Symbolic Weighted Parsing and Automated Music Transcription

Florent Jacquemard

May 3, 2021

Weighted parsing [9, 17, 16] consists in analyzing a word in input *wrt* a weighted language model, *e.g.* a probabilistic context-free grammar (CFG). In such a model, 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 (or *abstract syntax tree*, *i.e.* the result of an analyze). A second operator $\oplus$, associative and commutative, is used to handle ambiguity of the model, by summing the weights of the possibly several (in general exponentially many) syntax trees associated to a given input word. Intuitively, $\oplus$ aims at selecting, or ranking, the syntax trees. The weight domain, equipped with these two operators is assumed, at minima, to form a *semiring* $\mathbb{S}$, such as a min-plus algebra of domain $\mathbb{R} \cup \{+\infty\}$, where $\oplus$ is min and $\otimes$ is plus. Some efficient specialized parsing algorithms [11] have been proposed in this context in order to compute the n best syntax trees of a given input word without having to enumerate them all. Generally based on dynamic programming, these algorithms rely on additional algebraic properties of $\mathbb{S}$. – see *e.g.* [11] for some NLP applications.

The various approaches to weighted parsing usually assume a finite input alphabet. Considering infinite or large alphabets can however be of practical interest, for instance in the context of processing strings in UTF16 encodings, for vulnerability detection in Web-applications (see *e.g.* [6]), for the representation of execution traces in model checking, or when dealing with sequences of timestamped events – we describe briefly at the end of the paper a parsing based approach to of automated music transcription, that is the conversion of a symbolic representation of a music performance into a structured music score.

We present here some weighted language models computing on words over infinite input alphabets - but with finite sets of states and transitions rules. In their transition rules, input symbols appear as variables and the weight associated to a transition rule is a function of these variables. This approach is close to the case of Symbolic Automata (SA) [6], except that the domain for weight values is not restricted to be Boolean, like for the guards in the rules of SA, but can be an arbitrary commutative semiring (assuming some

restrictions). The models defined here are finite automata (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 expressiveness, they correspond to a strict restriction of weighted CFG. Let A be a sw-VPA, associating $A(w) \in \mathbb{S}$ to a given a nested word w (representing a parse tree), and let a swT compute a distance d, in $\mathbb{S}$, between 2 strings over respectively an infinite input alphabet and the same (infinite) alphabet of A. Then, the problem of Symbolic Weighted Parsing is, given an input string s, to find a nested word w minimizing (according to the ranking defined by $\oplus$) the distance $d(s, w) \otimes A(w)$ between s and A, as defined in [15].

In Section 1 we introduce swA and swT, then sw-VPA are defined in Section **??**, where a polynomial 1-best algorithm is described that can be use to solve Symbolic Weighted Parsing. Finally, , we present an application of this approach to automated music transcription that has been implemented.

1 SW Automata and Transducers

We follow the approach of [15] for the computation of distances between words with weighted transducers, and propose models of weighted automata and weighted transducers handling infinite alphabets.

These models generalize weighted automata and transducers over finite alphabets, see e.g. [15], by labeling each transition with a weight functions 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 [18], but they can return values in an arbitrary semiring, where the latter guards are restricted to the Boolean semiring.

1.1 Semirings

We shall consider semiring domains for weight values. 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, $\otimes$ distributes over $\oplus$: $\forall x, y, z \in \mathbb{S}$, $x \otimes (y \oplus z) = (x \otimes y) \oplus (x \otimes z)$, and $\mathbb{0}$ is absorbing for $\otimes$: $\forall x \in \mathbb{S}$, $\mathbb{0} \otimes x = x \otimes \mathbb{0} = \mathbb{0}$.

A semiring $\mathbb{S}$ is *monotonic wrt* a partial ordering $\leq$ iff for all $x, y, z \in \mathbb{S}$, $x \leq y$ implies $x \oplus z \leq y \oplus z$, $x \otimes z \leq y \otimes z$ and $z \otimes x \leq z \otimes y$, and it is *superior wrt* $\leq$ iff for all $x, y \in \mathbb{S}$, $x \leq x \otimes y$ and $y \leq x \otimes y$ [10]. The latter property corresponds to the *non-negative weights* condition in shortest-path algorithms [7]. Intuitively, it means that combining elements always increase their weight. Note that when $\mathbb{S}$ is superior *wrt* $\leq$, then $\mathbb{1} \leq \mathbb{0}$ and moreover, for all $x \in \mathbb{S}$, $\mathbb{1} \leq x \leq \mathbb{0}$.

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 $\mathbb{S}$ and defined by: for all x and y, $x \leq_\oplus y$ iff $x \oplus y = x$. This ordering is sometimes defined in the opposite direction [8]; The above definition follows [14], and coincides than the usual ordering on the Tropical semiring (*min-plus*). It holds that $\mathbb{S}$ is monotonic *wrt* $\leq_\oplus$. 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 consider below infinite sums with $\oplus$. A semiring $\mathbb{S}$ is called *complete* 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 properties hold:

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 *semirings*

1.2 Label Theory

Let Σ and Δ be respectively an input and output *alphabets*, which are countable (finite or infinite) sets of symbols, 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 4 recursively enumerable sets: $\Phi_\epsilon \subseteq \mathbb{S}$, 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 $\Sigma \times \Delta \to \mathbb{S}$. Moreover, we assume that these sets are closed under the operators $\oplus$ and $\otimes$ of $\mathbb{S}$. More precisely, 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 $\Phi_{\Sigma,\Delta}$.

The same also holds for the binary sum operator $\oplus$.

Finally, it is assumed that the codomain of every function of Φ_Σ and Φ_Δ is a subset of Φ_ϵ. and 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.

necessary?

1.3 Definitions

Definition 2 *A* symbolic-weighted transducer *(*swT*)* T *over the input and output alphabet* Σ *and* Δ *and the semiring* $\mathbb{S}$ *is a tuple* $T = \langle Q, \mathsf{in}, \mathsf{w}, \mathsf{out}\rangle$*, 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* w *is a transition function from* $Q \times Q$ *into* $\langle \Phi_\epsilon, \Phi_\Sigma, \Phi_\Delta, \Phi_{\Sigma,\Delta}\rangle$*.*

We extend the above transition function into a function from $Q \times (\Sigma \cup \{\epsilon\}) \times (\Delta \cup \{\epsilon\}) \times Q$ into $\mathbb{S}$, also called w for simplicity, such that for all $q, q' \in Q$, $a \in \Sigma$, $b \in \Delta$, and with $\langle \phi_\epsilon, \phi_\Sigma, \phi_\Delta, \phi_{\Sigma,\Delta}\rangle = \mathsf{w}(q, q')$,

$$\begin{array}{rcl} \mathsf{w}(q, \epsilon, \epsilon, q') & = & \phi_\epsilon \\ \mathsf{w}(q, a, \epsilon, q') & = & \phi_\Sigma(a) \\ \mathsf{w}(q, \epsilon, b, q') & = & \phi_\Delta(b) \\ \mathsf{w}(q, a, b, q') & = & \phi_{\Sigma,\Delta}(a, b) \end{array}$$

These functions ϕ act as guards for the transducer's transitions, preventing a transition when they return the absorbing $\mathbb{0}$ of $\mathbb{S}$.

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 $s \in \Sigma^*$, $t \in \Delta^*$:

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

Recall that by convention, an empty sum with $\oplus$ is $\mathbb{0}$. The weight associated by T to $\langle s,t\rangle \in \Sigma^* \times \Delta^*$ is then defined as follows:

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

Example 3 *comparison of two sequences of timestamped events*

A *symbolic weighted automata* (swA) $A = \langle Q, \mathsf{in}, \mathsf{weight}, \mathsf{out}\rangle$ over Σ and $\mathbb{S}$ is defined in a similar way by simply omitting the output symbols, *i.e.* w is a function from $Q \times Q$ into $\langle \Phi_\epsilon, \Phi_\Sigma\rangle$, or equivalently from $Q \times (\Sigma \cup \{\epsilon\}) \times Q$ into $\mathbb{S}$.

1.4 Properties

Proposition 4 *Given a swT T over Σ, Δ and $\mathbb{S}$, and a word $s \in \Sigma^*$, one can construct a swA $A_{s,T}$ such that for all $t \in \Delta^*$, $A_{s,T}(t) = T(s,t)$.*

The construction time and size for $A_{s,T}$ are $O(|s|.\|T\|)$.

Proposition 5 *...removal of ϵ transitions for swA...*

2 SW Visibly Pushdown Automata

The following model generalizes Symbolic VPA [5] from Boolean semirings to arbitrary semiring weight domains.

2.1 Definition

Let Σ be a countable alphabet that we assume partitioned into three subsets $\Sigma_\mathsf{i} \uplus \Sigma_\mathsf{c} \uplus \Sigma_\mathsf{c}$ respectively called of *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 Φ_{Σ_c}, Φ_{Σ_r} and $\Phi_{\Sigma_\mathsf{c},\Sigma_\mathsf{r}}$. Moreover, we extend this theory with a set Φ_i of unary functions in $\Sigma_\mathsf{i} \to \mathbb{S}$, closed under $\oplus$ and $\otimes$.

Definition 6 *A* Symbolic Weighted Visibly Pushdown Automata *(sw-VPA) A over $\Sigma = \Sigma_\mathsf{i} \uplus \Sigma_\mathsf{c} \uplus \Sigma_\mathsf{r}$ and $\mathbb{S}$ is a tuple $T = \langle Q, P, \mathsf{in}, \mathsf{w_i}, \mathsf{w_c}, \mathsf{w_r}, \mathsf{w_e}, \mathsf{out}\rangle$, 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 1, we extend the above transition functions as follows for all $q, q' \in Q$, $p \in P$, $a \in \Sigma_{\mathsf{i}}$, $c \in \Sigma_{\mathsf{c}}$, $r \in \Sigma_{\mathsf{r}}$, overloading their names:

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

The intuition is the following for the above transitions.

w_{i} : read the input internal symbol a, change state to q'.

w_{c} : read the input call symbol c, push it to the stack along with p, change state to q'.

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. This value might be $\mathbb{0}$ in order to express that the symbols do not match.

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. In the case of a pushdown automaton, a configuration is composed of a state $q \in Q$ and a stack content $\theta \in \Theta^*$, where $\Theta = \Sigma_{\mathsf{c}} \times P$. Therefore, weight_A is a function from $Q \times \Theta^* \times \Sigma^* \times Q \times \Theta^*$ into $\mathbb{S}$.

$$
\begin{aligned}
\mathsf{weight}_A\big(\left[\begin{smallmatrix} q \\ \theta \end{smallmatrix}\right], a\,u, \left[\begin{smallmatrix} q' \\ \theta' \end{smallmatrix}\right]\big) &= \bigoplus_{q'' \in Q} \mathsf{w}_{\mathsf{i}}(q, a, q'') \otimes \mathsf{weight}_A\big(\left[\begin{smallmatrix} q'' \\ \theta \end{smallmatrix}\right], u, \left[\begin{smallmatrix} q' \\ \theta' \end{smallmatrix}\right]\big) \\
\mathsf{weight}_A\big(\left[\begin{smallmatrix} q \\ \theta \end{smallmatrix}\right], c\,u, \left[\begin{smallmatrix} q' \\ \theta' \end{smallmatrix}\right]\big) &= \bigoplus_{\substack{q'' \in Q \\ p \in P}} \mathsf{w}_{\mathsf{c}}\big(q, c, q'', p\big) \otimes \mathsf{weight}_A\big(\left[\begin{smallmatrix} q'' \\ c\,p \cdot \theta \end{smallmatrix}\right], u, \left[\begin{smallmatrix} q' \\ \theta' \end{smallmatrix}\right]\big) \\
\mathsf{weight}_A\big(\left[\begin{smallmatrix} q \\ c\,p \cdot \theta \end{smallmatrix}\right], r\,u, \left[\begin{smallmatrix} q' \\ \theta' \end{smallmatrix}\right]\big) &= \bigoplus_{q'' \in Q} \mathsf{w}_{\mathsf{r}}\big(q, c, p, r, q''\big) \otimes \mathsf{weight}_A\big(\left[\begin{smallmatrix} q'' \\ \theta \end{smallmatrix}\right], u, \left[\begin{smallmatrix} q' \\ \theta' \end{smallmatrix}\right]\big) \\
\mathsf{weight}_A\big(\left[\begin{smallmatrix} q \\ \bot \end{smallmatrix}\right], r\,u, \left[\begin{smallmatrix} q' \\ \theta' \end{smallmatrix}\right]\big) &= \bigoplus_{q'' \in Q} \mathsf{w}_{\mathsf{e}}(q, r, q'') \otimes \mathsf{weight}_A\big(\left[\begin{smallmatrix} q'' \\ \bot \end{smallmatrix}\right], u, \left[\begin{smallmatrix} q' \\ \theta' \end{smallmatrix}\right]\big)
\end{aligned}
$$

where $\bot$ denotes the empty stack and $c\,p \cdot \theta$ denotes a stack with the pair made of c and p on its top and θ as the rest of stack.
The weight associated by A to $s \in \Sigma^*$ 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 7 *structured words... intro language of music notation ?*

2.2 Properties

The class of sw-VPA is closed under the binary operators of the underlying semiring.

Proposition 8 *Let A_1 and A_2 be two (sw-VPA) over the same Σ 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 $s \in \Sigma^*$, $(A_1 \oplus A_2)(s) = A_1(s) \oplus A_2(s)$ and $(A_1 \otimes A_2)(s) = A_1(s) \otimes A_2(s)$.*

The construction is essentially the same as the one of [5], in the case of the Boolean semiring.

2.3 Best-first Search

Assume that the semiring $\mathbb{S}$ is commutative, idempotent, superior, total and complete. 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 Θ, and for every two states $q, q' \in Q$ and $\sigma \in \{\bot, \top\}$, let

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

Since $\mathbb{S}$ is complete, this infinite sum is well defined, and since $\leq_\oplus$ is assumed total, it is the minimum in Σ^* 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 $\theta \in \Theta^*$ (possibly empty), and ending in state q with the same stack θ, such that moreover the computation does not touch a symbol of θ. 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 θ, 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 θ.

The algorithm 1 constructs iteratively a 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

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

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 9 *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,

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

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

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

Proposition 11 *For a sw-VPA A over a commutative, idempotent, superior, total and complete semiring and an alphabet Σ with a k-convex label theory, one can construct in PTIME a word $s\in\Sigma^*$ such that $A(s)$ is minimal wrt the natural ordering for $\mathbb{S}$.*

3 Application

Symbolic Automated Music Transcription and analysis of music performances

3.1 Time Scales

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

3.2 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*.

3.3 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 2. 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 [12]...). 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 12 ...

3.4 Performance/Score Distance Computation

We define a distance between performance and score representations by a 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 [13] (also used in the context of score following [4]).

3.5 Transcription by Best-first Search

We assume a score language defined by a sw-VPA over the semiring $\mathbb{S}$ of Section 3.4.

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] 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.

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

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

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

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

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

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

[11] 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.

[12] 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.

[13] E. W. Large and M. R. Jones. The dynamics of attending: How people track time-varying events. *Psychological review*, 106(1):119, 1999.

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

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

[16] 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.

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

[18] M. Veanes. Applications of symbolic finite automata. In *International Conference on Implementation and Application of Automata*, pages 16–23. Springer, 2013.

A Edit-Distance

...algebraic definition of edit-distance of Mohri, in [15] 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 13 *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 $(a, b) \in \Omega$.