Semirings, Generalized Effect Algebras, and Weighted Language Equivalence

Clovis Eberhart
under the supervision of Marcello Bonsangue,
Jan Rutten, and Alexandra Silva

September 9, 2014

Contents

1 Introduction

Weighted automata are automata where the transition and output functions, instead of being 0-1 functions (acceptance-or-rejection of a word and existence-or-absence of a transition), have "weights" associated to them. These weights can represent the cost to pay to do a certain action, the profit gained from doing a certain action, the duration of an action, the probability of that action happening, etc. Since they are so versatile, they are used in different fields of computer science, such as natural language processing, speech processing, image processing, quantitative modeling, etc. They are also interesting from another point of view, namely that they are mathematically more challenging

than classic automata. Basically, instead of associating 0 (reject) or 1 (accept) to each word, a state associates a quantity from a semiring $\mathbb{S}$ (which will be defined later) to each word, which can express several notions (cost, probability, etc.), based on $\mathbb{S}$'s structure. Therefore, there is a function from words to $\mathbb{S}$ associated to each state of a weighted automaton, called the *weighted language* that state recognizes. Two states are *weighted language equivalent* if they recognize the same weighted language.

In [1], the authors give several ways to decide weighted language equivalence for automata with weights on fields. However, these methods use linear algebra tools that do not exist in general for semirings, while many of the most important weighted automata have weights on semirings that are not fields. Furthermore, it is also well-known that weighted language equivalence is not decidable for all semirings. The goal of this work was to use the base results on fields and adapt them to a larger class of automata to push further the limits of decidability.

We assume the reader to be familiar with basic category theory. The introduction will contain the mathematical and categorical notions that the reader will need to know in order to understand this work, as well as a description of automata from a categorical point of view. In the first part, we first discuss a first approach that was tried and failed, then state and prove the main result, and explain its limitations. In the second part, we discuss some possible generalizations of the approach used to prove the main result, especially to structures where the addition is defined only partially. In the last part, we show some sound and complete proof systems that we get "for free" from the previous results.

1.1 Semirings and Power Series

It is natural in many fields of mathematics and computer science to have a structure with addition and multiplication that "behave well", such as the structure of ring, but where subtraction is not needed or cannot be defined. Such a structure is called a semiring and is defined as follows:

Definition 1 (Semiring)**.** *A semiring* $(\mathbb{S}, +, \cdot, 0, 1)$ *is a tuple of a set* $\mathbb{S}$*, together with elements* 0 *and* 1 *of* $\mathbb{S}$ *and binary operations* $+$ *and* $\cdot$ *such that:*

- $(\mathbb{S}, +, 0)$ *is a commutative monoid,*
- $(\mathbb{S}, \cdot, 1)$ *is a monoid,*
- 0 *is absorbing for* $\cdot$*,*
- $\cdot$ *distributes over* $+$*.*

Example 1. *Notable examples of semirings:*

- *rings and fields, such as* $\mathbb{Z}$*,* $\mathbb{Q}$*, and* $\mathbb{R}$*, are special cases of semirings,*
- *the Boolean semiring* $\{0, 1\}$*, with* $1 + 1 = 1$*,*
- $\mathbb{N}$ *with standard addition and multiplication,*
- *the tropical semiring* $(\mathbb{N} \cup \{\infty\}, \min, +, \infty, 0)$ *and arctic semiring* $(\mathbb{N} \cup \{-\infty\}, \max, +, -\infty, 0)$*,*

- *any Boolean algebra, in particular, for any set X, the powerset Boolean algebra $(2^X, \cup, \cap, \emptyset, X)$,*
- *the semiring of languages, with union as addition and concatenation as multiplication...*

Moreover, semirings have some nice properties that resemble those of rings. For example, matrices (and $n \times n$ matrices) of elements of a semiring are again semirings, functions (and finite support functions) from a set to a semiring again form a semiring (with pointwise addition and multiplication), and much of the theory of modules can be transferred to semimodules. Here, we will only be interested in basic semimodule theory.

Definition 2 (Semimodule)**.** *A semimodule $(M, +, \cdot, 0)$ on a semiring $\mathbb{S}$ is:*

- *a commutative monoid $(M, +, 0)$ equipped with*
- *a scalar multiplication $\cdot : \mathbb{S} \times M \to M$ such that:*
- *$s \cdot 0 = 0$, $s \cdot (x + y) = s \cdot x + s \cdot y$, $(s + t) \cdot x = s \cdot x + t \cdot x$, $(st) \cdot x = s \cdot (t \cdot x)$, and $1 \cdot x = x$.*

Semimodules are to semirings what vector spaces are to fields. The morphisms of semimodules are the linear functions between them, i.e., the functions f such that $f(\sum_{i \in I} \lambda_i \cdot x_i) = \sum_{i \in I} \lambda_i \cdot f(x_i)$, for all λ_i's in $\mathbb{S}$ and x_i's in M. Just like the free vector space on a field $\mathbb{F}$ associated to X is $\mathbb{F}(X)$ the set of all functions from X to $\mathbb{F}$ with finite support, the free semimodule on a semiring $\mathbb{S}$ associated to a set X is $\mathbb{S}(X)$, defined the same way. These functions will be written as formal sums $\sum_{i=1}^{n} \lambda_i \cdot \eta(x_i)$, where $\eta(x_i)$ maps x_i to 1 and all other x_j's to 0.

Later on, we will also be interested in the notion of (formal) *power series*. If Σ is an alphabet and $\mathbb{S}$ is a semiring, a power series r is simply a function from the set of words Σ^* to $\mathbb{S}$. The reason why we will be interested in them is because they can be seen as a semantics for some classes of automata. The set of all power series is denoted $\mathbb{S}\langle\langle\Sigma^*\rangle\rangle$. The value of r on a word w is noted (r, w), and the functions themselves are denoted as formal sums (hence their name): $r = \sum_{w \in \Sigma^*} (r, w)w$. The sum of power series is the pointwise sum, and the product of power series is the Cauchy product: $(r \cdot r', w) = \sum_{w_1 w_2 = w} (r, w_1) \cdot (r', w_2)$. The 0 power series is defined by $(0, w) = 0$, and the 1 power series is defined by $(1, \varepsilon) = 1$ and $(1, aw) = 0$. Power series again form a semiring with these definitions.

A power series r is *proper* if $(r, \varepsilon) = 0$, and *cycle-free* if there is a $k > 0$ such that $(r, \varepsilon)^k = 0$. If r is cycle-free, then the star of r is defined to be $r^* = \sum_{n \in \mathbb{N}} r^n$ (this sum is well defined because r is cycle-free). In [3], we can find the following theorem that will be used later:

Theorem 1. *If r is cycle-free, then the equation $y = ry + r'$ has a unique solution $\sigma = r^* r'$.*

1.2 Coinductive Reasoning

Here, we present the reader the basics of and some intuition about coinductive reasoning, which will be crucial to understand this work, and we illustrate them with the particular example of streams.

The most basic thing that can be said about coalgeraic reasoning is that it is reasoning about state systems (labelled transition systems, automata, etc.). A stream σ of elements in O is an infinite list $\sigma(0)$, $\sigma(1)$, $\sigma(2)$ of elements of O... However, that is only one possible view of streams. The coinductive view of a stream is the following: a stream is an output value $o(\sigma) = \sigma(0)$ in O and a derivative σ' such that $\sigma'(n) = \sigma(n+1)$. Such a view is very useful when the stream shows some regularity. For example, the stream σ such that $\sigma(n) = 1$ can be represented as $o(\sigma) = 1$ and $\sigma' = \sigma$. Coinductive reasoning is powerful for many reasons, such as allowing coinductive definitions, i.e., defining a stream by defining its output and derivative, coinductive calculus, coinductive proofs, the level of generality of some parts of the reasoning...

Coalgebras for a functor $F : \mathcal{C} \to \mathcal{C}$ is a pair of an object c of $\mathcal{C}$ and a morphism $\alpha : c \to Fc$. They will be denoted $\alpha : c \to Fc$ or $c \xrightarrow{\alpha} Fc$. Streams can be seen as coalgebras for the functor $F : \mathbf{Set} \to \mathbf{Set}$, $X \mapsto O \times X$. Indeed, the morphism $\alpha : X \to \mathbb{N} \times X$ is just a pair $\langle o, d \rangle$ of an output function $o : X \to O$ and a derivation function $d : X \to X$. The representation of a stream σ as a coalgebra $\langle X, \alpha \rangle$ is therefore: X as the set of all derivatives $\sigma^{(n)}$, $o : \sigma^{(n)} \mapsto \sigma^{(n)}(0) = \sigma(n)$ and $d : \sigma^{(n)} \mapsto \sigma^{(n+1)}$.

A morphism of coalgebras from $\alpha : c \to Fc$ to $\beta : d \to Fd$ is a morphism $f : c \to d$ such that $F(f) \circ \alpha = \beta \circ f$, i.e. that

$$\begin{array}{ccc} c & \xrightarrow{f} & d \\ {\scriptstyle\alpha}\downarrow & & \downarrow{\scriptstyle\beta} \\ Fc & \xrightarrow[Ff]{} & Fd \end{array}$$

commutes. In loose terms, it means that f is a morphism from c to d that is compatible with the coalgebra structures α and β. In the case of streams, a morphism from a stream σ to a stream τ is simply a function that preserves outputs and derivatives (for each $\sigma^{(n)}$), since

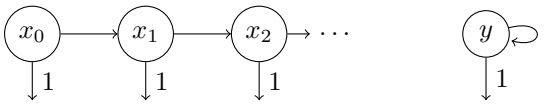
$$\begin{array}{ccc} X & \xrightarrow{f} & Y \\ {\scriptstyle\langle o, d \rangle}\downarrow & & \downarrow{\scriptstyle\langle o', d' \rangle} \\ O \times X & \xrightarrow[O \times f]{} & O \times Y \end{array}$$

commutes, i.e., $f \circ d = d' \circ f$ and $o = o'$. Coalgebras and their morphisms form a category.

Bisimulation is a notion that stems from concurrency theory and is about whether two concurrent systems behave similarly. If F is a **Set** functor, a bisimulation between two coalgebras $\alpha : X \to FX$ and $\beta : Y \to FY$ is a set $R \subseteq X \times Y$ such that there is another coalgebra $\gamma : R \to FR$ and coalgebra morphisms $f : \gamma \to \alpha$ and $g : \gamma \to \beta$, i.e. such that

$$\begin{array}{ccccc} X & \xleftarrow{f} & R & \xrightarrow{g} & Y \\ {\scriptstyle\alpha}\downarrow & & {\scriptstyle\gamma}\downarrow & & \downarrow{\scriptstyle\beta} \\ FX & \xleftarrow[Ff]{} & FR & \xrightarrow[Fg]{} & FY \end{array}$$

commutes. It basically means that R is a relation between the states of α and the states of β that is compatible with their coalgebra structure. A bisimulation on $\alpha : X \to PX$ (or bisimulation on X, if it is unambiguous) is a bisimulation between α and α. A *bisimulation equivalence* is a bisimulation that is also an equivalence relation. There is a reason why bisimulation does not make much sense on streams (which will be explained later on). However, coalgebras for the functor $X \to O \times X$ are more generally deterministic transition systems with output. Indeed, X is the set of states of the transition system, o an output function of each state, and d a transition function. In this case, it can be showed that any state x_i of the transition system on the left is bisimilar to y (i.e., there is a bisimulation R such that $(x, y) \in R$).



Sometimes, the category of coalgebras $\mathbf{Coal}(F)$ of a functor F have a terminal object, i.e., an object $*$ such that, for every other object c, there exists a unique morphism from c to $*$. In the case of coalgebras, these objects are called *final coalgebras*. All the functors we will study here possess final coalgebras. This is due to the fact that all the functors that we will study here are bounded, and any bounded functor has a final coalgebra [5]. Final coalgebras can also be seen as a semantics for the systems described by the functor F. If $\alpha : X \to FX$ is a coalgebra and f is the F-morphism to the final coalgebra, then two states x and y are said to be F-behaviorally equivalent if $f(x) = f(y)$. Final coalgebras also have other interesting properties:

Theorem 2. *If $\pi : P \to FP$ is a final coalgebra, then π is an isomorphism.*

Theorem 3 (Coinduction Proof Principle)**.** *If $\pi : P \to FP$ is a final coalgebra and R is a bisimulation on P, then R is included in the equality relation.*

In the case of the functor $F : X \mapsto O \times X$, the final coalgebra is the set of streams of elements in O. This is the reason why bisimulation does not make sense on streams: bisimulations on final coalgebras are always included in the equality relation. Therefore, since the equality relation is always a bisimulation, bisimulations and bisimilarity do not make much sense on streams, which are final. However, it also means that streams are a semantics for deterministic transition systems with output, and indeed, if a system has an output function o and a state x_0 that goes to x_1, then x_2, etc., then x_0 can be mapped to the stream $(o(x_0), o(x_1), o(x_2), \ldots)$, and that is the only way to map it to a stream that is compatible with the output and transition structure.

The coinduction proof principle is very interesting, as it states that, in order to prove that two prove that two states are equal, it is sufficient to prove that they are bisimilar, i.e., it is sufficient to find a bisimulation that relates them. In the case of streams, it means that, to prove that two streams are equal, it is sufficient to find a bisimulation that relates them, which is very useful when there is no known formula or no formula that is easy to manipulate to describe then nth term of the stream.

We will also use some very basic coinductive calculus on power series later, so let us first illustrate it on the simpler example of streams. Define X to be

the stream $(0, 1, 0, 0, 0, \ldots)$. All the reader will need to know is the very base of coinductive calculus, which comes from a very simple observation:

Theorem 4 (Fundamental Theorem of Stream Calculus)**.** *For any stream σ:*
$\sigma = \sigma(0) + X \times \sigma'$

This generalizes very naturally to power series. If we note $(\sigma)_a$ the derivative of σ in a (the power series that associates (σ, aw) to w) and X_a the power series defined by $(X_a, a) = 1$, $(X_a, w) = 0$ for $w \neq a$:

Theorem 5 (Fundamental Theorem of Power Series Calculus)**.** *For any power series σ:*
$\sigma = \sigma(\varepsilon) + \sum_{a \in \Sigma} X_a \times (\sigma)_a$

1.3 Automata from a Coinductive Point of View

Different classes of automata can be seen as coalgebras for different functors. The strength of reasoning on coalgebras is that, once a property is proved for all (or a significant class of) coalgebras, it can then be applied to specific functors, such as the different classes f automata and transition systems.

For example, classic deterministic (complete) automata with input on an alphabet Σ can be seen as coalgebras for the functor $F = \{0, 1\} \times -^{\Sigma}$. Indeed, such a coalgebra is a pair $\langle X, \alpha \rangle$ of a set (of states) X and a function $\alpha : X \to \{0, 1\} \times X^{\Sigma}$, i.e., $\alpha = \langle o, t \rangle$, where $o : X \to \{0, 1\}$ is the output function that tells whether a state is accepting or not (and can be interpreted as a subset F of final states of X), and $t : X \to X^{\Sigma}$ is the transition function that tells, given a state x and letter a, the state reached from x after reading a (note that the only difference between these automata and classic automata is the absence of an initial state, which is not very interesting from this point of view, since each state has its own semantics). A morphism f from an automaton $\langle Q, F, t \rangle$ to an automaton $\langle Q', F', t' \rangle$ is simply a function from Q to Q' that maps any state in F to a state in F' and that respects the transition structure, i.e., $f(t(x)(a)) = t'(f(x))(a)$.

The final coalgebra of this functor is the set of languages $\{0, 1\}^{\Sigma^*}$, equipped with functions:

$$o : \left| \begin{array}{ccc} \{0,1\}^{\Sigma^*} & \to & \{0,1\} \\ L & \mapsto & \begin{cases} 1 \text{ if } \varepsilon \in L \\ 0 \text{ otherwise} \end{cases} \end{array} \right. \quad \text{and } t : \left| \begin{array}{ccc} \{0,1\}^{\Sigma^*} & \to & (\{0,1\}^{\Sigma^*})^{\Sigma} \\ L & \mapsto & a \mapsto w \mapsto L(aw) \end{array} \right.$$

The semantics of a state of an automaton is the language of words that state recognizes (in the sense of classic automata theory), and two states are bisimilar if and only if they recognize the same language.

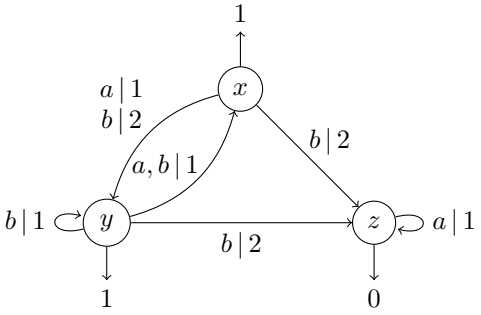
Non-deterministic (finitely-branching) automata can be seen as coalgebras for the functor $F = \{0, 1\} \times \mathcal{P}_\omega(-)^{\Sigma}$, where $\mathcal{P}_\omega$ is the finite powerset functor that associate to any set the set of its finite subsets. Indeed, here, a coalgebra $\langle X, \alpha \rangle$ is a pair of a set of states X together with two functions: an output function $o : X \to \{0, 1\}$ which, just like in the case of deterministic automata, is interpreted as a subset of finite states, and a transition function $t : X \to \mathcal{P}_\omega(X)^{\Sigma}$ which says, given a state x and a letter a, the set of states that can be reached from x by reading a. This functor also has a final coalgebra (since it is bounded),

but we will not make it explicit here, as it is not enlightening and will not be used again later.

The automata we are interested in here are not these automata, but *weighted automata*. It means that the output and transition functions have "weights" that can represent cost, probability, time, etc. To make the definition as broad as possible, we have to understand what is needed to define such an automaton. Let us take the example of non-deterministic automata, which can be seen as weighted automata with weights in $\{0, 1\}$, and a word is accepted if and only if its weight is 1. Computing the weight $f(x)(w)$ of a word w in a state x is easy: if the w is ε, then its weight is $f(x)(\varepsilon) = o(x)$, otherwise w is aw' for some a and w', and it is accepted if and only if w' is accepted from any of the states reached after reading a in x, i.e., $f(x)(aw') = \sum_{x' \in t(x)(a)} f(x')(w')$ (where $1 + 1 = 1$). Now, if instead of seeing $t(x)(a)$ as a subset of X, we see it as a function t that associates 1 to (x, a, x') if and only if x' is in $t(x)(a)$ and 0 otherwise, this expression can be seen as $f(x)(aw') = \sum_{x' \in X} t(x, a, x') \cdot f(x')(w')$. Therefore, if we take this formula and try to make it as general as possible, we see that we need the weights to have a structure of semiring. Weighted automata with weights in $\mathbb{S}$ are simply coalgebras for the functor $F = \mathbb{S} \times \mathbb{S}(-)^{\Sigma}$, where $\mathbb{S}(X)$ is the set of functions from X to $\mathbb{S}$ with finite support, i.e., $f(x) = 0_{\mathbb{S}}$, except for a finite number of x's. Though this functor possesses a final coalgebra, it is once again not enlightening, and will not be made explicit or used.

The states of a weighted automata do not recognize words. Instead, to each word they associate a weight, as described above: $f(x)(\varepsilon) = o(x)$ and $f(x)(aw') = \sum_{x' \in X} t(x, a, x') \cdot f(x')(w')$. This is the reason why they do not recognize a language, but what is called a *weighted language*. A weighted language is simply a function that associates to each word on an alphabet Σ a weight in $\mathbb{S}$, i.e., a weighted language is a power series in $\mathbb{S}\langle\langle\Sigma^*\rangle\rangle$. We say that two states are *weighted language equivalent* if they recognize the same weighted language.

Example 2. *Let us examine the following example:*



We want to show that x and y "behave the same way", i.e., recognize the same weighted language. There is a "semantic" morphism to the final coalgebra:

$$\begin{array}{ccc} \mathbb{S}(X) & \xrightarrow{\quad [\![-]\!] \quad} & \mathbb{S}^{\Sigma^*} \\ {\scriptstyle \langle \overline{o}, \overline{t} \rangle}\downarrow & & \downarrow{\scriptstyle \langle \varepsilon, d \rangle} \\ \mathbb{S} \times \mathbb{S}(X)^\Sigma & \xrightarrow[\mathbb{S} \times ([\![-]\!])^\Sigma]{} & \mathbb{S} \times (\mathbb{S}^{\Sigma^*})^\Sigma. \end{array}$$

To prove that x and y are equivalent is exactly to show that they have the same image through $[\![-]\!]$ (more precisely, that $\eta(x)$ and $\eta(y)$ have the same images, but we will not write the η's for conciseness). However, since $\mathbb{S}^{\Sigma^*}$ is final, it is sufficient to prove that $[\![x]\!]$ and $[\![y]\!]$ are bisimilar, since every bisimulation is included in the equality relation. Therefore, it is sufficient to find a bisimulation that relates $[\![x]\!]$ and $[\![y]\!]$. Here, we will use the bisimulation R that relates $\lambda [\![x]\!] + \mu [\![y]\!] + \nu [\![z]\!]$ and $\lambda' [\![x]\!] + \mu' [\![y]\!] + \nu [\![z]\!]$ if $\lambda + \mu = \lambda' + \mu'$ (note that it does relate $[\![x]\!]$ and $[\![y]\!]$). Unfolding the definition of bisimulation for this particular functor gives us that R is a bisimulation if for all pairs (x, y) in R, $\varepsilon(x) = \varepsilon(y)$, and for all a in Σ, $d(x)(a)\, R\, d(y)(a)$. But (dropping the $[\![-]\!]$ for conciseness) $\varepsilon(\lambda x + \mu y + \nu z) = \lambda o(x) + \mu o(y) + \nu o(z) = \lambda + \mu = \lambda' + \mu'$, $d(\lambda x + \mu y + \nu z)(a) = \mu x + \lambda y + \nu z\, R\, \mu' x + \lambda' y + \nu z$, and $d(\lambda x + \mu y + \nu z)(b) = \mu x + (2\lambda + \mu) y + 2(\lambda + \mu) z\, R\, \mu' x + (2\lambda' + \mu') y + 2(\lambda' + \mu') z$, so R is indeed a bisimulation, and x and y are weighted language equivalent.

The weighted language recognized by a state can be expressed in terms of final coalgebras. Take a weighted automaton with weights on a semiring $\mathbb{S}$, i.e., a coalgebra $\langle o, t \rangle : X \to \mathbb{S} \times \mathbb{S}(X)^\Sigma$. The process of *linearization* transforms it into a deterministic *linear weighted automaton* which is an automataon that lives in the category **SMod** of semimodules: its set of states is now $\mathbb{S}(X)$ (which is indeed a semimodule), $\eta(x)$ outputs $o(x)$ and makes a transition by reading a to $\sum_{j=1}^{n} t(x, a, x_j) \cdot x_j$. Since the output and transition functions are semimodule morphisms, the output and transition function of any state is defined by linearity. Note that its set of states is not finite, but still of finite dimension if X is finite. This automaton is now a coalgebra for the functor $F = \mathbb{S} \times (-)^\Sigma$ (since it is a morphism $\mathbb{S}(X) \to \mathbb{S} \times \mathbb{S}(X)^\Sigma$), so it has a unique morphism to the set of power series $\mathbb{S}\langle\langle \Sigma^* \rangle\rangle$, which maps each state to the weighted language it recognizes.

2 From Fields to Semirings

The main goal of this work is to extend a decidability result that is known for automata with weights on a field to a larger class of automata. The result in itself is that weighted language equivalence is decidable (for weighted automata with weights on a field).

2.1 A First Attempt: Computing the Star

The first approach that we tried was to use the notion of *star*, which is very close to the notion of inverse in linear algebra. Assume that $\langle o, t \rangle : X \to \mathbb{S} \times \mathbb{S}(X)$ is an automaton with a finite number of states n, and take its linearization $\langle \overline{o}, \overline{t} \rangle : \mathbb{S}(X) \to \mathbb{S} \times \mathbb{S}(X)$. If we note $[\![x]\!]_{\mathbb{S}(X)}$ the weighted language recognized by state x, we have that

$$\begin{array}{ccc} \mathbb{S}(X) & \xrightarrow{[\![-]\!]_{\mathbb{S}(X)}} & \mathbb{S}^{\Sigma^*} \\ {\scriptstyle \langle \overline{o}, \overline{t} \rangle}\downarrow & & \downarrow{\scriptstyle \langle \varepsilon, d \rangle} \\ \mathbb{S} \times \mathbb{S}(X)^\Sigma & \xrightarrow[\mathbb{S} \times ([\![-]\!]_{\mathbb{S}(X)})^\Sigma]{} & \mathbb{S} \times (\mathbb{S}^{\Sigma^*})^\Sigma \end{array}$$

commutes. Call $\sigma(i)$ the power series $[\![\eta(x_i)]\!]_{\mathbb{S}(X)}$. Then, by commutativity of the diagram above, we have $\sigma_i(\varepsilon) = \overline{o}(\eta_X(x_i)) = o(x_i)$ (where $\eta_X(x)$ is the function that maps x to 1 and the rest to 0) and $(\sigma_i)_a = ([\![\overline{t}(\eta_X(x_i))]\!]_{\mathbb{S}(X)})^\Sigma(a) = [\![t(x_i)(a)]\!]_{\mathbb{S}(X)} = \left[\!\!\left[\sum_{j=1}^n t_{i,a,j} \cdot \eta_X(x_j)\right]\!\!\right]_{\mathbb{S}(X)} = \sum_{j=1}^n t_{i,a,j} \cdot \sigma_j$. We thus have n pairs of power series equations relating the σ_i's and their derivatives, as well as the output and transition from the automaton. By using the fundamental theorem of power series calculus, we have that $\sigma = o(x_i) + \sum_{a \in \Sigma} X_a \cdot \sum_{j=1}^n t_{i,a,j} \cdot \sigma_j$. By putting all n of these equations together, we find the following matrix equation: $\mathfrak{S} = O + (\sum_{a \in \Sigma} X_a T_a)\mathfrak{S}$, where $\mathfrak{S}$ is the vertical vector of σ_i's, O the vertical vector of $o(x_i)$'s, and T_a is the matrix of $t_{i,a,j}$'s. While viewing them as matrices of power series is fine, it is also possible to see them as power series of matrices (while the reader may think the multiplication would not behave the same way with both views, it actually does).

Seeing them as power series, and noticing the fact that $\sum_{a \in \Sigma} X_a T_a$ is proper (and therefore cycle-free), we deduce that there is a unique solution to this equation, which is $\mathfrak{S} = (\sum_{a \in \Sigma} X_a T_a)^* O$, and now seeing them as matrices, all that is left is to compute the star of a matrix (defined by $M^* = \sum_{n \in \mathbb{N}} M^n$, when the sum is well-defined).

However, it is not clear under which conditions the star can easily be computed. It can obviously be computed when the semiring is a field (because M^* is $(I - M)^{-1}$). However, in a more general case, it is not obvious what can be done to compute the star. A possible step towards computing it would be to find a global formula for M^n, which is already difficult enough. Indeed, finding a global formula for M^n when M is a matrix whose elements are on a field already uses mildly advanced mathematics, such as the Cayley–Hamilton theorem, which already requires the ring to be commutative, and makes use of notions that become much trickier once they are extended from fields to rings. We were therefore unable to advance any further down this path, and had to try another approach.

2.2 Another Approach: Embeddings

Retrospectively, the answer we came up with was very simple. The idea is that if a semiring can be embedded into a field (or any semiring where we know weighted language equivalence to be decidable), then its weighted language equivalence should be the same as the field's.

Theorem 6. *Let $\mathbb{S}$ be a semiring, $\mathbb{T}$ another semiring, on which weighted language equivalence is decidable, and $i : \mathbb{S} \to \mathbb{T}$ be a semiring monomorphism. Weighted language equivalence is decidable on $\mathbb{S}$.*

Proof. Let $\langle o, t \rangle : X \to \mathbb{S} \times \mathbb{S}(X)^\Sigma$ (in **Set**) be a weighted automaton on $\mathbb{S}$. Its lineraization is $\langle \overline{o}, \overline{t} \rangle : \mathbb{S}(X) \to \mathbb{S} \times \mathbb{S}(X)^\Sigma$ (in **SMod**) where $\langle \overline{o}, \overline{t} \rangle$ is the

linearization (in the sense of semimodules) of $\langle o, t\rangle$, i.e., the only semimodule morphism that makes the following diagram commute in **Set** (this morphism exists as a universal property of $\mathbb{S}(X)$):

$$\begin{array}{ccc} X & \xrightarrow{\langle o,t\rangle} & \mathbb{S}\times\mathbb{S}(X)^\Sigma \\ {\scriptstyle \eta^{\mathbb{S}}_X}\downarrow & \nearrow_{\langle \overline{o},\overline{t}\rangle} & \\ \mathbb{S}(X), & & \end{array}$$

where $\eta^{\mathbb{S}}_X(x)$ is the function that maps x to $1_{\mathbb{S}}$ and all other elements of X to $0_{\mathbb{S}}$.

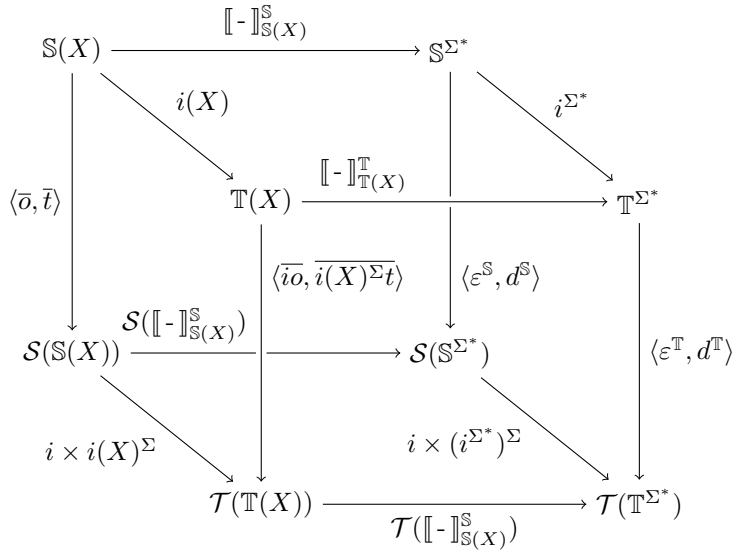
Similarly, one can define $\langle \overline{io}, \overline{i(X)^\Sigma t}\rangle : \mathbb{T}(X) \to \mathbb{T}\times\mathbb{T}(X)^\Sigma$ (in **SMod**), where $\langle \overline{io}, \overline{i(X)^\Sigma t}\rangle$ is the linearization of $\langle io, i(X)^\Sigma t\rangle$, i.e. the only semimodule morphism that makes the following diagram commute in **Set**:

$$\begin{array}{ccccc} X & \xrightarrow{\langle o,t\rangle} & \mathbb{S}\times\mathbb{S}(X)^\Sigma & \xrightarrow{i\times i(X)^\Sigma} & \mathbb{T}\times\mathbb{T}(X)^\Sigma \\ {\scriptstyle \eta^{\mathbb{T}}_X}\downarrow & & & \nearrow_{\langle \overline{io}, \overline{i(X)^\Sigma t}\rangle} & \\ \mathbb{T}(X). & & & & \end{array}$$

Now, we have two automata $\langle \overline{o}, \overline{t}\rangle : \mathbb{S}(X) \to \mathbb{S}\times\mathbb{S}(X)^\Sigma$ and $\langle \overline{io}, \overline{i(X)^\Sigma t}\rangle : \mathbb{T}(X) \to \mathbb{T}\times\mathbb{T}(X)^\Sigma$ that live in **SMod**. Therefore, we can define their semantics:

$$\begin{array}{ccc} \mathbb{S}(X) & \xrightarrow{[\![\text{-}]\!]^{\mathbb{S}}_{\mathbb{S}(X)}} & \mathbb{S}^{\Sigma^*} \\ {\scriptstyle \langle \overline{o},\overline{t}\rangle}\downarrow & & \downarrow{\scriptstyle \langle \varepsilon^{\mathbb{S}}, d^{\mathbb{S}}\rangle} \\ \mathcal{S}(S(X)) & \xrightarrow[\mathcal{S}([\![\text{-}]\!]^{\mathbb{S}}_{\mathbb{S}(X)})]{} & \mathcal{S}(\mathbb{S}^{\Sigma^*}), \end{array} \qquad \begin{array}{ccc} \mathbb{T}(X) & \xrightarrow{[\![\text{-}]\!]^{\mathbb{T}}_{\mathbb{T}(X)}} & \mathbb{T}^{\Sigma^*} \\ {\scriptstyle \langle \overline{io}, \overline{i(X)^\Sigma t}\rangle}\downarrow & & \downarrow{\scriptstyle \langle \varepsilon^{\mathbb{T}}, d^{\mathbb{T}}\rangle} \\ \mathcal{T}(\mathbb{T}(X)) & \xrightarrow[\mathcal{T}([\![\text{-}]\!]^{\mathbb{T}}_{\mathbb{T}(X)})]{} & \mathcal{T}(\mathbb{T}^{\Sigma^*}), \end{array}$$

where $\mathcal{S} : X \mapsto \mathbb{S}\times X^\Sigma$ and $\mathcal{T} : X \mapsto \mathbb{T}\times X^\Sigma$. Moreover, since $i : \mathbb{S} \to \mathbb{T}$, we can draw the following cube in **Set**:



and show that every face commutes. We already know that the front and back faces commute. The left and right faces commute by trivial calculations. The only interesting face is the top face, which is shown to commute by coinduction (by finding a suitable bisimulation). The bottom face commutes as a direct consequence of the top face commuting.

Now, we prove that, two states x and y in $\mathbb{S}(X)$ are bisimilar if and only if $i(X)(x)$ and $i(X)(y)$ are, by showing both implications:

- if $x \sim y$, then $[\![x]\!]^{\mathbb{S}}_{\mathbb{S}(X)} = [\![y]\!]^{\mathbb{S}}_{\mathbb{S}(X)}$, so $i^{\Sigma^*}([\![x]\!]^{\mathbb{S}}_{\mathbb{S}(X)}) = i^{\Sigma^*}([\![y]\!]^{\mathbb{S}}_{\mathbb{S}(X)})$, which brings by commutation of the top face of the cube: $[\![i(X)(x)]\!]^{\mathbb{F}}_{\mathbb{F}(X)} = [\![i(X)(y)]\!]^{\mathbb{F}}_{\mathbb{F}(X)}$, therefore $i(X)(x) \sim i(X)(y)$;
- if $i(X)(x) \sim i(X)(y)$, then $[\![i(X)(x)]\!]^{\mathbb{F}}_{\mathbb{F}(X)} = [\![i(X)(y)]\!]^{\mathbb{F}}_{\mathbb{F}(X)}$, so by commutation of the top face of the cube, we have $i^{\Sigma^*}([\![x]\!]^{\mathbb{S}}_{\mathbb{S}(X)}) = i^{\Sigma^*}([\![y]\!]^{\mathbb{S}}_{\mathbb{S}(X)})$, and since i^{Σ^*} is mono (because i is), it brings $[\![x]\!]^{\mathbb{S}}_{\mathbb{S}(X)} = [\![y]\!]^{\mathbb{S}}_{\mathbb{S}(X)}$, so $s \sim y$.

Since weighted language equivalence is decidable for automata with weights on fields, it is also decidable for automata with weights on $\mathbb{S}$. □

Remark 1. *For a field $\mathbb{F}$, weighted language equivalence does not depend on whether $\mathbb{F}$ is seen as a semiring or as a field.*

Corollary 1. *If $\mathbb{S}$ is a semiring that is embeddable into a field and equality is decidable on that field, since weighted language equivalence is decidable for automata with weights on a field, weighted language equivalence is decidable for automata with weights on $\mathbb{S}$.*

Corollary 2. *Language equivalence is decidable for any integral domain (which can be embedded into its field of fractions – if equality is decidable on this field) and $\mathbb{N}$ (which can be embedded into $\mathbb{Q}$).*

2.3 Embedding a Semiring into a Field

Corollary 1 tells us that weighted language equivalence is decidable if the semiring $\mathbb{S}$ is embeddable into a field. Therefore, a logical question to ask is which semirings can be embedded into a field, and that is what we discuss here.

First of all, there are some obvious necessary conditions for a semiring $\mathbb{S}$ to be embeddable into a field:

- the addition must be cancellative: if $a + b = a + c$, then $b = c$,
- the multiplication must be commutative,
- there must be no non-zero divisors of 0.

Example 3. *Here are some examples that do not enter the scope of corollary 1:*

- *the tropical semiring* $(\mathbb{N} \cup \{\infty\}, \min, +, \infty, 0)$ *cannot be embedded into a field because its addition (*min*) is not cancellative, and therefore no additive inverse can be defined: assume* $i : \mathbb{N} \cup \{\infty\} \to \mathbb{F}$ *(where addition in* $\mathbb{F}$ *is noted* min *and multiplication* $+$*) is a semiring morphism, then any* $i(x)$ *has an additive inverse* $\overline{i(x)}$ *in* $\mathbb{F}$*, and in* $\mathbb{F}$ *we have that:*

$$\min(i(x), \min(i(x), \overline{i(x)})) = \min(i(x), i(\infty)) = i(\min(x, \infty)) = i(x),$$

and by associativity of min*,*

$$\begin{aligned}(i(x), \min(i(x), \overline{i(x)})) &= \min(\min(i(x), i(x)), \overline{i(x)}) \\ &= \min(i(\min(x, x)), \overline{i(x)}) = \min(i(x), \overline{i(x)}) \\ &= i(\infty),\end{aligned}$$

so x *and* ∞ *are mapped by* i *to the same element in* $\mathbb{F}$*, so* i *cannot be monic;*

- *no non-commutative ring* $\mathbb{R}$ *can be embedded into a field because, if* $i : \mathbb{R} \to \mathbb{F}$ *is a semiring monomorphism, then for all* x *and* y *in* $\mathbb{R}$*, we have* $i(xy) = i(x)i(y) = i(y)i(x) = i(yx)$*, so* $xy = yx$*;*
- *the rings* $\mathbb{Z}/n\mathbb{Z}$ *for* n *not prime cannot be embedded into a field because there are non-zero divisors of* 0*, but weighted language equivalence is decidable because* $\mathbb{Z}/n\mathbb{Z}$ *is finite.*

Now, take a semiring $\mathbb{S}$ that meets all these conditions. To see whether $\mathbb{S}$ can be embedded into a field, we first embed it into its *ring of differences*, which is the "smallest" ring that contains it, then try to see whether that ring can be embedded into a field. By "smallest", we mean that this ring will only contain the elements of $\mathbb{S}$, their additive inverses, and the sums of these elements. This ring exists because $\mathbb{S}$'s addition is cancellative. Note that this is a general method to know whether a semiring is embeddable into a field, because every semiring embeddable into a field is embeddable into a ring (e.g., the field itself), and that ring must at least contain the ring of differences.

The rings that can be embedded into fields are exactly the integral domains. A ring is an integral domain if it is commutative and the product of two non-zero elements is non-zero. A ring that is not integral can obviously not be embedded

into a field, and any integral domain can be embedded into its field of fractions. Therefore, the only thing left to do is compute the ring of differences and see whether it is an integral domain or not.

The embedding of semirings into rings is discussed in [6], and we only write about it here to make the construction explicit. Let $S^* = \{(a, b) \mid a, b \in \mathbb{S}\}$, and define the addition of (a, b) and (a', b') to be $(a + a', b + b')$, their multiplication $(aa' + bb', ab' + a'b)$ (this makes sense because (a, b) is an element that represents $a - b$), and define $(a, b) \sim (a', b')$ if $a + b' = a' + b$. Now, we take $\overline{S}$ the ring of differences to be the equivalence classes of $\sim$ (note that $\sim$ is indeed an equivalence relation because the addition on $\mathbb{S}$ is cancellative). Proving that $\overline{S}$ is a ring is not difficult. There is an obvious semiring monomorphism which embeds S into $\overline{S}$, namely the one that maps s to $(s, 0)$.

Now, $\mathbb{S}$ is embeddable into a field if and only if $\overline{S}$ is an integral domain. Its multiplication is commutative, but it is not clear when the product of two non-zero elements is guaranteed to be non-zero. Indeed, let us multiply two elements and assume the result is 0:

$$\begin{aligned}(a, b)(a', b') \sim (0, 0) &\Leftrightarrow aa' + bb' = ab' + a'b \\ &\Leftrightarrow aa' + bb' + ab + a'b' = ab' + a'b + ab + a'b' \\ &\Leftrightarrow (a + b')(a' + b) = (a + a')(b + b')\end{aligned}$$

(the second line uses the cancellativity of addition). The most interesting lines are the first and third ones, but even using these, it is not clear under which conditions on $\mathbb{S}$ these equations imply that $a = b$ or $a' = b'$.

However, it is a fact that these three conditions are not sufficient. Here is the example of a semiring that has the three properties but is not embeddable into a field. Take $\mathbb{S} = \mathbb{N}[X, Y]$ the polynomials in two variables X and Y with coefficients in $\mathbb{N}$, and quotient it by the equation $X^2 = Y^2$. The addition is cancellative (this is very easy to prove, using the fact that any element of $\mathbb{S}$ can be written $P + QY$, where P and Q are polynomials in X), multiplication is obviously commutative, and there are no non-zero divisors of zero (because the equation $X^2 = Y^2$ may not decrease the degree of a polynomial). However, its ring of differences is not an integral domain. Indeed, it contains additive inverses, so the polynomials $X - Y$ and $X + Y$ exist, and $(X + Y)(X - Y) = X^2 - Y^2 = 0$.

Therefore, even though we have an interesting example of a semiring that can be embedded into a field ($\mathbb{N}$), it is not clear which semirings can actually be embedded into a field, and therefore when exactly corollary 1 can be used.

3 Generalizations

In this section, we use the same kind of technique to show decidability of weighted language equivalence for another class of automata, for which addition is defined only partially. The main example that we have in mind when we talk about automata with a partial sum is the example of probabilistic automata. Indeed, the sum of probabilities can never be greater than 1, so it is only defined partially. We want to use a model that captures at least probabilistic automata. We also show that the same technique cannot be applied for some another functor, which contains that finite powerset construction.

3.1 Powerset

We hoped that the same kind of technique would work on automata of the form $X \to \mathbb{N} \times \mathcal{P}_\omega(\mathbb{N}(X))^\Sigma$, which represent a more general form of non-determinism. The idea is that $\mathbb{N} \times \mathcal{P}_\omega(\mathbb{N}(X))^\Sigma$ can be structured as a semimodule, and if there is a vector space in which it can be embedded, then weighted language equivalence would also be decidable for automata of this form, by drawing a cube similar to that of the proof of theorem 6 and proving that weighted language equivalence is the same as in that vector space. However, it proves impossible to actually embed $\mathcal{P}_\omega(\mathbb{N}(X))$ into a vector space.

Indeed, the structure of semimodule of $\mathcal{P}_\omega(M)$ for any semimodule (or actually even any set) M is the following: addition is union (with the empty set as neutral element), and multiplication is iterated addition (which gives that $0 \cdot S = \emptyset$ and $n \cdot S = S$ otherwise). The problem here is that addition is idempotent. Assume that there is indeed an embedding i of $\mathcal{P}_\omega(M)$ into some vector space. Then the image $i(S)$ of each element S of $\mathcal{P}_\omega(M)$ has an additive inverse $-i(S)$, and we have: $(i(S) + i(S)) - i(S) = i(S) - i(S) = \emptyset$ (because $i(S) + i(S) = i(S \cup S) = i(S)$), but by associativity of addition in vector spaces, we also have: $(i(S) + i(S)) - i(S) = i(S) + (i(S) - i(S)) = i(S) + \emptyset = i(S)$, which contradicts injectivity of i when M is not empty.

There is another way to structure $\mathcal{P}_\omega(\mathbb{N}(X))$ into an $\mathbb{N}$-semimodule, that works only for $\mathbb{N}(X)$, and not for any semimodule M. The structure is the following: $S + S' = \{s + s' \mid s \in S,\, s' \in S'\}$ (with $\{0\}$ as neural element), and multiplication is iterated addition. However, even with this other structure, embedding $\mathcal{P}_\omega(\mathbb{N}(X))$ into a vector space is impossible. The problem here is that $\emptyset$ is absorbing for addition. Assume there is an embedding i of $\mathcal{P}_\omega(\mathbb{N}(X))$ into a vector space. Then $i(\emptyset)$ has an additive inverse, and $(-i(\emptyset) + i(\emptyset)) + i(S) = i(\{0\}) + i(S) = i(S)$, but by associativity $(-i(\emptyset) + i(\emptyset)) + i(S) = -i(\emptyset) + (i(\emptyset) + i(S)) = -i(\emptyset) + i(\emptyset) = i(\{0\})$, which contradicts injectivity of i when X is not empty.

3.2 Partial Semirings

Definition 3 (Partial Commutative Monoid)**.** *A partial commutative monoid [4] (PCM) is a tuple $(M, +, 0)$ such that:*

- $+ : M \times M \to M$ *is a partially defined operation such that:*
- *if $x + y$ is defined (hereafter noted $x \perp y$), then $y \perp x$, and $x + y = y + x$,*
- *if $x \perp y$ and $x + y \perp z$, then $y \perp z$, $x \perp y + z$, and $(x + y) + z = x + (y + z)$,*
- $x \perp 0$ *and $x + 0 = x$.*

A notable example of PCM, and which is the one that corresponds to probabilistic automata, is that of the interval $[0, 1]$. The addition $x + y$ is defined if and only if $x + y \leq 1$, which is why it is only partial.

A morphism of PCMs $f : (M, +, 0) \to (M', +', 0')$ is a function $f : M \to M'$ such that $f(0) = 0'$ and, if $x \perp y$, then $f(x) \perp f(y)$ and $f(x + y) = f(x) + f(y)$. PCMs and morphisms of PCMs form a category. This category can be equipped with a symmetric monoidal structure $(\otimes, I)$ where $M \otimes N$ is the PCM of finite formal sums of pairs $m \otimes n$ (with the obvious addition and 0), quotiented by

the congruence generated by the equations $(m + m') \otimes n = m \otimes n + m' \otimes n$, $m \otimes (n + n') = m \otimes n + m \otimes n'$, and $0 \otimes n = 0 = m \otimes 0$ (when the addition is defined), and I is the one-element PCM. Therefore, we can study the monoids in this category.

A *partial semiring* is a monoid in the category of PCMs, just like semirings are monoids in the category of commutative monoids. A partial semiring is therefore a PCM $(M, +, 0)$ with additional structure $M \otimes M \to M \leftarrow I$. Unfolding the definitions gives that a partial semiring is a PCM with an element 1 and a map $\cdot : M \times M \to M$ such that $(M, \cdot, 1)$ is a monoid, $m \cdot (n + n') = m \cdot n + m \cdot n'$, $(m + m') \cdot n = m \cdot n + m' \cdot n$, and $0 \cdot n = 0 = m \cdot 0$, when the addition is defined.

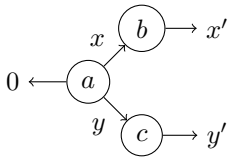
For a partial semiring $\mathbb{S}$, we can consider the category of its *partial modules*, which are $\mathbb{S}$-actions in the category of PCMs. By unfolding the definitions, we get that a partial module is a PCM $(M, +, 0)$ equipped with a map called scalar multiplication $\cdot : \mathbb{S} \times M \to M$ that verifies $s \cdot (x + x') = s \cdot x + s \cdot x'$, $(s + s') \cdot x = s \cdot x + s' \cdot x$, $0 \cdot x = 0 = s \cdot 0$, $s \cdot (s' \cdot x) = (ss') \cdot x$, and $1 \cdot x = x$, when addition is defined. A map f of partial modules from $(M, +, \cdot, 0)$ to $(M', +', \cdot', 0')$ is a PCM map from $(M, +, 0)$ to $(M', +', 0')$ that respects scalar multiplication: $f(s \cdot x) = s \cdot' f(x)$.

Example 4. *The sub-probability distribution functor $\mathcal{D}_{\leq}$ defined by $\mathcal{D}_{\leq}(X) = \{f : X \to [0, 1] \mid f$ has finite support, $\sum_{x \in X} f(x) \leq 1\}$ is a partial module for the partial semiring $[0, 1]$, with obvious the 0, addition, and scalar multiplication.*

3.3 Theorem for Partial Semirings

For the proof to go through, we have to ask one more property of our partial semiring $\mathbb{S}$. The property is the following: if $x \perp y$, then $x \cdot x' \perp y \cdot y'$. Let us explain why we need this property to hold. The proof is based on that of theorem 6, where the morphism $X \to \mathbb{S} \times \mathbb{S}(X)^{\Sigma}$ is factored through $\mathbb{S}(X)$. Here, we take an automaton $\langle o, t \rangle : X \to \mathbb{S} \times \mathbb{S}(X)^{\Sigma}$, where $\mathbb{S}(X)$ denotes $\{f : X \to \mathbb{S} \mid f$ has finite support, $\perp_{x \in X} f(x)\}$, which is the generalization of the sub-distribution functor. To factor o through $\mathbb{S}(X)$ is to find a partial module morphism $\overline{o} : \mathbb{S}(X) \to \mathbb{S}$ such that $\overline{o}\eta = o$, where $\eta(x)$ is the function that maps x to 1 and the rest to 0 (which is indeed in $\mathbb{S}(X)$). Therefore, $\overline{o}\left(\sum_{i=1}^{n} s_i \cdot \eta(x_i)\right) = \sum_{i=1}^{n} s_i \cdot o(x_i)$, and the property we asked is necessary and sufficient for this sum to be always defined, no matter the values $o(x_i)$.

Moreover, this property is actually very sound in the world of automata. Indeed, one of the strong points of automata is that they are easy to write. For example, to write a probabilistic automaton, everything one needs to check is that the sum of the weights of transitions from one state is not bigger than 1. However, for partial semirings that do not have this property, it is possible to write things that look like automata but actually are not. Indeed, assume that $x \perp y$, but $xx' \not\perp yy'$. Then



looks like a weighted automata (on an alphabet with a single letter a) but is not. Indeed, checking each state reveals nothing wrong, but there is no weight

associated to the word a^2 in x: it should be $xx' + yy'$, but this expression does not exist! Therefore, being an automaton is not a property that can be checked locally anymore. This is the reason why the property we ask for is not only useful in the proof, but also something we actually want to ask of the partial semiring.

Theorem 7. *Let $\mathbb{S}$ be a partial semiring that can be embedded into another partial semiring $\mathbb{T}$, and assume that both $\mathbb{S}$ and $\mathbb{T}$ are such that $x \perp y$ implies $xx' \perp yy'$, and weighted language equivalence is decidable for weighted automata with weights on $\mathbb{T}$. Then weighted language equivalence is decidable for weighted automata with weights on $\mathbb{S}$.*

Proof. As shown above, since the property holds for $\mathbb{S}$, o can be factorized through $\mathbb{S}(X)$. A similar computation shows that t can also be factorized through $\mathbb{S}(X)$ (using the fact that $\perp_{x' \in X} t(x)(a)(x')$ by definition of $\mathbb{S}(X)$). Similarly, io and $i(X)^\Sigma t$ can be factorized through $\mathbb{T}(X)$. Therefore, we can draw the exact same cube as in the proof of theorem 6, except that $\mathbb{S}(X)$, $\mathbb{T}(X)$, and $i(X)$ stand for slightly different constructions. However, they are not very different, and the same proof as in the case of semirings just goes through. □

4 Proof Systems

In this section, we show sound and complete proof systems for deciding weighted language equivalence for weighted automata with weights in a semiring $\mathbb{S}$ or a generalized effect monoid E embeddable into a field. Notable examples include weighted automata with weights on $\mathbb{N}$ and probabilistic automata. Even though sound and complete proof systems already exist for them, they are quite complex, and the ones we show here are much simpler. Actually, the proof systems we show here are the proof systems for weighted language equivalence on fields. However, since the proof of theorem 6 showed that weighted language equivalence on $\mathbb{S}$ (or E) is the same as the one on fields, these proof systems directly apply to these cases as well.

In [2], it is showed that the following proof system is sound and complete for weighted automata with weights on a Noetherian semiring $\mathbb{S}$ (and therefore in particular for fields). The expressions of the proof system are given by the grammar:

$$E ::= x \mid E + E \mid \underline{r} \mid a.(r \cdot E) \mid \mu x.E^g$$
$$E^g ::= E^g + E^g \mid \underline{r} \mid a.(r \cdot E) \mid \mu x.E^g$$

Where x ranges over a set of variables, r over $\mathbb{S}$, and a over Σ. Each expression denotes a weighted language: $E_1 + E_2$ is the union of E_1 and E_2, $\underline{r}$ associates r to ε and 0 to all other words, $a.(r \cdot E)$ associates 0 to ε and $r \cdot r_w$ to aw, where r_w is the weight of w in E, and μ is a fixpoint operator.

The calculus is equipped with the following rules:

$$\begin{array}{ccc}
E_1 \equiv E_2[E_1/x] \Rightarrow E_1 \equiv \mu x.E_2 & \underline{r} + \underline{s} \equiv \underline{r+s} & \underline{0} + E \equiv E \\
(E_1 + E_2) + E_3 \equiv E_1 + (E_2 + E_3) & E_1 + E_2 \equiv E_2 + E_1 & a.(0 \cdot E) = \underline{0} \\
a.(r \cdot E) + a.(s \cdot E) \equiv a.(r + s \cdot E) & \mu x.E \equiv E[\mu x.E/x] & a.(r \cdot \underline{0}) \equiv \underline{0} \\
a.(r \cdot (E_1 + E_2)) \equiv a.(r \cdot E_1) + a.(r \cdot E_2) & a.(r \cdot \underline{s}) \equiv a.(1 \cdot \underline{rs}) & a.(r \cdot \underline{0}) \equiv \underline{0} \\
a.(r \cdot b.(s \cdot E)) \equiv a.(rs \cdot b.(1 \cdot E)) & &
\end{array}$$

together with α-equivalence and the replacement rule $E_1 \equiv E_2 \Rightarrow E[E_1/x] \equiv E[E_2/x]$. This calculus is sound and complete for weighted language equivalence. Since these rules never introduce additive or multiplicative inverses, they can be performed inside a semiring $\mathbb{S}$ if $\mathbb{S}$ is embeddable into a field. However, it is not necessarily the case that they can be performed inside a partial semiring $\mathbb{S}$, since expressions of the form $r + s$ appear in the calculus, and while they exist in the field that contains $\mathbb{S}$, they may not exist in $\mathbb{S}$ itself. However, this is not a problem, as all the calculations may be done inside the field anyway.

5 Conclusion

The initial goal of this work was to extend decidability of weighted language equivalence for weighted automata with weights on a field to a larger class of automata. This was achieved through a fairly simple technique for weighted automata with weights on a semiring, and the same technique was adapted to other kinds of automata, with more or less success. The biggest achievement was probably the development of the theory of partial semirings (based on that of *effect algebras* [4], which it is inspired from), and the adaptation of the technique developed for semirings to partial semirings.

The direct application is that weighted language equivalence is decidable for two classes of automata: weighted automata with weights on $\mathbb{N}$, for which weighted language equivalence was already known to be decidable, but the proof was very complex, while this one (including the proof that weighted language equivalence is decidable on fields) is much more elementary; and probabilistic automata, for which weighted language equivalence was also known to be decidable. A more interesting result is that we have simple sound and complete proof systems for these two classes of automata, when the only known proof systems for them until now were much more complex.

There are several possible ways to continue this work. One possible way to continue it would be to understand the limits of this technique, e.g., by giving a necessary and sufficient condition for a semiring $\mathbb{S}$ to be embeddable into a field, or by investigating when a partial semiring has the property $x \perp y \Rightarrow xx' \perp yy'$. Another possibility, though maybe not as interesting, is to find other interesting examples that fit the scope of the theorems proven in this work. It is very probable that models of computation stemming from quantum theory actually fit this framework. Of course, there is still a large gap between the weighted automata for which we know weighted language equivalence and those for which we know it is undecidable, so this is also a possible direction to explore.

References

[1] Filippo Bonchi, Marcello Bonsangue, Michele Boreale, Jan Rutten, and Alexandra Silva. A coalgebraic perspective on linear weighted automata. *Information and Computation*, 211:77–105, 2012.

[2] Marcello M. Bonsangue, Stefan Milius, and Alexandra Silva. Sound and complete axiomatisations of coalgebraic language equivalence.

[3] Manfred Droste, Werner Kuich, and Heiko Vogler. *Handbook of Weighted Automata*. Springer.

[4] Bart Jacobs and Jorik Mandemaker. Relating Operator Spaces via Adjunctions. *Computing Research Repository*, abs/1201.1272, 2012.

[5] J. J. M. M. Rutten. Universal coalgebra: a theory of systems. *Theoretical Computer Science*, 249:3–80, 2000.

[6] Han-Lee Sang. Extending semiring homomorphisms to ring homomorphisms. *Communications of the Korean Mathematical Society*, 13(2):243–249, 1998.

A Proofs of commutations

Let us show that the left face

$$\begin{array}{ccc} \mathbb{S}(X) & \xrightarrow{\quad i(X) \quad} & \mathbb{T}(X) \\ {\scriptstyle \langle \overline{o}, \overline{t} \rangle}\downarrow & & \downarrow{\scriptstyle \langle \overline{io}, \overline{i(X)^\Sigma t} \rangle} \\ \mathcal{S}(\mathbb{S}(X)) & \xrightarrow[i \times i(X)^\Sigma]{} & \mathcal{T}(\mathbb{T}(X)) \end{array}$$

commutes, i.e., that $i \circ \overline{o} = \overline{io} \circ i(X)$ and $i(X)^\Sigma \circ \overline{t} = \overline{i(X)^\Sigma t} \circ i(X)$. Take f in $\mathbb{S}(X)$, f is by definition a finite sum $f = \sum_{i=0}^{n} s_i \cdot \eta_X^{\mathbb{S}}(x_i)$, where x_i are elements of X and s_i elements of $\mathbb{S}$. Now, we have:

$$\begin{aligned} (i \circ \overline{o}) \left(\sum_{i=0}^{n} s_i \cdot \eta_X^{\mathbb{S}}(x_i) \right) &= i \left(\sum_{i=0}^{n} s_i \cdot \overline{o}(\eta_X^{\mathbb{S}}(x_i)) \right) \\ &= \sum_{i=0}^{n} i(s_i) \cdot i(\overline{o}(\eta_X^{\mathbb{S}}(x_i))) \\ &= \sum_{i=0}^{n} i(s_i) \cdot i(o(x_i)) \\ (\overline{io} \circ i(X)) \left(\sum_{i=0}^{n} s_i \cdot \eta_X^{\mathbb{S}}(x_i) \right) &= \overline{io} \left(\sum_{i=0}^{n} i(s_i) \cdot \eta_X^{\mathbb{T}}(x_i) \right) \\ &= \sum_{i=0}^{n} i(s_i) \cdot \overline{io}(\eta_X^{\mathbb{T}}(x_i)) \\ &= \sum_{i=0}^{n} i(s_i) \cdot i(o(x_i)) \end{aligned}$$

and

$$\begin{aligned} (i(X)^\Sigma \circ \overline{t}) \left(\sum_{i=0}^{n} s_i \cdot \eta_X^{\mathbb{S}}(x_i) \right) &= i(X)^\Sigma \left(\sum_{i=0}^{n} s_i \cdot \overline{t}(\eta_X^{\mathbb{S}}(x_i)) \right) \\ &= \sum_{i=0}^{n} i(s_i) \cdot i(X)^\Sigma(\overline{t}(\eta_X^{\mathbb{S}}(x_i))) \\ &= \sum_{i=0}^{n} i(s_i) \cdot i(X)^\Sigma(t(x_i)) \\ (\overline{i(X)^\Sigma t} \circ i(X)) \left(\sum_{i=0}^{n} s_i \cdot \eta_X^{\mathbb{S}}(x_i) \right) &= \overline{i(X)^\Sigma t} \left(\sum_{i=0}^{n} i(s_i) \cdot \eta_X^{\mathbb{S}}(x_i) \right) \\ &= \sum_{i=0}^{n} i(s_i) \cdot \overline{i(X)^\Sigma t}(\eta_X^{\mathbb{S}}(x_i)) \\ &= \sum_{i=0}^{n} i(s_i) \cdot i(X)^\Sigma(t(x_i)). \end{aligned}$$

Now, let us show that the right face

$$\begin{array}{ccc} \mathbb{S}^{\Sigma^*} & \xrightarrow{\quad i^{\Sigma^*} \quad} & \mathbb{T}^{\Sigma^*} \\ {\scriptstyle \langle \varepsilon^{\mathbb{S}}, d^{\mathbb{S}} \rangle}\downarrow & & \downarrow{\scriptstyle \langle \varepsilon^{\mathbb{T}}, d^{\mathbb{T}} \rangle} \\ \mathcal{S}(\mathbb{S}^{\Sigma^*}) & \xrightarrow[i \times (i^{\Sigma^*})^{\Sigma}]{} & \mathcal{T}(\mathbb{T}^{\Sigma^*}) \end{array}$$

commutes. Take f in $\mathbb{S}^{\Sigma^*}$, we have:

$$\varepsilon^{\mathbb{T}}(i^{\Sigma^*}(f)) = i^{\Sigma^*}(f)(\varepsilon) = i(f(\varepsilon))$$
$$i(\varepsilon^{\mathbb{S}}(f)) = i(f(\varepsilon))$$

and

$$d^{\mathbb{T}}(i^{\Sigma^*}(f))(a) = w \mapsto i^{\Sigma^*}(f)(aw) = w \mapsto i(f(aw))$$
$$(i^{\Sigma^*})^{\Sigma}(d^{\mathbb{S}}(f))(a) = i^{\Sigma^*}(d^{\mathbb{S}}(f)(a)) = i^{\Sigma^*}(w \mapsto f(aw)) = w \mapsto i(f(aw))$$

Now, for the less trivial computation, let us show that the top face

$$\begin{array}{ccc} \mathbb{S}(X) & \xrightarrow{\quad [\![\text{-}]\!]^{\mathbb{S}}_{\mathbb{S}(X)} \quad} & \mathbb{S}^{\Sigma^*} \\ {\scriptstyle i(X)}\downarrow & & \downarrow{\scriptstyle i^{\Sigma^*}} \\ \mathbb{T}(X) & \xrightarrow[[\![\text{-}]\!]^{\mathbb{T}}_{\mathbb{T}(X)}]{} & \mathbb{T}^{\Sigma^*} \end{array}$$

commutes. Take f in $\mathbb{S}(X)$, we will show that $i^{\Sigma^*}([\![f]\!]^{\mathbb{S}}_{\mathbb{S}(X)}) = [\![i(X)(f)]\!]^{\mathbb{F}}_{\mathbb{F}(X)}$ by coinduction. Let $R \subseteq (\mathbb{T}^{\Sigma^*})^2$ be the set of pairs $\langle i^{\Sigma^*}([\![f]\!]^{\mathbb{S}}_{\mathbb{S}(X)}), [\![i(X)(f)]\!]^{\mathbb{F}}_{\mathbb{F}(X)} \rangle$, for all f in $\mathbb{S}^{\Sigma^*}$. We have:

$$\begin{aligned} i^{\Sigma^*}\left(\left[\!\!\left[\sum_{i=0}^{n} s_i \cdot \eta^{\mathbb{S}}(x_i)\right]\!\!\right]^{\mathbb{S}}_{\mathbb{S}(X)}\right) &= i^{\Sigma^*}\left(\sum_{i=0}^{n} s_i \cdot [\![\eta^{\mathbb{S}}(x_i)]\!]^{\mathbb{S}}_{\mathbb{S}(X)}\right) \\ &= \sum_{i=0}^{n} i(s_i) \cdot i^{\Sigma^*}([\![\eta^{\mathbb{S}}(x_i)]\!]^{\mathbb{S}}_{\mathbb{S}(X)}) \\ \left[\!\!\left[i(X)\left(\sum_{i=0}^{n} s_i \cdot \eta^{\mathbb{S}}(x_i)\right)\right]\!\!\right]^{\mathbb{T}}_{\mathbb{T}(X)} &= \left[\!\!\left[\sum_{i=0}^{n} i(s_i) \cdot \eta^{\mathbb{T}}(x_i)\right]\!\!\right]^{\mathbb{T}}_{\mathbb{T}(X)} \\ &= \sum_{i=0}^{n} i(s_i) \cdot [\![\eta^{\mathbb{T}}(x_i)]\!]^{\mathbb{T}}_{\mathbb{T}(X)}. \end{aligned}$$

Now, recall that, from the commutation of the front and back faces, we have:

$$\begin{cases} \begin{cases} [\![\eta^{\mathbb{S}}(x)]\!]^{\mathbb{S}}_{\mathbb{S}(X)}(\varepsilon) = o(x) \\ [\![\eta^{\mathbb{S}}(x)]\!]^{\mathbb{S}}_{\mathbb{S}(X)}(aw) = [\![t(x)(a)]\!]^{\mathbb{S}}_{\mathbb{S}(X)} \end{cases} \\ \begin{cases} [\![\eta^{\mathbb{T}}(x)]\!]^{\mathbb{F}}_{\mathbb{F}(X)}(\varepsilon) = i(o(x)) \\ [\![\eta^{\mathbb{T}}(x)]\!]^{\mathbb{F}}_{\mathbb{F}(X)}(aw) = [\![i(X)(t(x)(a))]\!]^{\mathbb{F}}_{\mathbb{F}(X)} \end{cases} \end{cases}$$

Using these identities, we obtain that:

$$\begin{aligned}
\varepsilon^{\mathbb{T}}(i^{\Sigma^*}(\llbracket f \rrbracket^{\mathbb{S}}_{\mathbb{S}(X)})) &= \varepsilon^{\mathbb{T}}\left(\sum_{i=0}^{n} i(s_i) \cdot i^{\Sigma^*}(\llbracket \eta^{\mathbb{S}}(x_i) \rrbracket^{\mathbb{S}}_{\mathbb{S}(X)})\right) \\
&= \sum_{i=0}^{n} i(s_i) \cdot i^{\Sigma^*}(\llbracket \eta^{\mathbb{S}}(x_i) \rrbracket^{\mathbb{S}}_{\mathbb{S}(X)})(\varepsilon) \\
&= \sum_{i=0}^{n} i(s_i) \cdot i(\llbracket \eta^{S}(x_i) \rrbracket^{\mathbb{S}}_{\mathbb{S}(X)}(\varepsilon)) \\
&= \sum_{i=0}^{n} i(s_i) \cdot i(o(x)) \\
\varepsilon^{\mathbb{T}}\left(\llbracket i(X)(f) \rrbracket^{\mathbb{T}}_{\mathbb{T}(X)}\right) &= \varepsilon^{\mathbb{T}}\left(\sum_{i=0}^{n} i(s_i) \cdot \llbracket \eta^{\mathbb{T}}(x_i) \rrbracket^{\mathbb{F}}_{\mathbb{F}(X)}\right) \\
&= \sum_{i=0}^{n} i(s_i) \cdot \llbracket \eta^{\mathbb{T}}(x_i) \rrbracket^{\mathbb{F}}_{\mathbb{F}(X)}(\varepsilon) \\
&= \sum_{i=0}^{n} i(s_i) \cdot i(o(x_i)),
\end{aligned}$$

i.e., that $\varepsilon^{\mathbb{T}}(i^{\Sigma^*}(\llbracket f \rrbracket^{\mathbb{S}}_{\mathbb{S}(X)})) = \varepsilon^{\mathbb{T}}\left(\llbracket i(X)\,(f) \rrbracket^{\mathbb{T}}_{\mathbb{T}(X)}\right)$, and that

$$\begin{aligned}
d^{\mathbb{T}}(i^{\Sigma^*}(\llbracket f \rrbracket^{\mathbb{S}}_{\mathbb{S}(X)}))(a) &= d^{\mathbb{T}}\left(\sum_{i=0}^{n} i(s_i) \cdot i^{\Sigma^*}(\llbracket \eta^{\mathbb{S}}(x_i) \rrbracket^{\mathbb{S}}_{\mathbb{S}(X)})\right)(a) \\
&= w \mapsto \sum_{i=0}^{n} i(s_i) \cdot i^{\Sigma^*}(\llbracket \eta^{\mathbb{S}}(x_i) \rrbracket^{\mathbb{S}}_{\mathbb{S}(X)})(aw) \\
&= w \mapsto \sum_{i=0}^{n} i(s_i) \cdot i(\llbracket \eta^{\mathbb{S}}(x_i) \rrbracket^{\mathbb{S}}_{\mathbb{S}(X)}\,(aw)) \\
&= w \mapsto \sum_{i=0}^{n} i(s_i) \cdot i(\llbracket t(x)(a) \rrbracket^{\mathbb{S}}_{\mathbb{S}(X)}) \\
&= i^{\Sigma^*}\left(\left[\!\!\left[\sum_{i=0}^{n} s_i \cdot t(x_i)(a)\right]\!\!\right]^{\mathbb{S}}_{\mathbb{S}(X)}\right) \\
d^{\mathbb{T}}\left(\llbracket i(X)\,(f) \rrbracket^{\mathbb{T}}_{\mathbb{T}(X)}\right)(a) &= d^{\mathbb{T}}\left(\sum_{i=0}^{n} i(s_i) \cdot \llbracket \eta^{\mathbb{T}}(x_i) \rrbracket^{\mathbb{F}}_{\mathbb{F}(X)}\right)(a) \\
&= \sum_{i=0}^{n} i(s_i) \cdot d^{\mathbb{T}}\left(\llbracket \eta^{\mathbb{T}}(x_i) \rrbracket^{\mathbb{F}}_{\mathbb{F}(X)}\right)(a) \\
&= w \mapsto \sum_{i=0}^{n} i(s_i) \cdot d^{\mathbb{T}}\left(\llbracket \eta^{\mathbb{T}}(x_i) \rrbracket^{\mathbb{F}}_{\mathbb{F}(X)}\right)(a)(w) \\
&= w \mapsto \sum_{i=0}^{n} i(s_i) \cdot \llbracket \eta^{\mathbb{T}}(x_i) \rrbracket^{\mathbb{F}}_{\mathbb{F}(X)}\,(aw) \\
&= w \mapsto \sum_{i=0}^{n} i(s_i) \cdot \llbracket i(X)(t(x_i)(a)) \rrbracket^{\mathbb{F}}_{\mathbb{F}(X)}\,(w) \\
&= \sum_{i=0}^{n} i(s_i) \cdot \llbracket i(X)(t(x_i)(a)) \rrbracket^{\mathbb{F}}_{\mathbb{F}(X)} \\
&= \left[\!\!\left[\sum_{i=0}^{n} i(s_i) \cdot i(X)(t(x_i)(a))\right]\!\!\right]^{\mathbb{F}}_{\mathbb{F}(X)} \\
&= \left[\!\!\left[i(X)\left(\sum_{i=0}^{n} s_i \cdot t(x_i)(a)\right)\right]\!\!\right]^{\mathbb{F}}_{\mathbb{F}(X)},
\end{aligned}$$

i.e., that $d^{\mathbb{T}}(i^{\Sigma^*}(\llbracket f \rrbracket^{\mathbb{S}}_{\mathbb{S}(X)}))(a)\ R\ d^{\mathbb{T}}\left(\llbracket i(X)\,(f) \rrbracket^{\mathbb{T}}_{\mathbb{T}(X)}\right)(a)$. Therefore R is a bisimulation, so the diagram commutes by coindution.

Commutation of the bottom face comes directly from commutation of the top face:

$$\begin{aligned}
(i \times (i^{\Sigma^*})^{\Sigma}) \circ \langle \mathrm{id}_{\mathbb{S}(X)}, (\llbracket \text{-} \rrbracket^{\mathbb{S}}_{\mathbb{S}(X)})^{\Sigma} \rangle &= \langle i, (i^{\Sigma^*})^{\Sigma} \circ (\llbracket \text{-} \rrbracket^{\mathbb{S}}_{\mathbb{S}(X)})^{\Sigma} \rangle \\
&= \langle i, (i^{\Sigma^*} \circ \llbracket \text{-} \rrbracket^{\mathbb{S}}_{\mathbb{S}(X)})^{\Sigma} \rangle \\
&= \langle i, (\llbracket i(X)(\text{-}) \rrbracket^{\mathbb{F}}_{\mathbb{F}(X)})^{\Sigma} \rangle \\
&= \langle \mathrm{id}_{\mathbb{T}(X)}, (\llbracket \text{-} \rrbracket^{\mathbb{F}}_{\mathbb{F}(X)})^{\Sigma} \rangle \circ \langle i, i(X)^{\Sigma} \rangle
\end{aligned}$$