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 (SWPDA) 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 WA are labeled by functions from an infinite alphabet into the weight domain. This is unlike WA 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 WA, WT and SWPDA 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: an 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 [17] 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 [11], for the analyse (*e.g.* validation or filtering) of data streams or serialization of structured documents (which may contain textual or numerical attributes) [31], or for processing timed execution traces [5]. Regarding the latter case, we describe at the end of the paper a parse-based approach to automated music transcription, where a symbolic music performance, presented as a sequence of timed musical events, is converted into a structured score in Common Western Music Notation [15].

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 [31] for a survey), with pebbles for marking positions [30], registers [21], or the possibility to compute

on subsequences with the same attribute values [4]. The automata at the core of model checkers compute on input symbols represented by large bitvectors [32] (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) [10,11], 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 [16,29,28]. In *weighted language models*, like *e.g.* probabilistic context-free grammars and weighted automata (WA) [14], 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 [26]. 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 [26]. Similarly to weighted-parsing methods [16,29,28], 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, a best AST t is found by applying to $A_{T,s}$ a best search algorithm similar to shortest distance in graphs [25,19].

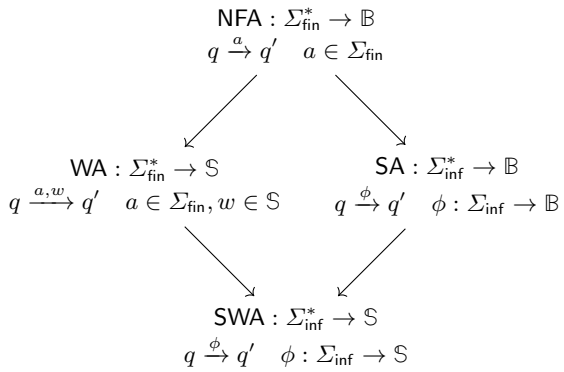

Fig. 1. Classes of Symbolic/Weighted Automata. Σ_{fin} is a finite alphabet, Σ_{inf} is a countable alphabet, $\mathbb{B}$ is the Boolean algebra, $\mathbb{S}$ is an arbitrary commutative semiring, $q \xrightarrow{\cdots} q'$ represents the form of a transition between states q and q'.

The main contributions of the paper are: (*i*) the introduction of automata, swA, transducers, swT (Section 2), and visibly pushdown automata sw-VPA (Section 3), generalizing the corresponding classes of symbolic and weighted models, (*ii*) a polynomial best-search algorithm for sw-VPA, and (*iii*) a uniform framework (Section 4) 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. Finally, Section 4.3 describes the implemented application to automated music transcription that motivated this study.

2 SW Automata and Transducers

We follow the approach of [26] for the computation of distances between words and languages, with weighted transducers, extending it to infinite alphabets. The models introduced in this section generalize weighted automata and transducers [14] by labeling each transition with a weight function that takes the input and output symbols as parameters, instead of a simple weight value. These functions are similar to the guards of symbolic automata [10,11], but they can return values in an arbitrary semiring, where the latter guards are restricted to the Boolean semiring.

2.1 Semirings

We shall consider semiring 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 $\mathbb{S}$ [25] 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 [13]; We follow here the direction that coincides with the usual ordering on the Tropical semiring *min-plus* (Figure 2).

Lemma 1 (Monotony, [25]). *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 [18], which corresponds to the *non-negative weights* condition in shortest-path algorithms [12]. 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. For $(ii) \Rightarrow (i) : x \oplus (x \otimes y) = x \otimes (\mathbb{1} \oplus y) = x$, 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 $y \leq_\oplus x \otimes y$. For $(i) \Rightarrow (ii)$: 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 [18], $\mathbb{S}$ with the property (i) is called *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 [25], 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. □

	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	$+$	$+\infty$	0

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

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* [22] if for every family $(x_i)_{i\in I}$ of elements of $dom(\mathbb{S})$ over an index set $I \subset \mathbb{N}$, the infinite sum $\bigoplus_{i\in I} x_i$ is well-defined and in $dom(\mathbb{S})$, and the following holds:

i. infinite sums extend finite sums: $\bigoplus_{i\in\emptyset} x_i = \mathbb{0}, \quad \forall j \in \mathbb{N}, \bigoplus_{i\in\{j\}} x_i = x_j, \quad \forall j,k \in \mathbb{N}, j \neq k, \bigoplus_{i\in\{j,k\}} x_i = x_j \oplus x_k,$

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 presented 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. Let Σ and Δ be two countable sets of symbols called respectively *input* and *output alphabets*, and let $\langle \mathbb{S}, \oplus, \mathbb{0}, \otimes, \mathbb{1}\rangle$ be a commutative semiring. A *label theory* over Σ and Δ is made of 3 recursively enumerable sets: Φ_Σ and Φ_Δ, containing unary functions in $\Sigma \to \mathbb{S}$, resp. $\Delta \to \mathbb{S}$, and $\Phi_{\Sigma,\Delta}$ containing binary functions in $\Sigma \times \Delta \to \mathbb{S}$, such that 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 Φ_Δ and $\Phi_{\Sigma,\Delta}$. Moreover, these sets are required to be closed under the following operators $\oplus$ and $\otimes$ of $\mathbb{S}$: for all $\phi, \phi' \in \Phi_\Sigma$, all $\psi, \psi' \in \Phi_\Delta$, and $\eta, \eta' \in \Phi_{\Sigma,\Delta}$, the function

$\phi \otimes \phi' : x \mapsto \phi(x) \otimes \phi'(x)$ belongs to Φ_Σ,
$\psi \otimes \psi' : y \mapsto \psi(y) \otimes \psi'(y)$ belongs to Φ_Δ,
$\phi \otimes \eta : x, y \mapsto \phi(x) \otimes \eta(x, y)$ belongs to $\Phi_{\Sigma,\Delta}$,
$\eta \otimes \psi : x, y \mapsto \eta(x, y) \otimes \psi(y)$ belongs to $\Phi_{\Sigma,\Delta}$,
$\eta \otimes \eta' : x, y \mapsto \eta(x, y) \otimes \eta'(x, y)$ belongs to $\Phi_{\Sigma,\Delta}$.

And the same also holds for the binary sum operator $\oplus$.
Finally, it is also required that all partial applications of functions $\Phi_{\Sigma,\Delta}$, resp. $f_a : y \mapsto f(a, y)$ for $a \in \Sigma$ and $y \in \Delta$ and $f_b : x \mapsto f(x, b)$ for $b \in \Delta$ and $x \in \Sigma$, belong resp. to Φ_Σ and Φ_Δ.

2.3 Definitions

Definition 1. *A* symbolic-weighted transducer *(*swT*) over the input and output alphabets Σ and Δ and the semiring $\mathbb{S}$ 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 tuplet made of the 4 of transition functions* w_ϵ*,* w_Σ*,* w_Δ*, and* $\mathsf{w}_{\Sigma,\Delta}$ *from $Q \times Q$ into respectively $\mathbb{S}$, Φ_Σ, Φ_Δ, and $\Phi_{\Sigma,\Delta}$.*

We summarize the above 4-uplet of transition functions into a unique function w from $Q \times (\Sigma \cup \{\epsilon\}) \times (\Delta \cup \{\epsilon\}) \times Q$ into $\mathbb{S}$, such that, for all $q, q' \in Q$, $a \in \Sigma$, and $b \in \Delta$,

$$\begin{array}{ll}
\mathsf{w}(q, \epsilon, \epsilon, q') = \mathsf{w}_\epsilon(q, q'), & \\
\mathsf{w}(q, a, \epsilon, q') = \phi_\Sigma(a) & \text{where } \phi_\Sigma = \mathsf{w}_\Sigma(q, q'), \\
\mathsf{w}(q, \epsilon, b, q') = \phi_\Delta(b) & \text{where } \phi_\Delta = \mathsf{w}_\Delta(q, q'), \\
\mathsf{w}(q, a, b, q') = \phi_{\Sigma,\Delta}(a, b) & \text{where } \phi_{\Sigma,\Delta} = \mathsf{w}_{\Sigma,\Delta}(q, q').
\end{array}$$

The symbolic-weighted transducer T defines a mapping from the pairs of strings of $\Sigma^* \times \Delta^*$ into the weights of $\mathbb{S}$, based on the following intermediate function weight_T defined recursively for every $q, q' \in Q$, for every strings of $s \in \Sigma^*$, $t \in \Delta^*$:

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

The sum above can be infinite because of the expression in the first line, however, $\mathcal{T}(\Omega)$ is well defined since $\mathbb{S}$ is complete. As we shall see, epsilon-transitions can be removed and hence the sum made finite. We recall that, by convention, an

empty sum with $\oplus$ is $\mathbb{0}$. Since $\mathbb{0}$ is absorbing for $\otimes$ in $\mathbb{S}$, one term $\mathsf{w}(q, a, b, q'')$ 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 writing b (including the case $a = \epsilon$ and $a = \epsilon$). This is analogous to a transition's guard not satisfied by $\langle a, b\rangle$ in the case of symbolic transducers.

The above 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 [27]. Intuitively, $\mathsf{w}(q, a, \epsilon, q'')$, in the third line, is the cost of the deletion of a symbol $a \in \Sigma$ in s, $\mathsf{w}(q, \epsilon, b, q'')$, in the fourth line, is the cost of the deletion of insertion of $b \in \Delta$ in s, and $\mathsf{w}(q, a, b, q'')$, in the last line, 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 then defined as follows:

$$T(s,t) = \bigoplus_{q,q' \in Q} \mathsf{in}(q) \otimes \mathsf{weight}_T(q, s, t, q') \otimes \mathsf{out}(q'). \tag{1}$$

Example 2. DTW for sequences of timestamped events **

Example 3. (simpler) pointwise distance between two sequences of timestamped events **

The *Symbolic Weighted Automata* over Σ and $\mathbb{S}$ are defined similarly as the transducers of Definition 1, by simply omitting the output symbols. In this case, a label theory over Σ is reduced to a set Φ_Σ closed under $\oplus$ and $\otimes$.

Definition 2. *A* symbolic-weighted automaton *(swA) over the input alphabet Σ and the commutative semiring $\mathbb{S}$ is a tuple $A = \langle Q, \mathsf{in}, \bar{\mathsf{w}}, \mathsf{out}\rangle$, 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 pair of transition functions w_ϵ and w_Σ from $Q \times Q$ into respectively $\mathbb{S}$ and Φ_Σ.*

The above transition functions w_ϵ and w_Σ are synthesized into a function w from $Q \times (\Sigma \cup \{\epsilon\}) \times Q$ into $\mathbb{S}$ as above. When $\mathsf{w}_\epsilon(q, q') = \mathbb{0}$ for all $q, q' \in Q$, the automaton A is called *without ϵ-transitions.*

2.4 Properties

The two following properties will be useful to our approach on symbolic weighted parsing in Section 4.

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

Proof. Let $T = \langle Q, \mathsf{in}, \bar{\mathsf{w}}, \mathsf{out}\rangle$ where $\bar{\mathsf{w}}\langle \mathsf{w}_\epsilon, \mathsf{w}_\Sigma, \mathsf{w}_\Delta, \mathsf{w}_{\Sigma,\Delta}\rangle$ has an associated synthesized form w, and let $s = s_1 \dots s_n$ with $s_1, \dots, s_n \in \Sigma$. Let $Q' = [1..n] \times Q$ be the state set of $A_{T,s}$, and let, for all $q \in Q$, $\mathsf{in}'(1, q) = \mathsf{in}(q)$ and $\mathsf{in}'(i, q) = \mathbb{0}$ for all $1 < i \leq n$, and $\mathsf{out}'(n, q) = \mathsf{out}(q)$ and $\mathsf{out}'(i, q) = \mathbb{0}$ for all $1 \leq i < n$. We define the functions w'_ϵ and w'_Σ for $A_{T,s}$ in their synthesized form: for all $1 \leq i, j \leq n$, $q, q' \in Q$ and $a \in \Sigma$, and $b \in \Delta$,

$$\begin{array}{lll}
\mathsf{w}'(\langle i, q\rangle, \epsilon, \langle i, q'\rangle) & = \mathsf{w}(q, \epsilon, \epsilon, q'), & \\
\mathsf{w}'(\langle i, q\rangle, \epsilon, \langle i+1, q'\rangle) & = \mathsf{w}(q, s_i, \epsilon, q') & \text{if } i < n, \\
\mathsf{w}'(\langle i, q\rangle, \epsilon, \langle j, q'\rangle) & = \mathbb{0} & \text{if } j \neq i, j \neq i+1, \\
\mathsf{w}'(\langle i, q\rangle, b, \langle i, q'\rangle) & = \mathsf{w}(q, \epsilon, b, q'), & \\
\mathsf{w}'(\langle i, q\rangle, b, \langle i+1, q'\rangle) & = \mathsf{w}(q, s_i, b, q') & \text{if } i < n, \\
\mathsf{w}'(\langle i, q\rangle, b, \langle j, q'\rangle) & = \mathbb{0} & \text{if } j \neq i, j \neq i+1.
\end{array}$$

Note that the functions defined above belong to the set Φ_Δ of the underlying label theory, by the hypothesis of closure under partial application. The swA wanted is $A_{T,s} = \langle Q', \mathsf{in}', \bar{\mathsf{w}}', \mathsf{out}'\rangle$. □

The construction time and size for $A_{T,s}$ are $O(\|T\|.|s|)$, where the size $\|T\|$ of T is its number of states $|Q|$.

Proposition 2. *Given a swA A over alphabet Σ and a commutative and bounded semiring $\mathbb{S}$, there exists an effectively constructible swA A' without ϵ-transitions such that for all $s \in \Sigma^*$, $A'(s) = A(s)$.*

Proof. ** Tb revised ** Let $A = \langle Q, \mathsf{in}, \langle \mathsf{w}_\epsilon, \mathsf{w}_\Sigma\rangle, \mathsf{out}\rangle$. The automaton A' is $\langle Q, \mathsf{in}, \langle \mathsf{w}'_\epsilon, \mathsf{w}'_\Sigma\rangle, \mathsf{out}\rangle$, where, for all $q, q' \in Q$, $\mathsf{w}'_\epsilon(q, q') = \mathbb{0}$ and

$$\mathsf{w}'_\Sigma(q, q') = \bigoplus_{q'' \in Q} \mathsf{w}_\epsilon(q, q'') \otimes \mathsf{w}_\Sigma(q'', q') \oplus \bigoplus_{q_1 \in Q} \mathsf{w}_\Sigma(q, q'') \otimes \mathsf{w}_\epsilon(q'', q')$$

In the above definition of w'_Σ we use the operator of product of function of Φ_Σ by $\mathbb{S}$ described in Section 2.2. By definition of weight_A and distributivity of $\otimes$ on $\oplus$, ** NO. TBC see [24] ** it holds that $\mathsf{weight}_A(q, s, q') = \mathsf{weight}_{A'}(q, s, q')$. □

3 SW Visibly Pushdown Automata

The following model generalizes Symbolic VPA [9] from Boolean semirings to arbitrary semiring weight domains.

3.1 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 semiring and let $(\Phi_\epsilon, \Phi_{\mathsf{c}}, \Phi_{\mathsf{r}}, \Phi_{\mathsf{cr}})$ be a label theory over $\mathbb{S}$ where Φ_{c}, Φ_{r} and Φ_{cr} stand respectively for $\Phi_{\Omega_{\mathsf{c}}}$, $\Phi_{\Omega_{\mathsf{r}}}$ and $\Phi_{\Omega_{\mathsf{c}},\Omega_{\mathsf{r}}}$. Moreover, we extend this theory with a set Φ_{i} of unary functions in $\Omega_{\mathsf{i}} \to \mathbb{S}$, closed under $\oplus$ and $\otimes$.

Definition 3. *A* Symbolic Weighted Visibly Pushdown Automata *(sw-VPA) over* $\Omega = \Omega_{\mathsf{i}} \uplus \Omega_{\mathsf{c}} \uplus \Omega_{\mathsf{r}}$ *and* $\mathbb{S}$ *is a tuple* $A = \langle Q, P, \mathsf{in}, \mathsf{w_i}, \mathsf{w_c}, \mathsf{w_r}, \mathsf{w_e}, \mathsf{out}\rangle$*, 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* $\mathsf{w_i} : Q \times Q \to \Phi_{\mathsf{i}}$*,* $\mathsf{w_c} : Q \times Q \times P \to \Phi_{\mathsf{c}}$*,* $\mathsf{w_r} : Q \times P \times Q \to \Phi_{\mathsf{cr}}$*,* $\mathsf{w_e} : Q \times Q \to \Phi_{\mathsf{r}}$*, are transition functions.*

Similarly as in Section 2, 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:

$$
\begin{array}{lll}
\mathsf{w_i} : Q \times \Omega_{\mathsf{i}} \times Q \to \mathbb{S} & \mathsf{w_i}(q, a, q') = \phi_{\mathsf{i}}(a) & \text{where } \phi_{\mathsf{i}} = \mathsf{w_i}(q, q'), \\
\mathsf{w_c} : Q \times \Omega_{\mathsf{c}} \times Q \times P \to \mathbb{S} & \mathsf{w_c}(q, c, q', p) = \phi_{\mathsf{c}}(c) & \text{where } \phi_{\mathsf{c}} = \mathsf{w_c}(q, q', p), \\
\mathsf{w_r} : Q \times \Omega_{\mathsf{c}} \times P \times \Omega_{\mathsf{r}} \times Q \to \mathbb{S} & \mathsf{w_r}(q, c, p, r, q') = \phi_{\mathsf{r}}(c, r) & \text{where } \phi_{\mathsf{r}} = \mathsf{w_r}(q, p, q'), \\
\mathsf{w_e} : Q \times \Omega_{\mathsf{r}} \times Q \to \mathbb{S} & \mathsf{w_e}(q, r, q') = \phi_{\mathsf{e}}(r) & \text{where } \phi_{\mathsf{e}} = \mathsf{w_e}(q, q').
\end{array}
$$

The intuition is the following for the above transitions.

- $\mathsf{w_i}$: read the input internal symbol a, change state to q' (stack is untouched).
- $\mathsf{w_c}$: read the input call symbol c, push it to the stack along with p, change state to q'.
- $\mathsf{w_r}$: when the stack is not empty, read and pop from stack a pair made of c and p, read the input return symbol r, change state to q'. In this case, the weight function ϕ_{r} computes a value of matching between the call and return symbols c and r. This value might be set to $\mathbb{0}$ in order to express that the symbols do not match.
- $\mathsf{w_e}$: when the stack is empty, read the input symbol r, change state to q'.

We give now a formal definition of these transitions of the automaton A in term of a weight value computed by an intermediate function weight_A, like in Section 2. 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 $\Gamma = \Omega_{\mathsf{c}} \times P$. Therefore, weight_A is a function from $Q \times \Gamma^* \times \Omega^* \times Q \times \Gamma^*$ into $\mathbb{S}$.

$$
\begin{aligned}
\mathsf{weight}_A\big(\left[\begin{smallmatrix} q \\ \gamma \end{smallmatrix}\right], a\,u, \left[\begin{smallmatrix} q' \\ \gamma' \end{smallmatrix}\right]\big) &= \bigoplus_{q'' \in Q} \mathsf{w_i}(q, a, q'') \otimes \mathsf{weight}_A\big(\left[\begin{smallmatrix} q'' \\ \gamma \end{smallmatrix}\right], u, \left[\begin{smallmatrix} q' \\ \gamma' \end{smallmatrix}\right]\big) \\
\mathsf{weight}_A\big(\left[\begin{smallmatrix} q \\ \gamma \end{smallmatrix}\right], c\,u, \left[\begin{smallmatrix} q' \\ \gamma' \end{smallmatrix}\right]\big) &= \bigoplus_{\substack{q'' \in Q \\ p \in P}} \mathsf{w_c}\big(q, c, q'', p\big) \otimes \mathsf{weight}_A\big(\left[\begin{smallmatrix} q'' \\ c\,p \cdot \gamma \end{smallmatrix}\right], u, \left[\begin{smallmatrix} q' \\ \gamma' \end{smallmatrix}\right]\big) \\
\mathsf{weight}_A\big(\left[\begin{smallmatrix} q \\ c\,p \cdot \gamma \end{smallmatrix}\right], r\,u, \left[\begin{smallmatrix} q' \\ \gamma' \end{smallmatrix}\right]\big) &= \bigoplus_{q'' \in Q} \mathsf{w_r}\big(q, c, p, r, q''\big) \otimes \mathsf{weight}_A\big(\left[\begin{smallmatrix} q'' \\ \gamma \end{smallmatrix}\right], u, \left[\begin{smallmatrix} q' \\ \gamma' \end{smallmatrix}\right]\big) \\
\mathsf{weight}_A\big(\left[\begin{smallmatrix} q \\ \bot \end{smallmatrix}\right], r\,u, \left[\begin{smallmatrix} q' \\ \gamma' \end{smallmatrix}\right]\big) &= \bigoplus_{q'' \in Q} \mathsf{w_e}(q, r, q'') \otimes \mathsf{weight}_A\big(\left[\begin{smallmatrix} q'' \\ \bot \end{smallmatrix}\right], u, \left[\begin{smallmatrix} q' \\ \gamma' \end{smallmatrix}\right]\big)
\end{aligned}
$$

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 then defined as follows, following empty stack semantics:

$$
A(s) = \bigoplus_{q, q' \in Q} \mathsf{in}(q) \otimes \mathsf{weight}_A\big(\left[\begin{smallmatrix} q \\ \bot \end{smallmatrix}\right], s, \left[\begin{smallmatrix} q' \\ \bot \end{smallmatrix}\right]\big) \otimes \mathsf{out}(q'). \tag{2}
$$

Example 4. structured words... intro language of music notation ?

3.2 Properties

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

Proposition 3. *Let A_1 and A_2 be two (sw-VPA) over the same Ω and $\mathbb{S}$. 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 [9].

3.3 Best-first Search

Let us assume that the semiring $\mathbb{S}$ is commutative, bounded, complete, and total. We propose a Dijkstra algorithm computing the minimal weight by A, *wrt* $\leq_\oplus$, for a word in $\mathbb{S}^*$.
More precisely, let $\top$ be a fresh stack symbol which does not belong to Γ, and for every two states $q, q' \in Q$ and $\sigma \in \{\bot, \top\}$, let

$$d_0(q, \sigma, q') = \bigoplus_{s \in \Omega^*} \mathsf{weight}_A\big(\left[\begin{smallmatrix} q \\ \sigma \end{smallmatrix}\right], s, \left[\begin{smallmatrix} q' \\ \sigma \end{smallmatrix}\right]\big).$$

Since $\mathbb{S}$ is complete, this infinite sum is well defined, and since $\leq_\oplus$ is assumed total, it is the minimum in Ω^* of $s \mapsto \mathsf{weight}_A(\left[\begin{smallmatrix} q \\ \sigma \end{smallmatrix}\right], s, \left[\begin{smallmatrix} q' \\ \sigma \end{smallmatrix}\right])$ *wrt* this ordering. When $\sigma = \bot$, $d_0(q, \sigma, q')$ is the central expression in a term of the definition (2) of $A(s_0)$ for the minimum s_0 (for the above function). When $\sigma = \bot$, intuitively, it is the minimum weight of a computation of A starting in state q with a stack $\gamma \in \Gamma^*$ (possibly empty), and ending in state q with the same stack γ, such that moreover the computation does not touch a symbol of γ. That means that during the computation, A may apply the first case of in the definition of weight_A (internal symbol), as well as the second case, to can push call symbols on the top of γ, and may pop these symbols with the third case (return symbol). However, it cannot apply one of the two last cases (return symbol and empty stack) when the current stack is γ.

The algorithm 1 constructs iteratively a marking $d : Q \times \{\bot, \top\} \times Q \to \mathbb{S}$ that converges eventually to $d_0(q, \sigma, q')$.

Algorithm 1 (1-best for sw-VPA)
initially let $P = Q \times \{\bot, \top\} \times Q$, and $d(q_1, \bot, q_2) = d(q_1, \top, q_2) = \mathbb{1}$ if $q_1 = q_2$ and $d(q_1, \bot, q_2) = d(q_1, \top, q_2) = \mathbb{0}$ otherwise.

while P is not empty
  extract $\langle q_1, \sigma, q_2 \rangle$ from P such that $d(q_1, \sigma, q_2)$ is minimal wrt $\leq_\oplus$.
  for all $q_0, q_3 \in Q$ and $p \in P$ do

$$
\begin{aligned}
d(q_1,\sigma,q_3) &\oplus= d(q_1,\sigma,q_2) \otimes \bigoplus_{a\in\Omega_{\mathsf{i}}} \mathsf{w_i}(q_2,a,q_3)\\
\text{if } \sigma=\top\ d(q_0,\top,q_3) &\oplus= d(q_1,\sigma,q_2) \otimes \bigoplus_{c\in\Omega_{\mathsf{c}}}\bigoplus_{r\in\Omega_{\mathsf{r}}} \eta(c,r)\\
\text{and } d(q_0,\bot,q_3) &\oplus= d(q_1,\sigma,q_2) \otimes \bigoplus_{c\in\Omega_{\mathsf{c}}}\bigoplus_{r\in\Omega_{\mathsf{r}}} \eta(c,r)\\
&\text{where } \eta = \mathsf{w_c}(q_0,q_1,p)\otimes \mathsf{w_r}(q_2,p,q_3)\\
\text{if } \sigma=\bot\ d(q_1,\bot,q_3) &\oplus= d(q_1,\sigma,q_2) \otimes \bigoplus_{r\in\Omega_{\mathsf{r}}} \mathsf{w_e}(q_2,r,q_3)\\
d(q_1,\bot,q_3) &\oplus= d(q_1,\sigma,q_2)\otimes d(q_2,\bot,q_3), \text{if } \langle q_2,\bot,q_3\rangle\notin P\\
\text{if } \sigma=\top\ d(q_1,\top,q_3) &\oplus= d(q_1,\sigma,q_2)\otimes d(q_2,\top,q_3), \text{if } \langle q_2,\top,q_3\rangle\notin P
\end{aligned}
$$

The infinite sums in the updates of d in Algorithm 1 are well defined since $\mathbb{S}$ is complete. 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 4. *For all $\langle q_1,\sigma,q_2\rangle \notin P$, $d(q_1,\sigma,q_2) = d_0(q_1,\sigma,q_2)$.*

The proof is by contradiction, assuming a counter-example minimal in the length of the witness word.
For computing the minimal weight of a computation of A, we use the fact that, at the termination of Algorithm 1,

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

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

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

Proposition 4. *For a sw-VPA A over a commutative, idempotent, superior, total and complete semiring and an alphabet Ω with a k-convex label theory, one can construct in PTIME a word $s\in\Omega^*$ such that $A(s)$ is minimal wrt the natural ordering for $\mathbb{S}$.*

3.4 Trees and Nested-Words

The hierarchical structure of nested-words, defined the *call* and *return* markup symbols of suggest a correspondence between these word and trees. The lifting

of this correspondence to languages, respective of tree automata and VPA have been discussed in [1], see also [6] for the weighted case. In this section, we discuss the 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 write Ω_k the subset of all symbols a of Ω with $\mathsf{rk}(a) = k$, where $0 \leq k \leq M$. $\mathcal{T}(\Omega)$ denotes the free Ω-algebra of finite, ordered, Ω-labeled trees, which 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 $\hat{\Omega}$ be the countable (unranked) alphabet obtained from Ω as follows: $\hat{\Omega} = \langle \Omega_{\mathsf{i}}, \Omega_{\mathsf{c}}, \Omega_{\mathsf{r}} \rangle$, with $\Omega_{\mathsf{i}} = \Omega_0$, $\Omega_{\mathsf{c}} = \{ \langle_a | \ a \in \Omega_{>0} \}$, $\Omega_{\mathsf{c}} = \{ \ {}_a\rangle \mid a \in \Omega_{>0} \}$, where $\Omega_{>0}$ denotes $\bigcup_{k=1}^{M} \Omega_k$.
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 $b \in \Omega_k$, $1 \leq k \leq M$.

Let us assume a label theory Φ_{Ω_k} for each $k \in [0..M]$.

Definition 5. *A* symbolic-weighted tree automaton *(swTA) over the ranked input alphabet Ω and the commutative semiring $\mathbb{S}$ 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 $M+2$-uplet of transition functions made of: w_ϵ from $Q \times Q$ into $\mathbb{S}$, and, for each $k \in [0..M]$, $\mathsf{w}_{\Omega,k}$ from $Q \times Q^k$ into Φ_{Ω_k}.*

Like in Section 2.3, we define from $\bar{\mathsf{w}}$ a transition function w, from $Q \times (\Sigma \cup \{\epsilon\}) \times \bigcup_{k=0}^{M} Q^k$ into $\mathbb{S}$:

$$\begin{aligned} \mathsf{w}(q_0, \epsilon, q_1) &= \mathsf{w}_\epsilon(q_0, q_1), \\ \mathsf{w}(q_0, a, q_1 \ldots q_k) &= \phi_{\Omega,k}(a) \qquad \text{where } \phi_{\Omega,k} = \mathsf{w}_{\Omega,k}(q_0, q_1 \ldots q_k). \end{aligned}$$

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 [7], 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 $t = b(t_1, \ldots, t_k) \in \mathcal{T}(\Omega)$, åwith $0 \leq k \leq M$:

$$\begin{aligned} \mathsf{weight}_A(q_0, t) = & \bigoplus_{q_1 \in Q} \mathsf{w}(q, \epsilon, q_1) \otimes \mathsf{weight}_A(q_1, t) \\ & \oplus \bigoplus_{q_1 \ldots q_k \in Q^k} \mathsf{w}(q_0, b, q_1 \ldots q_k) \otimes \bigotimes_{i=1}^{k} \mathsf{weight}_A(q_i, t_i). \end{aligned}$$

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

$$A(t) = \bigoplus_{q \in Q} \mathsf{in}(q) \otimes \mathsf{weight}_A(q, t). \tag{3}$$

Lemma 5. *For all* swTA A *over* Ω *and* $\mathbb{S}$*, without* ϵ*-transitions, there exists an effectively constructible* sw-VPA A' *over* $\hat{\Omega}$ *and* $\mathbb{S}$ *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 summarized as above by a function $\mathsf{w} : Q \times (\Sigma \cup \{\epsilon\}) \times \bigcup_{k=0}^{M} Q^k \to \mathbb{S}$.

We build $A' = \langle Q', P', \mathsf{in}', \mathsf{w_i}, \mathsf{w_c}, \mathsf{w_r}, \mathsf{w_e}, \mathsf{out}' \rangle$, computing over $\hat{\Omega} = \langle \Omega_\mathsf{i}, \Omega_\mathsf{c}, \Omega_\mathsf{r} \rangle$, where $Q' = \bigcup_{k=0}^{M} Q^k$ be the set of sequences of state symbols of A, of length at most M, including the empty sequence denoted by ϵ, and where $P' = Q'$.

$$
\begin{array}{ll}
\mathsf{w_i}(\bar{q}, a, \bar{q}q') = \mathsf{w}(q', a, \epsilon) & \text{for all } a \in \Omega_0, \\
\mathsf{w_c}(\bar{q}, \langle_b, \epsilon, \bar{q}) = \mathbb{1} & \text{for all } b \in \Omega_{>0}, \\
\mathsf{w_r}(\bar{q}, \langle_b, \bar{p}, {}_b\rangle, \bar{p}q') = \mathsf{w}(q', b, \bar{q}) & \text{for all } b \in \Omega_{>0}, \\
\mathsf{w_e}(\bar{p}, {}_b\rangle, \bar{q}) = \mathbb{0} & \text{for all } b \in \Omega_{>0}.
\end{array}
$$

In practice, it is sufficient to consider in Q' only the prefixes of sequences. □

4 Symbolic Weighted Parsing

Let us now use the models and results of the former sections in an approach 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 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.

4.1 Definition

Let Σ be a countable input alphabet and $\Omega = \Omega_\mathsf{i} \uplus \Omega_\mathsf{c} \uplus \Omega_\mathsf{r}$ an output countable alphabet, let $\langle \mathbb{S}, \oplus, \mathbb{0}, \otimes, \mathbb{1} \rangle$ be a commutative, bounded, complete and total semiring and let $(\Phi_\epsilon, \Phi_\mathsf{c}, \Phi_\mathsf{r}, \Phi_\mathsf{cr})$ be a label theory over $\mathbb{S}$, assumed computable and k-convex for some fixed k.
Assuming given in input:

- a swT T over Σ, Ω, and $\mathbb{S}$, defining a measure $T : \Sigma^* \times \Omega^* \to \mathbb{S}$,
- a sw-VPA A over Ω, and $\mathbb{S}$, defining a series of nested words $A : \Omega^* \to \mathbb{S}$,
- an input word $s \in \Sigma^*$,

the problem of *symbolic weighted parsing* is to find a nested word $t \in \Omega$ minimizing $T(s,t) \otimes A(t)$ *wrt* $\leq_\oplus$, *i.e.* such that $T(s,t) \otimes A(t) = \bigoplus_{t' \in \Omega^*} T(s,t') \otimes A(t')$.
Therefore, it is the problem of optimizing a measure called the *edit-distance between* s *and* A in [26]. The input language can also be expressed as a swTA, or, as a particular case, as a weighted context-free grammar, converted in turn

into a sw-VPA following Lemma 5. In the case of finite alphabets, the problem of searching, in a WTA language, for the best parse tree for a given, sometimes referred as *weighted parsing* (see [16,28] **more general problems**) is a particular case of SW parsing. Indeed, it corresponds to the case where T accepts only the pairs $\langle s, t\rangle$ such that s is the projection of t on Ω_{i}. This can be done with a single state q and with transition rules of the form:

$\mathsf{w}(q, \epsilon, a, q) = \mathbb{1}$ for all $a \in \Omega_{\mathsf{c}} \cup \Omega_{\mathsf{r}}$,
$\mathsf{w}(q, a, a, q) = \mathbb{1}$ for all $a \in \Omega_{\mathsf{i}}$,
$\mathsf{w}(q, a, b, q) = \mathbb{0}$ for all $a, b \in \Omega_{\mathsf{i}}$, $a \neq b$.

4.2 Computation

Proposition 5. *The problem of Symbolic Weighted parsing can be solved in PTIME in the size of the input swT, sw-VPA (or swTA) and input word, and the computation time of the functions of the label theory.*

Proof. Bar-Hillel construction □

4.3 Application to Automated Music Transcription

Symbolic Automated Music Transcription and analysis of music performances

Time Scales Real-Time Unit (RTU) = seconds
Musical-Time Unit (MTU) = number of measures
conversion via tempo value

Representation of Music Performances We consider symbolic representations of musical performances, as finite sequences of events. It corresponds to the concrete case of a MIDI file [2] recorded from an electronic keyboard, or the output of a transcription from audio files [3]. For the sake of simplicity, we shall only consider here the case of monophonic performances, where at most one note is sounding at a time. The approach however extends to the polyphonic case.

A music performance is a finite sequence of events in a set Σ. Every event $e \in \Sigma$ has attributes such from a finite domain, like a number of key for a note or a flag indicating that it is a rest (ON and OFF messages in [2]) and a velocity value (0..127 in [2])). Moreover, it contains a RTU value $\mathsf{ioi}(e)$ (real number) representing the time distance to the next event, or to the end of performance for the last event, also called *inter-onset interval*.

Representation of Music Scores Music score are represented as structured words made of timed events and parenthesized markups, akin of nested words [1].

We consider an alphabet Δ, every symbol of which is composed of a tag, in a finite set Ξ, and an MTU (rational) IOI duration value. It is partitioned into $\Delta = \Delta_{\mathsf{i}} \uplus \Delta_{\mathsf{c}} \uplus \Delta_{\mathsf{r}}$, like in Section 3. The symbols of Δ_{i} represent events: with tags

indicating a new note or grace-note (with null IOI), a rest or the continuation of the previous note (tie or dot). The elements of $\Delta_{\mathsf{c}} \uplus \Delta_{\mathsf{r}}$ are matched markups for describing the structure of the score, *i.e.* the hierarchical grouping of events, and also, importantly the division of time in measures, tuplets... (linearization of rhythm trees [20]...). They contain additional info such as tuple number, beaming policy...
The duration values of letters of Δ, in MTU (rational), can be computed with the markups and tags (*e.g.* grace note has duration 0).

Example 5. ...

Performance/Score Distance Computation We define a distance between performance and score representations by a $\mathsf{swT}\ T = \langle Q, \mathsf{in}, \mathsf{w}, \mathsf{out}\rangle$, over a semiring $\mathbb{S}$. ** detail the elements of $\mathbb{S}$ ....**

Every state of Q contains a tempo value in a finite domain (e.g. 30..300 bpm). This value can be fixed or recomputed by the T after reading each event, according to a perceptive/cognitive model of tempo such as [23] (also used in the context of score following [8]).

4.4 Transcription by SW Parsing

We assume a score language defined by a sw-VPA over the semiring $\mathbb{S}$ of Section 4.3.

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. association. Standard midi files specification.
3. E. Benetos, S. Dixon, Z. Duan, and S. Ewert. Automatic music transcription: An overview. *IEEE Signal Processing Magazine*, 36(1):20–30, 2018.
4. 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.
5. P. Bouyer, A. Petit, and D. Thérien. An algebraic approach to data languages and timed languages. *Information and Computation*, 182(2):137–162, 2003.

6. 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.
7. 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.
8. A. Cont. A coupled duration-focused architecture for realtime music to score alignment. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 32(6):974–987, 2010.
9. L. D'Antoni and R. Alur. Symbolic visibly pushdown automata. In *International Conference on Computer Aided Verification*, pages 209–225. Springer, 2014.
10. 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.
11. L. D'Antoni and M. Veanes. Automata modulo theories. *Communications of the ACM*, 64(5):86–95, 2021.
12. E. W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1(1):269–271, 1959.
13. M. Droste and W. Kuich. Semirings and formal power series. In *Handbook of Weighted Automata*, pages 3–28. Springer, 2009.
14. M. Droste, W. Kuich, and H. Vogler. *Handbook of weighted automata.* Springer Science & Business Media, 2009.
15. F. Foscarin, F. Jacquemard, P. Rigaux, and M. Sakai. A Parse-based Framework for Coupled Rhythm Quantization and Score Structuring. In *MCM 2019 - Mathematics and Computation in Music*, volume Lecture Notes in Computer Science of *Proceedings of the Seventh International Conference on Mathematics and Computation in Music (MCM 2019)*, Madrid, Spain, June 2019. Springer.
16. J. Goodman. Semiring parsing. *Computational Linguistics*, 25(4):573–606, 1999.
17. D. Grune and C. J. Jacobs. *Parsing Techniques.* Number 2nd edition in Monographs in Computer Science. Springer, 2008.
18. L. Huang. Advanced dynamic programming in semiring and hypergraph frameworks. In *In COLING*, 2008.
19. 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.
20. F. Jacquemard, P. Donat-Bouillud, and J. Bresson. A Structural Theory of Rhythm Notation based on Tree Representations and Term Rewriting. In D. M. Tom Collins and A. Volk, editors, *Mathematics and Computation in Music: 5th International Conference, MCM 2015*, volume 9110 of *Lecture Notes in Artificial Intelligence*, page 12, London, United Kingdom, June 2015. Oscar Bandtlow and Elaine Chew, Springer.
21. M. Kaminski and N. Francez. Finite-memory automata. *Theor. Comput. Sci.*, 134:329–363, November 1994.
22. W. Kuich. Semirings and formal power series: Their relevance to formal languages and automata. In *Handbook of formal languages*, pages 609–677. Springer, 1997.
23. E. W. Large and M. R. Jones. The dynamics of attending: How people track time-varying events. *Psychological review*, 106(1):119, 1999.
24. M. Mohri. Generic epsilon-removal and input epsilon-normalization algorithms for weighted transducers. *International Journal of Foundations of Computer Science*, 2002.

25. M. Mohri. Semiring frameworks and algorithms for shortest-distance problems. *Journal of Automata, Languages and Combinatorics*, 7(3):321–350, 2002.
26. M. Mohri. Edit-distance of weighted automata: General definitions and algorithms. *International Journal of Foundations of Computer Science*, 14(06):957–982, 2003.
27. M. Mohri. Edit-distance of weighted automata: General definitions and algorithms. *International Journal of Foundations of Computer Science*, 14(06):957–982, 2003.
28. 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.
29. M.-J. Nederhof. Weighted deductive parsing and Knuth's algorithm. *Computational Linguistics*, 29(1):135–143, 2003.
30. 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.
31. L. Segoufin. Automata and logics for words and trees over an infinite alphabet. In *Computer Science Logic*, volume 4207 of *LNCS*. Springer, 2006.
32. M. Y. Vardi. Linear-time model checking: automata theory in practice. In *International Conference on Implementation and Application of Automata*, pages 5–10. Springer, 2007.

A Edit-Distance

...algebraic definition of edit-distance of Mohri, in [26] distance d over $\Sigma^* \times \Sigma^*$ into a semiring $\mathbb{S} = (\mathbb{S}, \oplus, \mathbb{0}, \otimes, \mathbb{1})$.

Let $\Omega = \Sigma \cup \{\epsilon\} \times \Sigma \cup \{\epsilon\} \setminus \{(\epsilon, \epsilon)\}$, and let h be the morphism from Ω^* into $\Sigma^* \times \Sigma^*$ defined over the concatenation of strings of Σ^* (that removes the ϵ's). An *alignment* between 2 strings $s, t \in \Sigma^*$ is an element $\omega \in \Omega^*$ such that $h(\omega) = (s, t)$. We assume a base cost function $\Omega : \delta : \Omega \to S$, extended to Ω^* as follows (for $\omega \in \Omega^*$): $\delta(\omega) = \bigotimes_{0 \leq i < |\omega|} \delta(\omega_i)$.

Definition 6. *For $s, t \in \Sigma^*$, the edit-distance between s and t is $d(s, t) = \bigoplus_{\omega \in \Omega^* \; h(\omega) = (s,t)} \delta(\omega)$.*

e.g. Levenstein edit-distance: S is min-plus and $\delta(a, b) = 1$ for all $(a, b) \in \Omega$.