^{2^{[k]}} \times 2^{[k]}$ **do**
    $E_\varphi[\rho_{i \to j}(\sigma), \rho_{i \to j}(D)] \leftarrow E_\varphi[\rho_{i \to j}(\sigma), \rho_{i \to j}(D)] \uplus E_{\varphi'}[\sigma, D]$
  end
end
if $\varphi = \eta_{i,j}(\varphi')$*:* **then**
  Run **CONNECTED-DOM-SET**(φ')
  for $(\sigma, D) \in \overline{\mathbf{3}}^{2^{[k]}} \times 2^{[k]}$ **do**
    $E_\varphi[\eta_{i,j}(\sigma, D)] \leftarrow E_\varphi[\eta_{i,j}(\sigma, D)] \uplus E_{\varphi'}[\sigma, D]$
  end
end
if $\varphi = \varphi_1 \oplus \varphi_2$ **then**
  Run **CONNECTED-DOM-SET**(φ_1) and **CONNECTED-DOM-SET**(φ_2)
  for $(\sigma_1, D_1) \in \overline{\mathbf{3}}^{2^{[k]}} \times 2^{[k]}$ **do**
    for $(\sigma_2, D_2) \in \overline{\mathbf{3}}^{2^{[k]}} \times 2^{[k]}$ **do**
      $E_\varphi[\sigma_1 + \sigma_2, D_1 \cap D_2] \leftarrow$
        $E_\varphi[\sigma_1 + \sigma_2, D_1 \cap D_2] \uplus (E_{\varphi_1}[\sigma_1, D_1] \bowtie E_{\varphi_2}[\sigma_2, D_2])$
    end
  end
end

Algorithm 1: An algorithm that solves Semiring-Connected-Dominating-Set in time $O^*(3^{2^{k+1}} \times 4^k)$. The notations used are introduced in the proof of Theorem 7.

is $O^*((2^k \times 2^k)^2) = O^*(16^k)$ instead of $O^*((3^{2^k} \times 2^k)^2) = O^*((3^{2^{k+1}} \times 4^k)^2)$ (the square comes from the case where the k-expression φ is of the form $\varphi := \varphi_1 \oplus \varphi_2$).

If one is only interested in solving #Dominating-Set, the running time $O^*(16^k)$ is not optimal since there is a faster $O^*(4^k)$ time algorithm [6]. However, this algorithm could still be generalized while keeping the complexity $O^*(4^k)$ if we only consider *rings* instead of all semirings. The issue of semirings is that the algorithm requires the use of substractions, which is not always possible in semirings. If we allow a new operation $\setminus$ in the semiring expression, we could define the *ring-expressions*: the semantic $[E_1 \setminus E_2]$ of $E_1 \setminus E_2$ would be $[E_1] \setminus [E_2]$ if $[E_2] \subseteq [E_1]$ (assuming E_1 and E_2 are correct), and would fail otherwise. Note that then, for any measure μ taking its value into a ring, we would have the property that $\mu([E_1 \setminus E_2]) = \mu([E_1]) - \mu([E_2])$ for all E_1 and E_2 (assuming the expressions involved are correct), thus Lemma 1 could easily be extended to ring-expressions. This motivates the introduction of the Ring-$\mathbf{\Pi}$ problem with $\mathbf{\Pi} \in \mathsf{NP}$, where one is asked to compute a ring-expression of the set of solutions of $\mathbf{\Pi}$. The Ring-$\mathbf{\Pi}$ problem is easier than Semiring-$\mathbf{\Pi}$ since every semiring expression is in particular a ring-expression. However, it derives strictly less algorithmic applications, since Lemma 1 then only applies if the semiring considered is a ring.

To our knowledge, even if one is only interested in solving the decision problem Dominating-Set parameterized by clique-width, the algorithm running in time $O^*(4^k)$ referred to earlier [6] reaches the best time complexity proposed in the literature. Note that the extension to Ring-Dominating-Set that we propose would preserve this complexity.

6. Semiring-CSP, Sum-Product-CSP and Primal Treewidth

As a general application of our approach we turn to Semiring-CSP and the related problem Sum-Product-CSP (see Section 3.7 for a comparison). The goal of this section is to obtain a semiring expression of the set of solutions of an instance of Csp in FPT time parameterized by the so-called primal treewidth.

The contribution of this section to the semiring formalism presented in this paper is summarized in Figure 4.

6.1. Primal tree-width

Formally, the *Gaifman graph* [30] of an instance $\mathcal{I} = (V, \mathcal{C})$ of Csp(Γ) is the graph $G = (V, E_G)$ where $\{u, v\} \in E_G$ if there exists a constraint in $\mathcal{C}$ containing both u and v. Next, we recall the definition of a tree decomposition of a graph.

Definition 12. A *tree decomposition* [5] of a graph $G = (V_G, E_G)$ is a tuple (T, bag) where $T = (N_T, E_T)$ is a tree and $bag\colon N_T \mapsto 2^{V_G}$ is a function that satisfies the following properties.

1. For all $v \in V_G, \exists N \in N_T$ with $v \in bag(N)$.
2. For all $\{u, v\} \in E_G$, $\exists N \in N_T$ such that $\{u, v\} \subseteq bag(N)$.

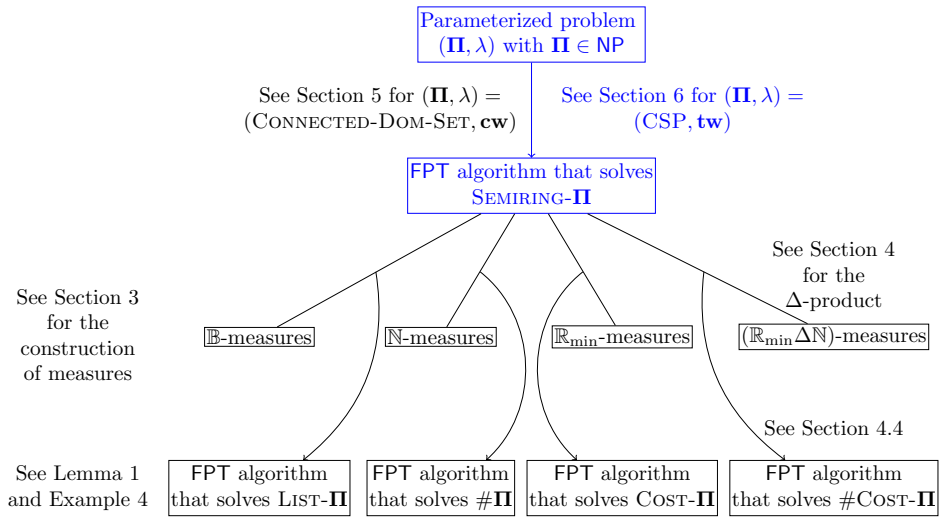

Figure 4: A general overview of our contributions where the contribution in this section is colored in blue.

3. For all $(N_1, N_2, N_3) \in (N_T)^3$, if N_2 is on the (unique) path between N_1 and N_3 in T, then $bag(N_1) \cap bag(N_3) \subseteq bag(N_2)$.

The *treewidth* of a tree decomposition $((N_T, E_T), bag)$ defined as the integer $\max_{N \in N_T} |bag(N)| - 1$. The *treewidth* of a graph G denoted by $\mathbf{tw}(G)$ is then the minimum of the treewidth of its tree decompositions. The *primal treewidth* of an instance of Csp is the treewidth of its Gaifman graph. Additionally, a *nice tree decomposition* [5] of a graph G is a rooted tree decomposition of G where every node N is of one of the four following types.

1. **Leaf:** N is a leaf of T and its bag is empty.
2. **Forget:** N has one child and the bag of N is of the form $bag(N) = bag(child(N)) \setminus \{v\}$ with $v \in bag(child(N))$.
3. **Introduce:** N has one child and the bag of N is of the form $bag(N) = bag(child(N)) \uplus \{v\}$ with $v \in V_G$ and $v \notin bag(child(N))$.
4. **Join:** N has two children N_1 and N_2 with $bag(N) = bag(N_1) = bag(N_2)$.

Here, $bag(N)$ designates the bag associated to a node N of the rooted tree decomposition. We will also use the notation

$$descbag(N) := \bigcup_{N' \text{descendant of } N} bag(N').$$

We remark that any tree decomposition of a graph can easily be transformed into a nice tree decomposition with the same treewidth and in linear time (in the size of the tree decomposition).

We notice in Lemma 2 that $descbag(N)$ and $bag(N)$ depend only on the type of the node N and on the $descbag(N')$, and $bag(N')$ for the children N' of N.

Lemma 2. *Let T be a nice tree decomposition of a graph G, and $N \in V_T$.*

1. *If N is a* **Leaf***: then $descbag(N) = bag(N) = \emptyset$*
2. *If N is of the type* **Forget**(v)*: then, for $v \in V_G$,*

$$descbag(N) = descbag(child(N)) \text{ and } bag(N) = bag(child(N)) \setminus \{v\}.$$

3. *If N is of the type* **Introduce**(v)*: then, for $v \in V_G$, $v \notin descbag(child(N))$, and*

$$descbag(N) = descbag(child(N)) \uplus \{v\} \text{ and } bag(N) = bag(child(N)) \uplus \{v\}.$$

   Moreover, for each edge $\{u, v\} \in E_G$ with $u \in descbag(N)$, $u \in bag(N)$.
4. *If N is of the type* **Join***, and its two children of N are $N_1 \in N_T$ and $N_2 \in N_T$, then $descbag(N) = descbag(N_1) \cup descbag(N_2)$ and*

$$bag(N) = descbag(N_1) \cap descbag(N_2) = bag(N_1) = bag(N_2).$$

   Moreover, there is no edge in G with an endpoint in $descbag(N_1) \setminus bag(N)$ and an endpoint in $descbag(N_2) \setminus bag(N)$.

Proof. We prove each item individually, depending on the type of N.

1. If N is a leaf, then $descbag(N) = bag(N) = \emptyset$ by definition of a nice tree decomposition.
2. If N is of the type **Forget**(v), the result is clear.
3. If N is of the type **Introduce**(v), we fist prove that $v \notin descbag(child(N))$. Otherwise, there exists $N_v \in desc(child(N))$ that contains v, and $child(N)$ is thus on the path between N and N_v. By definition of a tree decomposition, we would have $bag(N) \cap bag(N_v) \subseteq bag(child(N))$ and $v \in bag(child(N))$ which is false. Thus, $v \notin descbag(child(N))$.
   We now prove that for all edge $\{u, v\} \in E_G$ with $u \in descbag(N)$, we have $u \in bag(N)$. Let N_u be a descendant of N with $u \in bag(N_u)$. By definition of a tree decomposition, there exists a node $N_{u,v}$ whose bag contains both u and v. If $N_{u,v}$ is a descendant of N then $child(N)$ is on the path between N and $N_{u,v}$, and we have that $bag(N) \cap bag(N_{u,v}) \subseteq bag(child(N))$. This is a contradiction since $v \in bag(N) \cap bag(N_{u,v})$ but $v \notin bag(child(N))$, so this case is impossible. Thus, since $N_{u,v}$ is not a descendant of N, N is on the path between $child(N)$ and $N_{u,v}$ so $bag(N_u) \cap bag(N_{u,v}) \subseteq bag(N)$, which implies that $u \in bag(N)$.
4. If N is of the type **Join**, and its two children of N are $N_1 \in N_T$ and $N_2 \in N_T$, then $bag(N) = bag(N_1) = bag(N_2)$ by definition of a nice tree decomposition. It follows $descbag(N) = descbag(N_1) \cup descbag(N_2)$. Moreover, by definition of a tree decomposition, we get that $descbag(N_1) \cap descbag(N_2) \subseteq bag(N)$ through a similar reasoning than in the "Introduce" case. Moreover, $bag(N) = bag(N_1) \subseteq descbag(N_1)$, and $bag(N) =$

$bag(N_2) \subseteq descbag(N_2)$. It follows that $bag(N) \subseteq descbag(N_1) \cap descbag(N_2)$, and the equality follows.
We now prove that an edge $\{u, v\} \in E_G$ of G can not have an endpoint $u \in descbag(N_1) \setminus bag(N)$ and the other endpoint $v \in descbag(N_2) \setminus bag(N)$. Assume by contradiction that this is the case. Let N_u a descendant of N_1 with $u \in bag(N_u)$ and symmetrically N_v a descendant of N_2 with $v \in bag(N_v)$. By definition of a tree decomposition, there exists a node $N_{u,v}$ containing both u and v. We remark that N is either on the path between N_u and $N_{u,v}$ contradicting that $u \notin bag(N)$ or on the path between N_v and $N_{u,v}$, contradicting that $v \notin bag(N)$.

□

It can be useful to interpret a nice tree decomposition (T, bag) of a graph G as a construction of a graph with two labels on its vertices, say black and white. For every node N, we can associate a graph G_N built at the node N, whose vertices satisfy:

- The set of vertices of G_N is $descbag(N)$.
- The set of black vertices of G_N is $bag(N)$. The other vertices in $descbag(N) \setminus bag(N)$ are white.

With this interpretation, the treewidth of the tree decomposition is the maximum number of black vertices in a G_N (for N a node of T) minus 1. The interpretation G_N of a node N can be recursively derived from its children, based on the four possible types of N.

1. **Leaf:** Construct the empty graph.
2. **Forget**(v)**:** The black vertex v in $G_{child(N)}$ is now white in G_N.
3. **Introduce**(v)**:** To build G_N, we add a new black vertex v in $G_{child(N)}$, and we can build an edge between v and any black vertex.
4. **Join:** Letting N_1 and N_2 be the two children of N, G_{N_1} and G_{N_2} must have the same black graphs (i.e., their black vertices must induce the same graph). We then obtain G_N from G_{N_1} and G_{N_2} by identifying their black graphs. See Figure 5 for an illustration.

6.2. FPT algorithm

We now give a concrete application of the semiring formalism to the constraint satisfaction problem, by giving an FPT algorithm when parameterized by the primal tree-width. Note that Semiring-Csp subsumes Semiring-Sat, and that the $\mathcal{R}$-semiring extensions of Sat are $\mathsf{NP}(\mathcal{R})$-complete for any commutative semiring $\mathcal{R}$ [13]. Thus, we should not expect to solve these problems in polynomial time and an FPT algorithm is then highly desirable.

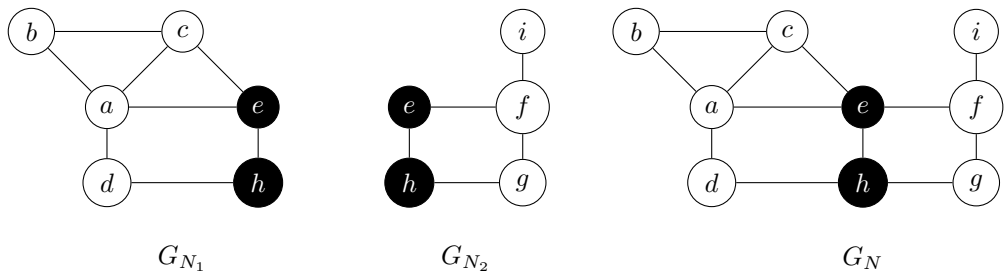

Figure 5: An example of N being a Join node with two children N_1 and N_2. Here, $descbag(N_1) = \{a, b, c, d, e, h\}, descbag(N_2) = \{e, f, g, h, i\}$ and thus $descbag(N) = descbag(N_1) \cup descbag(N_2) = \{a, b, c, d, e, f, g, h, i\}$ and $bag(N) = bag(N_1) = bag(N_2) = descbag(N_1) \cap descbag(N_2) = \{e, h\}$

.

Theorem 8. *For every finite set Γ of relation over a finite domain D,* Semiring-Csp(Γ) *is solvable in time $O^*(|D|^{\mathbf{tw}(G)})$ on any instance $\mathcal{I}$, with G the Gaifman graph of $\mathcal{I}$ (assuming that a tree decomposition of G is given).*

Proof. Let (T, bag) be a nice tree decomposition of the Gaifman graph $G = (V_G, E_G)$ of an instance $\mathcal{I}$ of Semiring-Csp(Γ).

Let $\lambda\colon V_G \to N_T$ be a function which maps every $v \in V_G$ to an arbitrary **Introduce**(v) node $\lambda(v) \in N_T$ with $v \in \lambda(v)$ (clearly, such a node exists for all $v \in V_G$). The interest of the function λ is to prevent the following issue:

In the example of Figure 5, if we had semiring expressions E_{N_1} and E_{N_2} of the set of solutions over $descbag(N_1)$ and $descbag(N_2)$, the semiring expression $E_{N_1} \bowtie E_{N_2}$ would not be correct, as the domains of E_{N_1} and E_{N_2}, which are $descbag(N_1)$ and $descbag(N_2)$, would not be disjoint since their intersection is $bag(N) = \{e, h\}$. To solve the problem, we only express restrictions of such solutions, and λ serves to decide (arbitrarly) who (if any) between N_1 and N_2 takes e and/or h in its domain.

Keeping this technical issue in mind, we introduce the following definitions for each node $N \in N_T$:

- $dom(N) := \{v \in V_G \mid \lambda(v) \in desc(N)\} \subseteq descbag(N)$.
- For each $f\colon bag(N) \to D$,

  $$F_N(f) := \{F : descbag(N) \to D \mid F \text{ is a partial solution s.t. } F|_{bag(N)} = f\},$$

  i.e., for every constraint C that involves only variables in $descbag(N)$, each $F \in \mathcal{F}_N(f)$ satisfies C and

$$E_N(f) := \{F|_{dom(N)} \mid F \in \mathcal{F}_N(f)\}.$$

According to Remark 3, when treating a node N, the "trace" of a partial solution $F : descbag(N) \to D$ will be its restriction to the current $bag(N)$ (i.e.,

its restriction to the "black vertices" of G_N). Then, $\mathcal{F}_N(f)$ is the set of partial solutions that have the trace f. We provide a semiring expression for $\mathcal{E}_N(f)$ instead, as discussed above.

The first item of Remark 3 is indeed respected, as there are at most $|D|^{\mathbf{tw}(G)}$ possible traces at every node (which is FPT with respect to the primal treewidth). Since we give a semiring expression of the sets of partial *solutions*, the second item of Remark 3 is irrelevant.

We now justify the third item of Remark 3 stating that $\mathcal{E}_N(f)$ can be inductively computed, when given a nice tree decomposition. We prove by induction (on the structure of T) that every node $N \in N_T$ satisfies the property:

$\mathcal{P}(N)$: "At the end of the treatment of the node N in Algorithm 2, for each $f\colon bag(N) \to D$, we have $[E_N(f)] = \mathcal{E}_N(f)$."

1. If N is a **Leaf**, $bag(N) = descbag(N) = dom(N) = \emptyset$: the only function f to consider is $f : \emptyset \to D$. Clearly $\mathcal{F}_N(f) = \mathcal{E}_N(f) = D^{\emptyset}$. Hence, $\mathcal{P}(N)$ is true.

2. If N is of the form **Forget**(v) with $v \in bag(child(N))$, assume that $\mathcal{P}(child(N))$ is true. Recall that $bag(N) = bag(child(N)) \setminus \{v\}$ and note that $descbag(N) = descbag(child(N))$ and $dom(N) = dom(child(N))$. We prove that, for each $f : bag(N) \to D$, we have

$$\mathcal{E}_N(f) = \biguplus_{\substack{f' : bag(N) \uplus \{v\} \to D \\ f'|_{bag(N)} = f}} \mathcal{E}_{child(N)}(f')$$

   which implies $\mathcal{P}(N)$ by $\mathcal{P}(child(N))$ and the definition of $E_N(f)$ in Algorithm 2.

   Firstly, notice that $v \in dom(N)$ (i.e., $\lambda(v) \in desc(N)$), because otherwise N is on the path between $child(N)$ and $\lambda(v)$ in T, and we obtain by definition of a tree decomposition that $bag(child(N)) \cap bag(\lambda(v)) \subseteq bag(N)$, which is a contradiction, since $v \in (bag(child(N)) \cap bag(\lambda(v))) \setminus bag(N)$.

   Then, the partitioning of $\mathcal{E}_N(f)$ into equivalence classes of $\sim$ defined by

$$\forall (F_1, F_2) \in (\mathcal{E}_N(f))^2, F_1 \sim F_2 \iff F_1(v) = F_2(v),$$

   gives the desired equality.

3. If N is of the form **Introduce**(v) with $v \notin bag(child(N))$, assume that $\mathcal{P}(child(N))$ is true. Recall that $bag(N) = bag(child(N)) \uplus \{v\}$ and that $descbag(N) = descbag(child(N)) \uplus \{v\}$, and that

$$dom(N) = \begin{cases} dom(child(N)) & \text{if } \lambda(v) \neq N \\ dom(child(N)) \uplus \{v\} & \text{if } \lambda(v) = N \end{cases}$$

Indeed, $v \notin descbag(child(N))$ by Lemma 2, which implies in particular that $v \notin dom(child(N))$.

Recall that proving $\mathcal{P}(N)$ knowing $\mathcal{P}(child(N))$, requires a relation between $\mathcal{E}_N(f)$ and $\mathcal{E}_{child(N)}(f|_{bag(child(N))})$. Since $\mathcal{E}_N(f)$ is defined using $\mathcal{F}_N(f)$, it will be usefull to compute $\mathcal{F}_N(f)$.

We prove:

$$\mathcal{F}_N(f) = \begin{cases} \emptyset & \text{if } f \text{ violates a constraint} \\ \mathcal{F}_{child(N)}(f|_{bag(child(N))}) \bowtie \{v \mapsto f(v)\} & \text{if } f \text{ respects every constraint} \end{cases}$$

Firstly, notice that $\mathcal{F}_{child(N)}(f|_{bag(child(N))}) \bowtie \{v \mapsto f(v)\}$ is a valid operation, since $v \notin descbag(N)$.

If f violates a constraint, every function that has f as a restriction also violates a constraint, and is thus not a partial solution. This justifies that $\mathcal{F}_N(f) = \emptyset$ if f violates a constraint.

If f respects every constraint, notice that every element $F : descbag(N) \mapsto D$ of $\mathcal{F}_N(f)$ is in particular in $\mathcal{F}_{child(N)}(f|_{bag(child(N))}) \bowtie \{v \mapsto f(v)\}$. Indeed, since F coincides with f on $bag(N) = bag(child(N)) \uplus \{v\}$, we have $F(v) = f(v)$, and $F|_{bag(child(N))}$ is a partial solution (it is a restriction of the partial solution F) and coincides with $f|_{bag(child(N))}$ on $bag(child(v))$.

Conversely, we prove that every map $F : descbag(N) \to D$ of

$$\mathcal{F}_{child(N)}(f|_{bag(child(N))}) \bowtie \{v \mapsto f(v)\}$$

is a partial solution.

Notice first that $F|_{bag(child(N))}$ is in $\mathcal{F}_{child(N)}(f|_{bag(child(N))})$, which is a solution. Therefore, F respects every constraint that is not involving v. We now prove that F respects every constraint involving v.

Let $u \in descbag(N)$ be such that $\{u, v\}$ is the edge of the Gaifman graph G. Let N_u be a descendant of N with $u \in bag(N_u)$. By Lemma 2, we have $u \in bag(N)$ and, by definition of the Gaifman graph G of the instance $\mathcal{I}$ of Csp, this proves that any constraint over $descbag(N)$ that involves v involves only variables of $bag(N)$.

Since $f : bag(N) \to D$ respects the constraints over $bag(N)$ (by assumption), and since F coincides with f on $bag(N)$, this shows that F respects every constraint involving v, and thus F respects every constraint over $descbag(N)$. Hence, F is a partial solution. This proves that

$$\mathcal{F}_N(f) = \mathcal{F}_{child(N)}(f|_{bag(child(N))}) \bowtie \{v \mapsto f(v)\}.$$

By restricting to $dom(N)$, we get that:

$$\mathcal{E}_N(f) = \begin{cases} \emptyset & \text{if } f \text{ violates a constraint} \\ \mathcal{E}_{child(N)}(f|_{bag(child(N))}) & \text{if } f \text{ respects every constraint and } \lambda(v) \neq N \\ \mathcal{E}_{child(N)}(f|_{bag(child(N))}) \bowtie \{v \mapsto f(v)\} & \text{if } f \text{ respects every constraint and } \lambda(v) = N \end{cases}$$

and thus $\mathcal{P}(N)$ conforms to $\mathcal{P}(child(N))$ and the definition of $E_N(f)$ in Algorithm 2.

4. If N is of the form $\mathbf{Join}(N_1, N_2)$, assume that $\mathcal{P}(N_1)$ and $\mathcal{P}(N_2)$ are true. Recall that $bag(N) = bag(N_1) = bag(N_2)$ and that $descbag(N) = descbag(N_1) \cup descbag(N_2)$, $bag(N) = descbag(N_1) \cap descbag(N_2)$ by Lemma 2. Note also that $dom(N) = dom(N_1) \uplus dom(N_2)$.

   Let $f\colon bag(N) \to D$. We prove that $\mathcal{F}_N(f) = \mathcal{F}_{N_1,N_2}(f)$, where

$$\begin{aligned} \mathcal{F}_{N_1,N_2}(f) &:= \{F_1|_{descbag(N_1)\setminus bag(N)} \mid F \in \mathcal{F}_{N_1}(f)\} \\ &\bowtie \{F_2|_{descbag(N_2)\setminus bag(N)} \mid F \in \mathcal{F}_{N_2}(f)\} \bowtie \{f\} \\ &= \mathcal{F}_{N_1}(f) \bowtie \{F_2|_{descbag(N_2)\setminus bag(N)} \mid F \in \mathcal{F}_{N_2}(f)\} \\ &= \{F_1|_{descbag(N_1)\setminus bag(N)} \mid F \in \mathcal{F}_{N_1}(f)\} \bowtie \mathcal{F}_{N_2}(f). \end{aligned}$$

   The idea is that every pair $(F_1, F_2) \in \mathcal{F}_{N_1}(f) \times \mathcal{F}_{N_2}(f)$ coincides on the intersection of their domain, since by definition, they both coincide with f on $bag(N) = descbag(N_1) \cap descbag(N_2)$. Note that we cannot use the notation $F_1 \bowtie F_2$ since their domain is not disjoint but that the sets $\mathcal{E}_N(f), \mathcal{E}_{N_1}(f)$ and $\mathcal{E}_{N_2}(f)$ are more convenient to use since $dom(N_1)$ and $dom(N_2)$ are disjoint.

   Since a restriction of a partial solution is still a partial solution, every element of $\mathcal{F}_N(f)$ is in $\mathcal{F}_{N_1,N_2}(f)$. We now prove that $\mathcal{F}_{N_1,N_2}(f) \subseteq \mathcal{F}_N(f)$.

   By Lemma 2, every edge $\{u, v\} \in E_G$ in $descbag(N)$ can not have an endpoint in $descbag(N_1) \setminus bag(N)$ and the other endpoint in $descbag(N_2) \setminus bag(N)$. So, by definition of the Gaifman graph, every constraint in $descbag(N)$ is either a constraint over $descbag(N_1)$ or over $descbag(N_2)$ (recall that $bag(N) = descbag(N_1) \cap descbag(N_2)$ by Lemma 2.

   Take $F : descbag(N) \to D$ of $\mathcal{F}_{N_1,N_2}(f)$. We have that F coincides with f on $bag(N) = bag(N_1) = bag(N_2)$, and thus F coincides with a partial solution $F_1 \in \mathcal{F}_{N_1}(f)$ on $descbag(N_1)$. This proves that F respects every constraints over $descbag(N_1)$. Symmetrically, F respects every constraints over $descbag(N_2)$. We have proven that F respects all the constraints, ie. F is a partial solution, and thus F is in $\mathcal{F}_N(f)$. This proves that $\mathcal{F}_N(f) = \mathcal{F}_{N_1,N_2}(f)$. We get as a corollary (by restricting to $dom(N) = dom(N_1) \uplus dom(N_2)$) that:

$$\mathcal{E}_N(f) = \mathcal{E}_{N_1}(f) \bowtie \mathcal{E}_{N_2}(f)$$

and $\mathcal{P}(N)$ follows.

This proves $\mathcal{P}(N)$, for every node $N \in N_T$.

Now, since $bag(root(T)) = \emptyset$, $dom(root(T)) = descbag(root(T)) = V_G$, $\mathcal{P}(root(T))$ justifies that $[E_{root(T)}(\emptyset \to D)]$ encodes the set of solutions of $\mathcal{I}$.

Lastly, it is straightforward to see that the complexity of Algorithm 2 is indeed $O^*(|D|^{\mathbf{tw}(G)})$. □

In particular, for Semiring-H-Coloring we obtain the following corollary.

Corollary 1. *For every graph H,* Semiring-H-Coloring *is solvable in time $O^*(|V_H|^{\mathbf{tw}(G)})$.*

In particular, Corollary 1 implies that H-Coloring, #H-Coloring, List-H-Coloring and #Cost-H-Coloring can be solved in $O^*(|V_H|^{\mathbf{tw}(G)})$ time. To the best of our knowledge, no general algorithm for counting H-colorings of minimal cost was previously known in the literature.

We now indicate how to generalize this algorithm to solve the even more general problem Sum-Product-Csp (Algorithm 3). If we restrict Algorithm 3 to solve only #Csp, this algorithm is a particular case of an already known algorithm [16]. In Theorem 9, we require that each constraint C is mapped to a unique **Introduce** node $\lambda(C)$ such that every vertex of the Gaifman graph (i.e., variable) involved in constraint C belongs to $bag(\lambda(C))$. This property can always be ensured because such a set of vertices is a clique (complete graph), and because a clique in a graph is necessarily contained in the bag of one node in every tree decomposition.

Theorem 9. *For every finite set Γ of constraints over a finite domain D,* Sum-Product-Csp(Γ) *is solvable in time $O^*(|D|^{\mathbf{tw}(G(\mathcal{I}))})$ on any instance $\mathcal{I}$, where $\mathbf{tw}(G(\mathcal{I}))$ is the treewidth of the Gaifman graph of $\mathcal{I}$ (assuming that a tree decomposition is given).*

Proof. Similarly to Theorem 8 we give a proof by induction on every node N of the rooted tree decomposition via the property:

$\mathcal{P}(N)$:"For every $f : bag(N) \to D$, in the execution of Algorithm 3, $E_N(f)$ contains

$$\sum_{\substack{F\,:\,descbag(N) \to D \\ F|_{bag(N)} = f}} \prod_{\substack{C \text{ contraint} \\ \lambda(C) \text{ is a descendant of } N}} C(F)." \quad \square$$

Moreover, this running time seems to be optimal at least in the case of k-Coloring and in most cases of H-Coloring, while giving access to every semiring extension. Indeed, we even have a lower bound for the k-Coloring problem, that relies on the *strong exponential time hypothesis* (SETH) [7]. The SETH is a well-known conjecture within complexity theory which states that for every $\varepsilon > 0$, there exists $k \geq 3$ such that k-Sat problem can not be solved in time $O^*((2-\varepsilon)^n)$, where n is the number of variables.

Data: An instance $\mathcal{I}$ of $\textsc{Csp}(\Gamma)$, a nice tree decomposition $T = (N_T, E_T)$ of its Gaifman graph G, whose root has empty bag (which can be done by forgetting every node of the bag of the root), and a function $\lambda : V_G \to N_T$ where $\forall v \in V_G$, $\lambda(v)$ is an Introduce node with $v \in bag(\lambda(v))$.

Result: A semiring expression E_{SOL} over (V_G, D) encoding the set of solutions of $\mathcal{I}$.

Run **Expressions**$(root(T))$ and return $E_{root(T)}(\emptyset \to D)$ with **Expressions**(N) being defined for each node N of T as:

Expressions(N):
if N *is a leaf* ($bag(N) = \emptyset$) **then**
  $E_N(f) := D^{\emptyset}$
end
if N *is Forget*(v) **then**
  Run **Expressions**$(child(N))$
  for $f : bag(N) \to D$ **do**
    $E_N(f) := \emptyset$
  end
  for $f' : bag(N) \cup \{v\} \to D$ **do**
    $E_N(f'|_{bag(N)}) \leftarrow E_N(f'|_{bag(N)}) \uplus E_{child(N)}(f')$
  end
end
if N *is Introduce*(v) **then**
  Run **Expressions**$(child(N))$
  for $f : bag(N) \to D$ **do**
    if f *respects every contraint and* $\lambda(v) = N$ **then**
      $E_N(f) := E_{child(N)}(f|_{bag(N)\setminus\{v\}}) \bowtie \{v \mapsto f(v)\}$
    end
    if f *respects every contraint and* $\lambda(v) \neq N$ **then**
      $E_N(f) := E_{child(N)}(f|_{bag(N)\setminus\{v\}})$
    end
    if f *violates a constraint* **then**
      $E_N(f) := \emptyset$
    end
  end
end
if N *is Join*(N_1, N_2) **then**
  Run **Expressions**(N_1) and **Expressions**(N_2)
  for $f : bag(N) \to D$ **do**
    $E_N(f) := E_{N_1}(f) \bowtie E_{N_2}(f)$
  end
end

Algorithm 2: Algorithm to construct an expression of the set of solutions of an instance of $\textsc{Csp}(\Gamma)$ in time $|D|^{\mathbf{tw}(G)}$ (with D being the domain of the relations in Γ).

Data: An instance $\mathcal{I}$ of Sum-Product-Csp(Γ), a nice tree decomposition T of its Gaifman graph G, whose root has empty bag, and such that every constraint C is mapped to a unique Introduce node $\lambda(C)$.

Result: An element E_{SOL} of the semiring containing

$$\sum_{F:V_G \to D} \prod_{C \text{ contraint}} C(F)$$

Run **Sum-Product**($root(T)$) and return $E_{root(T)}(\emptyset \to D)$ with **Sum-Product**(N) being defined for all node N of T as:

```
Sum-Product(N):
if N is a leaf (bag(N) = ∅) then
|  E_N(f) := 1
end
if N is Forget(v) then
|  Run Sum-Product(child(N))
|  for f : bag(N) → D do
|  |  E_N(f) := 0
|  end
|  for f' : bag(N) ∪ {v} → D do
|  |  E_N(f'|_bag(N)) ← E_N(f'|_bag(N)) + E_child(N)(f')
|  end
end
if N is Introduce(v) then
|  Run Sum-Product(child(N))
|  for f : bag(N) → D do
|  |  E_N(f) := E_child(N)(f|_bag(N)\{v})
|  |  for C constraint with λ(C) = N do
|  |  |  E_N(f) ← E_N(f) × C(f)
|  |  end
|  end
end
if N is Join(N_1, N_2) then
|  Run Sum-Product(N_1) and Sum-Product(N_2)
|  for f : bag(N) → D do
|  |  E_N(f) ← E_N_1(f) × E_N_2(f)
|  end
end
```

Algorithm 3: Algorithm to solve an instance of Sum-Product-Csp(Γ) in time $|D|^{\mathbf{tw}(G)}$ (with D being the domain of the relations of Γ).

Theorem 10. *[29] For all $k \geq 3$ and $\varepsilon > 0$, k-*Coloring *is not solvable in time $O^*((k-\varepsilon)^{\mathbf{tw}(G)})$ under the* SETH.

This lower bound applies to H-Coloring problems for a broad classes of graphs. Precisely, Theorem 11 applies as long as H is a so-called *projective core* on at least 3 vertices [29].

Theorem 11. *[29] If H is a projective core on at least three vertices, then, for every $\varepsilon > 0$, H-*Coloring *is not solvable in time $O^*((|V_H| - \varepsilon)^{\mathbf{tw}(G)})$ under the* SETH.

Thus running time obtained in Corollary 1 (and even more so in Theorem 8 and Theorem 9) cannot not be asymptotically improved under the SETH. Note also that, asymptotically, almost all graphs are projective cores [18, 27, 29], and thus Theorem 11 applies to almost all graphs.

7. Conclusion

In this article we explored semiring extensions for computational problems. We proposed a general approach (Section 3) that does not require particular a semiring structure, and that is applicable to any problem with a reasonable notion of a certificate (e.g., any problem in NP). In Section 4 we proceeded to defining a novel operation on semirings, the Δ-product, which increases the scope of semiring algorithms and, in particular, that solves the problem of counting solutions of minimal cost. To illustrate our framework, we turned our attention to two well-known problems with a vast number of applications: Connected-Dominating-Set and finite-domain Csps. For the former we successfully gave an algorithm that outputs a semiring extension in FPT time when parameterized by the clique-width of the input graph (Section 5). Despite being a problem with many practical applications [10], the counting extension of this problem was not known to be FPT. Moreover, we proved a similar result for the Csp problem parameterized by primal treewidth, and showed how to extend the algorithm to the more general Sum-Product-Csp problem (Section 6).

Let us now discuss some potential directions of future research.

7.1. Algorithmic applications

We have seen that existing algorithms in the literature (e.g., the algorithm by Ganian et al. [16] for #Csp) can be generalized (under some assumptions) to semiring extensions. Is this a common behaviour? Are there more examples of algorithms in the literature, whether it is for the counting, list, or cost version, which can be easily adapted to solving the semiring extension? Or are there problems where, e.g., counting the number of solutions of minimal cost is prohibitively more computationally expensive than merely counting solutions?

As discussed at the end of Section 5, our work opens up the study of a novel *ring formalism* built on *ring-expressions*, where a new operation $\setminus$ corresponding to the set difference is introduced. Thus, the semantics of $[E_1 \setminus E_2]$ for two

correct ring expressions E_1 and E_2 such that $[E_2] \subseteq [E_1]$, would be $[E_1] \setminus [E_2]$. Indeed, it is not hard to verify that if a semiring measure takes its value in a ring instead, Lemma 1 still applies even if a ring-expression is given (by mapping $\setminus$ to the additive inverse $-_A$ of the ring A), which enables similar algorithmic applications.

From a complexity perspective, it would be interesting to unify problems of the form Semiring-$\mathbf{\Pi}$ with $\mathbf{\Pi}$ in a new complexity class and, similarily, counting, optimizing, and counting solutions of optimal costs to NP problems belonging to the classes #P, OptP and #·OptP. Moreover, a convenient notion of "reduction" preserving this complexity class would be desirable, as well as a completeteness notion for Semiring problems. Note that some #P-complete problems of these classes are extensions of NP problems that are not NP-complete (unless P=NP). For instance, counting the number of perfect matchings of a bipartite graph is known to be #P-complete [32], whereas deciding if one exists is in P. Does there also exist easy problems whose semiring extensions become hard?

7.2. Sum-Product CSP

Even though our formalism extends decision problems (corresponding to the Boolean semiring $\mathbb{B}$) to arbitrary semirings, our formalism does not subsume "soft constraints" allowed, e.g., in the Sum-Product-Csp problem. Is there a possibility that any candidate solution f, instead of being either a solution ($f \in SOL$) or not a solution ($f \notin SOL$), could be mapped to a semiring value instead? Then, the formalism of "hard constraints" presented in this article would correspond to the particular case where $f \in SOL$ if and only if f is mapped to $\top$ (and $f \notin SOL$ if and only if f is mapped to $\bot$).

7.3. Combinatoric and algebraic results

The definition of a semiring expression and its algorithmic applications raises the question of which sets of functions that can be expressed efficiently. In particular, Theorem 1 and Theorem 2 suggest that the sets $\mathfrak{S}_n$ and $\mathfrak{C}_n$ can not be represented by an expression of polynomial size. Proving or disproving Conjecture 1 and Conjecture 2 would likely provide useful tools to better understand semiring expressions. As a first step, it would be interesting to see if we could determine at least if Conjecture 1 implies Conjecture 2, and vice versa. Note that we indeed have the bijection:

$$\Phi : \begin{array}{ccc} \mathfrak{S}_n & \to & \mathfrak{C}_{n+1} \\ \sigma & \mapsto & (\sigma(1)\ \sigma(2)\ \dots\ \sigma(n)\ n+1) \end{array}$$

for all $n \geq 1$, whose inverse is

$$\Psi : \begin{array}{ccc} \mathfrak{C}_{n+1} & \to & \mathfrak{S}_n \\ c & \mapsto & \begin{pmatrix} 1 & 2 & \dots & n \\ c(n+1) & c^2(n+1) & \dots & c^n(n+1) \end{pmatrix}. \end{array}$$

However, it is not clear if the bijection Φ (respectively Ψ) can be used to derive a semiring expression of $\mathfrak{C}_{n+1}$ (respectively $\mathfrak{S}_n$) from a semiring expression of $\mathfrak{S}_n$ (respectively $\mathfrak{C}_{n+1}$).

7.4. Sequential semiring expressions

It may be useful to introduce *sequences of semiring expressions*, i.e., sequences of the form $(\mathcal{E}_0, \ldots, \mathcal{E}_m)$ with $m \geq 0$, and where $\mathcal{E}_0$ is a set of semiring expressions and, for all $i \in [m]$, $\mathcal{E}_i$ is a set of semiring expressions whose leaves can also be expressions of $\mathcal{E}_j$ with $j < i$. We may then obtain an result analogous to Lemma 1 by iteratively computing $\mu([E])$ for $E \in \mathcal{E}_0$, and then $\mu([E])$ for $E \in \mathcal{E}_1, \ldots$. This may become relevant since, even if a semiring expression E is repeated multiple times, it would not be necessary to compute $\mu([E])$ several times, similarly to dynamic programming.

Considering that Theorem 1 and Theorem 2 also apply to sequential semiring expressions (since their proofs only exploit algorithmic applications), it could be interesting to answer Conjecture 1 and Conjecture 2 when sequential semiring expressions are considered.

Acknowledgement

The authors wish to thank Pierre Charbit, Eun Jung Kim and Noleen Köhler for the many fruitful discussions. We are particularly thankful for their ideas and suggestions for improving the contents of the paper.

References

[1] Bacchus, F., Dalmao, S., Pitassi, T., 2009. Solving sat and bayesian inference with backtracking search. Journal of Artificial Intelligence Research 34, 391–442.

[2] Bergougnoux, B., Kanté, M., 2019. Fast exact algorithms for some connectivity problems parameterized by clique-width. Theoretical Computer Science 782, 30–53.

[3] Berman, J., Blok, W., 2001. Stipulations, multi-valued logic and de morgan algebras. Multi-valued Logic 7, 391–416.

[4] Bistarelli, S., Montanari, U., Rossi, F., 1997. Semiring-based constraint satisfaction and optimization. Journal of the ACM (JACM) 44, 201–236.

[5] Bodlaender, H.L., Kloks, T., 1991. Better algorithms for the pathwidth and treewidth of graphs, in: Automata, Languages and Programming: 18th International Colloquium Madrid, Spain, July 8–12, 1991 Proceedings 18, Springer. pp. 544–555.

[6] Bodlaender, H.L., Van Leeuwen, E.J., Van Rooij, J.M., Vatshelle, M., 2010. Faster algorithms on branch and clique decompositions, in: Mathematical Foundations of Computer Science 2010: 35th International Symposium, MFCS 2010, Brno, Czech Republic, August 23-27, 2010. Proceedings 35, Springer. pp. 174–185.

[7] Calabro, C., Impagliazzo, R., Paturi, R., 2009. The complexity of satisfiability of small depth circuits, in: Parameterized and Exact Computation, 4th International Workshop (IWPEC 2009), pp. 75–85. URL: https://doi.org/10.1007/978-3-642-11269-0_6, doi:10.1007/978-3-642-11269-0_6.

[8] Courcelle, B., Makowsky, J.A., Rotics, U., 2000. Linear time solvable optimization problems on graphs of bounded clique-width. Theory of Computing Systems 33, 125–150.

[9] Courcelle, B., Olariu, S., 2000. Upper bounds to the clique width of graphs. Discrete Applied Mathematics 101, 77–114.

[10] Du, D.Z., Wan, P.J., 2012. Connected Dominating Set: Theory and Applications. Springer Publishing Company, Incorporated.

[11] Duan, R., Wu, H., Zhou, R., 2022. Faster matrix multiplication via asymmetric hashing. arXiv preprint arXiv:2210.10173 .

[12] Eiter, T., Kiesel, R., 2021. On the complexity of sum-of-products problems over semirings, in: Proceedings of the AAAI Conference on Artificial Intelligence, pp. 6304–6311.

[13] Eiter, T., Kiesel, R., 2023. Semiring reasoning frameworks in ai and their computational complexity. Journal of Artificial Intelligence Research 77, 207–293.

[14] Fan, A.Z., Koutris, P., Zhao, H., 2023. The fine-grained complexity of boolean conjunctive queries and sum-product problems. arXiv preprint arXiv:2304.14557 .

[15] Floyd, R.W., 1968. Algorithm 97 (shortest path). Communications of the ACM 18.

[16] Ganian, R., Ramanujan, M.S., Szeider, S., 2016. Combining treewidth and backdoors for csp. arXiv preprint arXiv:1610.03298 .

[17] Hegerfeld, F., Kratsch, S., 2023. Tight algorithms for connectivity problems parameterized by clique-width. arXiv preprint arXiv:2302.03627 .

[18] Hell, P., Nešetřil, J., 1992. The core of a graph. Discrete Mathematics 109, 117–126.

[19] Hermann, M., Pichler, R., 2008. Counting complexity of minimal cardinality and minimal weight abduction, in: European Workshop on Logics in Artificial Intelligence, Springer. pp. 206–218.

[20] Hermann, M., Pichler, R., 2009. Complexity of counting the optimal solutions. Theoretical computer science 410, 3814–3825.

[21] Hertrampf, U., 1990. Relations among mod-classes. Theoretical Computer Science 74, 325–328.

[22] Jensen, T.R., Toft, B., 2011. Graph coloring problems. John Wiley & Sons.

[23] Kleene, S.C., 1938. On notation for ordinal numbers. The Journal of Symbolic Logic 3, 150–155.

[24] Kleene, S.C., 1956. Representation of Events in Nerve Nets and Finite Automata. Princeton University Press. volume 34 of *Annals of Mathematics Studies*. pp. 3–42.

[25] Krentel, M.W., 1986. The complexity of optimization problems, in: Proceedings of the eighteenth annual ACM symposium on Theory of computing, pp. 69–76.

[26] Lehmann, D.J., 1977. Algebraic structures for transitive closures. Theoretical Computer Science 4, 59–76.

[27] Łuczak, T., Nešetřil, J., 2004. Note on projective graphs. Journal of Graph Theory 47, 81–86.

[28] Mohri, M., 2002. Semiring frameworks and algorithms for shortest-distance problems. Journal of Automata, Languages and Combinatorics 7, 321–350. URL: `https://doi.org/10.25596/jalc-2002-321`.

[29] Okrasa, K., ażewski, P.R., 2021. Fine-grained complexity of the graph homomorphism problem for bounded-treewidth graphs. SIAM J. Comput. 50, 487–508.

[30] Rossi, F., Van Beek, P., Walsh, T., 2006. Handbook of constraint programming. Elsevier.

[31] Thomassen, C., 1995. 3-list-coloring planar graphs of girth 5. Journal of Combinatorial Theory, Series B 64, 101–107.

[32] Valiant, L.G., 1979. The complexity of computing the permanent. Theoretical Computer Science 8, 189–201.

[33] Warshall, S., 1962. A theorem on boolean matrices. Journal of the ACM 9(1), 11–12.

Appendix A. Proofs of Section 4

We recall and prove Theorem 3.

Theorem 3. *The structure* $\mathcal{D}\Delta\mathcal{A} = (D\Delta A, \oplus, \otimes, (\infty_D, 0_A), (0_D, 1_A))$ *is a commutative semiring with:*

- $D\Delta A := \{d\Delta a \mid (d,a) \in D \times A\} = \{(d,a) \in D \times A \mid d \in reg(\mathcal{D})$ *or* $a = 0_A\}$.
- $\oplus : \begin{array}{ccc} (D\Delta A)^2 & \to & D\Delta A \\ ((d_1,a_1),(d_2,a_2)) & \mapsto & (\min(d_1,d_2), \left\{ \begin{array}{cc} a_1 & \textit{if } d_1 <_D d_2 \\ a_2 & \textit{if } d_2 <_D d_1 \\ a_1 +_A a_2 & \textit{if } d_1 = d_2 \end{array} \right\}) \end{array}$

  and
- $\otimes : \begin{array}{ccc} (D\Delta A)^2 & \to & D\Delta A \\ ((d_1,a_1),(d_2,a_2)) & \mapsto & ((d_1 +_D d_2),(a_1 \times_A a_2)) \end{array}$.

Proof. Note that we can reformulate the definition of $\oplus$ as:

$$\oplus : \begin{array}{ccc} (D\Delta A)^2 & \to & D\Delta A \\ ((d_1,a_1),(d_2,a_2)) & \mapsto & \left\{ \begin{array}{cc} (d_1,a_1) & \text{if } d_1 <_D d_2 \\ (d_2,a_2) & \text{if } d_2 <_D d_1 \\ (d_1, a_1 +_A a_2) & \text{if } d_1 = d_2 \end{array} \right\} \end{array}$$

by Property 1.
First, we need to prove that $D\Delta A$ is indeed stable by $\oplus$ and $\otimes$.

- We prove that $D\Delta A$ is stable by $\oplus$.

  Let $((d_1,a_1),(d_2,a_2)) \in (D\Delta A)^2$.

  - Assume that $d_1 \neq d_2$. Then $(d_1,a_1) \oplus (d_2,a_2) \in \{(d_1,a_1),(d_2,a_2)\}$. It follows that $(d_1,a_1) \oplus (d_2,a_2) \in D\Delta A$.
  - Assume that $d_1 = d_2$ is multiplicatively regular. Then, $(d_1,a_1) \oplus (d_2,a_2) = (d_1, a_1 +_A a_2) \in D\Delta A$ because d_1 is regular.
  - Assume that $d_1 = d_2$ is not multiplicatively regular. Then, since $((d_1,a_1),(d_2,a_2)) \in (D\Delta A)^2$, it follows that $a_1 = a_2 = 0_A$. Then, $(d_1,a_1) \oplus (d_2,a_2) = (d_1, a_1 +_A a_2) \in D\Delta A$ because $a_1 +_A a_2 = 0_A +_A 0_A = 0_A$.

  $D\Delta A$ is stable by $\oplus$.

- We prove that $D\Delta A$ is stable by $\otimes$.

  Let $((d_1,a_1),(d_2,a_2)) \in (D\Delta A)^2$.

  - Assume that d_1 and d_2 are multiplicatively regular. Then, by Property 3, $d_1 +_D d_2$ is multiplicatively regular, and thus $(d_1,a_1) \otimes (d_2,a_2) = (d_1 +_D d_2, a_1 \times_A a_2) \in D\Delta A$.

- Assume that d_1 is not multiplicatively regular, then since $(d_1, a_1) \in D\Delta A$, $a_1 = 0_A$, and thus $(d_1, a_1) \otimes (d_2, a_2) = (d_1 +_D d_2, a_1 \times a_2) \in D\Delta A$ because $a_1 \times_A a_2 = 0_A \times a_2 = 0_A$.
- Similar reasoning and conclusion if d_2 is not multiplicatively regular.

$D\Delta A$ is stable by $\otimes$.

- $\oplus$ and $\otimes$ are clearly commutative by the commutativity of $\min_D, +_D, +_A$ and $\times_A$.
- We now prove the associativity of $\oplus$:

  Let $((d_1, a_1), (d_2, a_2), (d_3, a_3)) \in (D\Delta A)^3$. We prove that $(d_1, a_1) \oplus ((d_2, a_2) \oplus (d_3, a_3)) = ((d_1, a_1) \oplus (d_2, a_2)) \oplus (d_3, a_3)$.

  We only prove the result when $d_1 \leq_D d_2 \leq_D d_3$. The other cases are similar.

  - Assume that $d_1 <_D d_2$. Then there exists a such that $(d_2, a_2) \oplus (d_3, a_3) = (d_2, a)$. Indeed, we take $a := a_2$ if $d_2 < d_3$ and $a := a_2 +_A a_3$ if $d_2 = d_3$. Hence,

    $$(d_1, a_1) \oplus ((d_2, a_2) \oplus (d_3, a_3)) = (d_1, a_1) \oplus (d_2, a) = (d_1, a_1)$$

    and

    $$((d_1, a_1) \oplus (d_2, a_2)) \oplus (d_3, a_3) = (d_1, a_1) \oplus (d_3, a_3) = (d_1, a_1),$$

    because $d_1 <_D d_3$.
  - Assume that $d_1 = d_2 <_D d_3$. Then

    $$(d_1, a_1) \oplus ((d_2, a_2) \oplus (d_3, a_3)) = (d_1, a_1) \oplus (d_2, a_2) = (d_1, a_1 +_A a_2)$$

    and

    $$((d_1, a_1) \oplus (d_2, a_2)) \oplus (d_3, a_3) = (d_1, a_1 +_A a_2) \oplus (d_3, a_3) = (d_1, a_1 +_A a_2).$$

  - Assume that $d_1 = d_2 = d_3$. Then,

    $$\begin{aligned} (d_1, a_1) \oplus ((d_2, a_2) \oplus (d_3, a_3)) &= (d_1, a_1) \oplus (d_2, a_2 +_A a_3) \\ &= (d_1, a_1 +_A (a_2 +_A a_3)) \end{aligned}$$

    and

    $$\begin{aligned} ((d_1, a_1) \oplus (d_2, a_2)) \oplus (d_3, a_3) &= (d_1, a_1 +_A a_2) \oplus (d_3, a_3) \\ &= (d_1, a_1 +_A (a_2 +_A a_3)) \\ &= (d_1, (a_1 +_A a_2) +_A a_3), \end{aligned}$$

    by the associativity of $+_A$.

- We now prove the associativity of $\otimes$. Let $((d_1, a_1), (d_2, a_2), (d_3, a_3)) \in (D\Delta A)^3$. Then

$$\begin{aligned}
(d_1, a_1) \otimes ((d_2, a_2) \otimes (d_3, a_3)) &= (d_1, a_1) \otimes (d_2 +_D d_3, a_2 \times_A a_3) \\
&= (d_1 +_D (d_2 +_D d_3), a_1 \times_A (a_2 \times_A a_3)) \\
&= ((d_1 +_D d_2) +_D d_3, (a_1 \times_A a_2) \times_A a_3) \\
&= (d_1, a_1) \otimes ((d_2, a_2) \otimes (d_3, a_3)) \\
&= (d_1, a_1) \otimes (d_2 +_D d_3, a_2 \times_A a_3) \\
&= (d_1 +_D (d_2 +_D d_3), a_1 \times_A (a_2 \times_A a_3)) \\
&= ((d_1 +_D d_2) +_D d_3, (a_1 \times_A a_2) \times_A a_3) \\
&= ((d_1, a_1) \otimes (d_2, a_2)) \otimes (d_3, a_3),
\end{aligned}$$

  by associativity of $+_D$ and $\times_A$.

- $(\infty_D, 0_A)$ is clearly neutral for $\oplus$ because ∞_D is neutral for $\min_D$, and 0_A is neutral for $+_A$.

- $(0_D, 1_A)$ is clearly neutral for $\otimes$, because 0_D is neutral for $+_D$, and because 1_A is neutral for $\times_A$.

- $(\infty_D, 0_A)$ is clearly absorbing for $\otimes$, because ∞_D is absorbing for $+_D$ and because 0_A is absorbing for $\times_A$.

- We now prove that $\otimes$ is distributive over $\oplus$.

  Let $((d_1, a_1), (d_2, a_2), (d_3, a_3)) \in (D\Delta A)^3$. On the one hand,

$$\begin{aligned}
((d_1, a_1) \oplus (d_2, a_2)) \quad &\otimes \quad (d_3, a_3) \\
&= \left\{ \begin{array}{cc} (d_1, a_1) & \text{if } d_1 <_D d_2 \\ (d_2, a_2) & \text{if } d_2 <_D d_1 \\ (d_1, a_1 +_A a_2) & \text{if } d_1 = d_2 \end{array} \right\} \otimes (d_3, a_3) \\
&= \left\{ \begin{array}{cc} (d_1 +_D d_3, a_1 \times_A a_3) & \text{if } d_1 <_D d_2 \\ (d_2 +_D d_3, a_2 \times_A a_3) & \text{if } d_2 <_D d_1 \\ (d_1 +_D d_3, (a_1 +_A a_2) \times_A a_3) & \text{if } d_1 = d_2 \end{array} \right\}.
\end{aligned}$$

  On the other hand,

$$\begin{aligned}
((d_1, a_1) \quad &\otimes \quad (d_3, a_3)) \oplus ((d_2, a_2) \otimes (d_3, a_3)) \\
&= (d_1 +_D d_3, a_1 \times_1 a_3) \oplus (d_2 +_D d_3, a_2 \times_A a_3) \\
&= \left\{ \begin{array}{cc} (d_1 +_D d_3, a_1 \times_A a_3) & \text{if } d_1 +_D d_3 <_D d_2 +_D d_3 \\ (d_2 +_D d_3, a_2 \times_A a_3) & \text{if } d_2 +_D d_3 <_D d_1 +_D d_3 \\ (d_1 +_D d_3, (a_1 \times_A a_3) +_A (a_2 \times a_3)) & \text{if } d_1 +_D d_3 = d_2 +_D d_3 \end{array} \right\}.
\end{aligned}$$

  - Assume that d_3 is multiplicatively regular. Then, by Property 4 (second item),

$$\begin{aligned}
d_1 <_D d_2 \quad &\Longleftrightarrow \quad d_1 +_D d_3 <_D d_2 +_D d_3, \\
d_2 <_D d_1 \quad &\Longleftrightarrow \quad d_2 +_D d_3 <_D d_1 +_D d_3, \text{ and} \\
d_1 = d_2 \quad &\Longleftrightarrow \quad d_1 +_D d_3 = d_2 +_D d_3.
\end{aligned}$$

Since $\times_A$ is distributive over $+_A$, we also have

$$(a_1 +_A a_2) \times_A a_3 = (a_1 \times_A a_3) +_A (a_2 \times a_3).$$

This proves that $((d_1, a_1) \oplus (d_2, a_2)) \otimes (d_3, a_3) = ((d_1, a_1) \otimes (d_3, a_3)) \oplus ((d_2, a_2) \otimes (d_3, a_3))$.

– Assume that d_3 is not multiplicatively regular. Since $(d_3, a_3) \in D\Delta A$, $a_3 = 0_A$.

Assume by symmetry that $d_1 \leq_D d_2$. Then, by Property 4 (first item), $d_1 +_D d_3 \leq_D d_2 +_D d_3$. Since $a_3 = 0_A$, we deduce that:

$$\begin{aligned} ((d_1, a_1) \oplus (d_2, a_2)) \otimes (d_3, a_3) &= (d_1 +_D d_3, 0_A) \\ (d_1, a_1) \otimes (d_3, a_3)) \oplus ((d_2, a_2) \otimes (d_3, a_3)) &= (d_1 +_D d_3, 0_A). \end{aligned}$$

Hence, we have proved that

$$((d_1, a_1) \oplus (d_2, a_2)) \otimes (d_3, a_3) = ((d_1, a_1) \otimes (d_3, a_3)) \oplus ((d_2, a_2) \otimes (d_3, a_3)),$$

thus showing that $\otimes$ is indeed distributive over $\oplus$. □

Theorem 4. *The semiring $\mathcal{D}_1 \Delta \mathcal{D}_2$ is also a commutative totally ordered idempotent dioid, and the associated order $\leq_{\mathcal{D}_1 \Delta \mathcal{D}_2}$ is the lexicographical order $\leq_{\mathcal{D}_1 lex \mathcal{D}_2}$.*

Proof. Since $\mathcal{D}_1$ and $\mathcal{D}_2$ are semirings, $\mathcal{D}_1 \Delta \mathcal{D}_2$ is a semiring. It is idempotent: indeed, let $(d_1, d_2) \in D_1 \times D_2$, then $(d_1, d_2) \oplus (d_1, d_2) \underset{\text{Because } d_1 = d_1}{=} (\min_1(d_1, d_1), \min_2(d_2, d_2)) = (d_1, d_2)$ by idempotence of $\mathcal{D}_1$ and $\mathcal{D}_2$.

It only remains to prove that the relation $\leq_{\mathcal{D}_1 \Delta \mathcal{D}_2}$ defined by:

$\forall ((a_1, a_2), (b_1, b_2)) \in (D_1 \times D_2)^2, (a_1, a_2) \leq_{\mathcal{D}_1 \Delta \mathcal{D}_2} (b_1, b_2) \iff \exists (c_1, c_2) \in D_1 \times D_2, (a_1, a_2) = (b_1, b_2) \oplus (c_1, c_2)$

is the same relation as $\leq_{\mathcal{D}_1 lex \mathcal{D}_2}$, defined as the lexicographical order with respects to $\leq_{\mathcal{D}_1}$ and $\leq_{\mathcal{D}_2}$.

Indeed, since $\leq_{\mathcal{D}_1 lex \mathcal{D}_2}$ is a total order (because $\leq_{\mathcal{D}_1}$ and $\leq_{\mathcal{D}_2}$ are), this will prove that $\mathcal{D}_1 \Delta \mathcal{D}_2$ is an idempotent totally-ordered dioid.

Let $((a_1, a_2), (b_1, b_2)) \in (D_1 \times D_2)^2$.

- Assume that $(a_1, a_2) \leq_{\mathcal{D}_1 \Delta \mathcal{D}_2} (b_1, b_2)$, ie. $\exists (c_1, c_2) \in D_1 \times D_2, (a_1, a_2) = (b_1, b_2) \oplus (c_1, c_2)$.
  - If $b_1 <_{\mathcal{D}_1} c_1$, then $(a_1, a_2) = (b_1, b_2)$. In particular, $(a_1, a_2) \leq_{\mathcal{D}_1 lex \mathcal{D}_2} (b_1, b_2)$.
  - If $c_1 <_{\mathcal{D}_1} b_1$, then $(a_1, a_2) = (c_1, c_2) <_{\mathcal{D}_1 lex \mathcal{D}_2} (b_1, b_2)$.
  - If $b_1 = c_1$, then $(a_1, a_2) = (b_1, \min_2(b_2, c_2)) \leq_{\mathcal{D}_1 lex \mathcal{D}_2} (b_1, b_2)$
- Assume that $(a_1, a_2) \leq_{\mathcal{D}_1 lex \mathcal{D}_2} (b_1, b_2)$.
  - If $a_1 <_{\mathcal{D}_1} b_1$, then $(a_1, a_2) = (b_1, b_2) \oplus (a_1, a_2)$, which proves that $(a_1, a_2) \leq_{\mathcal{D}_1 \Delta \mathcal{D}_2} (b_1, b_2)$.

– Else, $a_1 = b_1$ and $a_2 \leq_{\mathcal{D}_2} b_2$. Then, $(a_1, a_2) = (b_1, b_2) \oplus (a_1, a_2)$, which proves that $(a_1, a_2) \leq_{\mathcal{D}_1 \Delta \mathcal{D}_2} (b_1, b_2)$.

$\leq_{\mathcal{D}_1 \Delta \mathcal{D}_2}$ and $\leq_{\mathcal{D}_1 lex \mathcal{D}_2}$ are the same relation, which concludes the proof.

□

Lemma 3. *Let $(a_1, a_2) \in D_1 \Delta D_2$. Then, $(a_1, a_2) \in reg(\mathcal{D}_1 \Delta \mathcal{D}_2)$ if and only if $a_1 \in reg(\mathcal{D}_1)$ and $a_2 \in reg(\mathcal{D}_2)$.*

Proof.

- Assume that a_1 and a_2 are regular.

  Let $((b_1, b_2), (c_1, c_2)) \in (D_1 \times D_2)^2$ be such that

  $$(a_1, a_2) \otimes (b_1, b_2) = (a_1, a_2) \otimes (c_1, c_2).$$

  By definition of $\otimes$,

  $$(a_1 +_1 b_1, a_2 +_2 b_2) = (a_1 +_1 c_1, a_2 +_2 c_2).$$

  From the fact that a_1 and a_2 are regular, it follows that $b_1 = c_1$ and $b_2 = c_2$, ie. $(b_1, b_2) = (c_1, c_2)$. This proves that (a_1, a_2) is regular.

- Assume that (a_1, a_2) is regular. Let $(b_1, c_1) \in A^2$ be such that $a_1 +_1 b_1 = a_1 +_1 c_1$. Then, $(a_1, a_2) \otimes (b_1, 0_2) = (a_1 +_1 b_1, a_2) = (a_1 +_1 c_1, a_2) = (a_1, a_2) \otimes (c_1, 0_2)$. Since (a_1, a_2) is regular, we have that $(b_1, 0_2) = (c_1, 0_2)$. In particular, $b_1 = c_1$, which proves that a_1 is regular. A similar argument shows that a_2 is also regular.

□

Lemma 4. *For all $(d_1, d_2, a) \in D_1 \times D_2 \times A$, $(d_1 \Delta d_2) \Delta a = d_1 \Delta (d_2 \Delta a)$.*

Proof.

- Assume that $d_1 \notin reg(\mathcal{D}_1)$. Then,

  $$(d_1 \Delta d_2) \Delta a = (d_1, \infty_2) \Delta a = ((d_1, \infty_2), 0_A)$$

  by Lemma 3, and

  $$d_1 \Delta (d_2 \Delta a) = (d_1, (\infty_2, 0_A)),$$

  because $d_1 \notin reg(\mathcal{D}_1)$.

- Assume that $d_1 \in reg(\mathcal{D}_1)$ and $d_2 \notin reg(\mathcal{D}_2)$. Then,

  $$(d_1 \Delta d_2) \Delta a = (d_1, d_2) \Delta a = ((d_1, d_2), 0_A)$$

  by Lemma 3, and $d_1 \Delta (d_2 \Delta a) = (d_1, d_2 \Delta a) = (d_1, (d_2, 0_A))$.

- Assume that $d_1 \in reg(\mathcal{D}_1)$ and $d_2 \in reg(\mathcal{D}_2)$. Then,

  $$(d_1 \Delta d_2) \Delta a = (d_1, d_2) \Delta a = ((d_1, d_2), a)$$

  by Lemma 3, and $d_1 \Delta (d_2 \Delta a) = (d_1, d_2 \Delta a) = (d_1, (d_2, a))$. This shows that $(d_1 \Delta d_2) \Delta a = d_1 \Delta (d_2 \Delta a)$.

□

Theorem 5. *The Δ-product of semirings is associative. More precisely, up to identifying $(D_1 \times D_2) \times A$ with $D_1 \times (D_2 \times A)$, $(\mathcal{D}_1 \Delta \mathcal{D}_2)\Delta\mathcal{A} = \mathcal{D}_1(\Delta\mathcal{D}_2\Delta\mathcal{A})$.*

Proof. Notice that the product, zero element, and unity elements of the two structures $(\mathcal{D}_1\Delta\mathcal{D}_2)\Delta\mathcal{A}$ and $\mathcal{D}_1\Delta(\mathcal{D}_2\Delta\mathcal{A})$ correspond through the identification $(D_1 \times D_2) \times A = D_1 \times (D_2 \times A)$. We only have to verify that the two structures $(\mathcal{D}_1\Delta\mathcal{D}_2)\Delta\mathcal{A}$ and $\mathcal{D}_1\Delta(\mathcal{D}_2\Delta\mathcal{A})$ have the same sets and the same addition.

- To prove that $(D_1\Delta D_2)\Delta A = D_1\Delta(D_2\Delta A)$, it is sufficient to prove that $\forall (d_1, d_2, a) \in D_1 \times D_2 \times A$, $(d_1\Delta d_2)\Delta a = d_1\Delta(d_2\Delta a)$, which is true by Lemma 4.
- We now show that the two srtuctures have the same addition.
  Let $((d_1, d_2, a), (d'_1, d'_2, a')) \in (D_1 \times D_2 \times A)^2$. By Theorem 4,

$$
\begin{aligned}
(d_1,(d_2,a)) \underset{\mathcal{D}_1\Delta(\mathcal{D}_2\Delta\mathcal{A})}{\oplus} (d'_1,(d'_2,a')) &= \left\{ \begin{array}{cc} (d_1,(d_2,a)) & \text{if } d_1 <_{\mathcal{D}_1} d'_1 \\ (d'_1,(d'_2,a')) & \text{if } d'_1 <_{\mathcal{D}_1} d_1 \\ (d_1,(d_2,a) \underset{\mathcal{D}_2\Delta\mathcal{A}}{\oplus} (d'_2,a')) & \text{if } d_1 = d'_1 \end{array} \right\} \\
&= \left\{ \begin{array}{cc} (d_1,(d_2,a)) & \text{if } d_1 <_{\mathcal{D}_1} d'_1 \\ (d'_1,(d'_2,a')) & \text{if } d'_1 <_{\mathcal{D}_1} d_1 \\ (d_1,(d_2,a)) & \text{if } d_1 = d'_1 \text{ and } d_2 <_{\mathcal{D}_2} d'_2 \\ (d'_1,(d'_2,a')) & \text{if } d_1 = d'_1 \text{ and } d'_2 <_{\mathcal{D}_2} d_2 \\ (d_1,(d_2,a +_A a')) & \text{if } d_1 = d'_1 \text{ and } d_2 = d'_2 \end{array} \right\} \\
&= \left\{ \begin{array}{cc} (d_1,(d_2,a)) & \text{if } (d_1,d_2) <_{\mathcal{D}_1 lex \mathcal{D}_2} (d'_1,d'_2) \\ (d'_1,(d'_2,a')) & \text{if } (d'_1,d'_2) <_{\mathcal{D}_1 lex \mathcal{D}_2} (d_1,d_2) \\ (d_1,(d_2,a +_A a')) & \text{if } (d_1,d_2) = (d'_1,d'_2) \end{array} \right\}.
\end{aligned}
$$

We see that the two structures $(\mathcal{D}_1\Delta\mathcal{D}_2)\Delta\mathcal{A}$ and $\mathcal{D}_1\Delta(\mathcal{D}_2\Delta\mathcal{A})$ have the same addition, under the identification $D_1 \times (D_2 \times A) = (D_1 \times D_2) \times A$.

□

Theorem 6. *The function $w\Delta\mu$ that associates to every $\mathcal{F} \subseteq T^{S'}$ with $S' \subseteq S$:*

$$
\begin{aligned}
(w\Delta\mu)(\mathcal{F}) &:= \left\{ \begin{array}{cc} (w(\mathcal{F}), \mu(\mathrm{argmin}_w(\mathcal{F}))) & \textit{if } w(\mathcal{F}) \in reg(D) \\ (w(\mathcal{F}), 0_A) & \textit{if } w(\mathcal{F}) \notin reg(D) \end{array} \right. \\
&= (w(\mathcal{F}))\Delta(\mu(\mathrm{argmin}_w(\mathcal{F})))
\end{aligned}
$$

with $\mathrm{argmin}_w(\mathcal{F}) := \{f \in \mathcal{F}, w(\{f\}) = w(\mathcal{F})\}$ *is a $(\mathcal{D}\Delta\mathcal{A})$-measure.*

Proof.
- **zero axiom:** $(w\Delta\mu)(\emptyset) = (w(\emptyset), 0_A) = (\infty_D, 0_A)$ because w is a measure and ∞_D is not regular.

- **unit axiom:** $(w\Delta\mu)(T^{\emptyset}) = (w(T^{\emptyset}), \mu(T^{\emptyset})) = (0_D, 1_A)$ because w and μ are measures, and 0_D is regular.

- **additivity:** Let $\mathcal{F}_1, \mathcal{F}_2 \subseteq T^{S'}$ be disjoint and $S' \subseteq S$. Since w is a measure, we have

$$w(\mathcal{F}_1 \uplus \mathcal{F}_2) = \min(w(\mathcal{F}_1), w(\mathcal{F}_2)). \tag{A.1}$$

  Suppose first that $w(\mathcal{F}_1) = w(\mathcal{F}_2)$. By idempotence of min and (A.1), $w(\mathcal{F}_1 \uplus \mathcal{F}_2) = w(\mathcal{F}_1) = w(\mathcal{F}_2)$, and

$$\text{argmin}_w(\mathcal{F}_1 \uplus \mathcal{F}_2) = \text{argmin}_w(\mathcal{F}_1) \uplus \text{argmin}_w(\mathcal{F}_2).$$

  Since μ is a measure, it is not difficult to verify that

$$\begin{aligned}
(w\Delta\mu)(\mathcal{F}_1 \uplus \mathcal{F}_2) &= (w(\mathcal{F}_1 \uplus \mathcal{F}_2))\Delta(\mu(\text{argmin}_w(\mathcal{F}_1 \uplus \mathcal{F}_2))) \\
&= (\min(w(\mathcal{F}_1), w(\mathcal{F}_2)))\Delta(\mu(\text{argmin}_w(\mathcal{F}_1) \uplus \text{argmin}_w(\mathcal{F}_2))) \\
&= (\min(w(\mathcal{F}_1), w(\mathcal{F}_2)))\Delta(\mu(\text{argmin}_w(\mathcal{F}_1)) +_A \mu(\text{argmin}_w(\mathcal{F}_2))) \\
&= ((w(\mathcal{F}_1))\Delta(\mu(\text{argmin}_w(\mathcal{F}_1)))) \oplus ((w(\mathcal{F}_2))\Delta(\mu(\text{argmin}_w(\mathcal{F}_2)))) \\
&= (w\Delta\mu)(\mathcal{F}_1) \oplus (w\Delta\mu)(\mathcal{F}_2).
\end{aligned}$$

  Suppose now that $w(\mathcal{F}_1) <_D w(\mathcal{F}_2)$. By Property 1 and (A.1), we have $w(\mathcal{F}_1 \uplus \mathcal{F}_2) = w(\mathcal{F}_1)$, and thus $\text{argmin}_w(\mathcal{F}_1 \uplus \mathcal{F}_2) = \text{argmin}_w(\mathcal{F}_1)$, and:

$$\begin{aligned}
(w\Delta\mu)(\mathcal{F}_1 \uplus \mathcal{F}_2) &= (w(\mathcal{F}_1 \uplus \mathcal{F}_2))\Delta(\mu(\text{argmin}_w(\mathcal{F}_1 \uplus \mathcal{F}_2))) \\
&= (w(\mathcal{F}_1))\Delta(\mu(\text{argmin}_w(\mathcal{F}_1))) = (w\Delta\mu)(\mathcal{F}_1) \\
&= ((w(\mathcal{F}_1))\Delta\mu(\text{argmin}_w(\mathcal{F}_1))) \oplus ((w(\mathcal{F}_2))\Delta\mu(\text{argmin}_w(\mathcal{F}_2))) \\
&= (w\Delta\mu)(\mathcal{F}_1) \oplus (w\Delta\mu)(\mathcal{F}_2).
\end{aligned}$$

  The proof in the case when $w(\mathcal{F}_2) <_D w(\mathcal{F}_1)$ follows similarly.

- **elementary muliplicativity:** Let $f_1 \in T^{S_1}$ and $f_2 \in T^{S_2}$, for disjoint subsets S_1 and S_2 of S. By elementary multiplicativity of w,

$$w(\{f_1 \bowtie f_2\}) = w(\{f_1\}) +_D w(\{f_2\}).$$

  Also, $\text{argmin}_w(\{f_1\}) = \{f_1\}$ and $\text{argmin}_w(\{f_2\}) = \{f_2\}$.

  Suppose that $w(\{f_1\})$ and $w(\{f_2\})$ are multiplicatively regular. By Property 3, it follows that $w(\{f_1 \bowtie f_2\})$ is multiplicatively regular, and

$$\begin{aligned}
(w\Delta\mu)(\{f_1 \bowtie f_2\}) &= (w(\{f_1 \bowtie f_2\}), \mu(\{f_1 \bowtie f_2\})) \\
&= (w(\{f_1\}) +_D w(\{f_2\}), \mu(\{f_1\}) \times_A \mu(\{f_2\})) \\
&= (w(\{f_1\}), \mu(\{f_1\})) \otimes (w(\{f_2\}), \mu(\{f_2\})) \\
&= (w\Delta\mu)(\{f_1\}) \otimes (w\Delta\mu)(\{f_2\}),
\end{aligned}$$

  since $w(f_1) \in reg(\mathcal{D})$ and $w(f_2) \in reg(\mathcal{D})$.

Suppose that one of $w(\{f_1\})$ and $w(f_2)$ is not multiplicatively regular. Without loss of generality, assume that it is the former case. Then

$$w(\{f_1 \bowtie f_2\}) = w(\{f_1\}) +_D w(\{f_2\})$$

is not mulitplicatively multiplicatively regular by Property 3.

There exists $a_2 \in A$ such that $(w\Delta\mu)(\{f_2\}) = (w(\{f_2\}), a_2)$. Indeed, we take $a_2 := \mu(\{f_2\})$ if $w(\{f_2\}) \in reg(\mathcal{D})$ and $a_2 := 0_A$ if $w(\{f_2\}) \notin reg(\mathcal{D})$. Then

$$\begin{aligned}
(w\Delta\mu)(\{f_1 \bowtie f_2\}) &= (w(\{f_1 \bowtie f_2\}), 0_A) \\
&= (w(\{f_1\} +_D w(\{f_2\}), 0_A \times a_2) \\
&= (w(\{f_1\}), 0_A) \otimes (w(\{f_2\}), a_2) \\
&= (w\Delta\mu)(\{f_1\}) \otimes (w\Delta\mu)(\{f_2\}),
\end{aligned}$$

by definition of a_2 and $w\Delta\mu$, since $w(\{f_1\})$ is not regular.

□