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 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.

1 Introduction

Parsing is the problem of structuring a linear representation in input (a finite word over an alphabet) according to a language model. Most of the context-free parsing approaches [14] assume a finite and reasonably small input alphabet. Such a restriction makes perfectly sense in the context of NLP tasks like constituency parsing or of programming languages compilers or interpreters. Considering large or infinite alphabets can however be of practical interest in other cases. For instance, when dealing with large characters encodings such as UTF-16, *e.g.* for vulnerability detection in Web-applications [8], for the analyse (*e.g.* validation or filtering) of data streams or serialization of structured documents (which may contain textual or numerical attributes) [24], or for processing timed execution traces [3]. Regarding the latter case, we briefly mention at the end of the paper a parse-based approach to automated music transcription [12] that motivated the present work. In this problem, a symbolic music performance, presented as a sequence of timed musical events, is converted into a structured score in Common Western Music Notation.

Various extensions of language models for handling infinite alphabets have been studied. For instance, some automata with memory extensions allow restricted storage and comparison of input symbols, (see [24] for a survey), with

pebbles for marking positions [23], registers [17], 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 [25] (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 closure conditions on the sets of such predicates, all the good properties enjoyed by automata over finite alphabets are preserved.

Other extensions of language models help in dealing with non-determinism, by the computation of weight values. With an ambiguous grammar, there may exists several derivations (*abstract syntax trees* – AST), yielding one input word. The association of one weight value to each derivation permits to select a best one (or n bests). This is roughly the principle of *weighted parsing* approaches [13,22,21]. 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 a associative product operator $\otimes$ into the weight of a derivation. A second operator $\oplus$, associative and commutative, is moreover used to handle ambiguity of the model, by summing the weights of the possibly 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 is assumed, at minima, to form a *semiring* where $\oplus$ can be extended to infinite sums, such as the Viterbi semiring and the tropical min-plus algebra, see Figure 2.

In this paper, we present a uniform framework for weighted parsing over infinite input alphabets. It is based on weighted language models generalizing both sA, with functions in an arbitrary semiring instead of Boolean guards, and wA, by handling infinite alphabets – Figure 1. In their transition rules, input symbols appear as variables and the weight associated to a transition rule is a function of these variables. The models presented here are finite automata called symbolic-weighted (swA), transducers (swT) and pushdown automata, with a visibly restriction [1] (sw-VPA). The latter model of automata computes on *nested words* [1], a structured form of words parenthesized with markup symbols, corresponding to a linearization of trees. In the context of parsing, they are used here to represent (weighted) AST of CF grammars. More precisely, a sw-VPA A associates a weight value $A(t)$ to a given a nested word t, which is the linearization of an AST. On the other hand, a swT is used to define a distance $T(s,t)$ between two finite words s and t over an infinite alphabet, following [19]. Then, the *SW-parsing* problem aims at finding t minimizing $T(s,t) \otimes A(t)$ (*wrt* the ranking defined by $\oplus$) – this value is called the distance between s and A in [19]. Similarly to weighted-parsing methods [13,22,21], our approach proceeds in two steps, based on properties of the SW models. The first step is a Bar-Hillel construction where, given a swT T, a sw-VPA A, and an input word s, a sw-VPA $A_{T,s}$ is built, such that for all t, $A_{T,s}(t) = T(s,t) \otimes A(t)$. In the second step,

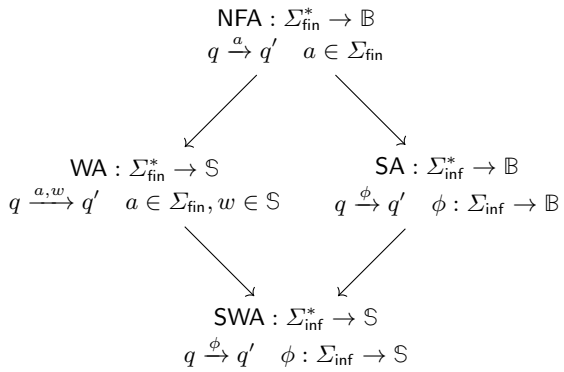

Fig. 1. Classes of Symbolic/Weighted Automata. Σ_{fin} is a finite alphabet, Σ_{inf} is a countable alphabet, $\mathbb{B}$ is the Boolean algebra, $\mathbb{S}$ is an arbitrary commutative semiring, $q \xrightarrow{\cdots} q'$ represents the form of a transition between states q and q'.

a best AST t is found by applying to $A_{T,s}$ a best search algorithm similar to shortest distance in graphs [18,16].

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 words, and (*iii.b*) the use of nested words, and sw-VPA, instead of syntax trees and grammars.

2 Preliminary Notions

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

2.1 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, in a chaining of transitions.

A semiring $\mathbb{S}$ is *commutative* if $\otimes$ is commutative. It is *idempotent* if for each $x \in dom(\mathbb{S})$, $x \oplus x = x$. Every idempotent semiring $\mathbb{S}$ induces a partial ordering $\leq_\oplus$ called the *natural ordering* of $\mathbb{S}$ [18] and defined, by: for all x and y, $x \leq_\oplus y$ 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).

Lemma 1 (Monotony, [18]). *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 $z \otimes x \leq_\oplus z \otimes y$.*

When the property of Lemma 1 holds, $\mathbb{S}$ is called *monotonic*. Another important semiring property in the context of optimization is superiority [15], 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 2 (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 [15], the property (i) is called $\mathbb{S}$ *superior wrt* the ordering $\leq_\oplus$. We have seen in the proof of Lemma 2 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 [18], the property (ii) of Lemma 2 is called *boundedness* of $\mathbb{S}$ – 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.

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. □

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 $x \oplus y = y$.

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 the following holds:

	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

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

i. infinite sums extend finite sums:
$\bigoplus_{i\in\emptyset} x_i = \mathbb{0}$, $\forall j \in \mathbb{N}, \bigoplus_{i\in\{j\}} x_i = x_j$, $\forall j,k \in \mathbb{N}, j \neq k, \bigoplus_{i\in\{j,k\}} x_i = x_j \oplus x_k$,
ii. associativity and commutativity:
for all $I \subseteq \mathbb{N}$ and all partition $(I_j)_{j\in J}$ of I, $\bigoplus_{j\in J}\bigoplus_{i\in I_j} x_i = \bigoplus_{i\in I} x_i$,
iii. distributivity of product over infinite sum:
for all $I \subseteq \mathbb{N}$, $\bigoplus_{i\in I}(x \otimes y_i) = x \otimes \bigoplus_{i\in I} y_i$, and $\bigoplus_{i\in I}(x_i \otimes y) = (\bigoplus_{i\in I} x_i) \otimes y$.

Example 1. Figure 2 presents examples of semirings interesting in practice and enjoying the above properties.

2.2 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 tuplet $\bar{\Phi}$ of recursively enumerable sets denoted Φ_Σ, containing unary functions of type $\Sigma \to \mathbb{S}$, and $\Phi_{\Sigma,\Delta}$, containing binary functions $\Sigma \times \Delta \to \mathbb{S}$, and such that:
– for all $\Phi_{\Sigma,\Delta} \in \bar{\Phi}$, we have $\Phi_\Sigma \in \bar{\Phi}$ and $\Phi_\Delta \in \bar{\Phi}$
– every Φ_Σ 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 $\oplus$
– the partial applications $\eta \in \Phi_{\Sigma,\Delta}$ and $\eta_a : y \mapsto \eta(a,y)$ for $a \in \Sigma$ and $\eta_b : x \mapsto \eta(x,b)$ for $b \in \Delta$ belong respectively to Φ_Δ and Φ_Σ. partial appli needed?

In what follows, we shall omit the subscripts in $\otimes_1$, $\otimes_2$, $\oplus_1$, $\oplus_2$ and reserve them to the special case $\Sigma = \Delta$, *i.e.* $\eta \in \Phi_{\Sigma,\Sigma}$ for $\otimes_1$ above, and $\eta \in \Phi_{\Delta,\Delta}$ for $\otimes_2$.

When the semiring $\mathbb{S}$ is complete, let us consider the following operators on the functions of a label theory (we use overloading to simplify notations):

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

Similarly as for the notation of product and sum of functions above, the superscripts in $\bigoplus^1_\Sigma$ and $\bigoplus^2_\Sigma$ shall be reserved to the ambiguous case of $\Phi_{\Sigma,\Sigma}$, in order to to distinguish between the first and the second argument. A label theory $\bar\Phi$ is called *complete* when the underlying semiring $\mathbb{S}$ is complete, and for all $\Phi_{\Sigma,\Delta} \in \bar\Phi$ and all $\eta \in \Phi_{\Sigma,\Delta}$, $\bigoplus_\Sigma \eta \in \Phi_\Delta$ and $\bigoplus_\Delta \eta \in \Phi_\Sigma$.
The following facts are immediate.

Lemma 4. *For $\bar\Phi$ complete $\alpha \in \mathbb{S}$, $\phi, \phi' \in \Phi_\Sigma$, $\psi \in \Phi_\Delta$, and $\eta \in \Phi_{\Sigma,\Delta}$:*

i. $\bigoplus_\Sigma \bigoplus_\Delta \eta = \bigoplus_\Delta \bigoplus_\Sigma \eta$
ii. $\alpha \otimes \bigoplus_\Sigma \phi = \bigoplus_\Sigma(\alpha \otimes \phi)$ *and* $\big(\bigoplus_\Sigma \phi\big) \otimes \alpha = \bigoplus_\Sigma(\phi \otimes \alpha)$, *and similarly for* $\oplus$
iii. $\big(\bigoplus_\Sigma \phi\big) \oplus \big(\bigoplus_\Sigma \phi'\big) = \bigoplus_\Sigma(\phi \oplus \phi')$ *and* $\big(\bigoplus_\Sigma \phi\big) \otimes \big(\bigoplus_\Sigma \phi'\big) = \bigoplus_\Sigma(\phi \otimes \phi')$
iv. $\big(\bigoplus_\Delta \eta\big) \oplus \big(\bigoplus_\Delta \eta'\big) = \bigoplus_\Delta(\eta \oplus \eta')$, *and* $\big(\bigoplus_\Delta \eta\big) \otimes \big(\bigoplus_\Delta \eta'\big) = \bigoplus_\Delta(\eta \otimes \eta')$
v. $\phi \otimes \big(\bigoplus_\Delta \eta\big) = \bigoplus_\Delta(\phi \otimes \eta)$, *and* $\big(\bigoplus_\Delta \eta\big) \otimes \phi = \bigoplus_\Delta(\eta \otimes \phi)$, *and simil. for* $\oplus$
vi. $\psi \otimes \big(\bigoplus_\Sigma \eta\big) = \bigoplus_\Sigma(\psi \otimes \eta)$, *and* $\big(\bigoplus_\Sigma \eta\big) \otimes \psi = \bigoplus_\Sigma(\eta \otimes \psi)$, *and simil. for* $\oplus$.

Intuitively, the operators $\bigoplus_\Sigma$ return global minimum, *wrt* $\leq_\oplus$, of functions of $\bar\Phi$.
A label theory is called *effective* when for all $\phi \in \Phi_\Sigma$ and $\eta \in \Phi_{\Sigma,\Delta}$, $\bigoplus_\Sigma \phi$, $\bigoplus_\Sigma \eta$, and $\bigoplus_\Delta \eta$ can be effectively computed from ϕ and η.

3 SW Automata and Transducers

We follow the approach of [19] 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 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 [7,8], 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 1. *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, 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 $\mathsf{w}_{11} : Q \times Q \to \Phi_{\Sigma,\Delta}$.*

For convenience, we shall the following presentation of transition functions 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 $q, q' \in Q$, $a \in \Sigma$, $b \in \Delta$,

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

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 $q, q' \in Q$, $a \in \Sigma$, $u \in \Sigma^*$, $b \in \Delta$, $v \in \Delta^*$,

$$\begin{aligned} \mathsf{weight}_T(q, \varepsilon, \varepsilon, q) &= \mathsf{out}(q) \qquad (1) \\ \mathsf{weight}_T(q, \varepsilon, \varepsilon, q') &= \mathbb{0} \quad \text{if } q \neq q' \\ \mathsf{weight}_T(q, au, \varepsilon, q') &= \bigoplus_{q'' \in Q} \mathsf{w}_{10}(q, a, \varepsilon, q'') \otimes \mathsf{weight}_T(q'', u, \varepsilon, q') \\ \mathsf{weight}_T(q, \varepsilon, bv, q') &= \bigoplus_{q'' \in Q} \mathsf{w}_{01}(q, \varepsilon, b, q'') \otimes \mathsf{weight}_T(q'', \varepsilon, v, q') \\ \mathsf{weight}_T(q, au, bv, q') &= \bigoplus_{q'' \in Q} \mathsf{w}_{10}(q, a, \varepsilon, q'') \otimes \mathsf{weight}_T(q'', u, bv, q') \\ &\quad \oplus \bigoplus_{q'' \in Q} \mathsf{w}_{01}(q, \varepsilon, b, q'') \otimes \mathsf{weight}_T(q'', au, v, q') \\ &\quad \oplus \bigoplus_{q'' \in Q} \mathsf{w}_{11}(q, a, b, q'') \otimes \mathsf{weight}_T(q'', u, v, q') \end{aligned}$$

We recall that, by convention (Section 2.1), 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 $j = 1$ (otherwise, do not change it), 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 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 [20]. Intuitively, $\mathsf{w}_{10}(q, a, \varepsilon, r)$ is the cost of the deletion of the symbol $a \in \Sigma$ in s, $\mathsf{w01}(q, \varepsilon, b, r)$ 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 such product of costs.

The weight associated by T to $\langle s,t\rangle \in \Sigma^* \times \Delta^*$ is defined as follows:

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

Example 2. Let us consider the tropical (*min-plus*) semiring $\mathbb{S}$ of Figure 2 and let $\Sigma = \mathbb{R}_+$ be an input alphabet of event dates and $\Delta = \{\mathsf{e}, -\} \times \mathbb{R}_+$ be an output alphabet of symbols with timestamps. A symbol $\langle \mathsf{e}, d\rangle \in \Delta$ 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 ♩‿♪ 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 $\langle \mathsf{e}, d'_i\rangle \in \Sigma$, in which case the weight value computed by the swT is the time distance between both (see transitions w_{11} below). If $\langle \mathsf{e}, d'_i\rangle$ is followed by continuations $\langle -, d'_{i+1}\rangle$..., they are just skip with no cost (transitions w_{01} or weight $\mathbb{1}$).

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

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 $\mathsf{in}(q_0) = \mathsf{out}(q_0) = \mathbb{1}$, and $\mathsf{in}(q_1) = \mathsf{out}(q_1) = \mathbb{0}$. ◇

The *Symbolic Weighted Automata* are defined similarly as the transducers of Definition 1, 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 2. *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, a state, and w_1 is a transition functions 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 $q, q' \in Q$, $a \in \Sigma$, $u \in \Sigma^*$,

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

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

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

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

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 $T = \langle Q, \mathsf{in}, \bar{\mathsf{w}}, \mathsf{out}\rangle$, with $\bar{\mathsf{w}} = \langle \mathsf{w}_{10}, \mathsf{w}_{01}, \mathsf{w}_{11}\rangle$, 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 $q \in Q$ $\mathsf{in}'(1,q) = \mathsf{in}(q)$ and $\mathsf{in}'(i,q) = \mathbb{0}$ for all $1 < i \leq n+1$. Its state leaving weight function is defined by, for all $q \in Q$ $\mathsf{out}'(n+1,q) = \mathsf{out}(q)$, and $\mathsf{out}'(i,q) = \mathbb{0}$ for all $1 \leq i < n+1$.

Every non-null transition of $A_{T,s}$ will simulate a sequence of transitions of T performing the following steps: 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 first steps in the sequence correspond to ε-transitions of automata, and their total weight is computed by the following intermediate function $\mathsf{w}'_0 : Q' \times Q' \to \mathbb{S}$, defined for all $q, q' \in Q$: by:

$$\begin{aligned}
\mathsf{w}'_0\big(\langle i,q\rangle, \langle i,q\rangle\big) &= \mathbb{1} \quad \text{if } 1 \leq i \leq n+1,\\
\mathsf{w}'_0\big(\langle i,q\rangle, \langle i,q'\rangle\big) &= \mathbb{0} \quad \text{if } 1 \leq i \leq n+1 \text{ and } q \neq q',\\
\mathsf{w}'_0\big(\langle i,q\rangle, \langle i+k,q'\rangle\big) &= \bigoplus_{\substack{q_0,\dots,q_k \in Q\\ q_0 = q, q_k = q'}} \bigotimes_{j=0}^{k-1} \phi_{j,j+1}(s_{i+j}) \quad \text{if } 1 \leq i < n, \text{ and } 1 \leq k \leq n-i,\\
&\qquad\qquad \text{where } \phi_{j,j+1} = \mathsf{w}_{10}(q_j, q_{j+1})
\end{aligned}$$

The function w'_0 is defined thanks to the closure properties of the label theory $\bar{\Phi}$ (Section 2.2). 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 $\mathsf{w}'_1 : Q' \times Q' \to \Phi_\Sigma$ of $A_{T,s}$ as follows, for $q, q' \in Q$, $1 \leq i \leq n$, and $0 \leq k \leq n-i$:

$$\begin{aligned}
\mathsf{w}'_1\big(\langle i,q\rangle, \langle i,q'\rangle\big) &= \mathsf{w}_{01}(q,q')\\
\mathsf{w}'_1\big(\langle i,q\rangle, \langle i+k,q'\rangle\big) &= \bigoplus_{\substack{q'' \in Q\\ i+k<n}} \mathsf{w}'_0\big(\langle i,q\rangle, \langle i+k,q''\rangle\big) \otimes \psi \quad \text{where } \psi = \mathsf{w}_{01}(q'',q'),\\
&\quad \oplus \bigoplus_{q'' \in Q} \mathsf{w}'_0\big(\langle i,q\rangle, \langle i+k-1,q''\rangle\big) \otimes \eta_{s_{i+k}} \quad \text{where } \eta = \mathsf{w}_{11}(q'',q'),\\
\mathsf{w}'_1\big(\langle i,q\rangle, \langle j,q'\rangle\big) &= \mathbb{0} \quad \text{if } j < i.
\end{aligned}$$

In the second equation, $\eta_{s_{i+j}} \in \Phi_\Delta$ is a partial application for $\eta \in \Phi_{\Sigma,\Delta}$, $s_{i+j} \in \Sigma$, defined by $\eta_{s_{i+j}}(y) = \eta(s_{i+j}, y)$ for all $y \in \Sigma$.

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

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 Φ_Σ... ordering = summary.
The construction time and size for $A_{T,s}$ are $O(\|T\|.|s|)$, 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 [6] 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 Ω_i, Ω_c, Ω_c, 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_{Ω_i}, Φ_{Ω_c}, Φ_{Ω_r} and $\Phi_{\Omega_\mathsf{c},\Omega_\mathsf{x}}$.

Definition 3. *A* Symbolic Weighted Visibly Pushdown Automata *(sw-VPA) over $\Omega = \Omega_\mathsf{i} \uplus \Omega_\mathsf{c} \uplus \Omega_\mathsf{r}$, $\mathbb{S}$ 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 $q, q' \in Q$, $p \in P$, $a \in \Omega_\mathsf{i}$, $c \in \Omega_\mathsf{c}$, $r \in \Omega_\mathsf{r}$, overloading their names:

$\mathsf{w}_\mathsf{i} : Q \times \Omega_\mathsf{c} \times P \times \Omega_\mathsf{i} \times Q \to \mathbb{S}$	$\mathsf{w}_\mathsf{i}(q,c,p,a,q') = \eta_\mathsf{ci}(c,a)$	where $\eta_\mathsf{ci} = \mathsf{w}_\mathsf{i}(q,p,q')$,
$\mathsf{w}^\mathsf{e}_\mathsf{i} : Q \times \Omega_\mathsf{i} \times Q \to \mathbb{S}$	$\mathsf{w}^\mathsf{e}_\mathsf{i}(q,a,q') = \phi_\mathsf{i}(a)$	where $\phi_\mathsf{i} = \mathsf{w}^\mathsf{e}_\mathsf{i}(q,q')$.
$\mathsf{w}_\mathsf{c} : Q \times \Omega_\mathsf{c} \times P \times \Omega_\mathsf{c} \times P \times Q \to \mathbb{S}$	$\mathsf{w}_\mathsf{c}(q,c,p,c',p',q') = \eta_\mathsf{cc}(c,c')$	where $\eta_\mathsf{cc} = \mathsf{w}_\mathsf{c}(q,p,p',q')$,
$\mathsf{w}^\mathsf{e}_\mathsf{c} : Q \times \Omega_\mathsf{c} \times P \times Q \to \mathbb{S}$	$\mathsf{w}^\mathsf{e}_\mathsf{c}(q,c,p,q') = \phi_\mathsf{c}(c)$	where $\phi_\mathsf{c} = \mathsf{w}^\mathsf{e}_\mathsf{c}(q,p,q')$.
$\mathsf{w}_\mathsf{r} : Q \times \Omega_\mathsf{c} \times P \times \Omega_\mathsf{r} \times Q \to \mathbb{S}$	$\mathsf{w}_\mathsf{r}(q,c,p,r,q') = \eta_\mathsf{cr}(c,r)$	where $\eta_\mathsf{cr} = \mathsf{w}_\mathsf{r}(q,p,q')$,
$\mathsf{w}^\mathsf{e}_\mathsf{r} : Q \times \Omega_\mathsf{r} \times Q \to \mathbb{S}$	$\mathsf{w}^\mathsf{e}_\mathsf{r}(q,r,q') = \phi_\mathsf{r}(r)$	where $\phi_\mathsf{r} = \mathsf{w}^\mathsf{e}_\mathsf{r}(q,q')$.

The intuition is the following for the above transitions.
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 $c \in \Omega_\mathsf{c}$ and $p \in P$

at the top of the stack (c is compared to a by the weight function $\eta_{\mathsf{ci}} \in \Phi_{\mathsf{ci}}$) and $\mathsf{w}_{\mathsf{i}}^{\mathsf{e}}$ applies if and only if the stack is empty.
w_{c} and $\mathsf{w}_{\mathsf{c}}^{\mathsf{e}}$ 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'), and $\mathsf{w}_{\mathsf{c}}^{\mathsf{e}}$ applies iff the stack is empty.
w_{r} and $\mathsf{w}_{\mathsf{r}}^{\mathsf{e}}$ 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), and $\mathsf{w}_{\mathsf{r}}^{\mathsf{e}}$ 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 3. 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 $\Gamma = \Omega_{\mathsf{c}} \times P$. Hence, weight_A is a function from $[Q \times \Gamma^*] \times \Omega^* \times [Q \times \Gamma^*]$ into $\mathbb{S}$.

$$
\begin{aligned}
\mathsf{weight}_A\Big(\begin{bmatrix} q \\ c\,p\cdot\gamma \end{bmatrix}, a\,u, \begin{bmatrix} q' \\ \gamma' \end{bmatrix}\Big) &= \bigoplus_{q''\in Q} \mathsf{w}_{\mathsf{i}}(q,c,p,a,q'') \otimes \mathsf{weight}_A\Big(\begin{bmatrix} q'' \\ \gamma \end{bmatrix}, u, \begin{bmatrix} q' \\ \gamma' \end{bmatrix}\Big) \\
\mathsf{weight}_A\Big(\begin{bmatrix} q \\ \bot \end{bmatrix}, a\,u, \begin{bmatrix} q' \\ \gamma' \end{bmatrix}\Big) &= \bigoplus_{q''\in Q} \mathsf{w}_{\mathsf{i}}^{\mathsf{e}}(q,a,q'') \otimes \mathsf{weight}_A\Big(\begin{bmatrix} q'' \\ \bot \end{bmatrix}, u, \begin{bmatrix} q' \\ \gamma' \end{bmatrix}\Big) \\
\mathsf{weight}_A\Big(\begin{bmatrix} q \\ c\,p\cdot\gamma \end{bmatrix}, c'\,u, \begin{bmatrix} q' \\ \gamma' \end{bmatrix}\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(\begin{bmatrix} q'' \\ c'\,p'\cdot c\,p\cdot\gamma \end{bmatrix}, u, \begin{bmatrix} q' \\ \gamma' \end{bmatrix}\Big) \\
\mathsf{weight}_A\Big(\begin{bmatrix} q \\ \bot \end{bmatrix}, c\,u, \begin{bmatrix} q' \\ \gamma' \end{bmatrix}\Big) &= \bigoplus_{q''\in Q} \mathsf{w}_{\mathsf{c}}^{\mathsf{e}}(q,c,p,q'') \otimes \mathsf{weight}_A\Big(\begin{bmatrix} q'' \\ c\,p \end{bmatrix}, u, \begin{bmatrix} q' \\ \gamma' \end{bmatrix}\Big) \\
\mathsf{weight}_A\Big(\begin{bmatrix} q \\ c\,p\cdot\gamma \end{bmatrix}, r\,u, \begin{bmatrix} q' \\ \gamma' \end{bmatrix}\Big) &= \bigoplus_{q''\in Q} \mathsf{w}_{\mathsf{r}}\big(q,c,p,r,q''\big) \otimes \mathsf{weight}_A\Big(\begin{bmatrix} q'' \\ \gamma \end{bmatrix}, u, \begin{bmatrix} q' \\ \gamma' \end{bmatrix}\Big) \\
\mathsf{weight}_A\Big(\begin{bmatrix} q \\ \bot \end{bmatrix}, r\,u, \begin{bmatrix} q' \\ \gamma' \end{bmatrix}\Big) &= \bigoplus_{q''\in Q} \mathsf{w}_{\mathsf{r}}^{\mathsf{e}}(q,r,q'') \otimes \mathsf{weight}_A\Big(\begin{bmatrix} q'' \\ \bot \end{bmatrix}, u, \begin{bmatrix} q' \\ \gamma' \end{bmatrix}\Big) \qquad (5)
\end{aligned}
$$

where $\bot$ denotes the empty stack and $c\,p\cdot\gamma$ denotes a stack where the pair made of $c \in \Omega_{\mathsf{c}}$ 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:

$$
A(s) = \bigoplus_{q,q'\in Q} \mathsf{in}(q) \otimes \mathsf{weight}_A\big(\big[\begin{smallmatrix} q \\ \bot \end{smallmatrix}\big], s, \big[\begin{smallmatrix} q' \\ \bot \end{smallmatrix}\big]\big) \otimes \mathsf{out}(q'). \qquad (6)
$$

Example 3. 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 [6].

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 $b_\perp : Q \times Q \to \mathbb{S}$ be the function:

$$b_\perp(q, q') = \bigoplus_{s \in \Omega^*} \mathsf{weight}_A\big(\left[\begin{smallmatrix} q \\ \perp \end{smallmatrix}\right], s, \left[\begin{smallmatrix} q' \\ \perp \end{smallmatrix}\right]\big) \tag{7}$$

Since $\mathbb{S}$ is complete, the infinite sum in (7) is well defined, and by totality of $\leq_\oplus$, it is the minimum in Ω^* of the fonction $s \mapsto \mathsf{weight}_A(\left[\begin{smallmatrix} q \\ \sigma \end{smallmatrix}\right], s, \left[\begin{smallmatrix} q' \\ \sigma \end{smallmatrix}\right])$ *wrt* this ordering. The term $\left[\begin{smallmatrix} q \\ \perp \end{smallmatrix}\right], s, \left[\begin{smallmatrix} q' \\ \perp \end{smallmatrix}\right]$ of this sum is the central expression in the definition (6) 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 $b_\top : Q \times P \times Q \to \Phi_{\mathsf{c}}$ be such that, for every two states $q, q' \in Q$ and stack symbol $p \in P$:

$$b_\top(q, p, q') : c \mapsto \bigoplus_{s \in \Omega^*} \mathsf{weight}_A\big(\left[\begin{smallmatrix} q \\ c\,p\cdot\top \end{smallmatrix}\right], s, \left[\begin{smallmatrix} q' \\ c\,p\cdot\top \end{smallmatrix}\right]\big). \tag{8}$$

Intuitively, the function defined in (8) associateds to $c \in \Omega_{\mathsf{c}}$ 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 (5) of weight_A, and may pop $\langle c', p' \rangle$ later, following the fifth case of (5) (return symbol).

Algorithm 1 constructs iteratively 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 $b_\perp$.

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

The infinite sums in the updates of d in Algorithm 1, Figure 3 are well defined since $\mathbb{S}$ is complete. ** effectively computable by hypothese that the label theory

For all $q_0, q_3 \in Q$,

$$
\begin{aligned}
d_\top(q_1,p,q_3) &\oplus= d_\top(q_1,p,q_2) \otimes \bigoplus_{\Omega_\mathsf{i}} \mathsf{w_i}(q_2,p,q_3)\\
d_\bot(q_1,p,q_3) &\oplus= d_\bot(q_1,q_2) \otimes \bigoplus_{\Omega_\mathsf{i}} \mathsf{w_i^e}(q_2,q_3)\\
d_\top(q_0,p,q_3) &\oplus= \bigoplus_{\Omega_\mathsf{c}}{}^2 \left[\left(\mathsf{w_c}(q_0,p,p',q_1) \otimes_2 d_\top(q_1,p',q_2)\right) \otimes_2 \bigoplus_{\Omega_\mathsf{r}} \mathsf{w_r}(q_2,p',q_3)\right]\\
d_\bot(q_0,q_3) &\oplus= \bigoplus_{\Omega_\mathsf{c}} \left(\mathsf{w_c^e}(q_0,p,q_1) \otimes d_\top(q_1,p,q_2) \otimes \bigoplus_{\Omega_\mathsf{r}} \mathsf{w_r}(q_2,p,q_3)\right)\\
d_\bot(q_1,q_3) &\oplus= d_\bot(q_1,q_2) \otimes \bigoplus_{\Omega_\mathsf{r}} \mathsf{w_r^e}(q_2,q_3)\\
d_\top(q_1,p,q_3) &\oplus= d_\top(q_1,p,q_2) \otimes d_\top(q_2,p,q_3), \text{if } \langle q_2,\top,q_3\rangle \notin P\\
d_\bot(q_1,q_3) &\oplus= d_\bot(q_1,q_2) \otimes d_\bot(q_2,q_3), \text{if } \langle q_2,\bot,q_3\rangle \notin P
\end{aligned}
$$

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

is effective** The algorithm performs $2.|Q|^2$ iterations until P is empty, and each iteration has a time complexity $O(|Q|^2.|P|)$. This 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 ***complete***.

The correctness of Algorithm 1 is ensured by the invariant expressed in the following lemma.

Lemma 5. *For all $\langle q_1,q_2\rangle \notin \mathcal{Q}$, $d_\bot(q_1,q_2) = b_\bot(q_1,q_2)/$*

The proof is by contradiction, assuming a counter-example minimal in the length of the witness word.

Lemma 6. *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 1,

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

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 4. *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* $card\{\mathbf{a} \in \Omega^n \mid \phi(\mathbf{a}) = \bigoplus_{\mathbf{p}\in\Omega^n} \phi(\mathbf{p})\} \leq k$.

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 [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 $\Omega_{>0} = \Omega \backslash \Omega_0$. 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 $k \in [0..M]$.

Let $\hat{\Omega}$ be the countable (unranked) alphabet obtained from Ω by: $\hat{\Omega} = \Omega_{\mathsf{i}} \uplus \Omega_{\mathsf{c}} \uplus \Omega_{\mathsf{r}}$, with $\Omega_{\mathsf{i}} = \Omega_0$, $\Omega_{\mathsf{c}} = \{ \langle_a | \ a \in \Omega_{>0} \}$, $\Omega_{\mathsf{r}} = \{ \ {}_a\rangle \mid a \in \Omega_{>0} \}$. We associate to $\hat{\Omega}$ a label theory $\hat{\Phi}$ like in Section 4.1.
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 $b \in \Omega_k$, $1 \leq k \leq M$.

Definition 5. *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, $\mathsf{in} : Q \to \mathbb{S}$ is the starting weight function, and $\bar{\mathsf{w}}$ is a tuplet of transition functions containing $\mathsf{w}_k : Q \times Q^k \to \Phi_{\Omega_k}$ for each $k \in [0..M]$.*

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

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

Intuitively, $\mathsf{w}(q_0, a, q_1 \ldots q_k)$ can be seen as the weight of a production rule $q_0 \to a(q_1, \ldots, q_k)$ of a regular tree grammar [5], that replaces the non-terminal symbol q_0 by $a(q_1, \ldots, q_k)$. 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 5 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 $t = b(t_1, \ldots, t_k) \in \mathcal{T}(\Omega)$, with $0 \leq k \leq M$:

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

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

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

Lemma 7. *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* $t \in \mathcal{T}(\Omega)$, $A'\big(\mathsf{lin}(t)\big) = A(t)$.

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:

$$
\begin{array}{ll}
\mathsf{w}_{\mathsf{i}}(\bar{q}, \langle_b, \bar{p}, a, \bar{q}q'\rangle) = \mathsf{w}(q', a, \varepsilon) & \text{for all } b \in \Omega_{>0}, \bar{p} \in P', a \in \Omega_0 \\
\mathsf{w}^{\mathsf{e}}_{\mathsf{i}}(\bar{q}, a, \bar{q}q') = \mathsf{w}(q', a, \varepsilon) & \text{for all } a \in \Omega_0 \\
\mathsf{w}_{\mathsf{c}}(\bar{q}, \langle_b, \bar{p}, \langle_{b'}, \varepsilon, \bar{q}) = \mathbb{1} & \text{for all } b \in \Omega_{>0}, \bar{p} \in P', b, b' \in \Omega_{>0} \\
\mathsf{w}^{\mathsf{e}}_{\mathsf{c}}(\bar{q}, \langle_b, \varepsilon, \bar{q}) = \mathbb{1} & \text{for all } b \in \Omega_{>0} \\
\mathsf{w}_{\mathsf{r}}(\bar{q}, \langle_b, \bar{p}, {}_b\rangle, \bar{p}q') = \mathsf{w}(q', b, \bar{q}) & \text{for all } b \in \Omega_{>0}, \bar{p} \in P' \\
\mathsf{w}^{\mathsf{e}}_{\mathsf{r}}(\bar{p}, {}_b\rangle, \bar{q}) = \mathbb{0} & \text{for all } b \in \Omega_{>0}
\end{array}
$$

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

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 $\hat{\Omega} = \Omega_{\mathsf{i}} \uplus \Omega_{\mathsf{c}} \uplus \Omega_{\mathsf{r}}$ be the alphabet with nesting symbols, associated like Section 4.4, for the linearization of trees of $\mathcal{T}(\Omega)$ (remember that $\Omega_{\mathsf{i}} = \Omega_0$). Let $\langle \mathbb{S}, \oplus, \mathbb{0}, \otimes, \mathbb{1}\rangle$ be a commutative, bounded, complete and total semiring and let $\bar{\Phi}$ be a label theory over $\mathbb{S}$ containing Φ_Σ, $\Phi_{\Sigma,\Omega_{\mathsf{i}}}$, as well as Φ_{i}, Φ_{c}, Φ_{r}, Φ_{cr} (following the notations of Section 4.1). 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 $T : \Sigma^* \times {\Omega_{\mathsf{i}}}^* \to \mathbb{S}$,
- a swTA A over Ω, $\mathbb{S}$, and $\bar{\Phi}$, defining a measure $A : \mathcal{T}(\Omega) \to \mathbb{S}$,
- an input word $s \in \Sigma^*$.

As explained in Section 4.4, $A = \langle Q, \mathsf{in}, \bar{\mathsf{w}} \rangle$ 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 $a(q_1, \ldots, q_k)$, where $k = \mathsf{rk}(a)$. A replacement rule $q_0 \to a(q_1, \ldots, q_k)$, of weight $\mathsf{w}(q_0, a, q_1 \ldots q_k) \in \mathbb{S}$ according to Definition 5, corresponds to the production rule $q_0 := a(q_1, \ldots, q_k)$ 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 $d : \Sigma^* \times \mathcal{T}(\Omega) \to \mathbb{S}$ as follows. Given a word $w \in \hat{\Omega}^*$, the projection of w onto Ω_{i}, denoted $w|_{\Omega_{\mathsf{i}}}$, is the word of $\Omega_{\mathsf{i}}{}^*$ obtained from w by removing all symbols in $\hat{\Omega} \setminus \Omega_{\mathsf{i}}$. Using this notation and the tree linearization operator defined in Section 4.4, d is defined by:

$$d(s,t) = T\big(s, \mathsf{lin}(t)|_{\Omega_{\mathsf{i}}}\big) \text{ for } s \in \Sigma^*, t \in \mathcal{T}(\Omega) \tag{11}$$

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:

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

The measure expressed in (12) is called called edit-distance between s and A in [19]. 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.* [13] and [21] 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* swT *T,* 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 swT T over Σ, Ω_{i}, $\mathbb{S}$, and $\bar{\Phi}$, into a swT T' over Σ and $\hat{\Omega}$ (and the same semiring and label theory), such that for all $s \in \Sigma^*$, and $u \in \hat{\Omega}^*$, $T'(s,u) = T(s, u|_{\Omega_{\mathsf{i}}})$. The transducer T' simplify skips every symbol $b \in \hat{\Omega} \setminus \Omega_{\mathsf{i}}$ in input, with new transitions of the form $\mathsf{w}_{01}(q, \varepsilon, b, q')$.
Then, given an input word $s \in \Sigma^*$, we compute the swA $A_{T',s}$, using Proposition 1. This automaton is such that for all $t \in \mathcal{T}(\Omega)$, $A_{T',s}\big(\mathsf{lin}(t)\big) = T'\big(s, \mathsf{lin}(t)\big) = T'\big(s, \mathsf{lin}(t)|_{\Omega_{\mathsf{i}}}\big) = d(s,t)$.
Next, we convert the input swTA A over Ω into a sw-VPA A' over $\hat{\Omega}$, using Lemma 7, and we compute the sw-VPA $A_{T',s} \otimes A'$, using Proposition 2.
It remains to compute a best nested-word $w \in \hat{\Omega}^*$ using the best-search procedure of Proposition 3, and convert it into a best tree in $\mathcal{T}(\Omega)$ in order to solve SW parsing for T, A and s. □

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)

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. D. Grune and C. J. Jacobs. *Parsing Techniques*. Number 2nd edition in Monographs in Computer Science. Springer, 2008.
15. L. Huang. Advanced dynamic programming in semiring and hypergraph frameworks. In *In COLING*, 2008.

16. 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.
17. M. Kaminski and N. Francez. Finite-memory automata. *Theor. Comput. Sci.*, 134:329–363, November 1994.
18. M. Mohri. Semiring frameworks and algorithms for shortest-distance problems. *Journal of Automata, Languages and Combinatorics*, 7(3):321–350, 2002.
19. M. Mohri. Edit-distance of weighted automata: General definitions and algorithms. *International Journal of Foundations of Computer Science*, 14(06):957–982, 2003.
20. M. Mohri. Edit-distance of weighted automata: General definitions and algorithms. *International Journal of Foundations of Computer Science*, 14(06):957–982, 2003.
21. 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.
22. M.-J. Nederhof. Weighted deductive parsing and Knuth's algorithm. *Computational Linguistics*, 29(1):135–143, 2003.
23. 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.
24. L. Segoufin. Automata and logics for words and trees over an infinite alphabet. In *Computer Science Logic*, volume 4207 of *LNCS*. Springer, 2006.
25. 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.