Weighted Visibly Pushdown Automata and Automated Music Transcription

April 29, 2021

Abstract

...

Symbolic Weighted (SW) extension of symbolic automata where...

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 $\otimes$: $\forall x \in \mathbb{S}$, $\mathbb{0} \otimes x = x \otimes \mathbb{0} = \mathbb{0}$. In the application presented in this paper, intuitively, $\oplus$ selects an optimal value amongst two values and $\otimes$ combines two values into a single value.

and let $\langle \mathbb{S}, \oplus, \mathbb{0}, \otimes, \mathbb{1} \rangle$ be a semiring,

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 $y \leq x \otimes y$ [5]. The latter property corresponds to the *non-negative weights* condition in shortest-path algorithms [3]. 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 $x \in \mathbb{S}$, $\mathbb{1} \leq x \leq \mathbb{0}$.

Every idempotent semiring $\mathbb{S}$ induces a partial ordering $\leq_{\mathbb{S}}$ called the *natural ordering* of $\mathbb{S}$ and defined by: for all x and y, $x \leq_{\mathbb{S}} y$ iff $x \oplus y = x$. This ordering is sometimes defined in the opposite direction [4]; The above definition follows [7], and coincides than the usual ordering on the Tropical semiring (*min-plus*). It holds that $\mathbb{S}$ is monotonic *wrt* $\leq_{\mathbb{S}}$. An idempotent Semiring $\mathbb{S}$ is called *total* if it $\leq_{\mathbb{S}}$ 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: $\displaystyle\bigoplus_{i \in \emptyset} x_i = \mathbb{0}$, $\quad \forall j \in \mathbb{N}, \displaystyle\bigoplus_{i \in \{j\}} x_i = x_j$,

$\forall j, k \in \mathbb{N}, j \neq k, \displaystyle\bigoplus_{i \in \{j,k\}} x_i = x_j \oplus x_k$, *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 $\bigoplus_{i\in I}(x_i \otimes y) = (\bigoplus_{i\in I} x_i) \otimes y$.

1 SW Automata and Transducers

We follow the approach of [8] for the computation of distances between words with transducers.

The following definition of weighted transducers over infinite alphabets generalizes weighted transducers over finite alphabets, see e.g. [8], by considering weight functions generalizing the guards of symbolic automata

Let Σ and Δ be respectively an input and output *alphabets*, which are finite or infinite sets of symbols, and let $\mathbb{S}$ be a semiring. A *label theory* is a 4-uplet of recursively enumerable sets: Φ_0 containing constant functions valued in $\mathbb{S}$, 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 each of these sets is closed under $\oplus$ and $\otimes$, 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.

Definition 1 *A* symbolic-weighted transducer 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* $\langle \Phi_0, \Phi_\Sigma, \Phi_\Delta, \Phi_{\Sigma,\Delta}\rangle$.

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

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

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:

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

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 of $Q \times Q$ into $\langle \Phi_0, \Phi_\Sigma\rangle$, or equivalently from $Q \times (\Sigma \cup \{\epsilon\}) \times Q$ into $\mathbb{S}$.

Proposition 2 *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 of $A_{s,T}$ are $O(|s|.\|T\|)$.

2 SW Visibly Pushdown Automata

The following model generalizes Symbolic VPA [2] from Boolean semirings to arbitrary semiring weight domains.

Let Σ be an input alphabet, finite (large) or infinite, that we assume partitioned into :

- a set Σ_{i} of *internal symbols* denoted a,
- a set Σ_{c} of *call symbols* denoted $\langle_a$,
- a set Σ_{r} of *return symbols* denoted $_a\rangle$.

In order to simplify notations, and following the definition of Section 1, we shall write respectively Φ_{i}, Φ_{c}, Φ_{r} and Φ_{cr} for $\Phi_{\Sigma_{\mathsf{i}}}$, $\Phi_{\Sigma_{\mathsf{c}}}$, $\Phi_{\Sigma_{\mathsf{r}}}$ and $\Phi_{\Sigma_{\mathsf{c}}, \Sigma_{\mathsf{r}}}$,

Definition 3 *A* Symbolic Weighted Visibly Pushdown Automata *(SWVPA) A over the input $\Sigma = \Sigma_{\mathsf{i}} \uplus \Sigma_{\mathsf{c}} \uplus \Sigma_{\mathsf{r}}$ and the semiring $\mathbb{S}$ 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 $q, q' \in Q$, $p \in P$, $a \in \Sigma_\mathsf{i}$, $\langle_c \in \Sigma_\mathsf{c}$, ${}_r\rangle \in \Sigma_\mathsf{r}$, overloading the names for simplicity:

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

The intuition is the following for the above transitions.

- $\mathsf{w_i}$: read the input internal symbol a, change state to q'.
- $\mathsf{w_c}$: read the input symbol $\langle_c$, 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 $\langle_c$ and p, read the input symbol ${}_r\rangle$, change state to q'. In this case, the weight function ϕ_r checks a matching between the call and return symbols.
- $\mathsf{w_e}$: when the stack is empty, read the input symbol $\langle_r$, 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 $\mathbb{S}$.

$$
\begin{array}{rcl}
\mathsf{weight}_A\big(\big[\begin{smallmatrix} q \\ \theta \end{smallmatrix}\big], au, \big[\begin{smallmatrix} q' \\ \theta' \end{smallmatrix}\big]\big) & = & \displaystyle\bigoplus_{q'' \in Q} \mathsf{w_i}(q, a, q'') \otimes \mathsf{weight}_A\big(\big[\begin{smallmatrix} q'' \\ \theta \end{smallmatrix}\big], u, \big[\begin{smallmatrix} q' \\ \theta' \end{smallmatrix}\big]\big) \\
\mathsf{weight}_A\big(\big[\begin{smallmatrix} q \\ \theta \end{smallmatrix}\big], \langle_c u, \big[\begin{smallmatrix} q' \\ \theta' \end{smallmatrix}\big]\big) & = & \displaystyle\bigoplus_{\substack{q'' \in Q \\ p \in P}} \mathsf{w_c}\big(q, \langle_c, q'', p\big) \otimes \mathsf{weight}_A\big(\big[\begin{smallmatrix} q'' \\ \langle_c\, p \cdot \theta \end{smallmatrix}\big], u, \big[\begin{smallmatrix} q' \\ \theta' \end{smallmatrix}\big]\big) \\
\mathsf{weight}_A\big(\big[\begin{smallmatrix} q \\ \langle_c\, p \cdot \theta \end{smallmatrix}\big], {}_r\rangle u, \big[\begin{smallmatrix} q' \\ \theta' \end{smallmatrix}\big]\big) & = & \displaystyle\bigoplus_{q'' \in Q} \mathsf{w_r}\big(q, \langle_c, p, {}_r\rangle, q''\big) \otimes \mathsf{weight}_A\big(\big[\begin{smallmatrix} q'' \\ \theta \end{smallmatrix}\big], u, \big[\begin{smallmatrix} q' \\ \theta' \end{smallmatrix}\big]\big) \\
\mathsf{weight}_A\big(\big[\begin{smallmatrix} q \\ \bot \end{smallmatrix}\big], {}_r\rangle u, \big[\begin{smallmatrix} q' \\ \theta' \end{smallmatrix}\big]\big) & = & \displaystyle\bigoplus_{q'' \in Q} \mathsf{w_e}(q, {}_r\rangle, q'') \otimes \mathsf{weight}_A\big(\big[\begin{smallmatrix} q'' \\ \bot \end{smallmatrix}\big], u, \big[\begin{smallmatrix} q' \\ \theta' \end{smallmatrix}\big]\big)
\end{array}
$$

where $\bot$ denotes the empty stack and $\langle_c\, p \cdot \theta$ denotes a stack with the pair made of $\langle_c$ 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 by empty stack computation:

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

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

A musical performance is represented symbolically as a finite sequence of timestamped events. similar to piano roll representation [9] chap.1
We consider an infinite alphabet Σ of MIDI-like events made of:

- a timestamp in RTU
  or IOI in RTU
- a pitch value in 0..127
- ON | OFF flag
- a velocity value in 0..127

3.3 Representation of Music Scores

we consider here the case of monodies for the sake of simplificity.

A music score ris epresented as a structured word = sequence of quantified events + markups, see nested words [1].

We consider an alphabet Δ, every symbol of which is composed of a tag, in a finite set Ξ, and an IOI duration value. Moreover, Δ is partitioned into $\Delta = \Delta_{\mathsf{i}} \uplus \Delta_{\mathsf{c}} \uplus \Delta_{\mathsf{r}}$, like in Section 2.
The elements of Δ_{i} represent events: whose tags are one of:

- continuation (0) : tie or dot
- note, grace note (pitch) or chord (pitch+)
- rest
- ...

The elements of Δ_{c} and Δ_{r} are markups for the representation of groups of events (linearization of rhythm trees [6]...) - parentheses for time divisions : tuplets, bars... tag contain info such as tuple number, beaming policy...
The date or duration of events, in MTU (rational), can be computed with the markups and tags (e.g. grace note has duration 0).
There are simultaneous events, since grace notes has duration 0. They are ordered.
Finite bound on the number of duration ratio. ?

3.4 Performance/Score Distance Computation

with a transducer

References

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

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

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

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

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

[6] 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.

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

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

[9] M. Müller. *Fundamentals of music processing: Audio, analysis, algorithms, applications.* Springer, 2015.

A Edit-Distance

...algebraic definition of edit-distance of Mohri, in [8] 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 Ω : $\delta : \Omega \to S$, extended to Ω^* as follows (for $\omega \in \Omega^*$): $\delta(\omega) = \bigotimes_{0 \leq i < |\omega|} \delta(\omega_i)$.

Definition 4 *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 $(a, b) \in \Omega$.