Symbolic Weighted Language Models and Quantitative Parsing over Infinite Alphabets

Florent Jacquemard
Inria & CNAM, Paris, France

Abstract

We propose a framework for weighted parsing over infinite alphabets. It is based on language models called Symbolic Weighted Automata (swA) at the joint between Symbolic Automata (sA) and Weighted Automata (wA), as well as Transducers (swT) and Visibly Pushdown (sw-VPA) variants. Like sA, swA deal with large or infinite input alphabets, and like wA, they output a weight value in a semiring domain. The transitions of swA are labeled by functions from an infinite alphabet into the weight domain. This is unlike sA whose transitions are guarded by boolean predicates overs symbols in an infinite alphabet and also unlike wA whose transitions are labeled by constant weight values, and who deal only with finite automata. We present some properties of swA, swT and sw-VPA models, that we use to define and solve a variant of parsing over infinite alphabets. We also briefly describe the application that motivated the introduction of these models: a parse-based approach to automated music transcription.

2012 ACM Subject Classification Theory of computation → Quantitative automata

Keywords and phrases Weighted Automata, Symbolic Automata, Visibly Pushdown, Parsing

Digital Object Identifier 10.4230/LIPIcs...

Funding *Florent Jacquemard*: Inria AEx Codex, ANR Collabscore, EU H2020 Polifonia

Acknowledgements I want to thank ...

indep. counters Def. Prop. Ex...

1 Introduction

Parsing is the problem of structuring a linear representation on input (a finite word), according to a language model. Most of the context-free parsing approaches [15] assume a finite and reasonably small input alphabet. Such a restriction makes perfect sense in the context of NLP tasks such as constituency parsing, or of programming languages compilers or interpreters. Considering large or infinite alphabets can however be of practical interest, for instance, when dealing with large characters encodings such as UTF-16, *e.g.* for vulnerability detection in Web-applications [8], for the analysis (*e.g.* validation or filtering) of data streams or serialization of structured documents (with textual or numerical attributes) [26], or for processing timed execution traces [3].

The latter case is related to a study that motivated the present work: automated music transcription. Most representations of music are essentially linear. This is true for audio files, but also for widely used symbolic representations like MIDI. Such representations ignore the hierarchical structures that frame the conception of music, at least in the western area. These structures, on the other hand, are present, either explicitly or implicitly, in music notation [14]: music scores are partitioned in measures, measures in beats, and beats can be further recursively divided. It follows that music events do not occur at arbitrary timestamps, but respect a discrete partitioning of the timeline incurred by these recursive divisions. The *transcription problem* takes as input a linear representation (audio or MIDI) and aims at re-constructing these structures by mapping input events to this hierarchical rhythmic space. It can therefore be stated as a parsing problem [12], over an infinite alphabet of timed events.

Various extensions of language models for handling infinite alphabets have been studied.

$$
\begin{tikzcd}[column sep=small]
 & \begin{array}{c}\mathsf{FA} : \Sigma_{\mathsf{fin}}^* \to \mathbb{B} \\ q \xrightarrow{a} q' \quad a \in \Sigma_{\mathsf{fin}}\end{array} \arrow[dl] \arrow[dr] & \\
\begin{array}{c}\mathsf{WA} : \Sigma_{\mathsf{fin}}^* \to \mathbb{S} \\ q \xrightarrow{a,w} q' \quad a \in \Sigma_{\mathsf{fin}}, w \in \mathbb{S}\end{array} \arrow[dr] & & \begin{array}{c}\mathsf{SA} : \Sigma_{\mathsf{inf}}^* \to \mathbb{B} \\ q \xrightarrow{\phi} q' \quad \phi : \Sigma_{\mathsf{inf}} \to \mathbb{B}\end{array} \arrow[dl] \\
 & \begin{array}{c}\mathsf{SWA} : \Sigma_{\mathsf{inf}}^* \to \mathbb{S} \\ q \xrightarrow{\phi} q' \quad \phi : \Sigma_{\mathsf{inf}} \to \mathbb{S}\end{array} &
\end{tikzcd}
$$

Figure 1 Classes of Symbolic/Weighted Automata. Σ_{fin} is a finite alphabet, Σ_{inf} is a countable alphabet, $\mathbb{B}$ is the Boolean algebra, $\mathbb{S}$ is a commutative semiring, $q \xrightarrow{\cdots} q'$ is a transition between states q and q'.

register: skip refs and details, add Mikolaj recent

For instance, some automata with memory extensions allow restricted storage and comparison of input symbols, (see [26] for a survey), with pebbles for marking positions [25], registers [18], or the possibility to compute on subsequences with the same attribute values [2]. The automata at the core of model checkers compute on input symbols represented by large bitvectors [27] (sets of assignments of Boolean variables) and in practice, each transition accepts a set of such symbols (instead of an individual symbol), represented by Boolean formula or Binary Decision Diagrams. Following a similar idea, in symbolic automata (sA) [7, 8], the transitions are guarded by predicates over infinite alphabet domains. With appropriate closure conditions on the sets of such predicates, all the good properties enjoyed by automata over finite alphabets are preserved.

WARNING: [23] much is more general that weighted parsing

Other extensions of language models help in dealing with non-determinism, by the computation of weight values. With an ambiguous grammar, there may exist several derivations (*abstract syntax trees* – AST) yielding one input word. The association of one weight value to each AST permits to select a best one (or n bests). This is roughly the principle of *weighted parsing* approaches [13, 24, 23]. In *weighted language models*, like *e.g.* probabilistic context-free grammars and weighted automata (wA) [11], a weight value is associated to each transition rule, and the rule's weights can be combined with an associative product operator $\otimes$ into the weight of an AST. A second operator $\oplus$, associative and commutative, is moreover used to resolve the ambiguity raised by the existence of several (in general exponentially many) AST associated to a given input word. Typically, $\oplus$ will select the best of two weight values. The weight domain, equipped with these two operators shall be, at minima, a *semiring* where $\oplus$ can be extended to infinite sums, such as the Viterbi semiring and the tropical min-plus algebra

In this paper, we present a uniform framework for weighted parsing over infinite input alphabets. It is based on *symbolic weighted* finite states language models (swM), generalizing the Boolean guards of sA into functions into an arbitrary semiring, and generalizing also wA, by handling infinite alphabets, see Figure 1.

In short, a transition rule $q \xrightarrow{\phi} q'$ from state q to q' of a swA, is labeled by a function ϕ associating to every input symbol a a weight value $\phi(a)$ in a semiring domain. The framework relies on several language models: finite automata called symbolic-weighted (swA), transducers (swT), and pushdown automata with a visibly restriction [1] (sw-VPA). The latter model of automata operates sequentially on *nested words* [1], structured with markup symbols (parentheses) and able to describe linearizations of trees. In the context of parsing, they can represent (weighted) AST of CF grammars. More precisely, a sw-VPA A associates a weight value $A(t)$ to a given nested word t, which is the linearization of an AST. On the other hand, a swT can define a distance $T(s,t)$ between finite words s and t over infinite alphabets. Then, the *SW-parsing* problem aims at finding t minimizing $T(s,t) \otimes A(t)$ (*wrt* the ranking defined by $\oplus$), given an input word s. The latter product is called the distance between s and A in [21]. Like weighted-parsing methods [13, 24, 23], our approach proceeds in two steps, based on properties of the swM. The first step is an intersection (Bar-Hillel construction [15]) where, given a swT T, a sw-VPA A, and an input word s, a sw-VPA B is built, such that for all t, $B(t) = T(s,t) \otimes A(t)$.In the second step, a best AST t is found by applying to B a best search algorithm similar to the shortest distance in graphs [20, 17].

WARNING: [23] much is more general that weighted parsing

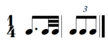
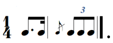
chap. intersection in [15]

The main contributions of the paper are: (i) the introduction of automata, swA, transducers, swT (Section 3), and visibly pushdown automata sw-VPA (Section 4), generalizing the corresponding classes of symbolic and weighted models, (ii) a polynomial best-search algorithm for sw-VPA, and (iii) a uniform framework (Section 5) for parsing over infinite alphabets, the keys to which are $(iii.a)$ the swT-based definition of generic edit distances between input and output (yield) words, and $(iii.b)$ the use, convenient in this context, of nested words, and sw-VPA, instead of syntax trees and grammars.

expressiveness: VPA have restricted equality test. comparable to pebble automata? → conclusion

▶ **Example 1** (Running example). Throughout the paper we illustrate our framework with music transcription examples: Given a *timeline* of musical events with arbitrary timestamps as input, parse it into a structured music score. In our example, input events are pairs $\langle \eta, \tau \rangle$ made of a symbol $\eta \in \Sigma$, where Σ stands for the set of MIDI message symbols [**?**] and $\tau \in \mathbb{Q}$ is a timestamp. The output of parsing is a representation of the sequence in Common Western Music Notation (CWMN) [14] where event symbols belong to the domain Δ of *pitches* (e.g., A4, G5, etc.), temporal information is encoded as *durations* (whole 𝅝, quarter ♩, eight ♪, etc), and notes are grouped in high-level structures (beams, measures, tuplets). The following inputs will be used:

1. $I_1 = [\langle e_1, 0.07 \rangle, \langle e_2, 0.72 \rangle, \langle e_3, 0.91 \rangle]$, over interval $[0, 1[$
2. $I_2 = [\langle e_3, 1.05 \rangle, \langle e_4, 1.36 \rangle, \langle e_5, 1.71 \rangle]$, over interval $[1, 2[$

There exists many possible parsings of $I_1 \cup I_2$ in music notation, among which  and . SW-parsing associates a cost to each solution, and our framework aims at selecting the best one with respect to this cost. ◇

weight value

2 Preliminary Notions

Semirings

We shall consider semirings for the weight values of our language models. A *semiring* $\langle \mathbb{S}, \oplus, \mathbb{0}, \otimes, \mathbb{1} \rangle$ is a structure with a domain $\mathbb{S}$, equipped with two associative binary operators $\oplus$ and $\otimes$, with respective neutral elements $\mathbb{0}$ and $\mathbb{1}$, and such that:

- $\oplus$ is commutative: $\langle \mathbb{S}, \oplus, \mathbb{0} \rangle$ is a commutative monoid and $\langle \mathbb{S}, \otimes, \mathbb{1} \rangle$ a monoid,
- $\otimes$ distributes over $\oplus$: $\forall x, y, z \in \mathbb{S}$, $x \otimes (y \oplus z) = (x \otimes y) \oplus (x \otimes z)$, and $(x \oplus y) \otimes z = (x \otimes z) \oplus (y \otimes z)$,
- $\mathbb{0}$ is absorbing for $\otimes$: $\forall x \in \mathbb{S}$, $\mathbb{0} \otimes x = x \otimes \mathbb{0} = \mathbb{0}$.

Intuitively, in the models presented in this paper, $\oplus$ selects an optimal value from two given values, in order to handle non-determinism, and $\otimes$ combines two values into a single value.

A semiring $\mathbb{S}$ is *commutative* if $\otimes$ is commutative. It is *idempotent* if for all $x \in \mathbb{S}$, $x \oplus x = x$. Every idempotent semiring $\mathbb{S}$ induces a partial ordering $\leq_{\oplus}$ called the *natural* *ordering* of $\mathbb{S}$ [20] defined, by: for all $x, y \in \mathbb{S}$, $x \leq_\oplus y$ iff $x \oplus y = x$. The natural ordering is sometimes defined in the opposite direction [10]; We follow here the direction that coincides with the usual ordering on the Tropical semiring *min-plus* (Figure 2). An idempotent semiring $\mathbb{S}$ is called *total* if it $\leq_\oplus$ is total *i.e.* when for all $x, y \in \mathbb{S}$, either $x \oplus y = x$ or $x \oplus y = y$.

is total necessary?

▶ **Lemma 2** (Monotony, [20]). *Let $\langle \mathbb{S}, \oplus, \mathbb{0}, \otimes, \mathbb{1} \rangle$ be an idempotent semiring. For all $x, y, z \in \mathbb{S}$, if $x \leq_\oplus y$ then $x \oplus z \leq_\oplus y \oplus z$, $x \otimes z \leq_\oplus y \otimes z$ and $z \otimes x \leq_\oplus z \otimes y$.*

To express the property of Lemma 2, we call $\mathbb{S}$ *monotonic wrt* $\leq_\oplus$. Another important semiring property in the context of optimization is superiority [16], which corresponds to the *non-negative weights* condition in shortest-path algorithms [9]. Intuitively, it means that combining elements with $\otimes$ always increase their weight. Formally, it is defined as the property (i) below.

▶ **Lemma 3** (Superiority, Boundedness). *Let $\langle \mathbb{S}, \oplus, \mathbb{0}, \otimes, \mathbb{1} \rangle$ be an idempotent semiring. The two following statements are equivalent:*

i. for all $x, y \in \mathbb{S}$, $x \leq_\oplus x \otimes y$ and $y \leq_\oplus x \otimes y$

ii. for all $x \in \mathbb{S}$, $\mathbb{1} \oplus x = \mathbb{1}$.

Proof. $(ii) \Rightarrow (i)$: $x \oplus (x \otimes y) = x \otimes (\mathbb{1} \oplus y) = x$, by distributivity of $\otimes$ over $\oplus$. Hence $x \leq_\oplus x \otimes y$. Similarly, $y \oplus (x \otimes y) = (\mathbb{1} \oplus x) \otimes y = y$, hence $y \leq_\oplus x \otimes y$. $(i) \Rightarrow (ii)$: by the second inequality of (i), with $y = \mathbb{1}$, $\mathbb{1} \leq_\oplus x \otimes \mathbb{1} = x$, *i.e.*, by definition of $\leq_\oplus$, $\mathbb{1} \oplus x = \mathbb{1}$. ◀

In [16], when the property (i) holds, $\mathbb{S}$ is called *superior wrt* the ordering $\leq_\oplus$. We have seen in the proof of Lemma 3 that it implies that $\mathbb{1} \leq_\oplus x$ for all $x \in \mathbb{S}$. Similarly, by the first inequality of (i) with $y = \mathbb{0}$, $x \leq_\oplus x \otimes \mathbb{0} = \mathbb{0}$. Hence, in a superior semiring, it holds that for all $x \in \mathbb{S}$, $\mathbb{1} \leq_\oplus x \leq_\oplus \mathbb{0}$. Intuitively, from an optimization point of view, it means that $\mathbb{1}$ is the best value, and $\mathbb{0}$ the worst. In [20], $\mathbb{S}$ with the property (ii) of Lemma 3 is called *bounded* – we shall use this term in the rest of the paper. It implies that, when looking for a best path in a graph whose edges are weighted by values of $\mathbb{S}$, the loops can be safely avoided, because, for all $x \in \mathbb{S}$ and $n \geq 1$, $x \oplus x^n = x \otimes (\mathbb{1} \oplus x^{n-1}) = x$.

Ca j'ai pas compris

▶ **Lemma 4.** *Every bounded semiring is idempotent.*

Proof. By boundedness, $\mathbb{1} \oplus \mathbb{1} = \mathbb{1}$, and idempotency follows by multiplying both sides by x and distributing. ◀

Here the difference between $\mathbb{S}$ as a structure and as a domain is blurred.

$j \in \mathbb{N}$: j is en element of $\mathbb{N}$, not the same s $j \subset \mathbb{N}$

We shall need below infinite sums with $\oplus$. A semiring $\mathbb{S}$ is called *complete* [11] if it has an operation $\bigoplus_{i \in I} x_i$ for every family $(x_i)_{i \in I}$ of elements of $dom(\mathbb{S})$ over an index set $I \subset \mathbb{N}$, such that:

i. infinite sums extend finite sums:

$$\bigoplus_{i \in \emptyset} x_i = \mathbb{0}, \quad \forall j \in \mathbb{N}, \bigoplus_{i \in \{j\}} x_i = x_j, \forall j, k \in \mathbb{N}, j \neq k, \bigoplus_{i \in \{j,k\}} x_i = x_j \oplus x_k,$$

ii. associativity and commutativity:

for all $I \subseteq \mathbb{N}$ and all partition $(I_j)_{j \in J}$ of I, $\displaystyle\bigoplus_{j \in J} \bigoplus_{i \in I_j} x_i = \bigoplus_{i \in I} x_i$,

iii. distributivity of product over infinite sum:

for all $I \subseteq \mathbb{N}$, $\displaystyle\bigoplus_{i \in I} (x \otimes y_i) = x \otimes \bigoplus_{i \in I} y_i$, and $\displaystyle\bigoplus_{i \in I} (x_i \otimes y) = (\bigoplus_{i \in I} x_i) \otimes y$.

results of this paper for semirings commutative, bounded, total and complete

▶ **Example 5.** The recursive subdivision of time that leads to hierarchichal structures of music notation can be modeled as production rules. Since there exists several possible division, rules can be weighted in the tropical semiring whose domain $\mathbb{R}_+ \cup \{+\infty\}$, $\oplus$ is

	domain	$\oplus$	$\otimes$	$\mathbb{0}$	$\mathbb{1}$
Boolean	$\{\bot, \top\}$	$\vee$	$\wedge$	$\bot$	$\top$
Counting	$\mathbb{N}$	$+$	$\times$	0	1
Viterbi	$[0,1] \subset \mathbb{R}$	*max*	$\times$	0	1
Tropical min-plus	$\mathbb{R}_+ \cup \{\infty\}$	*min*	$+$	∞	0

Figure 2 Some commutative, bounded, total and complete semirings.

min, $\mathbb{0} = +\infty$, $\otimes$ is sum, and $\mathbb{1} = 0$. For instance, the following production rules define two possible divisions of a bounded time interval into respectively a duplet and a triplet.

$$\rho_1 : q_0 \xrightarrow{0.06} \langle q_1, q_2 \rangle, \ \rho_2 : q_0 \xrightarrow{0.12} \langle q_1, q_2, q_2 \rangle.$$

Further binary divisions of time sub-intervals are possible with:

$$\rho_3 : q_2 \xrightarrow{0.1} \langle q_3, q_3 \rangle, \ \rho_4 : q_3 \xrightarrow{0.11} \langle q_4, q_4 \rangle.$$

◇

Label Theory

We shall now define the functions labeling the transitions of SW automata and transducers, generalizing the Boolean algebras of [7] from Boolean to other semiring domains. We consider *alphabets*, which are countable sets of symbols denoted Σ, Δ,... Given a semiring $\langle \mathbb{S}, \oplus, \mathbb{0}, \otimes, \mathbb{1} \rangle$, a *label theory* over $\mathbb{S}$ is a set $\bar{\Phi}$ of recursively enumerable sets denoted Φ_Σ, containing unary functions of type $\Sigma \to \mathbb{S}$, or $\Phi_{\Sigma,\Delta}$, containing binary functions $\Sigma \times \Delta \to \mathbb{S}$, and such that:

ADD: notations Σ^*, ε...

unary for swA (weight depends on input symbol) and binary for transducers and VPA (weight depends on input symbol AND output or stack symbol)

- for all $\Phi_{\Sigma,\Delta} \in \bar{\Phi}$, we have $\Phi_\Sigma \in \bar{\Phi}$ and $\Phi_\Delta \in \bar{\Phi}$
- every $\Phi_\Sigma \in \bar{\Phi}$ contains all the constant functions from Σ into $\mathbb{S}$,
- for all $\alpha \in \mathbb{S}$ and $\phi \in \Phi_\Sigma$, $\alpha \otimes \phi : x \mapsto \alpha \otimes \phi(x)$, and $\phi \otimes \alpha : x \mapsto \phi(x) \otimes \alpha$ belong to Φ_Σ, and similarly for $\oplus$ and for $\Phi_{\Sigma,\Delta}$
- for all $\phi, \phi' \in \Phi_\Sigma$, $\phi \otimes \phi' : x \mapsto \phi(x) \otimes \phi'(x)$ belongs to Φ_Σ
- for all $\eta, \eta' \in \Phi_{\Sigma,\Delta}$ $\eta \otimes \eta' : x, y \mapsto \eta(x,y) \otimes \eta'(x,y)$ belongs to $\Phi_{\Sigma,\Delta}$
- for all $\phi \in \Phi_\Sigma$ and $\eta \in \Phi_{\Sigma,\Delta}$, $\phi \otimes_1 \eta : x, y \mapsto \phi(x) \otimes \eta(x,y)$ and $\eta \otimes_1 \phi : x, y \mapsto \eta(x,y) \otimes \phi(x)$ belong to $\Phi_{\Sigma,\Delta}$
- for all $\psi \in \Phi_\Delta$ and $\eta \in \Phi_{\Sigma,\Delta}$, $\psi \otimes_2 \eta : x, y \mapsto \psi(y) \otimes \eta(x,y)$ and $\eta \otimes_2 \psi : x, y \mapsto \eta(x,y) \otimes \psi(y)$ belong to $\Phi_{\Sigma,\Delta}$
- similar closures hold for $\oplus$.

partial application is needed?

Intuitively, the operators $\bigoplus_\Sigma$ return global minimum, *wrt* $\leq_\oplus$, of functions of Φ_Σ. When the semiring $\mathbb{S}$ is complete, we consider the following operators on the functions of $\bar{\Phi}$.

$$\begin{gathered}\bigoplus_\Sigma : \Phi_\Sigma \to \mathbb{S}, \ \phi \mapsto \bigoplus_{a \in \Sigma} \phi(a) \\ \bigoplus_\Sigma^1 : \Phi_{\Sigma,\Delta} \to \Phi_\Delta, \ \eta \mapsto \big(y \mapsto \bigoplus_{a \in \Sigma} \eta(a,y)\big) \quad \bigoplus_\Delta^2 : \Phi_{\Sigma,\Delta} \to \Phi_\Sigma, \ \eta \mapsto \big(x \mapsto \bigoplus_{b \in \Delta} \eta(x,b)\big)\end{gathered}$$

In what follows, we might omit the sub- and superscripts in $\otimes_1$, $\bigoplus_\Sigma^1$..., when there is no ambiguity. We shall keep them only for the special case $\Sigma = \Delta$, *i.e.* $\eta \in \Phi_{\Sigma,\Sigma}$, in order to be able to distinguish between the first and the second argument.

▶ **Definition 6.** *A label theory $\bar{\Phi}$ is* complete *when the underlying semiring $\mathbb{S}$ is complete, and for all $\Phi_{\Sigma,\Delta} \in \bar{\Phi}$ and all $\eta \in \Phi_{\Sigma,\Delta}$, $\bigoplus^1_\Sigma \eta \in \Phi_\Delta$ and $\bigoplus^2_\Delta \eta \in \Phi_\Sigma$.*

notion of diagram of functions akin BDD for transitions in practice

The following facts are immediate.

mv appendix?

▶ **Lemma 7.** *For $\bar{\Phi}$ complete $\alpha \in \mathbb{S}$, $\phi, \phi' \in \Phi_\Sigma$, $\psi \in \Phi_\Delta$, and $\eta \in \Phi_{\Sigma,\Delta}$:*

i. $\bigoplus_\Sigma \bigoplus^2_\Delta \eta = \bigoplus_\Delta \bigoplus^1_\Sigma \eta$

ii. $\alpha \otimes \bigoplus_\Sigma \phi = \bigoplus_\Sigma (\alpha \otimes \phi)$ *and* $\left(\bigoplus_\Sigma \phi\right) \otimes \alpha = \bigoplus_\Sigma (\phi \otimes \alpha)$, *and similarly for* $\oplus$

iii. $\left(\bigoplus_\Sigma \phi\right) \oplus \left(\bigoplus_\Sigma \phi'\right) = \bigoplus_\Sigma (\phi \oplus \phi')$ *and* $\left(\bigoplus_\Sigma \phi\right) \otimes \left(\bigoplus_\Sigma \phi'\right) = \bigoplus_\Sigma (\phi \otimes \phi')$

iv. $\left(\bigoplus^2_\Delta \eta\right) \oplus \left(\bigoplus^2_\Delta \eta'\right) = \bigoplus^2_\Delta (\eta \oplus \eta')$, *and* $\left(\bigoplus^2_\Delta \eta\right) \otimes \left(\bigoplus^2_\Delta \eta'\right) = \bigoplus^2_\Delta (\eta \otimes \eta')$

v. $\phi \otimes \left(\bigoplus^2_\Delta \eta\right) = \bigoplus_\Delta (\phi \otimes_1 \eta)$, *and* $\left(\bigoplus^2_\Delta \eta\right) \otimes \phi = \bigoplus_\Delta (\eta \otimes_1 \phi)$, *and similarly for* $\oplus$

vi. $\psi \otimes \left(\bigoplus^1_\Sigma \eta\right) = \bigoplus_\Sigma (\psi \otimes_2 \eta)$, *and* $\left(\bigoplus^1_\Sigma \eta\right) \otimes \psi = \bigoplus_\Sigma (\eta \otimes_2 \psi)$, *and similarly for* $\oplus$

Je trouve qu'il y a beaucoup de notions à retenir (complete, effective) et ça devient difficile pour un lecteur non spécialiste. Est-ce que tout est nécessaire (je ne sais plus qui m'avait dit: un concept en plus, un point en moins.

A label theory is called *effective* when for all $\phi \in \Phi_\Sigma$ and $\eta \in \Phi_{\Sigma,\Delta}$, $\bigoplus_\Sigma \phi$, $\bigoplus_\Delta \bigoplus_\Sigma \eta$, and $\bigoplus_\Sigma \bigoplus_\Delta \eta$ can be effectively computed from ϕ and η.

∃ oracle returning … in worst time complexity T.

▶ **Example 8.** Consider the music transcription problem, with an input representing a music performance. In order to align the input with a music score, we must take into consideration the expressive timing of human performance that results in small time shifts between an input event and the corresponding notation event. These shifts can be weighted as the time distance between both, computed in the tropical semiring with a base function based on a given $\delta \in \Phi_{\Sigma,\Delta}$.

$$\delta(< e_1, t1_>, < e_2, t_2 >) = \begin{cases} |t_1 - t_2| & if e_1 = e_2 \\ \mathbb{0} & otherwise \end{cases}$$

For the sake of concreteness, consider our running example 1 with $s = I_1 \cup I_2 = [< e_1, 0.07 >, < e_2, 0.72 >, < e_3, 0.91 >] \cup [< e_3, 1.05 >, < e_4, 1.36 >, < e_5, 1.71 >]$ and the music notation $t =$ . The latter shows a subdivision of the temporal interval $[1, 2[$ based on the rules from Example 5. The first measure (I_1) results from the successive applications of rules ρ_1 (division in two) and rules ρ_3 (division of the second half) in two. The second measure is a division in three obtained by rule ρ_3. It follows that the notation defines the sequence of timestamps $0, \frac{3}{4}, \frac{7}{8}, 1, \frac{4}{3}, \frac{5}{3}$. The distance between s and t is the pairwise difference between the timestamps from s and t, 0.255. ◇

3 SW Automata and Transducers

We follow the approach of [21] for the computation of distances, between words and languages, using weighted transducers, and extend it to infinite alphabets. The models introduced in this section generalize weighted automata and transducers [11] by labeling each transition with a weight function (instead of a simple weight value), that takes the input and output symbols as parameters. These functions are similar to the guards of symbolic automata [7, 8], but they can return values in a generic semiring, whereas the latter guards are restricted to the Boolean semiring.
Let $\mathbb{S}$ be a commutative semiring, Σ and Δ be alphabets called respectively *input* and *output*, and $\bar{\Phi}$ be a label theory over $\mathbb{S}$ containing Φ_Σ, Φ_Δ, $\Phi_{\Sigma,\Delta}$.

▶ **Definition 9.** *A* symbolic-weighted transducer *(swT) over Σ, Δ, $\mathbb{S}$ and $\bar{\Phi}$ is a tuple $T = \langle Q, \mathsf{in}, \bar{\mathsf{w}}, \mathsf{out} \rangle$, where Q is a finite set of states, $\mathsf{in} : Q \to \mathbb{S}$ (respectively $\mathsf{out} : Q \to \mathbb{S}$) are functions defining the weight for entering (respectively leaving) computation in a state, and $\bar{\mathsf{w}}$ is a triplet of transition functions $\mathsf{w}_{10} : Q \times Q \to \Phi_\Sigma$, $\mathsf{w}_{01} : Q \times Q \to \Phi_\Delta$, and $\mathsf{w}_{11} : Q \times Q \to \Phi_{\Sigma,\Delta}$.*

We call *number of transitions* of T the number of pairs of states $q, q' \in Q$ such that w_{10} or w_{01} or w_{11} is not the constant $\mathbb{0}$. For convenience, we shall sometimes present transitions as functions of $Q \times (\Sigma \cup \{\varepsilon\}) \times (\Delta \cup \{\varepsilon\}) \times Q \to \mathbb{S}$, overloading the function names, such that, for all $q, q' \in Q$, $a \in \Sigma$, $b \in \Delta$:

$$\begin{aligned}
\mathsf{w}_{10}(q, a, \varepsilon, q') &= \phi(a) && \text{where } \phi = \mathsf{w}_{10}(q, q') \in \Phi_\Sigma,\\
\mathsf{w}_{01}(q, \varepsilon, b, q') &= \psi(b) && \text{where } \psi = \mathsf{w}_{01}(q, q') \in \Phi_\Delta,\\
\mathsf{w}_{11}(q, a, b, q') &= \eta(a, b) && \text{where } \eta = \mathsf{w}_{11}(q, q') \in \Phi_{\Sigma,\Delta}.
\end{aligned}$$

The swT T computes on pairs of words $\langle s, t\rangle \in \Sigma^* \times \Delta^*$, s and t, being respectively called *input* and *output* word. More precisely, T defines a mapping from $\Sigma^* \times \Delta^*$ into $\mathbb{S}$, based on an intermediate function weight_T defined recursively, for every states $q, q' \in Q$, and every pairs of strings $\langle s, t\rangle \in \Sigma^* \times \Delta^*$, where au, and bv, denote the concatenation of the symbol $a \in \Sigma$ (resp. $b \in \Delta$) with a word $u \in \Sigma^*$ (resp. $v \in \Delta^*$).

It compute asynchronously: advance in s and not in t (w_{10}), or the opposite (w_{01}), or advance in both s and t (w_{11}).

added u and v def

$$\mathsf{weight}_T(q, \varepsilon, \varepsilon, q') = \mathbb{1} \quad \text{if } q = q' \text{ and } \mathbb{0} \text{ otherwise} \tag{1}$$

$$\begin{aligned}
\mathsf{weight}_T(q, s, t, q') = & \bigoplus_{\substack{q'' \in Q\\ s = au,\, a \in \Sigma}} \mathsf{w}_{10}(q, a, \varepsilon, q'') \otimes \mathsf{weight}_T(q'', u, t, q')\\
& \oplus \bigoplus_{\substack{q'' \in Q\\ t = bv,\, b \in \Delta}} \mathsf{w}_{01}(q, \varepsilon, b, q'') \otimes \mathsf{weight}_T(q'', s, v, q')\\
& \oplus \bigoplus_{\substack{q'' \in Q\\ s = au,\, t = bv}} \mathsf{w}_{11}(q, a, b, q'') \otimes \mathsf{weight}_T(q'', u, v, q')
\end{aligned}$$

We recall that, by convention (Section 2), an empty sum with $\bigoplus$ is equal to $\mathbb{0}$. Intuitively, using a transition $\mathsf{w}_{ij}(q, a, b, q')$ means for T: when reading respectively a and b at the current positions in the input and output words, increment the current position in the input word if and only if $i = 1$, and in the output word iff $j = 1$, and change state from q to q'. When $a = \varepsilon$ (resp. $b = \varepsilon$), the current symbol in the input (resp. output) is not read. Since $\mathbb{0}$ is absorbing for $\otimes$ in $\mathbb{S}$, one term $\mathsf{w}_{ij}(q, a, b, q'')$ equal to $\mathbb{0}$ in the above expression will be ignored in the sum, meaning that there is no possible transition from state q into state q' while reading a and b. This is analogous to the case of a transition's guard not satisfied by $\langle a, b\rangle$ for symbolic transducers.

The expression (1) can be seen as a stateful definition of an edit-distance between a word $s \in \Sigma^*$ and a word $t \in \Delta^*$, see also [22]. Intuitively, $\mathsf{w}_{10}(q, a, \varepsilon, r)$ is the cost of the deletion of the symbol $a \in \Sigma$ in s, $\mathsf{w}_{01}(q, \varepsilon, b, r)$ is the cost of the insertion of $b \in \Delta$ in t, and $\mathsf{w}_{11}(q, a, b, r)$ is the cost of the substitution of $a \in \Sigma$ by $b \in \Delta$. The cost of a sequence of such operations transforming s into t, is the product, with $\otimes$, of the individual costs of the operations involved; and the distance between s and t is the sum, with $\oplus$, of all possible products. Formally, the weight associated by T to $\langle s, t\rangle \in \Sigma^* \times \Delta^*$ is:

$$T(s, t) = \bigoplus_{q, q' \in Q} \mathsf{in}(q) \otimes \mathsf{weight}_T(q, s, t, q') \otimes \mathsf{out}(q') \tag{2}$$

▶ **Example 10.** Let us develop the example of comparison between music played by a performer, represented as a sequence $s \in \Sigma^*$ of events in the MIDI alphabet Σ, and a music score represented as a sequence $t \in \Delta^*$ in the CWMN alphabet Δ. We build a small weighted transducer model with two states q_0 and q_1 that calculates the distance between s and t.

If one performed event s_i corresponds to one notated event t_1 (for instance MIDI code 61 and pitch A4), the weight value computed by the swT is the time distance between both, as in Example 8, and is modeled by transitions w_{11} below. If we meet the music notation symbol '-' that represents continuation (such as instance in *ties* ♩‿♪, or *dots* ♩.), it is skipped with no cost (transitions w_{01} or weight $\mathbb{1}$).

$$\begin{array}{rclrcl}
\mathsf{w}_{11}(q_0, d, \langle \mathsf{e}, d' \rangle, q_0) & = & |d' - d| & \mathsf{w}_{11}(q_1, d, \langle \mathsf{e}, d' \rangle, q_0) & = & |d' - d| \\
\mathsf{w}_{01}(q_0, \varepsilon, \langle -, d' \rangle, q_0) & = & \mathbb{1} & \mathsf{w}_{01}(q_1, \varepsilon, \langle -, d' \rangle, q_0) & = & \mathbb{1} \\
\mathsf{w}_{10}(q_0, d, \varepsilon, q_1) & = & \alpha & & &
\end{array}$$

We also must be able to take performing errors into account, while still being able to compare with the score, since a performer could, for example, play an unwritten extra note. This is modelled by the transition w_{10} with an arbitrary weight value $\alpha \in \mathbb{S}$, switching from state q_0 (normal) to q_1 (error). The transitions in the second column below switch back to the normal state q_0. At last, we let q_0 be the only initial and final state, with $\mathsf{in}(q_0) = \mathsf{out}(q_0) = \mathbb{1}$, and $\mathsf{in}(q_1) = \mathsf{out}(q_1) = \mathbb{0}$.

reformulated this sentence

Comprends pas cette phrase

ccl to the ex

That way, an swT is capable of evaluating the differences between a score and a performance, all the while ensuring that performance errors are plausible. ◇

The *Symbolic Weighted Automata* are defined similarly as the transducers of Definition 9, by simply omitting the output symbols.

▶ **Definition 11.** *A* symbolic-weighted automaton *(swA) over Σ, $\mathbb{S}$ and $\bar{\Phi}$ is a tuple $A = \langle Q, \mathsf{in}, \mathsf{w}_1, \mathsf{out} \rangle$, where Q is a finite set of states,* $\mathsf{in} : Q \to \mathbb{S}$ *(respectively* $\mathsf{out} : Q \to \mathbb{S}$*) are functions defining the weight for entering (respectively leaving) computation in a state, and* w_1 *is a transition function from $Q \times Q$ into Φ_Σ.*

As above in the case of swT, when $\mathsf{w}_1(q, q') = \phi \in \Phi_\Sigma$, we may write $\mathsf{w}_1(q, a, q')$ for $\phi(a)$. The computation of A on words $s \in \Sigma^*$ is defined with an intermediate function weight_A, defined as follows for $q, q' \in Q$, $a \in \Sigma$, $u \in \Sigma^*$,

$$\begin{aligned}
\mathsf{weight}_A(q, \varepsilon, q) &= \mathbb{1} \\
\mathsf{weight}_A(q, \varepsilon, q') &= \mathbb{0} \quad \text{if } q \neq q' \\
\mathsf{weight}_A(q, au, q') &= \bigoplus_{q'' \in Q} \mathsf{w}_1(q, a, q'') \otimes \mathsf{weight}_A(q'', u, q')
\end{aligned} \tag{3}$$

and the weight value associated by A to $s \in \Sigma^*$ is defined as follows:

$$A(s) = \bigoplus_{q, q' \in Q} \mathsf{in}(q) \otimes \mathsf{weight}_A(q, s, q') \otimes \mathsf{out}(q') \tag{4}$$

Il me manque une explication: on construit un automate qui, étant donnée une partition t, renvoie la distance minimale avec n'importe quelle performance (distance donnée par un transducer)? Quel est le rôle de A(s)?

The following property will be useful to the approach on symbolic weighted parsing presented in Section 5.

▶ **Proposition 12.** *Given a swT T over Σ, Δ, $\mathbb{S}$ commutative, bounded and complete, and $\bar{\Phi}$ effective, and a swA A over Σ, $\mathbb{S}$ and $\bar{\Phi}$, there exists an effectively constructible swA $B_{A,T}$ over Δ, $\mathbb{S}$ and $\bar{\Phi}$, such that for all $t \in \Delta^*$, $B_{A,T}(t) = \bigoplus_{s \in \Sigma^*} A(s) \otimes T(s, t)$.*

Proof. Let $T = \langle Q, \mathsf{in}_T, \bar{\mathsf{w}}, \mathsf{out}_T \rangle$, where $\bar{\mathsf{w}}$ contains w_{10}, w_{01}, and w_{11}, from $Q \times Q$ into respectively Φ_Σ, Φ_Δ, and $\Phi_{\Sigma,\Delta}$, and let $A = \langle P, \mathsf{in}_A, \mathsf{w}_1, \mathsf{out}_A \rangle$ with $\mathsf{w}_1 : Q \times Q \to \Phi_\Sigma$. The

state set of $B_{A,T}$ will be $Q' = P \times Q$. The entering, leaving and transition functions of $B_{A,T}$ will simulate synchronized computations of A and T, while reading an output word of Δ^*. Its state entering functions is defined for all $p \in P$, $q \in Q$ by $\mathsf{in}'(p, q) = \mathsf{in}_A(p) \otimes \mathsf{in}_T(q)$. The transition function w'_1 will roughly perform a synchronized product of transitions defined by w_1, w_{01} (T reading in output word and not an input word) and w_{11} (T reading both an input word and an output word). Moreover, w'_1 also needs to simulate transitions defined by w_{10}: T reading in input word and not an output word. Since $B_{A,T}$ will read only in the output word, such a transition corresponds to an ε-transition of swA, but swA have been defined without ε-transitions. Therefore, in order to take care of this case, we perform an on-the-fly suppression of ε-transition in the swA in construction, following the algorithm of [19].
Initially, for all $p_1, p_2 \in P$, and $q_1, q_2 \in Q$, let

$$\mathsf{w}'_1\big(\langle p_1, q_1\rangle, \langle p_2, q_2\rangle\big) = \mathsf{w}_1(p_1, p_2) \otimes \big[\mathsf{w}_{01}(q_1, q_2) \oplus \bigoplus_{\Sigma} \mathsf{w}_{11}(q_1, q_2)\big].$$

Iterate the following for all $p_1 \in P$ and $q_1, q_2 \in Q$: for all $p_2 \in P$ and $q_3 \in Q$,

$$\mathsf{w}'_1\big(\langle p_1, q_1\rangle, \langle p_2, q_3\rangle\big) \oplus\!= \bigoplus_{\Sigma} \mathsf{w}_{10}(q_1, q_2) \otimes \mathsf{w}'_1\big(\langle p_1, q_2\rangle, \langle p_2, q_3\rangle\big)$$

and $\mathsf{out}'(p_1, q_1) \oplus\!= \bigoplus_{\Sigma} \mathsf{w}_{10}(q_1, q_2) \otimes \mathsf{out}'(p_1, q_2)$ ◀

proof correctness

The construction time and size for $B_{A,T}$ are $O(\|T\|^3.\|A\|^2)$, where the sizes $\|T\|$ and $\|A\|$ are their number of states.

revise with nb of tr. and states

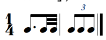
▶ **Corollary 13.** *Given a swT T over Σ, Δ, $\mathbb{S}$ commutative, bounded and complete, and $\bar{\Phi}$ effective, and $s \in \Sigma^+$, there exists an effectively constructible swA $B_{s,T}$ over Δ, $\mathbb{S}$ and $\bar{\Phi}$, such that for all $t \in \Delta^*$, $B_{s,T}(t) = T(s, t)$.*

4 SW Visibly Pushdown Automata

The model presented in this section generalizes symbolic VPA (sVPA [6], generalizing themselves VPA [1] to infinite alphabets) from Boolean semirings to arbitrary semiring weight domains. It will compute on nested words over infinite alphabets, associating to every such word a weight value. Nested words are able to describe structures of labeled trees, and in the context of parsing, they will be useful to represent AST.

see §5 and App.A

Let Δ be a countable alphabet that we assume partitioned into three subsets Δ_i, Δ_c, Δ_r, whose elements are respectively called *internal*, *call* and *return* symbols. Let $\langle \mathbb{S}, \oplus, \mathbb{0}, \otimes, \mathbb{1}\rangle$ be a commutative and complete semiring and let $\bar{\Phi} = \langle \Phi_\mathsf{i}, \Phi_\mathsf{c}, \Phi_\mathsf{r}, \Phi_\mathsf{ci}, \Phi_\mathsf{cc}, \Phi_\mathsf{cr}\rangle$ be a label theory over $\mathbb{S}$ where Φ_i, Φ_c, Φ_r and Φ_cx (with $\mathsf{x} \in \{\mathsf{i}, \mathsf{c}, \mathsf{r}\}$) stand respectively for Φ_{Δ_i}, Φ_{Δ_c}, Φ_{Δ_r} and $\Phi_{\Delta_\mathsf{c}, \Delta_\mathsf{x}}$.

▶ **Example 14.** Consider once more the scores of Exemple 1. Δ_i represents the set of music events of the form $< n, t >$ where n is either a note (e.g. A4), the continuation symbol $-$, or a rest symbol R. Δ_c and Δ_r are used to encode the score structure: Let $\Delta_\mathsf{c} = \{[, <]\}$ and $\Delta_\mathsf{r} = \{], >]\}$ where [(rep.]) and $<$ (resp. $>$) correspond to start/end of measures, and start/end of beams. Then  is encoded with the nested word $[0, \frac{3}{4}, \frac{7}{8} >][< 1, \frac{4}{3}, \frac{5}{3}]$ ◇

▶ **Definition 15.** *A Symbolic Weighted Visibly Pushdown Automata (sw-VPA) over $\Delta = \Delta_\mathsf{i} \uplus \Delta_\mathsf{c} \uplus \Delta_\mathsf{r}$, $\mathbb{S}$ and $\bar{\Phi}$ is a tuple $A = \langle Q, P, \mathsf{in}, \bar{\mathsf{w}}, \mathsf{out}\rangle$, where Q is a finite set of states, P is a finite set of stack symbols,* $\mathsf{in} : Q \to \mathbb{S}$ *(respectively* $\mathsf{out} : Q \to \mathbb{S}$*) are functions defining* *the weight for entering (respectively leaving) a state, and* $\bar{\mathsf{w}}$ *is a sextuplet composed of the transition functions :* $\mathsf{w}_\mathsf{i} : Q \times P \times Q \to \Phi_{\mathsf{ci}}$, $\mathsf{w}^\mathsf{e}_\mathsf{i} : Q \times Q \to \Phi_\mathsf{i}$, $\mathsf{w}_\mathsf{c} : Q \times P \times Q \times P \to \Phi_{\mathsf{cc}}$, $\mathsf{w}^\mathsf{e}_\mathsf{c} : Q \times P \times Q \to \Phi_\mathsf{c}$, $\mathsf{w}_\mathsf{r} : Q \times P \times Q \to \Phi_{\mathsf{cr}}$, $\mathsf{w}^\mathsf{e}_\mathsf{r} : Q \times Q \to \Phi_\mathsf{r}$.

Similarly as in Section 3, we extend the above transition functions as follows for all $q, q' \in Q$, $p \in P$, $a \in \Delta_\mathsf{i}$, $c \in \Delta_\mathsf{c}$, $r \in \Delta_\mathsf{r}$, overloading their names:

$$
\begin{array}{lll}
\mathsf{w}_\mathsf{i} : Q \times [\Delta_\mathsf{c} \times P] \times \Delta_\mathsf{i} \times Q \to \mathbb{S} & \mathsf{w}_\mathsf{i}(q, c, p, a, q') = \eta_{\mathsf{ci}}(c, a) & \text{where } \eta_{\mathsf{ci}} = \mathsf{w}_\mathsf{i}(q, p, q'), \\
\mathsf{w}^\mathsf{e}_\mathsf{i} : Q \times \Delta_\mathsf{i} \times Q \to \mathbb{S} & \mathsf{w}^\mathsf{e}_\mathsf{i}(q, a, q') = \phi_\mathsf{i}(a) & \text{where } \phi_\mathsf{i} = \mathsf{w}^\mathsf{e}_\mathsf{i}(q, q'). \\
\mathsf{w}_\mathsf{c} : Q \times [\Delta_\mathsf{c} \times P] \times [\Delta_\mathsf{c} \times P] \times Q \to \mathbb{S} & \mathsf{w}_\mathsf{c}(q, c, p, c', p', q') = \eta_{\mathsf{cc}}(c, c') & \text{where } \eta_{\mathsf{cc}} = \mathsf{w}_\mathsf{c}(q, p, p', q'), \\
\mathsf{w}^\mathsf{e}_\mathsf{c} : Q \times [\Delta_\mathsf{c} \times P] \times Q \to \mathbb{S} & \mathsf{w}^\mathsf{e}_\mathsf{c}(q, c, p, q') = \phi_\mathsf{c}(c) & \text{where } \phi_\mathsf{c} = \mathsf{w}^\mathsf{e}_\mathsf{c}(q, p, q'). \\
\mathsf{w}_\mathsf{r} : Q \times [\Delta_\mathsf{c} \times P] \times \Delta_\mathsf{r} \times Q \to \mathbb{S} & \mathsf{w}_\mathsf{r}(q, c, p, r, q') = \eta_{\mathsf{cr}}(c, r) & \text{where } \eta_{\mathsf{cr}} = \mathsf{w}_\mathsf{r}(q, p, q'), \\
\mathsf{w}^\mathsf{e}_\mathsf{r} : Q \times \Delta_\mathsf{r} \times Q \to \mathbb{S} & \mathsf{w}^\mathsf{e}_\mathsf{r}(q, r, q') = \phi_\mathsf{r}(r) & \text{where } \phi_\mathsf{r} = \mathsf{w}^\mathsf{e}_\mathsf{r}(q, q').
\end{array}
$$

The intuition is the following for the above transitions. $\mathsf{w}^\mathsf{e}_\mathsf{i}$, $\mathsf{w}^\mathsf{e}_\mathsf{c}$, and $\mathsf{w}^\mathsf{e}_\mathsf{r}$ describe the cases where the stack is empty. w_i and $\mathsf{w}^\mathsf{e}_\mathsf{i}$ both read an input internal symbol a and change state from q to q', without changing the stack. Moreover, w_i reads a pair made of $c \in \Delta_\mathsf{c}$ and $p \in P$ on the top of the stack (c is compared to a by the weight function $\eta_{\mathsf{ci}} \in \Phi_{\mathsf{ci}}$). w_c and $\mathsf{w}^\mathsf{e}_\mathsf{c}$ read the input call symbol c', push it to the stack along with p', and change state from q to to q'. Moreover, w_c reads c and p at the top of the stack (c is compared to c'). w_r and $\mathsf{w}^\mathsf{e}_\mathsf{r}$ read the input return symbol r, and change state from q to to q'. Moreover, w_r reads and pop from stack a pair made of c and p, (c is compared to r).

moved this to the beginning

Formally, the transitions of the automaton A are defined in term of an intermediate function weight_A, like in Section 3. A configuration, denoted by $q[\gamma]$, is here composed of a state $q \in Q$ and a stack content $\gamma \in \Gamma^*$, where $\Gamma = \Delta_\mathsf{c} \times P$. Hence, weight_A is a function from $[Q \times \Gamma^*] \times \Delta^* \times [Q \times \Gamma^*]$ into $\mathbb{S}$. The empty stack is denoted by $\perp$, and the upmost symbol is the last pushed content. The following functions illustrate each of the possible cases, being : reading $a \in \Delta_\mathsf{i}$, or $c \in \Delta_\mathsf{c}$, or $r \in \Delta_\mathsf{r}$ for each possible state of the stack (empty or not), to add to $u \in \Delta^*$.

intro to func

introduced the 6 cases

notation cp for $\langle c, p \rangle$?

$$\mathsf{weight}_A\big(q[\perp], \varepsilon, q'[\perp]\big) = \mathbb{1} \text{ if } q = q' \text{ and } \mathbb{0} \text{ otherwise} \tag{5}$$

$$\mathsf{weight}_A\Big(q\begin{bmatrix}\langle c, p\rangle \\ \gamma\end{bmatrix}, a\,u, q'[\gamma']\Big) = \bigoplus_{q'' \in Q} \mathsf{w}_\mathsf{i}(q, c, p, a, q'') \otimes \mathsf{weight}_A\Big(q''\begin{bmatrix}\langle c, p\rangle \\ \gamma\end{bmatrix}, u, q'[\gamma']\Big)$$

$$\mathsf{weight}_A\big(q[\perp], a\,u, q'[\gamma']\big) = \bigoplus_{q'' \in Q} \mathsf{w}^\mathsf{e}_\mathsf{i}(q, a, q'') \otimes \mathsf{weight}_A\big(q''[\perp], u, q'[\gamma']\big)$$

$$\mathsf{weight}_A\Big(q\begin{bmatrix}\langle c, p\rangle \\ \gamma\end{bmatrix}, c'\,u, q'[\gamma']\Big) = \bigoplus_{\substack{q'' \in Q \\ p' \in P}} \mathsf{w}_\mathsf{c}\big(q, c, p, c', p', q''\big) \otimes \mathsf{weight}_A\Big(q''\begin{bmatrix}\langle c', p'\rangle \\ \langle c, p\rangle \\ \gamma\end{bmatrix}, u, q'[\gamma']\Big)$$

$$\mathsf{weight}_A\big(q[\perp], c\,u, q'[\gamma']\big) = \bigoplus_{\substack{q'' \in Q \\ p \in P}} \mathsf{w}^\mathsf{e}_\mathsf{c}(q, c, p, q'') \otimes \mathsf{weight}_A\big(q''[\langle c, p\rangle], u, q'[\gamma']\big)$$

$$\mathsf{weight}_A\Big(q\begin{bmatrix}\langle c, p\rangle \\ \gamma\end{bmatrix}, r\,u, q'[\gamma']\Big) = \bigoplus_{q'' \in Q} \mathsf{w}_\mathsf{r}\big(q, c, p, r, q''\big) \otimes \mathsf{weight}_A\big(q''[\gamma], u, q'[\gamma']\big)$$

$$\mathsf{weight}_A\big(q[\perp], r\,u, q'[\gamma']\big) = \bigoplus_{q'' \in Q} \mathsf{w}^\mathsf{e}_\mathsf{r}(q, r, q'') \otimes \mathsf{weight}_A\big(q''[\perp], u, q'[\gamma']\big)$$

c p to <c, p>

The weight associated by A to $t \in \Delta^*$ is defined according to empty stack semantics:

$$A(t) = \bigoplus_{q,q' \in Q} \mathsf{in}(q) \otimes \mathsf{weight}_A\big(q[\bot], t, q'[\bot]\big) \otimes \mathsf{out}(q'). \tag{6}$$

Every swA $A = \langle Q, \mathsf{in}, \mathsf{w}_1, \mathsf{out}\rangle$, over Σ, $\mathbb{S}$ and $\bar{\Phi}$ is a particular case of sw-VPA $\langle Q, \emptyset, \mathsf{in}, \bar{\mathsf{w}}, \mathsf{out}\rangle$ over Δ, $\mathbb{S}$ and $\bar{\Phi}$ with $\Delta_{\mathsf{i}} = \Sigma$ and $\Delta_{\mathsf{c}} = \Delta_{\mathsf{r}} = \emptyset$, and computing with an always empty stack: $\mathsf{w}_{\mathsf{i}}^{\mathsf{e}} = \mathsf{w}_1$ and all the other functions of $\bar{\mathsf{w}}$ are the constant $\mathbb{0}$.

Similarly to VPA [1] and sVPA [6], the class of sw-VPA is closed under the binary operators of the underlying semiring.

▶ **Proposition 16.** *Let A_1 and A_2 be two sw-VPA over the same Δ, $\mathbb{S}$ and $\bar{\Phi}$. There exists two effectively constructible sw-VPA $A_1 \oplus A_2$ and $A_1 \otimes A_2$, such that for all $s \in \Delta^*$, $(A_1 \oplus A_2)(s) = A_1(s) \oplus A_2(s)$ and $(A_1 \otimes A_2)(s) = A_1(s) \otimes A_2(s)$.*

Proof. The construction is essentially the same as in the case of the Boolean semiring [6]. ◀

complete proof

We shall now present a procedure for searching, for a sw-VPA A, a word of minimal weight for A, as stated in the following proposition.

▶ **Proposition 17.** *For a sw-VPA A over Δ, $\mathbb{S}$ commutative, bounded, total and complete, and $\bar{\Phi}$ effective, one can construct in PTIME a word $t \in \Delta^*$ such that $A(t)$ is minimal wrt the natural ordering for $\mathbb{S}$.*

Let $A = \langle Q, P, \mathsf{in}, \bar{\mathsf{w}}, \mathsf{out}\rangle$. We propose a Dijkstra algorithm computing, for every $q, q' \in Q$, the minimum, *wrt* $\leq_\oplus$, of the function $\beta_{q,q'} : t \mapsto \mathsf{weight}_A(q[\bot], t, q'[\bot])$. Let us denote by $b_\bot(q, q')$ this minimum. By definition of $\leq_\oplus$, and since $\mathbb{S}$ is total, it holds that:

$$b_\bot(q, q') = \bigoplus_{t \in \Delta^*} \mathsf{weight}_A\big(q[\bot], t, q'[\bot]\big). \tag{7}$$

Since $\mathbb{S}$ is complete, the infinite sum in (7) is well defined. Following (6), and the associativity and commutativity and distributivity for $\otimes$ and $\oplus$, the minimum of $A(t)$ is:

$$\bigoplus_{t \in \Delta^*} A(t) = \bigoplus_{t \in \Delta^*} \bigoplus_{q,q' \in Q} \mathsf{in}(q) \otimes \beta_{q,q'}(t) \otimes \mathsf{out}(q') = \bigoplus_{q,q' \in Q} \mathsf{in}(q) \otimes b_\bot(q, q') \otimes \mathsf{out}(q') \tag{8}$$

In order to compute the above function $b_\bot : Q \times Q \to \mathbb{S}$, we shall consider an auxiliary function $b_\top : Q \times P \times Q \to \Phi_{\mathsf{c}}$. Intuitively, $b_\top(q, p, q')$ is a function of Φ_{c}, mapping every $c \in \Delta_{\mathsf{c}}$ to the minimum weight of a computation of A starting in state q with a non-empty stack $\gamma' = \langle c, p\rangle\, \gamma \in \Gamma^+$, and ending in state q' with the same stack γ', such that moreover, the computation does not pop the pair $\langle c, p\rangle$ at the top of γ' (*i.e.* γ' is left untouched during the computation). However, the computation can read $\langle c, p\rangle$ at the top of γ', and can also push another pair $\langle c', p'\rangle \in \Gamma$ on γ', following the third case of in the definition (5) of weight_A (call symbol). The pair $\langle c', p'\rangle$ can be pop later, during the computation from q to q', following the fifth case of (5) (return symbol). Formally, in order to define $b_\top$, we consider a fresh stack symbol $\top \notin \Gamma$, representing the above untouched stack, and let:

$$b_\top(q, p, q') : c \mapsto \bigoplus_{s \in \Delta^*} \mathsf{weight}_A\Big(q \begin{bmatrix} \langle c, p\rangle \\ \top \end{bmatrix}, s, q' \begin{bmatrix} \langle c, p\rangle \\ \top \end{bmatrix}\Big) \quad \text{for all } c \in \Delta_{\mathsf{c}} \tag{9}$$

By definition of weight_A in (5), using the symbol $\top$ for the part of the stack below $\langle c, p\rangle$ (*i.e.* the substack γ in the above $\gamma' = \langle c, p\rangle\, \gamma$) ensures that this part γ' is not touched during the

Algorithm 1 Best search for sw-VPA

---

initially let $\mathcal{Q} = (Q \times Q) \cup (Q \times P \times Q)$, and let $d_\perp(q_1, q_2) = d_\top(q_1, p, q_2) = \mathbb{1}$ if $q_1 = q_2$ and $d_\perp(q_1, q_2) = d_\top(q_1, p, q_2) = \mathbb{0}$ otherwise

while $\mathcal{Q} \neq \emptyset$ **do**

  extract $\langle q_1, q_2\rangle$ or $\langle q_1, p, q_2\rangle$ from $\mathcal{Q}$ such that $d_\perp(q_1, q_2)$, resp. $\bigoplus_{c\in\Delta_\mathsf{c}} d_\top(q_1, p, q_2)(c)$, is minimal in $\mathbb{S}$ *wrt* $\leq_\oplus$

  update $d_\perp$ with $\langle q_1, q_2\rangle$ or $d_\top$ with $\langle q_1, p, q_2\rangle$ (Figure 3).

For all $q_0, q_3 \in Q$,

$$\begin{array}{lcl}
d_\top(q_1, p, q_3) & \oplus= & d_\top(q_1, p, q_2) \otimes \bigoplus_{\Delta_\mathsf{i}} \mathsf{w}_\mathsf{i}(q_2, p, q_3) \\
d_\perp(q_1, p, q_3) & \oplus= & d_\perp(q_1, q_2) \otimes \bigoplus_{\Delta_\mathsf{i}} \mathsf{w}^\mathsf{e}_\mathsf{i}(q_2, q_3) \\
d_\top(q_0, p, q_3) & \oplus= & \bigoplus_{\Delta_\mathsf{c}}^2 \big[\big(\mathsf{w}_\mathsf{c}(q_0, p, p', q_1) \otimes_2 d_\top(q_1, p', q_2)\big) \otimes_2 \bigoplus_{\Delta_\mathsf{r}} \mathsf{w}_\mathsf{r}(q_2, p', q_3)\big] \\
d_\perp(q_0, q_3) & \oplus= & \bigoplus_{\Delta_\mathsf{c}} \big(\mathsf{w}^\mathsf{e}_\mathsf{c}(q_0, p, q_1) \otimes d_\top(q_1, p, q_2) \otimes \bigoplus_{\Delta_\mathsf{r}} \mathsf{w}_\mathsf{r}(q_2, p, q_3)\big) \\
d_\perp(q_1, q_3) & \oplus= & d_\perp(q_1, q_2) \otimes \bigoplus_{\Delta_\mathsf{r}} \mathsf{w}^\mathsf{e}_\mathsf{r}(q_2, q_3) \\
d_\top(q_1, p, q_3) & \oplus= & d_\top(q_1, p, q_2) \otimes d_\top(q_2, p, q_3), \text{if } \langle q_2, \top, q_3\rangle \notin P \\
d_\perp(q_1, q_3) & \oplus= & d_\perp(q_1, q_2) \otimes d_\perp(q_2, q_3), \text{if } \langle q_2, \perp, q_3\rangle \notin P
\end{array}$$

Figure 3 Update $d_\perp$ with $\langle q_1, q_2\rangle$ or $d_\top$ with $\langle q_1, p, q_2\rangle$.

computation. This ensures in particular that the subword read during the computation is well parenthesized (every symbol in Δ_c has a matching symbol in Δ_r).

Algorithm 1 constructs iteratively two markings $d_\perp : Q \times Q \to \mathbb{S}$ and $d_\top : Q \times P \times Q \to \Phi_\mathsf{c}$, that converges eventually to $b_\top$ and $b_\perp$. The infinite sums in the updates of d in Algorithm 1, Figure 3 are well defined since $\mathbb{S}$ is complete.

explication Fig. 3 suivant cas de (5)

complete **

** effectively computable by hypothesis that the label theory is effective** termination: The algorithm performs $2.|Q|^2$ iterations until P is empty, and each iteration has a time complexity $O(|Q|^2.|P|)$. That gives a time complexity $O(|Q|^4.|P|)$. It can be reduced by implementing P as a priority queue, prioritized by the value returned by d. At termination, $d_\perp = b_\perp$ and $d_\top = b_\top$ (see Appendix B). With (8), this ensures the correctness of Algorithm 1. In order to obtain effectively a witness (word of Δ^* with a computation of A of minimal weight), we require the additional property of convexity of weight functions.***

detail with nb tr. and states

completer

▶ **Proposition 18.** *For a sw-VPA A over Δ, $\mathbb{S}$ commutative, bounded, total and complete, and $\bar{\Phi}$ effective, one can construct in PTIME a word $t \in \Delta^*$ such that $A(t)$ is minimal wrt the natural ordering for $\mathbb{S}$.*

5 Symbolic Weighted Parsing

Let us now apply the models and results of the previous sections to the problem of parsing over an infinite alphabet. Let Σ and $\Delta = \Delta_\mathsf{i} \uplus \Delta_\mathsf{c} \uplus \Delta_\mathsf{r}$ be countable input and output alphabets, let $\langle \mathbb{S}, \oplus, \mathbb{0}, \otimes, \mathbb{1}\rangle$ be a commutative, bounded, and complete semiring and let $\bar{\Phi}$

total?

be an effective label theory over $\mathbb{S}$, containing Φ_Σ, $\Phi_{\Sigma,\Delta_\mathsf{i}}$, as well as Φ_i, Φ_c, Φ_r, Φ_cr (following the notations of Section 4). We assume given the following input:

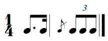
- a swT T over Σ, Δ_i, $\mathbb{S}$, and $\bar{\Phi}$, defining a measure $T : \Sigma^* \times {\Delta_\mathsf{i}}^* \to \mathbb{S}$,
- a sw-VPA A over Δ, $\mathbb{S}$, and $\bar{\Phi}$, defining a measure $A : \Delta^* \to \mathbb{S}$,
- an input word $s \in \Sigma^*$.

For all $u \in \Sigma^*$ and $t \in \Delta^*$, let $d(u,t) = T\big(u, t|_{\Delta_\mathsf{i}}\big)$, where $t|_{\Delta_\mathsf{i}} \in {\Delta_\mathsf{i}}^*$ is the projection of t onto Δ_i, obtained from t by removing all symbols in $\Delta \setminus \Delta_\mathsf{i}$. *Symbolic weighted parsing* is the problem, given the above input, to find $t \in \Delta^*$ minimizing $d(s,t) \otimes A(t)$ *wrt* $\leq_\oplus$, *i.e.* s.t.

$$d(s,t) \otimes A(t) = \bigoplus_{t' \in \Delta^*} d(s,t') \otimes A(t') \qquad (10)$$

Following the terminology of [21], sw-parsing is the problem of computing the distance (10) between the input s and the output weighted language of A, and returning a witness t.

▶ **Example 19** (Symbolic Weighted Parsing and the transcription problem)**.** Applied to the music transcription problem, the above formalism is interpreted as follows:

1. The swT T evaluates a "fitness measure" that expresses a correspondance between a performance and its notation. See Example **??**.
2. The sw-VPA A expresses a cost related to the music notation. One possibility is to relate this cost to the structural complexity: given a set of equivalent representations, we aim at choosing the simpler one. For instance  should be favored on , although both yield a same linearization. Costs can be expressed in the grammar with weight associated to each production rule. See Example 5.

Therefore, the framework, applied to the transcription problem, allows to find an optimal solution that considers both the fitness of the result to the input, and the complexity of the former.

▶ **Proposition 20.** *The problem of Symbolic Weighted parsing can be solved in PTIME in the size of the input* swT *T,* sw-VPA *A and input word s, and the computation time of the functions and operators of the label theory.*

Proof. (sketch) We follow a *Bar-Hillel* construction, for parsing by intersection. Let us first extend the swT T over Σ, Δ_i into a swT T' over Σ and Δ (and the same semiring and label theory $\mathbb{S}$ and $\bar{\Phi}$), such that for all $u \in \Sigma^*$, and $t \in \Delta^*$, $T'(u,t) = T(u, t|_{\Delta_\mathsf{i}})$. The transducer T' simply skips every symbol $b \in \Delta \setminus \Delta_\mathsf{i}$, by the addition to T, of new transitions of the form $\mathsf{w}_{01}(q, \varepsilon, b, q')$. Then, using Corolary 13, we construct from the input word $s \in \Sigma^*$ and T' a swA $B_{s,T'}$, such that for all $t \in \Delta^*$, $B_{s,T'}(t) = d(s,t)$. Next, we compute the sw-VPA $B_{s,T'} \otimes A$, using Proposition 16. It remains to compute a best nested-word $t \in \Delta^*$ using the best-search procedure of Proposition 18. ◀

The sw-parsing generalizes the problem of searching the best derivation (AST) of a weighted CF-grammar G that yields a given input word w. The latter problem, sometimes called *weighted parsing*, (see *e.g.* [13] and [23] for general weighted parsing frameworks) corresponds to sw-parsing in the case of finite alphabets, a transducer T computing the identity and some sw-VPA A obtained from G. Indeed, the *depth-first* traversal of an AST τ yields a well-parenthesised word $\mathsf{lin}(\tau)$ over an alphabet $\Delta = \Delta_\mathsf{i} \uplus \Delta_\mathsf{c} \uplus \Delta_\mathsf{r}$, assuming *e.g.* that Δ_i contains the symbols labelling the leaves of τ (symbols of rank 0), and Δ_c and Δ_r contain respectively one left and right parenthesis $\langle_b$ and $_b\rangle$ for each symbol b labelling inner nodes of τ (symbols of rank > 0). We show in Appendix A how to construct a sw-VPA A such that $A(\mathsf{lin}(\tau))$ is the weight the AST τ of G.

In practice, the functions in transition might be represented by diagram such as Algebraic Decision Diagrams [?].

sic Transcription: implementation $\neq$ but same principle, on-the-fly automata construction during best search, for efficiency.

Conclusion

We have introduced weighted language models (SW transducers and visibly pushdown automata) computing over infinite alphabets, and applied them to the problem of parsing with infinitely many possible input symbols (typically timed events). This approach extends conventional parsing and weighted parsing by computing a derivation tree modulo a generic distance between words, defined by a SW transducer given in input. This enables to consider finer word relationships than strict equality, opening possibilities of quantitative analysis via this method.

TODO future work

Ongoing and future work include

– The study of other theoretical properties of SW models, such as the extension of the best search algorithm from 1-best to n-best [17], and to k-*closed* semirings [20] (instead of *bounded*, which corresponds to 0-*closed*).

– ...there is room to improve the complexity bounds for the algorithms ... modular approach with oracles ...

– present here an offline algorithm for best search, semi-online implementation for AMT (bar-by-bar approach) with an on-the-fly automata construction.

References

1 R. Alur and P. Madhusudan. Adding nesting structure to words. *Journal of the ACM (JACM)*, 56(3):1–43, 2009.

2 M. Bojańczyk, C. David, A. Muscholl, T. Schwentick, and L. Segoufin. Two-variable logic on data words. *ACM Transactions on Computational Logic (TOCL)*, 12(4):1–26, 2011.

3 P. Bouyer, A. Petit, and D. Thérien. An algebraic approach to data languages and timed languages. *Information and Computation*, 182(2):137–162, 2003.

4 M. Caralp, P.-A. Reynier, and J.-M. Talbot. Visibly pushdown automata with multiplicities: finiteness and k-boundedness. In *International Conference on Developments in Language Theory*, pages 226–238. Springer, 2012.

5 H. Comon, M. Dauchet, R. Gilleron, F. Jacquemard, C. Löding, D. Lugiez, S. Tison, and M. Tommasi. *Tree Automata Techniques and Applications*. `http://tata.gforge.inria.fr`, 2007.

6 L. D'Antoni and R. Alur. Symbolic visibly pushdown automata. In *International Conference on Computer Aided Verification*, pages 209–225. Springer, 2014.

7 L. D'Antoni and M. Veanes. The power of symbolic automata and transducers. In *International Conference on Computer Aided Verification*, pages 47–67. Springer, 2017.

8 L. D'Antoni and M. Veanes. Automata modulo theories. *Communications of the ACM*, 64(5):86–95, 2021.

9 E. W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1(1):269–271, 1959.

10 M. Droste and W. Kuich. Semirings and formal power series. In *Handbook of Weighted Automata*, pages 3–28. Springer, 2009.

11 M. Droste, W. Kuich, and H. Vogler. *Handbook of weighted automata*. Springer Science & Business Media, 2009.

12 F. Foscarin, F. Jacquemard, P. Rigaux, and M. Sakai. A Parse-based Framework for Coupled Rhythm Quantization and Score Structuring. In *Mathematics and Computation in Music (MCM)*, volume 11502 of *Lecture Notes in Artificial Intelligence*, Madrid, Spain, 2019. Springer.

13 J. Goodman. Semiring parsing. *Computational Linguistics*, 25(4):573–606, 1999.

14 E. Gould. *Behind Bars: The Definitive Guide to Music Notation*. Faber Music, 2011.

15 D. Grune and C. J. Jacobs. *Parsing Techniques*. Number 2nd edition in Monographs in Computer Science. Springer, 2008.

16 L. Huang. Advanced dynamic programming in semiring and hypergraph frameworks. In *In COLING*, 2008.

17 L. Huang and D. Chiang. Better k-best parsing. In *Proceedings of the Ninth International Workshop on Parsing Technology*, Parsing '05, pages 53–64, Stroudsburg, PA, USA, 2005. Association for Computational Linguistics.

18 M. Kaminski and N. Francez. Finite-memory automata. *Theor. Comput. Sci.*, 134:329–363, November 1994.

19 S. Lombardy and J. Sakarovitch. The removal of weighted ε-transitions. In *International Conference on Implementation and Application of Automata*, pages 345–352. Springer, 2012.

20 M. Mohri. Semiring frameworks and algorithms for shortest-distance problems. *Journal of Automata, Languages and Combinatorics*, 7(3):321–350, 2002.

21 M. Mohri. Edit-distance of weighted automata: General definitions and algorithms. *International Journal of Foundations of Computer Science*, 14(06):957–982, 2003.

22 M. Mohri. Edit-distance of weighted automata: General definitions and algorithms. *International Journal of Foundations of Computer Science*, 14(06):957–982, 2003.

23 R. Mörbitz and H. Vogler. Weighted parsing for grammar-based language models. In *Proceedings of the 14th International Conference on Finite-State Methods and Natural Language Processing*, pages 46–55, Dresden, Germany, Sept. 2019. Association for Computational Linguistics.

24 M.-J. Nederhof. Weighted deductive parsing and Knuth's algorithm. *Computational Linguistics*, 29(1):135–143, 2003.

25 F. Neven, T. Schwentick, and V. Vianu. Finite state machines for strings over infinite alphabets. *ACM Trans. Comput. Logic*, 5(3):403–435, July 2004.

26 L. Segoufin. Automata and logics for words and trees over an infinite alphabet. In *Computer Science Logic*, volume 4207 of *LNCS*. Springer, 2006.

27 M. Y. Vardi. Linear-time model checking: automata theory in practice. In *International Conference on Implementation and Application of Automata*, pages 5–10. Springer, 2007.

A Nested-Words and Parse-Trees

The hierarchical structure of nested-words, defined with the *call* and *return* markup symbols suggest a correspondence with trees. The lifting of this correspondence to languages, of tree automata and VPA, has been discussed in [1], and [4] for the weighted case. In this section, we describe a correspondence between the symbolic-weighted extensions of tree automata and VPA.

Let Ω be a countable ranked alphabet, such that every symbol $a \in \Omega$ has a rank $\mathsf{rk}(a) \in [0..M]$ where M is a fixed natural number. We denote by Ω_k the subset of all symbols a of Ω with $\mathsf{rk}(a) = k$, where $0 \leq k \leq M$, and $\Omega_{>0} = \Omega \setminus \Omega_0$. The free Ω-algebra of finite, ordered, Ω-labeled trees is denoted by $\mathcal{T}_\Omega$. It is the smallest set such that $\Omega_0 \subset \mathcal{T}_\Omega$ and for all $1 \leq k \leq M$, all $a \in \Omega_k$, and all $t_1, \ldots, t_k \in \mathcal{T}_\Omega$, $a(t_1, \ldots, t_k) \in \mathcal{T}_\Omega$. Let us assume a commutative semiring $\mathbb{S}$ and a label theory $\bar{\Phi}$ over $\mathbb{S}$ containing one set Φ_{Ω_k} for each $k \in [0..M]$.

▶ **Definition 21.** *A symbolic-weighted tree automaton (swTA) over Ω, $\mathbb{S}$, and $\bar{\Phi}$ is a triplet $A = \langle Q, \mathsf{in}, \bar{\mathsf{w}} \rangle$ where Q is a finite set of states, $\mathsf{in} : Q \to \Phi_\Omega$ is the starting weight function, and $\bar{\mathsf{w}}$ is a tuplet of transition functions containing, for each $k \in [0..M]$, the functions $\mathsf{w}_k : Q \times Q^k \to \Phi_{\Omega_{>0}, \Omega_k}$ and $\mathsf{w}^{\mathsf{e}}_k : Q \times Q^k \to \Phi_{\Omega_k}$.*

We define a transition function $\mathsf{w} : Q \times (\Omega_{>0} \cup \{\varepsilon\}) \times \Omega \times \bigcup_{k=0}^{M} Q^k \to \mathbb{S}$ by:

$$\begin{array}{rcll} \mathsf{w}(q_0, a, b, q_1 \ldots q_k) & = & \eta(a, b) & \text{where } \eta = \mathsf{w}_k(q_0, q_1 \ldots q_k) \\ \mathsf{w}(q_0, \varepsilon, b, q_1 \ldots q_k) & = & \phi(b) & \text{where } \phi = \mathsf{w}^{\mathsf{e}}_k(q_0, q_1 \ldots q_k). \end{array}$$

where $q_1 \ldots q_k$ is ε if $k = 0$. The first case deals with a strict subtree, with a parent node labeled by a, and the second case is for a root tree.
Every swTA defines a mapping from trees of $\mathcal{T}_\Omega$ into $\mathbb{S}$, based on the following intermediate function $\mathsf{weight}_A : Q \times (\Omega \cup \{\varepsilon\}) \times \mathcal{T}_\Omega \to \mathbb{S}$

$$\mathsf{weight}_A(q_0, a, t) = \bigoplus_{q_1 \ldots q_k \in Q^k} \mathsf{w}(q_0, a, b, q_1 \ldots q_k) \otimes \bigotimes_{i=1}^{k} \mathsf{weight}_A(q_i, b, t_i) \tag{11}$$

where $q_0 \in Q$, $a \in \Omega_{>0} \cup \{\varepsilon\}$ and $t = b(t_1, \ldots, t_k) \in \mathcal{T}_\Omega$, $0 \leq k \leq M$.

Finally, the weight associated by A to $t \in \mathcal{T}_\Omega$ is

$$A(t) = \bigoplus_{q \in Q} \mathsf{in}(q) \otimes \mathsf{weight}_A(q, \varepsilon, t) \tag{12}$$

Intuitively, $\mathsf{w}(q_0, a, b, q_1 \ldots q_k)$ can be seen as the weight of a production rule $q_0 \to b(q_1, \ldots, q_k)$ of a regular tree grammar [5], that replaces the non-terminal symbol q_0 by $b(q_1, \ldots, q_k)$, provided that the parent of q_0 is labeled by a (or q_0 is the root node if $a = \varepsilon$). The above production rule can also be seen as a rule of a weighted CF grammar, of the form $[a, b]\, q_0 := q_1 \ldots q_k$ if $k > 0$, and $[a]\, q_0 := b$ if $k = 0$. In the first case, b is a label of the rule, and in the second case, it is a terminal symbol. And in both cases, a is a constraint on the label of rule applied on the parent node in the derivation tree. This features of observing the parent's label are useful in the case of infinite alphabet, where it is not possible to memorize a label with the states. The weight of a labeled derivation tree t of the weighted CF grammar associated to A as above, is $\mathsf{weight}_A(q, t)$, when q is the start non-terminal. We shall now establish a correspondence between such a derivation tree t and some word describing a linearization of t, in a way that $\mathsf{weight}_A(q, t)$ can be computed by a sw-VPA.

Let $\hat{\Omega}$ be the countable (unranked) alphabet obtained from Ω by: $\hat{\Omega} = \Delta_\mathsf{i} \uplus \Delta_\mathsf{c} \uplus \Delta_\mathsf{r}$, with $\Delta_\mathsf{i} = \Omega_0$, $\Delta_\mathsf{c} = \{ \langle_a | \; a \in \Omega_{>0}\}$, $\Delta_\mathsf{r} = \{ {}_a\rangle \mid a \in \Omega_{>0}\}$.
We associate to $\hat{\Omega}$ a label theory $\hat{\bar{\Phi}}$ like in Section 4, and we define a linearization of trees of $\mathcal{T}_\Omega$ into words of $\hat{\Omega}^*$ as follows:
$\mathsf{lin}(a) = a$ for all $a \in \Omega_0$,
$\mathsf{lin}\big(b(t_1, \ldots, t_k)\big) = \langle_b \, \mathsf{lin}(t_1) \ldots \mathsf{lin}(t_k) \, {}_b\rangle$ when $b \in \Omega_k$ for $1 \leq k \leq M$.

▶ **Proposition 22.** *For all swTA A over Ω, $\mathbb{S}$ commutative, and $\bar{\Phi}$, there exists an effectively constructible sw-VPA A' over $\hat{\Omega}$, $\mathbb{S}$ and $\hat{\bar{\Phi}}$ such that for all $t \in \mathcal{T}_\Omega$, $A'\big(\mathsf{lin}(t)\big) = A(t)$.*

Proof. Let $A = \langle Q, \mathsf{in}, \bar{\mathsf{w}}\rangle$ where $\bar{\mathsf{w}}$ is presented as above by a function We build $A' = \langle Q', P', \mathsf{in}', \bar{\mathsf{w}}', \mathsf{out}'\rangle$, where $Q' = \bigcup_{k=0}^{M} Q^k$ is the set of sequences of state symbols of A, of length at most M, including the empty sequence denoted by ε, and where $P' = Q'$ and $\bar{\mathsf{w}}$ is defined by:

$$
\begin{array}{lcll}
\mathsf{w_i}(q_0\,\bar{u}, \langle_c, \bar{p}, a, \bar{u}) & = & \mathsf{w}(q_0, c, a, \varepsilon) & \text{for all } c \in \Omega_{>0}, a \in \Omega_0\\
\mathsf{w_i^e}(q_0\,\bar{u}, a, \bar{u}) & = & \mathsf{w}(q_0, \varepsilon, a, \varepsilon) & \text{for all } a \in \Omega_0\\
\mathsf{w_c}(q_0\,\bar{u}, \langle_c, \bar{p}, \langle_d, \bar{u}, \bar{q}) & = & \mathsf{w}(q_0, c, d, \bar{q}) & \text{for all } c, d \in \Omega_{>0}\\
\mathsf{w_c^e}(q_0\,\bar{u}, \langle_c, \bar{u}, \bar{q}) & = & \mathsf{w}(q_0, \varepsilon, c, \bar{q}) & \text{for all } c \in \Omega_{>0}\\
\mathsf{w_r}(\varepsilon, \langle_c, \bar{p}, {}_c\rangle, \bar{p}) & = & \mathbb{1} & \text{for all } c \in \Omega_{>0}\\
\mathsf{w_r^e}(\bar{u}, {}_c\rangle, \bar{q}) & = & \mathbb{0} & \text{for all } c \in \Omega_{>0}
\end{array}
\quad (603)
$$

All cases not matched by one of the above equations have a weight $\mathbb{0}$, for instance $\mathsf{w_r}(\bar{u}, \langle_c, \bar{p}, {}_d\rangle, \bar{q}) = \mathbb{0}$ if $c \neq d$ or $\bar{u} \neq \varepsilon$ or $\bar{q} \neq \bar{p}$. ◀

B Correctness of the Best-Search Algorithm

The correctness of Algorithm 1 is stated by the following lemma.

▶ **Lemma 23.** *At the termination of Algorithm 1, for all $\langle q_1, q_2\rangle \notin \mathcal{Q}$, $d_\perp(q_1, q_2) = b_\perp(q_1, q_2)$.*

The proof is by contradiction, assuming a counter-example minimal in the length of the witness word.
is ensured by the invariant... expressed in the following lemma.

▶ **Lemma 24.** *At every step of Algorithm 1, for all $\langle q_1, p, q_2\rangle \notin \mathcal{Q}$, $d_\top(q_1, p, q_2) = b_\top(q_1, p, q_2)$.*

For computing the minimal weight of a computation of A, we use the equality (8), which, with Lemma 22 implies that at the termination of Algorithm 1, $\bigoplus_{s \in \Delta^*} A(s) = \bigoplus_{q,q' \in Q} \mathsf{in}(q) \otimes d_\perp(q, q') \otimes \mathsf{out}(q')$.

Todo list