

Symbolic Weighted Language Models and Parsing over Infinite Alphabets

Florent Jacquemard

INRIA & CNAM, Paris, France
florent.jacquemard@inria.fr

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 generalizes **sA**, whose transitions are guarded by Boolean predicates over symbols in an infinite alphabet, and also **wA**, whose transitions are labeled by constant weight values, and which deal only with finite alphabets. 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 illustrate the model with a motivating application to automated music transcription.

1 Introduction

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

The latter case is related to a motivation of the present work: automated music transcription. Representations that capture music performances are essentially linear: audio files or the widely used MIDI format [?]. 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 Common Western Music Notation [?]: 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 [?] over an infinite alphabet of timed events.

Various extensions of language models for handling infinite alphabets have been studied. Some automata with memory extensions allow restricted storage and comparison of input symbols (see [?] for a survey) register: skip refs and details, add Mikolaj recent using pebbles for marking positions [?], registers [?], or the possibility to compute on subsequences with the same attribute values [?]. The automata at the core of model checkers compute on input symbols represented by large bitvectors [?] (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 Formulas or Binary Decision Diagrams. Following a similar idea, in symbolic automata (sA) [?,?], transitions are guarded by predicates over infinite domains. With appropriate closure conditions on the sets of such predicates, all the good properties enjoyed over finite alphabets are preserved.

Other extensions of language models help in dealing with non-determinism by computing 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). WARNING: [?] much is more general than weighted parsing In *weighted language models* [?,?,?], like *e.g.* probabilistic context-free grammars and weighted automata (wA) [?], a weight is associated to each transition rule, and the rule's weights can be combined with an associative product operator \otimes to yield the weight of an AST. A second operator \oplus 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 selects the best of two weight values. The weight domain, equipped with these two operators is typically 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 framework for weighted parsing over infinite input alphabets. It is based on *symbolic weighted* finite states language models (SFL), generalizing the Boolean guards of sA to functions into an arbitrary semiring, and generalizing also wA, by handling infinite alphabets (Figure ??). In short, a transition rule $q \xrightarrow{\phi} q'$, from state q to q' , is labeled by a function ϕ associating to every input symbol a a weight value $\phi(a)$ in a semiring S .

The framework relies on several language models: symbolic-weighted automata (swA), transducers (swT), and pushdown automata with a visibility restriction [?] (sw-VPA). A swT defines a distance $T(s, t)$ between finite words s and t over infinite alphabets. A sw-VPA operates sequentially on *nested words* [?], structured with markup symbols (parentheses) and describing linearizations of trees. A sw-VPA A associates a weight value $A(t)$ to a given nested word t , which is itself the linearization of a weighted AST. Then, given an input word s , the *SW-parsing* problem aims at finding t minimizing $T(s, t) \otimes A(t)$, called the distance between s and A in [?], wrt the ranking defined by \oplus . Like weighted-parsing methods [?,?,?], WARNING: [?] much is more general than weighted parsing

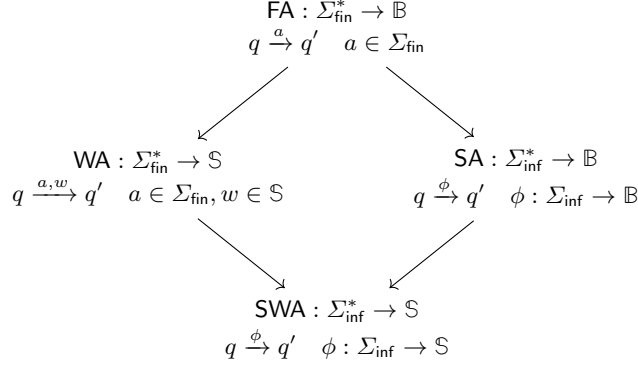



Fig. 1. Classes of Symbolic/Weighted Automata. Here, Σ_{fin} and Σ_{inf} denote finite/countable alphabets, \mathbb{B} the Boolean algebra, \mathbb{S} a commutative semiring. $q \xrightarrow{\cdot} q'$ is a transition between states q and q' .

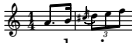
our approach proceeds in two steps. The first step is an intersection (Bar-Hillel construction [?]) chap. intersection in [?] where, given a **swT** T , a **sw-VPA** A , and an input word s , a **sw-VPA** B is built, such that for all t , $B(t) = T(s, t) \otimes A(t)$. In the second step, a best AST t is found by applying to B a best search algorithm similar to the shortest distance in graphs [?, ?].

expressiveness: VPA have restricted equality test. comparable to pebble automata? \rightarrow conclusion The main contributions of the paper are: (i) the introduction of automata, **swA**, transducers, **swT** (Section ??), and visibly pushdown automata **sw-VPA** (Section ??), generalizing the corresponding classes of symbolic and weighted models, (ii) a polynomial best-search algorithm for **sw-VPA**, and (iii) a uniform framework (Section ??) 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, for convenience in this context, of nested words, and **sw-VPA**, instead of syntax trees and grammars.

Example 1. We illustrate our framework with a very simplified running example of *music transcription*: a given input *timeline* of musical events from an infinite alphabet Σ , is parsed into a structured music score. Input events of Σ are pairs $\mu\tau$, where μ is a MIDI pitch [?], and $\tau \in \mathbb{Q}$ is a timestamp in seconds. Such inputs typically correspond to the recording of a live performance, e.g. $I = (69\ 0.07), (71\ 0.72), (73\ 0.91), (74\ 1.05), (76\ 1.36), (77\ 1.71)$.

The output of parsing is a sequence of timed symbols $\nu\tau'$ in an alphabet Δ , where ν represents an *event* (or *note*), specified by its *pitch name* (e.g., A4, G5, etc.), an event *continuation* (symbol ‘-’, see Example ??), or a *markup* (opening or closing parenthesis). The temporal information τ' is either a time interval, for the opening parentheses (representing the duration between the parenthesis and the matching closing one), or a timestamp, for the other symbols. The time points in τ' belong to a rhythmic “grid” obtained from recursive divisions: whole

notes (\circ) split in halves (\flat), halves in quarters (\flat), eights (\flat), *etc.* For instance, the output score , corresponds to a hierarchical structure that can be linearized as the sequence $O = m[0, 1], 2[0, 1], A40, 2[\frac{1}{2}, 1], -\frac{1}{2}, 2[\frac{3}{4}, 1], B4\frac{3}{4}, C\sharp5\frac{7}{8}, 21, 21, 21, m[1, 2], 3[1, 2], D51, E5\frac{4}{3}, F5\frac{5}{3}, 32, m2, m2$. The opening markups m delimit *measures*, which are time intervals of duration 1 in this example, while the subsequences of O between markups d and d , for some natural number d , represent a division of the current time interval into d sub-intervals of equal duration $\frac{\ell}{d}$ where ℓ is the length of the time interval attached to d .

We will show that O is a solution for the parsing of I . Note that several other parsings are possible like *e.g.* . SW-parsing associates a weight value to each solution, and our framework aims at selecting the best one with respect to this weight. \diamond

2 Preliminary Notions

notations: for set $S : S^*$ and S^+ . interval $[i..j]$ of natural numbers

2.1 Semirings

Semirings. A *semiring* $\langle \mathbb{S}, \oplus, \otimes, \mathbb{0}, \mathbb{1} \rangle$ is a structure with a domain \mathbb{S} , equipped with two associative binary operators \oplus and \otimes , with respective neutral elements $\mathbb{0}$ and $\mathbb{1}$, 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 : $\forall x \in \mathbb{S}, \mathbb{0} \otimes x = x \otimes \mathbb{0} = \mathbb{0}$.

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

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

We thus say the \mathbb{S} is *monotonic wrt \leq_\oplus* . Another important semiring property in the context of optimization is superiority [?], which corresponds to the *non-negative weights* condition in shortest-path algorithms [?]. Intuitively, it means that combining elements with \otimes always increases their weight. Formally, it is defined as the property (i) below.

Lemma 2 (Superiority, Boundedness). *Let $\langle \mathbb{S}, \oplus, \emptyset, \otimes, \mathbb{1} \rangle$ be an idempotent semiring. The two following statements are equivalent:*

- i. *for every $x, y \in \mathbb{S}$, $x \leq_{\oplus} x \otimes y$ and $y \leq_{\oplus} x \otimes y$*
- ii. *for every $x \in \mathbb{S}$, $\mathbb{1} \oplus x = \mathbb{1}$.*

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 [?], when property (i) holds, \mathbb{S} is called *superior wrt \leq_{\oplus}* . It implies (proof of Lemma ??) that $\mathbb{1} \leq_{\oplus} x$ for every $x \in \mathbb{S}$. Similarly, by the first inequality of (i) with $y = \emptyset$, $x \leq_{\oplus} x \otimes \emptyset = \emptyset$. Hence, in a superior semiring, for every $x \in \mathbb{S}$, $\mathbb{1} \leq_{\oplus} x \leq_{\oplus} \emptyset$. From an optimization point of view, it means that $\mathbb{1}$ is the best value, and \emptyset the worst. In [?], \mathbb{S} with the property (ii) of Lemma ?? 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 every $x \in \mathbb{S}$ and $n \geq 1$, $x \oplus x^n = x \otimes (\mathbb{1} \oplus x^{n-1}) = x$.

Lemma 3 ([?], Lemma 3). *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.

We need to extend \oplus to *infinitely many operands*. A semiring \mathbb{S} is called *complete* [?] if it has an operation $\bigoplus_{i \in I} x_i$ for every family $(x_i)_{i \in I}$ of elements of $\text{dom}(\mathbb{S})$ over an index set $I \subseteq \mathbb{N}$, such that:

i. *infinite sums extend finite sums:*

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

ii. *associativity and commutativity:*

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

iii. *distributivity of products over infinite sums:*

$$\text{for every } I \subseteq \mathbb{N}, \quad \bigoplus_{i \in I} (x \otimes y_i) = x \otimes \bigoplus_{i \in I} y_i, \quad \text{and} \quad \bigoplus_{i \in I} (x_i \otimes y) = \left(\bigoplus_{i \in I} x_i \right) \otimes y.$$

2.2 Label Theory

Label theory. We now define the functions labeling the transitions of SW automata and transducers, generalizing the Boolean algebras of [?]. We consider *alphabets*, which are countable sets of symbols denoted by Σ, Δ, \dots . Moreover, Σ^* is the set of finite sequences (*words*) over Σ , ε the empty word, $\Sigma^+ = \Sigma^* \setminus \{\varepsilon\}$, and uv denotes the concatenation of $u, v \in \Sigma^*$.

| | domain | \oplus | \otimes | $\mathbb{0}$ | $\mathbb{1}$ |
|-------------------|--------------------------------|----------|-----------|--------------|--------------|
| Boolean | $\{\perp, \top\}$ | \vee | \wedge | \perp | \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 |

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

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 by Φ_Σ , containing unary functions of type $\Sigma \rightarrow \mathbb{S}$, or $\Phi_{\Sigma, \Delta}$, containing binary functions $\Sigma \times \Delta \rightarrow \mathbb{S}$, and such that: 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?

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

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

We assume that when the underlying semiring \mathbb{S} is complete, for all $\Phi_{\Sigma, \Delta} \in \bar{\Phi}$ and all $\eta \in \Phi_{\Sigma, \Delta}$, $\bigoplus_\Sigma^1 \eta \in \Phi_\Delta$ and $\bigoplus_\Delta^2 \eta \in \Phi_\Sigma$.

Example 2. We return to Example ???. Let Δ_i be the subset of Δ without markup symbols. In order to align the input in Σ^* with a music score, we must account for the expressive timing of human performance that results in small time shifts between an input event of Σ and the corresponding notation event in Δ_i . These shifts can be weighted as the time distance between both, computed in the tropical semiring by $\delta \in \Phi_{\Sigma, \Delta_i}$, defined as follows:

$$\delta(\mu \tau, \nu \tau') = \begin{cases} |\tau' - \tau| & \text{if } \nu \text{ corresponds to } \mu, \\ 0 & \text{otherwise} \end{cases}$$

I et O déjà donnés dans ex.?? The distance between I and O is the aggregation with \otimes of the pairwise differences between the timestamps. In the tropical semiring, this yields $|0.07-0|+|0.72-\frac{3}{4}|+|0.91-\frac{7}{8}|+|1.05-1|+|1.36-\frac{4}{3}|+|1.71-\frac{5}{3}| = 0.255$. \diamond

We will need guarantees on the computability of the above infinite sum operators.

Definition 1. 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 , and moreover, the number of symbols reaching these bounds is finite and can be effectively computed.

3 SW Automata and Transducers

We follow the approach of [?] 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 [?] by labelling 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 [?,?], but they can return values in an 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 $\bar{\Phi} = \langle \Phi_\Sigma, \Phi_\Delta, \Phi_{\Sigma,\Delta} \rangle$ be a label theory over \mathbb{S} .

Definition 2. A symbolic-weighted transducer (*swT*) over Σ , Δ , \mathbb{S} and $\bar{\Phi}$ is a tuple $T = \langle Q, \text{in}, \bar{\mathbf{w}}, \text{out} \rangle$, where Q is a finite set of states, $\text{in} : Q \rightarrow \Phi_{\Sigma,\Delta}$, respectively $\text{out} : Q \rightarrow \Phi_{\Sigma,\Delta}$, are functions defining the weight for entering, respectively leaving, a state, and $\bar{\mathbf{w}}$ is a triplet of transition functions \mathbf{w}_{10} , \mathbf{w}_{01} , and \mathbf{w}_{11} from $Q \times Q$ into $\Phi_{\Sigma,\Delta}$.

For convenience, when $\mathbf{w}_{ij}(q, q') = \eta \in \Phi_{\Sigma,\Delta}$, for $q, q' \in Q$ and $i, j \in \{0, 1\} \setminus \{\langle 0, 0 \rangle\}$, respectively $\text{in}(q) = \eta$, $\text{out}(q') = \eta$, we shall sometimes write, overloading the function names: $\mathbf{w}_{ij}(q, a, b, q') = \eta(a, b)$ for $a \in \Sigma$, $b \in \Delta$, respectively $\text{in}(q, a, b) = \eta(a, b)$, $\text{out}(q', a, b) = \eta(a, b)$.

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, the symbolic-weighted transducer T defines a mapping from the pairs of strings of $\Sigma^+ \times \Delta^+$ into \mathbb{S} ,

based on the following intermediate function weight_T defined recursively for every $q, q' \in Q$, $a \in \Sigma$, $u \in \Sigma^*$, $b \in \Delta$, $v \in \Delta^*$,

$$\begin{aligned} \text{weight}_T(q, au, bv, q') = & \bigoplus_{q'' \in Q, u \neq \epsilon} w_{10}(q, a, b, q'') \otimes \text{weight}_T(q'', u, bv, q') \\ & \oplus \bigoplus_{q'' \in Q, v \neq \epsilon} w_{01}(q, a, b, q'') \otimes \text{weight}_T(q'', au, v, q') \\ & \oplus \bigoplus_{q'' \in Q, u, v \neq \epsilon} w_{11}(q, a, b, q'') \otimes \text{weight}_T(q'', u, v, q') \\ & \oplus \bigoplus_{u, v = \epsilon} w_{11}(q, a, b, q') \end{aligned} \quad (1)$$

We recall that, by convention (Section ??), an empty sum with \bigoplus is equal to $\mathbb{0}$. Intuitively, using a transition $w_{ij}(q, a, b, q')$ 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 $i = 1$, and in the output word if $j = 1$ (otherwise, do not change it), and change state from q to q' . In contrast with the models of weighted transducers over finite alphabets [?], the input and output symbols at current positions are always read by transitions, even when they do not change the reading position the head's position. This is an important feature in the case of an infinite alphabet in order to compare input and output symbols.

Since $\mathbb{0}$ is absorbing for \otimes in \mathbb{S} , one term $w_{ij}(q, a, b, q')$ 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 of weight_T 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 [?]. Intuitively, $w_{10}(q, a, b, r)$ is the cost of the deletion of the symbol $a \in \Sigma$ in s , $w_{01}(q, a, b, r)$ is the cost of the insertion of $b \in \Delta$ in t , and $w_{11}(q, a, b, r)$ 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 such product of costs.

Let $\langle s, t \rangle \in \Sigma^+ \times \Delta^+$, with $s = s_1 \dots s_n$, and $t = t_1 \dots t_m$. The weight associated by T to $\langle s, t \rangle$ is defined as follows:

$$T(s, t) = \bigoplus_{q, q' \in Q} \text{in}(q, s_1, t_1) \otimes \text{weight}_T(q, s, t, q') \otimes \text{out}(q', s_n, t_m) \quad (2)$$

Example 3. We can define with a **swT** the computation of a similarity measure between timed sequences similar to dynamic time warping (DTW).

Let \mathbb{S} be the tropical (*min-plus*) semiring of Figure ?? and let $\Sigma = \Delta = \mathbb{R}_+$ be sets of timestamps. We consider a **swT** with one state q and transitions $w_{11}(q, d, d', q) = w_{01}(q, d, d', q) = w_{10}(q, d, d', q) = |d' - d|$, for all $d, d' \in \mathbb{R}_+$.

The recursive definition of weight_T correspond to the dynamic programming equations of DTW for the computation of an optimal match between words, the matching cost for two symbols being the the time distance between them. \diamond

Example 4. Let us consider the tropical (*min-plus*) semiring \mathbb{S} of Figure ?? and let $\Sigma = \mathbb{R}_+$ be an input alphabet of event dates and $\Delta = \{\mathbf{e}, -\} \times \mathbb{R}_+$ be an output alphabet of symbols with timestamps. A symbol $\langle \mathbf{e}, d \rangle \in \Delta$ represents an event starting at date d , and $\langle -, d \rangle$ is a continuation of the previous event. This example of Δ is motivated by the case of music notation, where several notated events (notes) can be tied together, with a *tie* or a *dot* (like in $\text{♪} \text{—} \text{♪}$ or equivalently ♪), meaning that they will be played as a unique sounding event.

We consider a swT with two states q_0 and q_1 whose purpose is to compare a recorded performance $s \in \Sigma^*$ with notated music sheet $t \in \Delta^*$. One timestamp $d_i \in \Sigma$ may corresponds to one notated event $\langle \mathbf{e}, d'_i \rangle \in \Sigma$, in which case the weight value computed by the swT is the time distance between both (see transitions w_{11} below). If $\langle \mathbf{e}, d'_i \rangle$ is followed by continuations $\langle -, d'_{i+1} \rangle \dots$, they are just skip with no cost (transitions w_{01} or weight $\mathbb{1}$).

$$\begin{aligned} w_{11}(q_0, d, \langle \mathbf{e}, d' \rangle, q_0) &= |d' - d| & w_{11}(q_1, d, \langle \mathbf{e}, d' \rangle, q_0) &= |d' - d| \\ w_{01}(q_0, d, \langle -, d' \rangle, q_0) &= \mathbb{1} & w_{01}(q_1, d, \langle -, d' \rangle, q_0) &= \mathbb{1} \\ w_{10}(q_0, d, b, q_1) &= \alpha & & \text{for all } b \in \Delta \end{aligned}$$

Moreover, it may happen that the performers plays an extra note accidentally, but only once in a row. 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 $\text{in}(q_0, d, b) = \text{out}(q_0, d, b) = \mathbb{1}$, and $\text{in}(q_1, d, b) = \text{out}(q_1, d, b) = \mathbb{0}$, for all $d \in \Sigma$ and $b \in \Delta$. \diamond

The *Symbolic Weighted Automata* are defined similarly as the transducers of Definition ??, by simply omitting the output symbols. In this case, the label theory $\bar{\Phi}$ can be reduced to a singleton $\langle \Phi_\Sigma \rangle$.

Definition 3. A symbolic-weighted automaton (*swA*) over Σ, \mathbb{S} and $\bar{\Phi}$ is a tuple $A = \langle Q, \text{in}, w_1, \text{out} \rangle$, where Q is a finite set of states, $\text{in} : Q \rightarrow \Phi_\Sigma$, respectively $\text{out} : Q \rightarrow \Phi_\Sigma$, are functions defining the weight for entering, respectively leaving, a state, and w_1 is a transition functions from $Q \times Q$ into Φ_Σ .

As above in the case of swT, when $w_1(q, q') = \phi \in \Phi_\Sigma$, respectively $\text{in}(q) = \phi$, $\text{out}(q') = \phi$, we may write $w_1(q, a, q')$ for $\phi(a)$, respectively $\text{in}(q, a) = \phi(a)$, $\text{out}(q', a) = \phi(a)$. The computation of A on words $s \in \Sigma^+$ is defined with an intermediate function weight_A , defined as follows for $q, q' \in Q, a \in \Sigma, u \in \Sigma^*$,

$$\begin{aligned} \text{weight}_A(q, au, q') &= \bigoplus_{r \in Q, u \neq \epsilon} w_1(q, a, r) \otimes \text{weight}_A(r, u, q') \\ &\oplus \bigoplus_{u = \epsilon} w_1(q, a, q') \end{aligned} \tag{3}$$

and the weight value associated by A to $s = s_1 \dots s_n \in \Sigma^+$ is:

$$A(s) = \bigoplus_{q, q' \in Q} \text{in}(q, s_1) \otimes \text{weight}_A(q, s, q') \otimes \text{out}(q', s_n) \quad (4)$$

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

Proposition 1. *Given a swT T over Σ , Δ , \mathbb{S} and $\bar{\Phi}$, and $s \in \Sigma^+$, there exists an effectively constructible swA $A_{T,s}$ over Δ , \mathbb{S} and $\bar{\Phi}$, such that for all $t \in \Delta^+$, $A_{T,s}(t) = T(s, t)$.*

Proof. Let $T = \langle Q, \text{in}, \bar{w}, \text{out} \rangle$, let $w : Q \times \Sigma \times \Delta \times Q \rightarrow \mathbb{S}$ be the synthesized form of $\bar{w} = \langle w_{10}, w_{01}, w_{11} \rangle$, and let $s = s_1 \dots s_n$ with $s_1, \dots, s_n \in \Sigma$ ($n \geq 1$).

The state set of $A_{T,s}$ will be $Q' = [1..n+1] \times Q$. Its state entering weight function is defined by, for all $q \in Q$ and $b \in \Delta$, $\text{in}'(\langle 1, q \rangle, b) = \text{in}(q, s_1, b)$ and $\text{in}'(\langle i, q \rangle, b) = \emptyset$ for all $1 < i \leq n+1$. Its state leaving weight function is defined by, for all $q \in Q$ and $b \in \Delta$, $\text{out}'(\langle n+1, q \rangle, b) = \text{out}(q, s_n, b)$, and $\text{out}'(\langle i, q \rangle, b) = \emptyset$ for all $1 \leq i < n+1$.

Every non-null transition of $A_{T,s}$ will simulate a sequence of transitions of T which advance in the input word while staying immobile in the output word, and then make one step in the output word (and advance in the input word or not). The above initial sequence correspond to ϵ -transitions of automata, and its total weight is computed by the following intermediate function $w'_0 : Q' \times Q' \rightarrow \Phi_\Sigma$, defined for all $q, q' \in Q$: by:

$$\begin{aligned} w'_0(\langle i, q \rangle, \langle i, q \rangle) &= 1 && \text{if } 1 \leq i \leq n+1, \\ w'_0(\langle i, q \rangle, \langle i, q' \rangle) &= \emptyset && \text{if } 1 \leq i \leq n+1 \text{ and } q \neq q', \\ w'_0(\langle i, q \rangle, \langle i+k, q' \rangle) &= \bigoplus_{\substack{q_0, \dots, q_k \in Q \\ q_0 = q, q_k = q'}} \bigotimes_{j=0}^{k-1} [w_{10}(q_j, q_{j+1})]_{s_{i+j}} && \begin{array}{l} \text{if } 1 \leq i < n \\ \text{and } 1 \leq k \leq n-i. \end{array} \end{aligned}$$

where $[w_{10}(q_j, q_{j+1})]_{s_{i+j}}$ is the partial application $\eta_{s_{i+j}} \in \Phi_\Delta$ for $\eta = w_{10}(q_j, q_{j+1}) \in \Phi_{\Sigma, \Delta}$ ($s_{i+j} \in \Sigma$). The function w'_0 is defined thanks to the closure properties of the label theory $\bar{\Phi}$ (Section ??). The sum and product in its definition are finite, we shall see below how to compute the first sum in polynomial time.

We define now the transition function $w'_1 : Q' \times Q' \rightarrow \Phi_\Sigma$ of $A_{T,s}$ as follows, for $q, q' \in Q$, $1 \leq i \leq n$, and $0 \leq k \leq n-i$:

$$\begin{aligned} w'_1(\langle i, q \rangle, \langle i, q' \rangle) &= w_{01}(q, q') \\ w'_1(\langle i, q \rangle, \langle i+k, q' \rangle) &= \bigoplus_{\substack{q'' \in Q \\ i+k < n}} w'_0(\langle i, q \rangle, \langle i+k, q'' \rangle) \otimes \eta_{s_{i+k}} \quad \text{where } \eta = w_{01}(q'', q'), \\ &\quad \bigoplus_{q'' \in Q} w'_0(\langle i, q \rangle, \langle i+k-1, q'' \rangle) \otimes \eta'_{s_{i+k}} \quad \text{where } \eta' = w_{11}(q'', q'), \\ w'_1(\langle i, q \rangle, \langle j, q' \rangle) &= \emptyset \quad \text{if } j < i. \end{aligned}$$

We can show that the swA $A_{T,s} = \langle Q', \text{in}', w'_1, \text{out}' \rangle$ has the expected property:
 $\forall t \in \Delta^+, A_{T,s}(t) = T(s, t)$. \square

On the quadratic computation of w'_0 . ** by a best path search in the graph with nodes $[i..i+k] \times Q$ and edges labeled in $\Phi_{\Sigma} \dots$ ordering = summary.
 The construction time and size for $A_{T,s}$ are $O(\|T\| \cdot |s|)$, where the size $\|T\|$ of T is its number of states $|Q|$.

4 SW Visibly Pushdown Automata

The model presented in this section generalizes Symbolic VPA [?] 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. Nest words are able to describe structures of labeled trees, and in the context of parsing, they will be useful to represent parse trees.

4.1 Definition

Let Ω be a countable alphabet that we assume partitioned into three subsets $\Omega_i, \Omega_c, \Omega_r$, whose elements are respectively called *internal*, *call* and *return* symbols. Let $\langle \mathbb{S}, \oplus, \emptyset, \otimes, \mathbb{1} \rangle$ be a commutative and complete semiring and let $\bar{\Phi} = \langle \Phi_i, \Phi_c, \Phi_r, \Phi_{ci}, \Phi_{cc}, \Phi_{cr} \rangle$ be a label theory over \mathbb{S} where Φ_i, Φ_c, Φ_r and Φ_{cx} (with $x \in \{i, c, r\}$) stand respectively for $\Phi_{\Omega_i}, \Phi_{\Omega_c}, \Phi_{\Omega_r}$ and $\Phi_{\Omega_c, \Omega_x}$.

Definition 4. A Symbolic Weighted Visibly Pushdown Automata (sw-VPA) over $\Omega = \Omega_i \uplus \Omega_c \uplus \Omega_r$, \mathbb{S} and $\bar{\Phi}$ is a tuple $A = \langle Q, P, \text{in}, \bar{w}, \text{out} \rangle$, where Q is a finite set of states, P is a finite set of stack symbols, $\text{in} : Q \rightarrow \mathbb{S}$, respectively $\text{out} : Q \rightarrow \mathbb{S}$, are functions defining the weight for entering, respectively leaving, a state, and \bar{w} is a sextuplet composed of the transition functions: $w_i : Q \times P \times Q \rightarrow \Phi_{ci}$, $w_i^e : Q \times Q \rightarrow \Phi_i$, $w_c : Q \times P \times Q \times P \rightarrow \Phi_{cc}$, $w_c^e : Q \times P \times Q \rightarrow \Phi_c$, $w_r : Q \times P \times Q \rightarrow \Phi_{cr}$, $w_r^e : Q \times Q \rightarrow \Phi_r$.

Similarly as in Section ??, we extend the above transition functions as follows for all $q, q' \in Q$, $p \in P$, $a \in \Omega_i$, $c \in \Omega_c$, $r \in \Omega_r$, overloading their names:

$$\begin{array}{lll}
 w_i : Q \times \Omega_c \times P \times \Omega_i \times Q \rightarrow \mathbb{S} & w_i(q, c, p, a, q') = \eta_{ci}(c, a) & \text{where } \eta_{ci} = w_i(q, p, q'), \\
 w_i^e : Q \times \Omega_i \times Q \rightarrow \mathbb{S} & w_i^e(q, a, q') = \phi_i(a) & \text{where } \phi_i = w_i^e(q, q'), \\
 w_c : Q \times \Omega_c \times P \times \Omega_c \times P \times Q \rightarrow \mathbb{S} & w_c(q, c, p, c', p', q') = \eta_{cc}(c, c') & \text{where } \eta_{cc} = w_c(q, p, p', q'), \\
 w_c^e : Q \times \Omega_c \times P \times Q \rightarrow \mathbb{S} & w_c^e(q, c, p, q') = \phi_c(c) & \text{where } \phi_c = w_c^e(q, p, q'), \\
 w_r : Q \times \Omega_c \times P \times \Omega_r \times Q \rightarrow \mathbb{S} & w_r(q, c, p, r, q') = \eta_{cr}(c, r) & \text{where } \eta_{cr} = w_r(q, p, q'), \\
 w_r^e : Q \times \Omega_r \times Q \rightarrow \mathbb{S} & w_r^e(q, r, q') = \phi_r(r) & \text{where } \phi_r = w_r^e(q, q').
 \end{array}$$

The intuition is the following for the above transitions.

w_i and w_i^e 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 $c \in \Omega_c$ and $p \in P$

at the top of the stack (c is compared to a by the weight function $\eta_{ci} \in \Phi_{ci}$) and w_i^e applies if and only if the stack is empty.

w_c and w_c^e read the input call symbol c' , push it to the stack along with p' , and change state from q to q' . Moreover, w_c reads c and p at the top of the stack (c is compared to c'), and w_c^e applies iff the stack is empty.

w_r and w_r^e read the input return symbol r , and change state from q to q' . Moreover, w_r reads and pop from stack a pair made of c and p , (c is compared to r), and w_r^e applies iff the stack is empty.

Formally, the transitions of the automaton A are defined in term of an intermediate function weight_A , like in Section ???. In the case of a pushdown automaton, a configuration is composed of a state $q \in Q$ and a stack content $\gamma \in \Gamma^*$, where $\Gamma = \Omega_c \times P$. Hence, weight_A is a function from $[Q \times \Gamma^*] \times \Omega^* \times [Q \times \Gamma^*]$ into \mathbb{S} .

$$\begin{aligned}
\text{weight}_A([c p \cdot \gamma], a u, [q']) &= \bigoplus_{q'' \in Q} w_i(q, c, p, a, q'') \otimes \text{weight}_A([q''], u, [q']) \\
\text{weight}_A([\perp], a u, [q']) &= \bigoplus_{q'' \in Q} w_i^e(q, a, q'') \otimes \text{weight}_A([q''], u, [q']) \\
\text{weight}_A([c p \cdot \gamma], c' u, [q']) &= \bigoplus_{\substack{q'' \in Q \\ p \in P}} w_c(q, c, p, c', p', q'') \otimes \text{weight}_A([c' p' \cdot c p \cdot \gamma], u, [q']) \\
\text{weight}_A([\perp], c u, [q']) &= \bigoplus_{q'' \in Q} w_c^e(q, c, p, q'') \otimes \text{weight}_A([c p], u, [q']) \\
\text{weight}_A([c p \cdot \gamma], r u, [q']) &= \bigoplus_{q'' \in Q} w_r(q, c, p, r, q'') \otimes \text{weight}_A([q''], u, [q']) \\
\text{weight}_A([\perp], r u, [q']) &= \bigoplus_{q'' \in Q} w_r^e(q, r, q'') \otimes \text{weight}_A([q''], u, [q'])
\end{aligned} \tag{5}$$

where \perp denotes the empty stack and $c p \cdot \gamma$ denotes a stack where the pair made of $c \in \Omega_c$ and $p \in P$ is the top symbol and γ is the rest of stack.

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

$$A(s) = \bigoplus_{q, q' \in Q} \text{in}(q) \otimes \text{weight}_A([\perp], s, [q']) \otimes \text{out}(q'). \tag{6}$$

Example 5. structured words with timed symbols... intro language of music notation? (markup = time division, leaves = events etc)

4.2 Properties

Like VPA and symbolic VPA, the class of **sw-VPA** is closed under the binary operators of the underlying semiring.

Proposition 2. *Let A_1 and A_2 be two (sw-VPA) over the same Ω , \mathbb{S} and $\bar{\Phi}$. There exists two sw-VPA $A_1 \oplus A_2$ and $A_1 \otimes A_2$, effectively constructible, such that for all $s \in \Omega^*$, $(A_1 \oplus A_2)(s) = A_1(s) \oplus A_2(s)$ and $(A_1 \otimes A_2)(s) = A_1(s) \otimes A_2(s)$.*

The construction is essentially the same as in the case of the Boolean semiring [?].

4.3 Best Search

Let us assume that the semiring \mathbb{S} is commutative, bounded, total, and complete. and assume an effective label theory.

We propose a Dijkstra algorithm computing, for a **sw-VPA** A over Ω , \mathbb{S} and $\bar{\Phi}$, the minimal weight $wrt \leq_{\oplus}$, for a word in Ω^* .

More precisely, let $b_{\perp} : Q \times Q \rightarrow \mathbb{S}$ be the function:

$$b_{\perp}(q, q') = \bigoplus_{s \in \Omega^*} \text{weight}_A\left(\begin{bmatrix} q \\ \perp \end{bmatrix}, s, \begin{bmatrix} q' \\ \perp \end{bmatrix}\right) \quad (7)$$

Since \mathbb{S} is complete, the infinite sum in (??) is well defined, and by totality of \leq_{\oplus} , it is the minimum in Ω^* of the function $s \mapsto \text{weight}_A\left(\begin{bmatrix} q \\ \perp \end{bmatrix}, s, \begin{bmatrix} q' \\ \perp \end{bmatrix}\right)$ wrt this ordering. The term $\begin{bmatrix} q \\ \perp \end{bmatrix}, s, \begin{bmatrix} q' \\ \perp \end{bmatrix}$ of this sum is the central expression in the definition (??) of $A(s_0)$, for the minimum s_0 of the function weight_A .

Let \top be a fresh stack symbol which does not belong to Γ , and let $b_{\top} : Q \times P \times Q \rightarrow \Phi_c$ be such that, for every two states $q, q' \in Q$ and stack symbol $p \in P$:

$$b_{\top}(q, p, q') : c \mapsto \bigoplus_{s \in \Omega^*} \text{weight}_A\left(\begin{bmatrix} q \\ c p \cdot \top \end{bmatrix}, s, \begin{bmatrix} q' \\ c p \cdot \top \end{bmatrix}\right). \quad (8)$$

Intuitively, the function defined in (??) associateds to $c \in \Omega_c$ the minimum weight of a computation of A starting in state q with a stack $cp \cdot \gamma \in \Gamma^+$ and ending in state q' with the same stack, such that the computation does pop the pair made of c and p at the top of this stack, but may read these symbols. Moreover, A may push another pair $\langle c', p' \rangle$ on the top of $cp \cdot \gamma$, following the the third case of in the definition (??) of weight_A , and may pop $\langle c', p' \rangle$ later, following the fifth case of (??) (return symbol).

Algorithm ?? constructs iteratively markings $d_{\perp} : Q \times Q \rightarrow \mathbb{S}$ and $d_{\top} : Q \times P \times Q \rightarrow \Phi_c$ that converges eventually to b_{\top} and b_{\perp} .

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} is not empty

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. $\bigoplus_{c \in \Omega_c} d_{\top}(q_1, p, q_2)(c)$, 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 ??).

The infinite sums in the updates of d in Algorithm ??, Figure ?? are well defined since \mathbb{S} is complete. ** effectively computable by hypothese that the label theory is effective** The algorithm performs $2 \cdot |Q|^2$ iterations until P is empty, and each iteration has a time complexity $O(|Q|^2 \cdot |P|)$. This gives a time complexity $O(|Q|^4 \cdot |P|)$. It can be reduced by implementing P as a priority queue, prioritized by the value returned by d ***complete***.

The correctness of Algorithm ?? is ensured by the invariant expressed in the following lemma.

For all $q_0, q_3 \in Q$,

$$\begin{aligned}
d_{\top}(q_1, p, q_3) \oplus &= d_{\top}(q_1, p, q_2) \otimes \bigoplus_{a \in \Omega_i} w_i(q_2, p, q_3)_a \\
&\text{where } w_i(q_2, p, q_3)_a \in \Phi_c \text{ is the partial application } x_c \mapsto w_i(q_2, x_c, p, a, q_3) \\
d_{\perp}(q_1, p, q_3) \oplus &= d_{\perp}(q_1, q_2) \otimes \bigoplus_{a \in \Omega_i} w_e(q_2, a, q_3) \\
d_{\top}(q_0, p, q_3) \oplus &= \bigoplus_{c' \in \Omega_c} \bigoplus_{r \in \Omega_r} w_c(q_0, p, p', q_1)_{c'} \otimes d_{\top}(q_1, p', q_2) \otimes w_r(q_2, c', p', r, q_3) \\
&\text{where } w_c(q_0, p, p', q_1)_{c'} \in \Phi_c \text{ is the partial application } x_c \mapsto w_c(q_0, x_c, p, c', p', q_1) \\
d_{\perp}(q_0, q_3) \oplus &= \bigoplus_{c \in \Omega_c} \bigoplus_{r \in \Omega_r} w_c^e(q_0, c, p, q_1) \otimes d_{\top}(q_1, p, q_2)(c) \otimes w_r(q_2, c, p, r, q_3) \\
d_{\perp}(q_1, q_3) \oplus &= d_{\perp}(q_1, q_2) \otimes \bigoplus_{r \in \Omega_r} w_r^e(q_2, r, q_3) \\
d_{\top}(q_1, p, q_3) \oplus &= d_{\top}(q_1, p, q_2) \otimes d_{\top}(q_2, p, q_3), \text{ if } \langle q_2, \top, q_3 \rangle \notin P \\
d_{\perp}(q_1, q_3) \oplus &= d_{\perp}(q_1, q_2) \otimes d_{\perp}(q_2, q_3), \text{ if } \langle q_2, \perp, q_3 \rangle \notin P
\end{aligned}$$

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

Lemma 4. 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.

Lemma 5. 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 fact that, at the termination of Algorithm ??,

$$\bigoplus_{s \in \Omega^*} A(s) = \bigoplus_{q, q' \in Q} \text{in}(q) \otimes d_{\perp}(q, q') \otimes \text{out}(q').$$

In order to obtain effectively a witness (word of \mathbb{S}^* with computation of A of minimal weight), we require an additional property the of weight functions.

Definition 5. Let Ω be an alphabet and \mathbb{S} a complete semiring. A function ϕ from Ω^n into \mathbb{S} is called k -convex for a natural number k iff $\text{card}\{\mathbf{a} \in \Omega^n \mid \phi(\mathbf{a}) = \bigoplus_{\mathbf{p} \in \Omega^n} \phi(\mathbf{p})\} \leq k$.

A label theory is k -convex if all its functions are k -convex.

Proposition 3. For a sw-VPA A over Ω , \mathbb{S} commutative, bounded, total and complete, and $\bar{\Phi}$ k -convex effective, one can construct in PTIME a word $s \in \Omega^*$ such that $A(s)$ is minimal wrt the natural ordering for \mathbb{S} .

4.4 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 [?], and [?] 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 $\text{rk}(a) \in [0..M]$ where M is a fixed natural number. We denote by Ω_k the subset of all symbols a of Ω with $\text{rk}(a) = k$, 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 $t_1, \dots, t_k \in \mathcal{T}(\Omega)$, $a(t_1, \dots, t_k) \in \mathcal{T}(\Omega)$. 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 $k \in [0..M]$.

Let $\hat{\Omega}$ be the countable (unranked) alphabet obtained from Ω by: $\hat{\Omega} = \Omega_i \uplus \Omega_c \uplus \Omega_r$, with $\Omega_i = \Omega_0$, $\Omega_c = \{ \langle a \mid a \in \Omega_{>0} \rangle \}$, $\Omega_r = \{ \langle a \rangle \mid a \in \Omega_{>0} \}$. We associate to $\hat{\Omega}$ a label theory $\hat{\Phi}$ like in Section ??.

We define a linearization of trees of $\mathcal{T}(\Omega)$ into words of $\hat{\Omega}^*$ as follows:

$$\begin{aligned} \text{lin}(a) &= a \text{ for all } a \in \Omega_0, \\ \text{lin}(b(t_1, \dots, t_k)) &= \langle_b \text{lin}(t_1) \dots \text{lin}(t_k) \rangle_b \text{ when } b \in \Omega_k, 1 \leq k \leq M. \end{aligned}$$

Definition 6. A symbolic-weighted tree automaton (*swTA*) over Ω , \mathbb{S} , and $\bar{\Phi}$ is a triplet $A = \langle Q, \text{in}, \bar{w} \rangle$ where Q is a finite set of states, $\text{in} : Q \rightarrow \mathbb{S}$ is the starting weight function, and \bar{w} is a tuple of transition functions containing $w_k : Q \times Q^k \rightarrow \Phi_{\Omega_k}$ for each $k \in [0..M]$.

We define from \bar{w} a transition function w , from $Q \times \Omega \times \bigcup_{k=0}^M Q^k$ into \mathbb{S} by:

$$w(q_0, a, q_1 \dots q_k) = \phi_{\Omega, k}(a) \quad \text{where } \phi_k = w_k(q_0, q_1 \dots q_k).$$

Intuitively, $w(q_0, a, q_1 \dots q_k)$ can be seen as the weight of a production rule $q_0 \rightarrow a(q_1, \dots, q_k)$ of a regular tree grammar [?], that replaces the non-terminal symbol q_0 by $a(q_1, \dots, q_k)$. Such a grammar computes the weights of the derivation trees of the Context-Free grammar obtained by forgetting the labeling symbols of $\Omega_{>0}$. The swTA of Definition ?? defines a mapping from trees of $\mathcal{T}(\Omega)$ into the weights of \mathbb{S} , based on the intermediate function weight_A defined as follows for $q_0 \in Q$ and $t = b(t_1, \dots, t_k) \in \mathcal{T}(\Omega)$, with $0 \leq k \leq M$:

$$\text{weight}_A(q_0, t) = \bigoplus_{q_1 \dots q_k \in Q^k} w(q_0, b, q_1 \dots q_k) \otimes \bigotimes_{i=1}^k \text{weight}_A(q_i, t_i) \quad (9)$$

The weight associated by A to $t \in \mathcal{T}(\Omega)$ is

$$A(t) = \bigoplus_{q \in Q} \text{in}(q) \otimes \text{weight}_A(q, t) \quad (10)$$

Lemma 6. *For all swTA A over Ω , \mathbb{S} commutative, and Φ , there exists an effectively constructible sw-VPA A' over $\hat{\Omega}$, \mathbb{S} and $\hat{\Phi}$ such that for all $t \in \mathcal{T}(\Omega)$, $A'(\text{lin}(t)) = A(t)$.*

Proof. Let $A = \langle Q, \text{in}, \bar{w} \rangle$ where \bar{w} is presented as above by a function. We build $A' = \langle Q', P', \text{in}', \bar{w}', \text{out}' \rangle$, 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{w}' is defined by:

$$\begin{aligned} w_i(\bar{q}, \langle b, \bar{p}, a, \bar{q}q' \rangle) &= w(q', a, \epsilon) && \text{for all } b \in \Omega_{>0}, \bar{p} \in P', a \in \Omega_0 \\ w_i^e(\bar{q}, a, \bar{q}q') &= w(q', a, \epsilon) && \text{for all } a \in \Omega_0 \\ w_c(\bar{q}, \langle b, \bar{p}, \langle b', \epsilon, \bar{q} \rangle \rangle) &= 1 && \text{for all } b \in \Omega_{>0}, \bar{p} \in P', b, b' \in \Omega_{>0} \\ w_c^e(\bar{q}, \langle b, \epsilon, \bar{q} \rangle) &= 1 && \text{for all } b \in \Omega_{>0} \\ w_r(\bar{q}, \langle b, \bar{p}, b \rangle, \bar{p}q') &= w(q', b, \bar{q}) && \text{for all } b \in \Omega_{>0}, \bar{p} \in P' \\ w_r^e(\bar{p}, b, \bar{q}) &= 0 && \text{for all } b \in \Omega_{>0} \end{aligned}$$

It is sufficient to consider in Q' only the prefixes of sequences in transition with a non-null weight. \square

5 Symbolic Weighted Parsing

Let us now apply the models and results of the previous sections to the problem of parsing over infinite alphabet. Besides considering infinitely many possible of input symbols, handled with suitable language formalisms, 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 considering finer word relationships than strict equality as in the conventional parsing approach, opening possibilities of quantitative analysis via this method.

5.1 Definition

Let Σ be a countable input alphabet and let Ω be a countable output ranked alphabet, with maximal rank value M , and $\hat{\Omega} = \Omega_i \uplus \Omega_c \uplus \Omega_r$ be the alphabet with nesting symbols, associated like Section ??, for the linearization of trees of $\mathcal{T}(\Omega)$ (remember that $\Omega_i = \Omega_0$). Let $\langle \mathbb{S}, \oplus, 0, \otimes, 1 \rangle$ be a commutative, bounded, complete and total semiring and let $\bar{\Phi}$ be a label theory over \mathbb{S} containing Φ_Σ , Φ_{Σ, Ω_i} , as well as Φ_i , Φ_c , Φ_r , Φ_{cr} (following the notations of Section ??). It is moreover assumed computable and k -convex for some fixed k .

Let us assume given the following input:

- a swT T over Σ , Ω_i , \mathbb{S} , and $\bar{\Phi}$, defining a measure $T : \Sigma^* \times \Omega_i^* \rightarrow \mathbb{S}$,
- a swTA A over Ω , \mathbb{S} , and $\bar{\Phi}$, defining a measure $A : \mathcal{T}(\Omega) \rightarrow \mathbb{S}$,
- an input word $s \in \Sigma^*$.

As explained in Section ??, $A = \langle Q, \text{in}, \bar{w} \rangle$ can be seen as a weighted regular tree grammar, that generates (weighted) trees by replacement of a state symbol q_0 (non-terminal), by a tree $a(q_1, \dots, q_k)$, where $k = \text{rk}(a)$. A replacement rule $q_0 \rightarrow a(q_1, \dots, q_k)$, of weight $w(q_0, a, q_1 \dots q_k) \in \mathbb{S}$ according to Definition ??, corresponds to the production rule $q_0 := a(q_1, \dots, q_k)$ of a weighted CF grammar, with set non-terminal symbols Q and set of terminal symbols Ω_0 . This actually is a slight generalization of CFG since each such production rule is labelled by a symbol of $\Omega_{>0}$, hence parse trees are trees of $\mathcal{T}(\Omega)$. Another (more original) generalization is that the set of terminal symbols Ω_0 may be infinite.

We extend the measure defined by T to $d : \Sigma^* \times \mathcal{T}(\Omega) \rightarrow \mathbb{S}$ as follows. Given a word $w \in \hat{\Omega}^*$, the projection of w onto Ω_i , denoted $w|_{\Omega_i}$, is the word of Ω_i^* obtained from w by removing all symbols in $\hat{\Omega} \setminus \Omega_i$. Using this notation and the tree linearization operator defined in Section ??, d is defined by:

$$d(s, t) = T(s, \text{lin}(t)|_{\Omega_i}) \text{ for } s \in \Sigma^*, t \in \mathcal{T}(\Omega) \quad (11)$$

Symbolic weighted parsing is the problem, given the above input, to find a tree $t \in \mathcal{T}(\Omega)$ minimizing $d(s, t) \otimes A(t)$ wrt \leq_{\oplus} , i.e. such that:

$$d(s, t) \otimes A(t) = \bigoplus_{t' \in \mathcal{T}(\Omega)} d(s, t') \otimes A(t') \quad (12)$$

The measure expressed in (??) is called edit-distance between s and A in [?]. The problem of searching, in a WTA language, the best parse tree matching a given input, sometimes referred as *weighted parsing* corresponds to SW parsing in the case of finite alphabets and when the transducer T characterizes identity see e.g. [?] and [?] for a more general weighted parsing framework.

5.2 Computation

Proposition 4. *The problem of Symbolic Weighted parsing can be solved in PTIME in the size of the input $\text{swT } T$, $\text{swTA } A$ and input word s , and the computation time of the functions of the label theory.*

Proof. (sketch) We follow a *Bar-Hillel* construction, also called parsing by intersection.

We first extend the $\text{swT } T$ over Σ , Ω_i , \mathbb{S} , and $\bar{\Phi}$, into a $\text{swT } T'$ over Σ and $\hat{\Omega}$ (and the same semiring and label theory), such that for all $s \in \Sigma^*$, and $u \in \hat{\Omega}^*$, $T'(s, u) = T(s, u|_{\Omega_i})$. The transducer T' simply skips every symbol $b \in \hat{\Omega} \setminus \Omega_i$ in input, with new transitions of the form $w_{01}(q, \epsilon, b, q')$.

Then, given an input word $s \in \Sigma^*$, we compute the $\text{swA } A_{T',s}$, using Proposition ???. This automaton is such that for all $t \in \mathcal{T}(\Omega)$, $A_{T',s}(\text{lin}(t)) = T'(s, \text{lin}(t)) = T'(s, \text{lin}(t)|_{\Omega_i}) = d(s, t)$.

Next, we convert the input $\text{swTA } A$ over Ω into a $\text{sw-VPA } A'$ over $\hat{\Omega}$, using Lemma ??, and we compute the $\text{sw-VPA } A_{T',s} \otimes A'$, using Proposition ??.

It remains to compute a best nested-word $w \in \hat{\Omega}^*$ using the best-search procedure of Proposition ??, and convert it into a best tree in $\mathcal{T}(\Omega)$ in order to solve SW parsing for T , A and s . \square

5.3 Application to Automated Music Transcription

Conclusion

- summary
- other theoretical properties of SW models
- room to improve complexity for best-search algorithm ... modular approach with oracles ...
 - and extention to n -best
- offline algorithm, semi-online implementation for AMT (bar-by-bar approach)