Symbolic Weighted Language Models and Quantitative Parsing over Infinite Alphabets

Florent Jacquemard
Inria & CNAM, Paris, France

Philippe Rigaux
CNAM, Paris, France

Lydia Rodrigez de la Nava
Inria & CNAM, Paris, France

Abstract

We propose a framework for weighted parsing over infinite alphabets. It is based on language models called Symbolic Weighted Automata (swA) at the joint between Symbolic Automata (sA) and Weighted Automata (wA), as well as Transducers (swT) and Visibly Pushdown (sw-VPA) variants. Like sA, swA deal with large or infinite input alphabets, and like wA, they output a weight value in a semiring domain. The transitions of swA are labeled by functions from an infinite alphabet into the weight domain. This generalizes sA, whose transitions are guarded by Boolean predicates overs symbols in an infinite alphabet, and also wA, whose transitions are labeled by constant weight values, and who deal only with finite alphabets. We present some properties of swA, swT and sw-VPA models, that we use to define and solve a variant of parsing over infinite alphabets. We illustrate the model with a motivating application to automated music transcription.

2012 ACM Subject Classification Theory of computation → Quantitative automata

Keywords and phrases weighted automata, symbolic automata, visibly pushdown automata, parsing

1 Introduction

Parsing is the problem of structuring a linear representation (a finite word) according to a language model. Most of the context-free parsing approaches [16] assume a finite and reasonably small input alphabet. Such a restriction makes perfect sense in the context of NLP tasks such as constituency parsing, programming languages compilers or interpreters. Considering large or infinite alphabets can however be of practical interest, when dealing with large characters encodings such as UTF-16 [9], analysis of data streams, serialization of structured documents [27, 26], or processing timed execution traces [4].

The latter case is related to a motivation of the present work: automated music transcription. Representations that capture music performances are essentially linear: audio files, or the widely used MIDI format [28]. Such representations ignore the hierarchical structures that frame the conception of music, at least in the western area. These structures, on the other hand, are present, either explicitly or implicitly, in Common Western Music Notation [15]: Music scores are partitioned in measures, measures in beats, and beats can be further recursively divided. It follows that music events do not occur at arbitrary timestamps, but respect a discrete partitioning of the timeline incurred by these recursive divisions. The *transcription problem* takes as input a linear representation (audio or MIDI) and aims at re-constructing these structures by mapping input events to this hierarchical rhythmic space. It can therefore be stated as a parsing problem [13] over an infinite alphabet of timed events.

Various extensions of language models for handling infinite alphabets have been studied. Some automata with memory extensions allow restricted storage and comparison of input symbols, (see [27] for a survey), with pebbles for marking positions [26], registers [19], or the possibility to compute on subsequences with the same attribute values [3]. The automata at the core of model checkers compute on input symbols represented by large bitvectors [29]

$$
\begin{tikzcd}
 & \begin{array}{c} \mathsf{FA} : \Sigma_{\mathsf{fin}}^{*} \to \mathbb{B} \\ q \xrightarrow{a} q' \quad a \in \Sigma_{\mathsf{fin}} \end{array} \arrow[dl] \arrow[dr] & \\
\begin{array}{c} \mathsf{WA} : \Sigma_{\mathsf{fin}}^{*} \to \mathbb{S} \\ q \xrightarrow{a,w} q' \quad a \in \Sigma_{\mathsf{fin}}, w \in \mathbb{S} \end{array} \arrow[dr] & & \begin{array}{c} \mathsf{SA} : \Sigma_{\mathsf{inf}}^{*} \to \mathbb{B} \\ q \xrightarrow{\phi} q' \quad \phi : \Sigma_{\mathsf{inf}} \to \mathbb{B} \end{array} \arrow[dl] \\
 & \begin{array}{c} \mathsf{SWA} : \Sigma_{\mathsf{inf}}^{*} \to \mathbb{S} \\ q \xrightarrow{\phi} q' \quad \phi : \Sigma_{\mathsf{inf}} \to \mathbb{S} \end{array} &
\end{tikzcd}
$$

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

(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) [8, 9], transitions are guarded by predicates over infinite domains. With appropriate closure conditions on the sets of such predicates, all the good properties enjoyed over finite alphabets are preserved.

Other extensions of language models help in dealing with non-determinism by computing weight values. With an ambiguous grammar, there may exist several derivations (*abstract syntax trees* – AST) yielding one input word. The association of one weight value to each AST permits to select a best one (or n bests). In *weighted language models* [14, 25, 24], like *e.g.* probabilistic context-free grammars and weighted automata (wA) [12], a weight is associated to each transition rule, and the rule's weights can be combined with an associative product operator $\otimes$ to yield the weight of an AST. A second operator $\oplus$ is moreover used to resolve the ambiguity raised by the existence of several (in general exponentially many) AST associated to a given input word. Typically, $\oplus$ selects the best of two weight values. The weight domain, equipped with these two operators is, at minima, a *semiring* where $\oplus$ can be extended to infinite sums, such as the Viterbi semiring and the tropical min-plus algebra

In this paper, we present a framework for weighted parsing over infinite input alphabets. It is based on *symbolic weighted* finite states language models (swM), generalizing the Boolean guards of sA to functions into an arbitrary semiring, and generalizing also wA, by handling infinite alphabets (Figure 1). In short, a transition rule $q \xrightarrow{\phi} q'$, from state q to q', is labeled by a function ϕ associating to every input symbol a a weight value $\phi(a)$ in a semiring $\mathbb{S}$.

The framework relies on several language models: symbolic-weighted automata (swA), transducers (swT), and pushdown automata with a visibility restriction [2] (sw-VPA). A swT defines a distance $T(s,t)$ between finite words s and t over infinite alphabets. A sw-VPA operates sequentially on *nested words* [2], structured with markup symbols (parentheses), and describing linearizations of trees. A sw-VPA A associates a weight value $A(t)$ to a given nested word t, which is itsef the linearization of a weighted AST. Then, given an input word s, the *SW-parsing* problem aims at finding t minimizing $T(s,t) \otimes A(t)$, called the distance between s and A in [22], *wrt* the ranking defined by $\oplus$Like weighted-parsing methods [14, 25, 24], our approach proceeds in two steps. The first step is an intersection (Bar-Hillel construction [16]) where, given a swT T, a sw-VPA A, and an input word s, a sw-VPA B is built, such that for all t, $B(t) = T(s,t) \otimes A(t)$.In the second step, a best AST t is found by applying to B a best search algorithm similar to the shortest distance in graphs [21, 18].

The main contributions of the paper are: (i) the introduction of automata, swA, transducers, swT (Section 3), and visibly pushdown automata sw-VPA (Section 4), generalizing the corresponding classes of symbolic and weighted models, (ii) a polynomial best-search algorithm for sw-VPA, and (iii) a uniform framework (Section 5) for parsing over infinite alphabets, the keys to which are ($iii.a$) the swT-based definition of generic edit distances between input and output (yield) words, and ($iii.b$) the use, convenient in this context, of nested words, and sw-VPA, instead of syntax trees and grammars.

▶ **Example 1.** We illustrate our framework with a very simplified running example of *music transcription*: a given *timeline* of musical events from an infinite alphabet Σ as input, is parsed into a structured music score. Input events of Σ are pairs $\mu : \tau$, where μ is a MIDI pitch [28], and $\tau \in \mathbb{Q}$ is a timestamp in seconds. Such inputs typically correspond to the recording of a live performance, *e.g.* $I = 69:0.07, 71:0.72, 73:0.91, 74:1.05, 76:1.36, 77:1.71$.

The output of parsing is a sequence of timed symbols $\nu : \tau'$ in an alphabet Δ, where ν represents an *event* (or *note*), specified by its *pitch name* (*e.g.*, A4, G5, *etc.*), an event *continuation* (symbol '$-$', see Example 8), or a *markup* (opening or closing parenthesis). The temporal information τ' is either a time interval, for the opening parentheses (representing the duration between the parenthesis and the matching closing one), or a timestamp, for the other symbols. The time points in τ' belong to a rhythmic "grid" obtained from recursive divisions: whole notes (𝅝) split in halves (𝅗𝅥), halves in quarters (♩), eights (♪), *etc.* For instance, the output score , corresponds to a hierarchical structure that can be linearized as the sequence $O = \langle_{\mathsf{m}} : [0,1],\ \langle_2 : [0,1],\ \mathrm{A4} : 0,\ \langle_2 : [\frac{1}{2}, 1],\ - : \frac{1}{2},\ \langle_2 : [\frac{3}{4}, 1],\ \mathrm{B4} : \frac{3}{4},\ \mathrm{C\sharp 5} : \frac{7}{8},\ \rangle_2 : 1,\ \rangle_2 : 1,\ \rangle_2 : 1,\ \langle_{\mathsf{m}} : [1,2],\ \langle_3 : [1,2],\ \mathrm{D5} : 1,\ \mathrm{E5} : \frac{4}{3},\ \mathrm{F5} : \frac{5}{3},\ \rangle_3 : 2,\ \rangle_{\mathsf{m}} : 2,\ \rangle_{\mathsf{m}} : 2$. The opening markups $\langle_{\mathsf{m}}$ delimit *measures*, which are time intervals of duration 1 in this example, while the subsequences of O between markups $\langle_d$ and $\rangle_d$, for some natural number d, represent a division of the current time interval into d sub-intervals of equal duration $\frac{\ell}{d}$ where ℓ is the length of the time interval attached to $\langle_d$.

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

2 Preliminary Notions

Semirings. 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 $\mathbb{1}$, such that:

- $\oplus$ is commutative: $\langle \mathbb{S}, \oplus, \mathbb{0} \rangle$ is a commutative monoid and $\langle \mathbb{S}, \otimes, \mathbb{1} \rangle$ a monoid,
- $\otimes$ distributes over $\oplus$:
  $\forall x, y, z \in \mathbb{S}$, $x \otimes (y \oplus z) = (x \otimes y) \oplus (x \otimes z)$, and $(x \oplus y) \otimes z = (x \otimes z) \oplus (y \otimes z)$,
- $\mathbb{0}$ is absorbing for $\otimes$: $\forall x \in \mathbb{S}$, $\mathbb{0} \otimes x = x \otimes \mathbb{0} = \mathbb{0}$.

In the models presented in this paper, $\oplus$ selects an optimal value from two given values, in order to handle non-determinism, and $\otimes$ combines two values into a single one. A semiring $\mathbb{S}$ is *commutative* if $\otimes$ is commutative. It is *idempotent* if for all $x \in \mathbb{S}$, $x \oplus x = x$. Every idempotent semiring $\mathbb{S}$ induces a partial ordering $\leq_{\oplus}$ called the *natural ordering* of $\mathbb{S}$ [21] defined, by: for all $x, y \in \mathbb{S}$, $x \leq_{\oplus} y$ iff $x \oplus y = x$. The natural ordering is sometimes defined in the opposite direction [11]; We follow here the direction that coincides with the usual ordering on the Tropical semiring *min-plus* (Figure 2). 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 $x \oplus y = y$.

▶ **Lemma 2** (Monotony, [21]). *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 $z \otimes x \leq_{\oplus} z \otimes y$.*

	domain	$\oplus$	$\otimes$	$\mathbb{0}$	$\mathbb{1}$
Boolean	$\{\bot, \top\}$	$\vee$	$\wedge$	$\bot$	$\top$
Counting	$\mathbb{N}$	$+$	$\times$	0	1
Viterbi	$[0,1] \subset \mathbb{R}$	max	$\times$	0	1
Tropical min-plus	$\mathbb{R}_+ \cup \{\infty\}$	min	$+$	∞	0

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

We then say the $\mathbb{S}$ is *monotonic wrt* $\leq_\oplus$. Another important semiring property in the context of optimization is superiority [17], which corresponds to the *non-negative weights* condition in shortest-path algorithms [10]. Intuitively, it means that combining elements with $\otimes$ always increase their weight. Formally, it is defined as the property (i) below.

▶ **Lemma 3** (Superiority, Boundedness). *Let $\langle \mathbb{S}, \oplus, \mathbb{0}, \otimes, \mathbb{1}\rangle$ be an idempotent semiring. The two following statements are equivalent:*
i. for all $x, y \in \mathbb{S}$, $x \leq_\oplus x \otimes y$ and $y \leq_\oplus x \otimes y$
ii. for all $x \in \mathbb{S}$, $\mathbb{1} \oplus x = \mathbb{1}$.

Proof. $(ii) \Rightarrow (i)$: $x \oplus (x \otimes y) = x \otimes (\mathbb{1} \oplus y) = x$, by distributivity of $\otimes$ over $\oplus$. Hence $x \leq_\oplus x \otimes y$. Similarly, $y \oplus (x \otimes y) = (\mathbb{1} \oplus x) \otimes y = y$, hence $y \leq_\oplus x \otimes y$. $(i) \Rightarrow (ii)$: by the second inequality of (i), with $y = \mathbb{1}$, $\mathbb{1} \leq_\oplus x \otimes \mathbb{1} = x$, *i.e.*, by definition of $\leq_\oplus$, $\mathbb{1} \oplus x = \mathbb{1}$. ◀

In [17], when property (i) holds, $\mathbb{S}$ is called *superior wrt* $\leq_\oplus$. It implies (proof of Lemma 3) 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, for all $x \in \mathbb{S}$, $\mathbb{1} \leq_\oplus x \leq_\oplus \mathbb{0}$. From an optimization point of view, it means that $\mathbb{1}$ is the best value, and $\mathbb{0}$ the worst. In [21], $\mathbb{S}$ with the property (ii) of Lemma 3 is called *bounded* – we shall use this term in the rest of the paper. It implies that, when looking for a best path in a graph whose edges are weighted by values of $\mathbb{S}$, the loops can be safely avoided, because, for all $x \in \mathbb{S}$ and $n \geq 1$, $x \oplus x^n = x \otimes (\mathbb{1} \oplus x^{n-1}) = x$.

▶ **Lemma 4.** *Every bounded semiring is idempotent.*

Proof. By boundedness, $\mathbb{1} \oplus \mathbb{1} = \mathbb{1}$, and idempotency follows by multiplying both sides by x and distributing. ◀

We need infinite sums with $\oplus$. A semiring $\mathbb{S}$ is called *complete* [12] if it has an operation $\bigoplus_{i \in I} x_i$ for every family $(x_i)_{i \in I}$ of elements of $dom(\mathbb{S})$ over an index set $I \subset \mathbb{N}$, such that:
i. infinite sums extend finite sums:
$$\bigoplus_{i \in \emptyset} x_i = \mathbb{0}, \quad \forall j \in \mathbb{N}, \bigoplus_{i \in \{j\}} x_i = x_j, \forall j,k \in \mathbb{N}, j \neq k, \bigoplus_{i \in \{j,k\}} x_i = x_j \oplus x_k,$$
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 $\bigoplus_{i \in I} (x_i \otimes y) = (\bigoplus_{i \in I} x_i) \otimes y$.

Label theory. We now define the functions labeling the transitions of SW automata and transducers, generalizing the Boolean algebras of [8]. We consider *alphabets*, which are countable sets of symbols denoted Σ, Δ,... Σ^* is the set of finite sequences (*words*) over Σ, ε the empty word, $\Sigma^+ = \Sigma^* \setminus \{\varepsilon\}$, and uv denotes the concatenation of $u, v \in \Sigma^*$.

Given a semiring $\langle \mathbb{S}, \oplus, \mathbb{0}, \otimes, \mathbb{1} \rangle$, a *label theory* over $\mathbb{S}$ is a set $\bar{\Phi}$ of recursively enumerable sets denoted Φ_Σ, containing unary functions of type $\Sigma \to \mathbb{S}$, or $\Phi_{\Sigma,\Delta}$, containing binary functions $\Sigma \times \Delta \to \mathbb{S}$, and such that:

- for all $\Phi_{\Sigma,\Delta} \in \bar{\Phi}$, we have $\Phi_\Sigma \in \bar{\Phi}$ and $\Phi_\Delta \in \bar{\Phi}$
- every $\Phi_\Sigma \in \bar{\Phi}$ contains all the constant functions from Σ into $\mathbb{S}$,
- for all $\alpha \in \mathbb{S}$ and $\phi \in \Phi_\Sigma$, $\alpha \otimes \phi : x \mapsto \alpha \otimes \phi(x)$, and $\phi \otimes \alpha : x \mapsto \phi(x) \otimes \alpha$ belong to Φ_Σ, and similarly for $\oplus$ and for $\Phi_{\Sigma,\Delta}$
- for all $\phi, \phi' \in \Phi_\Sigma$, $\phi \otimes \phi' : x \mapsto \phi(x) \otimes \phi'(x)$ belongs to Φ_Σ
- for all $\eta, \eta' \in \Phi_{\Sigma,\Delta}$ $\eta \otimes \eta' : x, y \mapsto \eta(x, y) \otimes \eta'(x, y)$ belongs to $\Phi_{\Sigma,\Delta}$
- for all $\phi \in \Phi_\Sigma$ and $\eta \in \Phi_{\Sigma,\Delta}$, $\phi \otimes_1 \eta : x, y \mapsto \phi(x) \otimes \eta(x, y)$ and $\eta \otimes_1 \phi : x, y \mapsto \eta(x, y) \otimes \phi(x)$ belong to $\Phi_{\Sigma,\Delta}$
- for all $\psi \in \Phi_\Delta$ and $\eta \in \Phi_{\Sigma,\Delta}$, $\psi \otimes_2 \eta : x, y \mapsto \psi(y) \otimes \eta(x, y)$ and $\eta \otimes_2 \psi : x, y \mapsto \eta(x, y) \otimes \psi(y)$ belong to $\Phi_{\Sigma,\Delta}$
- similar closures hold for $\oplus$.

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

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

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

▶ **Example 5.** We return to Example 1. Let Δ_i be the subset of Δ without markup symbols. In order to align the input in Σ^* with a music score, we must account for the expressive timing of human performance that results in small time shifts between an input event of Σ and the corresponding notation event in Δ_i. These shifts can be weighted as the time distance between both, computed in the tropical semiring by $\delta \in \Phi_{\Sigma,\Delta_\mathsf{i}}$, defined as follows:

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

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

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

▶ **Definition 6.** *A label theory is called* effective *when for all $\phi \in \Phi_\Sigma$ and $\eta \in \Phi_{\Sigma,\Delta}$, $\bigoplus_\Sigma \phi$, $\bigoplus_\Delta \bigoplus_\Sigma \eta$, and $\bigoplus_\Sigma \bigoplus_\Delta \eta$ can be effectively computed from ϕ and η, and moreover, the number of symbols reaching these bounds is finite and can be effectively computed.*

3 SW Automata and Transducers

We follow the approach of [22] for the computation of distances between words and languages and extend it to infinite alphabets. The models introduced in this section generalize weighted automata and transducers [12] by labeling each transition with a weight function that takes the input and output symbols as parameters. These functions are similar to the guards of symbolic automata [8, 9], but the latter guards are restricted to the Boolean semiring.
Let $\mathbb{S}$ be a commutative semiring, Σ and Δ be alphabets called respectively *input* and *output*, and $\bar{\Phi}$ be a label theory over $\mathbb{S}$ containing Φ_Σ, Φ_Δ, $\Phi_{\Sigma,\Delta}$.

▶ **Definition 7.** *A* symbolic-weighted transducer *(swT) over Σ, Δ, $\mathbb{S}$ and $\bar{\Phi}$ is a tuple $T = \langle Q, \mathsf{in}, \bar{\mathsf{w}}, \mathsf{out}\rangle$, where Q is a finite set of states, $\mathsf{in} : Q \to \mathbb{S}$ (respectively $\mathsf{out} : Q \to \mathbb{S}$) are functions defining the weight for entering (respectively leaving) computation in a state, and $\bar{\mathsf{w}}$ is a triplet of transition functions $\mathsf{w}_{10} : Q \times Q \to \Phi_\Sigma$, $\mathsf{w}_{01} : Q \times Q \to \Phi_\Delta$, and $\mathsf{w}_{11} : Q \times Q \to \Phi_{\Sigma,\Delta}$.*

We call *number of transitions* of T the number of pairs of states $q, q' \in Q$ such that w_{10} or w_{01} or w_{11} is not the constant $\mathbb{0}$. For convenience, we shall sometimes present transitions as functions of $Q \times (\Sigma \cup \{\varepsilon\}) \times (\Delta \cup \{\varepsilon\}) \times Q \to \mathbb{S}$, overloading the function names, such that, for all $q, q' \in Q$, $a \in \Sigma$, $b \in \Delta$:

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

The swT T computes on pairs $\langle s, t\rangle \in \Sigma^* \times \Delta^*$, s and t, being respectively called *input* and *output* word. T is based on an intermediate function weight_T defined recursively, for every states $q, q' \in Q$, and every pairs of strings $\langle s, t\rangle \in \Sigma^* \times \Delta^*$.

$$\mathsf{weight}_T(q, \varepsilon, \varepsilon, q') = \mathbb{1} \quad \text{if } q = q' \text{ and } \mathbb{0} \text{ otherwise} \tag{1}$$

$$\begin{aligned} \mathsf{weight}_T(q, s, t, q') = & \bigoplus_{\substack{q'' \in Q \\ s = au,\, a \in \Sigma}} \mathsf{w}_{10}(q, a, \varepsilon, q'') \otimes \mathsf{weight}_T(q'', u, t, q') \\ & \oplus \bigoplus_{\substack{q'' \in Q \\ t = bv,\, b \in \Delta}} \mathsf{w}_{01}(q, \varepsilon, b, q'') \otimes \mathsf{weight}_T(q'', s, v, q') \\ & \oplus \bigoplus_{\substack{q'' \in Q \\ s = au,\, t = bv}} \mathsf{w}_{11}(q, a, b, q'') \otimes \mathsf{weight}_T(q'', u, v, q') \end{aligned}$$

We recall that, by convention (Section 2), an empty sum with $\bigoplus$ is equal to $\mathbb{0}$. Intuitively, a transition $\mathsf{w}_{ij}(q, a, b, q')$ is interpreted as follows: when reading a and b in the input and output words, increment the current position in the input word if and only if $i = 1$, and in the output word iff $j = 1$, and change state from q to q'. When $a = \varepsilon$ (resp. $b = \varepsilon$), the current symbol in the input (resp. output) is not read. Since $\mathbb{0}$ is absorbing for $\otimes$ in $\mathbb{S}$, one term $\mathsf{w}_{ij}(q, a, b, q'')$ equal to $\mathbb{0}$ in the above expression will be ignored in the sum, meaning that there is no possible transition from state q into state q' while reading a and b. This is analogous to the case of a transition's guard not satisfied by $\langle a, b\rangle$ for symbolic transducers.

The expression (1) 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 [23]. Intuitively, $\mathsf{w}_{10}(q, a, \varepsilon, r)$ is the cost of the deletion of the symbol $a \in \Sigma$ in s, $\mathsf{w}_{01}(q, \varepsilon, b, r)$ is the cost of the insertion of $b \in \Delta$ in t, and $\mathsf{w}_{11}(q, a, b, r)$ is the cost of the substitution of $a \in \Sigma$ by $b \in \Delta$. The cost of a sequence of such operations transforming s into t, is the product, with $\otimes$, of the individual costs of the operations involved; and the distance between s and t is the sum, with $\oplus$, of all possible products. Formally, the weight associated by T to $\langle s, t\rangle \in \Sigma^* \times \Delta^*$ is:

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

▶ **Example 8.** We build a small swT over alphabets Σ and Δ_{i} (Ex. 1 and 5), with two states q_0 and q_1, that calculates the temporal distance between an input performance in Σ^* and the subsequence of Δ_{i} events in a score. Given a performed event μ and the corresponding notated event ν (*e.g.* MIDI pitch 69 and note A4), the weight computed by the swT is the time distance between both, as modeled by transitions w_{11} below. The continuation symbol $'-'$ (met for instance in *ties* ♩‿♪, or *dots* ♩.), is skipped with no cost (transitions w_{01}).

$$\begin{array}{lcllcll}
\mathsf{w}_{11}(q_0, \mu\!:\!\tau, \nu\!:\!\tau', q_0) & = & \delta(\mu\!:\!\tau, \nu\!:\!\tau') & \mathsf{w}_{11}(q_1, \mu\!:\!\tau, \nu\!:\!\tau', q_0) & = & \delta(\mu\!:\!\tau, \nu\!:\!\tau') & \text{if } \nu \neq - \\
\mathsf{w}_{01}(q_0, \varepsilon, -\!:\!\tau', q_0) & = & \mathbb{1} & \mathsf{w}_{01}(q_1, \varepsilon, -\!:\!\tau', q_0) & = & \mathbb{1} & \\
\mathsf{w}_{10}(q_0, \mu\!:\!\tau, \varepsilon, q_1) & = & \alpha & & & &
\end{array}$$

We also want to take performing errors into account, since a performer could, for example, play an unwritten extra note. The transition w_{10}, with an fixed weight value $\alpha \in \mathbb{S}$, switches from state q_0 (normal) to q_1 (error) when reading an extra note μ. The transitions in the second column below switch back to the normal state q_0. At last, we let q_0 be the only initial and final state, with $\mathsf{in}(q_0) = \mathsf{out}(q_0) = \mathbb{1}$, and $\mathsf{in}(q_1) = \mathsf{out}(q_1) = \mathbb{0}$. ◁

Symbolic Weighted Automata are defined as the transducers of Definition 7, by simply omitting the output symbols.

▶ **Definition 9.** *A* symbolic-weighted automaton *(swA) over Σ, $\mathbb{S}$ and $\bar{\Phi}$ is a tuple $A = \langle Q, \mathsf{in}, \mathsf{w}_1, \mathsf{out}\rangle$, where Q is a finite set of states, $\mathsf{in} : Q \to \mathbb{S}$ (respectively $\mathsf{out} : Q \to \mathbb{S}$) are functions defining the weight for entering (respectively leaving) computation in a state, and w_1 is a transition function from $Q \times Q$ into Φ_Σ.*

As above in the case of swT, when $\mathsf{w}_1(q, q') = \phi \in \Phi_\Sigma$, we may write $\mathsf{w}_1(q, a, q')$ for $\phi(a)$. The computation of A on words $s \in \Sigma^*$ is based with an intermediate function weight_A, defined as follows for $q, q' \in Q$, $a \in \Sigma$, $u \in \Sigma^*$

$$\mathsf{weight}_A(q, \varepsilon, q') = \mathbb{1} \text{ if } q = q', \text{ or } \mathbb{0} \text{ otherwise} \tag{3}$$

$$\mathsf{weight}_A(q, au, q') = \bigoplus_{q'' \in Q} \mathsf{w}_1(q, a, q'') \otimes \mathsf{weight}_A(q'', u, q')$$

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

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

The following property will be useful for symbolic weighted parsing (Section 5).

▶ **Proposition 10.** *Given a swT T over Σ, Δ, $\mathbb{S}$ commutative, bounded and complete, and $\bar{\Phi}$ effective, and a swA A over Σ, $\mathbb{S}$ and $\bar{\Phi}$, there exists a swA $B_{A,T}$ over Δ, $\mathbb{S}$ and $\bar{\Phi}$, effectively constructible in PTIME, such that for all $t \in \Delta^+$, $B_{A,T}(t) = \bigoplus_{s \in \Sigma^*} A(s) \otimes T(s, t)$.*

Proof. (sketch, see Appendix B for details). The state set of $B_{A,T}$ is the Cartesian product of the state sets of A and T and its transitions simulate, while reading an output word $t \in \Delta^*$, the synchronized behaviour of A and T on t and some input word $s \in \Sigma^*$. The weight for reading the input s is obtained with $\bigoplus^1_\Sigma$. The main difficulty comes from the transitions of the form w_{10}, which read in input and ignore the output. Such transition shall be simulated by ε-transitions in $B_{A,T}$, but ε-transitions are not defined for swA. Therefore, the ε-transitions are eliminated on-the-fly during the construction of $B_{A,T}$, following a procedure of [20]. ◀

The particular case of Proposition 10 with a singleton A, *i.e.* such that $A(s) = \mathbb{1}$ for a given $s \in \Sigma^*$ and $A(s') = \mathbb{0}$ for all $s' \neq s$, corresponds to a construction of a swA for the partial application of the swT T, fixing the first argument s.

▶ **Corollary 11.** *Given a swT T over Σ, Δ, $\mathbb{S}$ commutative, bounded and complete, and $\bar{\Phi}$ effective, and $s \in \Sigma^+$, there exists an effectively constructible swA $B_{s,T}$ over Δ, $\mathbb{S}$ and $\bar{\Phi}$, such that for all $t \in \Delta^+$, $B_{s,T}(t) = T(s,t)$.*

4 SW Visibly Pushdown Automata

The model presented now generalizes symbolic VPA (sVPA [7], generalizing themselves VPA [2] to infinite alphabets) from Boolean semirings to arbitrary semiring domains. It associates to every nested word over an infinite alphabet a weight value in a semiring. Nested words can describe structures of labeled trees. In the context of parsing, they will be useful to represent AST (see Section 5 and Appendix D).

Let Δ be a countable alphabet artitioned into three subsets Δ_i, Δ_c, Δ_r, whose elements are respectively called *internal*, *call* and *return* symbols [2]. Let $\langle \mathbb{S}, \oplus, \mathbb{0}, \otimes, \mathbb{1} \rangle$ be a commutative and complete semiring and let $\bar{\Phi} = \langle \Phi_\mathsf{i}, \Phi_\mathsf{c}, \Phi_\mathsf{r}, \Phi_\mathsf{ci}, \Phi_\mathsf{cc}, \Phi_\mathsf{cr} \rangle$ be a label theory over $\mathbb{S}$ where Φ_i, Φ_c, Φ_r and Φ_cx (with $\mathsf{x} \in \{\mathsf{i}, \mathsf{c}, \mathsf{r}\}$) stand respectively for Φ_{Δ_i}, Φ_{Δ_c}, Φ_{Δ_r} and $\Phi_{\Delta_\mathsf{c}, \Delta_\mathsf{x}}$.

▶ **Example 12.** In the nested score representation $O \in \Delta^*$ in Ex. 1, Δ_i corresponds to timed notes and continuations, and Δ_c and Δ_r contain respectively opening and closing parentheses. Another example is the other candidate  of transcription of I, linearized into $O' = \langle_\mathsf{m}\!: [0,1],\ \langle_2\!: [0,1],\ \mathrm{A4}: 0,\ \langle_2\!: [\frac{1}{2}, 1],\ - : \frac{1}{2},\ \mathrm{B4}: \frac{3}{4},\ \rangle_2 : 1,\ \rangle_2 : 1,\ \langle_\mathsf{m}\!: [1,2],\ \langle_3\!: [1,2],\ \text{‘C}\sharp\text{5‘} : 1,\ \mathrm{D5}: 1,$ $\mathrm{E5}: \frac{4}{3},\ \mathrm{F5}: \frac{5}{3},\ \rangle_3 : 2,\ \rangle_\mathsf{m} : 2,\ \rangle_\mathsf{m} : 2$ (see also Fig. 4). The symbol between quotes ‘C♯5‘ represents an *appogiatura*, *i.e.* an ornemental note with theoretical duration 0. ◁

▶ **Definition 13.** *A* Symbolic Weighted Visibly Pushdown Automata *(sw-VPA) over $\Delta = \Delta_\mathsf{i} \uplus \Delta_\mathsf{c} \uplus \Delta_\mathsf{r}$, $\mathbb{S}$ and $\bar{\Phi}$ is a tuple $A = \langle Q, P, \mathsf{in}, \bar{\mathsf{w}}, \mathsf{out} \rangle$, where Q is a finite set of states, P is a finite set of stack symbols, $\mathsf{in} : Q \to \mathbb{S}$ (respectively $\mathsf{out} : Q \to \mathbb{S}$) are functions defining the weight for entering (respectively leaving) a state, and $\bar{\mathsf{w}}$ is a sextuplet composed of the transition functions : $\mathsf{w}_\mathsf{i} : Q \times P \times Q \to \Phi_\mathsf{ci}$, $\mathsf{w}^\mathsf{e}_\mathsf{i} : Q \times Q \to \Phi_\mathsf{i}$, $\mathsf{w}_\mathsf{c} : Q \times P \times Q \times P \to \Phi_\mathsf{cc}$, $\mathsf{w}^\mathsf{e}_\mathsf{c} : Q \times P \times Q \to \Phi_\mathsf{c}$, $\mathsf{w}_\mathsf{r} : Q \times P \times Q \to \Phi_\mathsf{cr}$, $\mathsf{w}^\mathsf{e}_\mathsf{r} : Q \times Q \to \Phi_\mathsf{r}$.*

As in Section 3, we extend the above transition functions as follows for all $q, q' \in Q$, $p \in P$, $a \in \Delta_\mathsf{i}$, $c \in \Delta_\mathsf{c}$, $r \in \Delta_\mathsf{r}$, overloading their names:

$$
\begin{array}{lll}
\mathsf{w}_\mathsf{i} : Q \times [\Delta_\mathsf{c} \times P] \times \Delta_\mathsf{i} \times Q \to \mathbb{S} & \mathsf{w}_\mathsf{i}(q, c, p, a, q') = \eta_\mathsf{ci}(c, a) & \text{where } \eta_\mathsf{ci} = \mathsf{w}_\mathsf{i}(q, p, q'), \\
\mathsf{w}^\mathsf{e}_\mathsf{i} : Q \times \Delta_\mathsf{i} \times Q \to \mathbb{S} & \mathsf{w}^\mathsf{e}_\mathsf{i}(q, a, q') = \phi_\mathsf{i}(a) & \text{where } \phi_\mathsf{i} = \mathsf{w}^\mathsf{e}_\mathsf{i}(q, q'). \\
\mathsf{w}_\mathsf{c} : Q \times [\Delta_\mathsf{c} \times P] \times [\Delta_\mathsf{c} \times P] \times Q \to \mathbb{S} & \mathsf{w}_\mathsf{c}(q, c, p, c', p', q') = \eta_\mathsf{cc}(c, c') & \text{where } \eta_\mathsf{cc} = \mathsf{w}_\mathsf{c}(q, p, p', q'), \\
\mathsf{w}^\mathsf{e}_\mathsf{c} : Q \times [\Delta_\mathsf{c} \times P] \times Q \to \mathbb{S} & \mathsf{w}^\mathsf{e}_\mathsf{c}(q, c, p', q') = \phi_\mathsf{c}(c) & \text{where } \phi_\mathsf{c} = \mathsf{w}^\mathsf{e}_\mathsf{c}(q, p, q'). \\
\mathsf{w}_\mathsf{r} : Q \times [\Delta_\mathsf{c} \times P] \times \Delta_\mathsf{r} \times Q \to \mathbb{S} & \mathsf{w}_\mathsf{r}(q, c, p, r, q') = \eta_\mathsf{cr}(c, r) & \text{where } \eta_\mathsf{cr} = \mathsf{w}_\mathsf{r}(q, p, q'), \\
\mathsf{w}^\mathsf{e}_\mathsf{r} : Q \times \Delta_\mathsf{r} \times Q \to \mathbb{S} & \mathsf{w}^\mathsf{e}_\mathsf{r}(q, r, q') = \phi_\mathsf{r}(r) & \text{where } \phi_\mathsf{r} = \mathsf{w}^\mathsf{e}_\mathsf{r}(q, q').
\end{array}
$$

$\mathsf{w}^\mathsf{e}_\mathsf{i}$, $\mathsf{w}^\mathsf{e}_\mathsf{c}$, and $\mathsf{w}^\mathsf{e}_\mathsf{r}$ describe the cases where the stack is empty. w_i and $\mathsf{w}^\mathsf{e}_\mathsf{i}$ both read an input internal symbol a and change state from q to q', without changing the stack. Moreover, w_i reads a pair made of $c \in \Delta_\mathsf{c}$ and $p \in P$ on the top of the stack (c is compared to a by the weight function $\eta_\mathsf{ci} \in \Phi_\mathsf{ci}$). w_c and $\mathsf{w}^\mathsf{e}_\mathsf{c}$ read the input call symbol c', push it to the stack along with p', and change state from q to to q'. Moreover, w_c reads c and p at the top of the stack (c is compared to c'). w_r and $\mathsf{w}^\mathsf{e}_\mathsf{r}$ read the input return symbol r, and change state from q to to q'. Moreover, w_r reads and pop from stack a pair made of c and p, (c is compared to r).

Formally, the transitions of the automaton A are defined with an intermediate function weight_A, like in Section 3. A configuration $q[\gamma]$ is composed of a state $q \in Q$ and a stack content $\gamma \in \Gamma^*$, where $\Gamma = \Delta_{\mathsf{c}} \times P$. Hence, weight_A is a function from $[Q \times \Gamma^*] \times \Delta^* \times [Q \times \Gamma^*]$ into $\mathbb{S}$. The empty stack is denoted by $\perp$, and the topupmost symbol is the last pushed content. The recursive definition of weight_A enumerates each of the six possible cases: reading $a \in \Delta_{\mathsf{i}}$, or $c \in \Delta_{\mathsf{c}}$, or $r \in \Delta_{\mathsf{r}}$, for each possible state of the stack (empty or not).

$$\mathsf{weight}_A\big(q[\perp], \varepsilon, q'[\perp]\big) = \mathbb{1} \text{ if } q = q' \text{ and } \mathbb{0} \text{ otherwise} \tag{5}$$

$$\mathsf{weight}_A\big(q\begin{bmatrix}\langle c,p\rangle\\ \gamma\end{bmatrix}, a\,u, q'[\gamma']\big) = \bigoplus_{q''\in Q} \mathsf{w}_{\mathsf{i}}(q,c,p,a,q'') \otimes \mathsf{weight}_A\big(q''\begin{bmatrix}\langle c,p\rangle\\ \gamma\end{bmatrix}, u, q'[\gamma']\big)$$

$$\mathsf{weight}_A\big(q[\perp], a\,u, q'[\gamma']\big) = \bigoplus_{q''\in Q} \mathsf{w}^{\mathsf{e}}_{\mathsf{i}}(q,a,q'') \otimes \mathsf{weight}_A\big(q''[\perp], u, q'[\gamma']\big)$$

$$\mathsf{weight}_A\big(q\begin{bmatrix}\langle c,p\rangle\\ \gamma\end{bmatrix}, c'\,u, q'[\gamma']\big) = \bigoplus_{\substack{q''\in Q\\ p'\in P}} \mathsf{w}_{\mathsf{c}}\big(q,c,p,c',p',q''\big) \otimes \mathsf{weight}_A\big(q''\begin{bmatrix}\langle c',p'\rangle\\ \langle c,p\rangle\\ \gamma\end{bmatrix}, u, q'[\gamma']\big)$$

$$\mathsf{weight}_A\big(q[\perp], c\,u, q'[\gamma']\big) = \bigoplus_{\substack{q''\in Q\\ p\in P}} \mathsf{w}^{\mathsf{e}}_{\mathsf{c}}(q,c,p,q'') \otimes \mathsf{weight}_A\big(q''[\langle c,p\rangle], u, q'[\gamma']\big)$$

$$\mathsf{weight}_A\big(q\begin{bmatrix}\langle c,p\rangle\\ \gamma\end{bmatrix}, r\,u, q'[\gamma']\big) = \bigoplus_{q''\in Q} \mathsf{w}_{\mathsf{r}}\big(q,c,p,r,q''\big) \otimes \mathsf{weight}_A\big(q''[\gamma], u, q'[\gamma']\big)$$

$$\mathsf{weight}_A\big(q[\perp], r\,u, q'[\gamma']\big) = \bigoplus_{q''\in Q} \mathsf{w}^{\mathsf{e}}_{\mathsf{r}}(q,r,q'') \otimes \mathsf{weight}_A\big(q''[\perp], u, q'[\gamma']\big)$$

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

$$A(t) = \bigoplus_{q,q'\in Q} \mathsf{in}(q) \otimes \mathsf{weight}_A\big(q[\perp], t, q'[\perp]\big) \otimes \mathsf{out}(q'). \tag{6}$$

Every swA $A = \langle Q, \mathsf{in}, \mathsf{w}_1, \mathsf{out}\rangle$, over Σ, $\mathbb{S}$ and $\bar{\Phi}$ is a particular case of sw-VPA $\langle Q, \emptyset, \mathsf{in}, \bar{\mathsf{w}}, \mathsf{out}\rangle$ over Δ, $\mathbb{S}$ and $\bar{\Phi}$ with $\Delta_{\mathsf{i}} = \Sigma$ and $\Delta_{\mathsf{c}} = \Delta_{\mathsf{r}} = \emptyset$, and computing with an always empty stack: $\mathsf{w}^{\mathsf{e}}_{\mathsf{i}} = \mathsf{w}_1$ and all the other functions of $\bar{\mathsf{w}}$ are the constant $\mathbb{0}$.

▶ **Example 14.** We consider a sw-VPA over the alphabet of Example 12 that expresses a weight related to the music notation, or more precisely to its structural complexity. Given a set of equivalent representations, we aim at choosing the simpler one.

For instance, the following call transition starts in state $q_{i/c}$, meaning that the current interval has number $i < c$ amongst c sub-intervals of duration ℓ (as indicated by the stack top). It reads a new time-division symbol $\langle_d$: $\mathsf{w}_{\mathsf{c}}\big(q_{i/c}, \langle_c\!: [\tau, \tau+\ell], q, \langle_d\!: \iota, q_{i+1/c}, q_{0/d}\big) = \alpha_d$. The time interval ι attached to the $\langle_d$ read must have a starting time $\tau + \frac{i\ell}{c}$, where τ and ℓ are respectively the starting time and duration of the interval attached to the previous time-division symbol read $\langle_c$ (found on the stack). And it must have a duration $\frac{\ell}{d}$. The weight of the above transition is α_d. We can penalize *e.g.* triplets compared to duplets with $\alpha_2 < \alpha_3$. Along with $\langle_d$, the above transition pushes the state $q_{i+1/c}$ on the stack, in order to start the next sub-interval after reading a return symbol $\rangle_d$, with the transition: $\mathsf{w}_{\mathsf{r}}\big(q_{d/d}, \langle_d\!: [\tau, \tau+\ell], q_{i+1/c}, \rangle_d : \tau+\ell, q_{i+1/c}\big) = \mathbb{1}$.

Reading a musical event μ is done with: $\mathsf{w_i}\big(q_{i/d}, \langle_d\!: [\tau, \tau+\ell], q_{j/c}, \mu : \tau + \frac{i\ell}{d}, q_{i+1/d}\big) = \alpha_\mu$. The transition to start reading the first measure is: $\mathsf{w^e_c}\big(q_{1/1}, \langle_{\mathsf{m}}\!: [0,1], q_{1/1}, q_{1/1}\big) = \mathbb{1}$, and for a new measure: $\mathsf{w_c}\big(q_{1/1}, \langle_{\mathsf{m}}\!: [\tau-1, \tau], q_{1/1}, \langle_{\mathsf{m}}\!: [\tau, \tau+1], q_{1/1}, q_{1/1}\big) = \mathbb{1}$. ◁

Similarly to VPA [2] and sVPA [7], the class of sw-VPA is closed under the binary operators of the underlying semiring.

▶ **Proposition 15.** *Let A_1 and A_2 be two sw-VPA over the same Δ, commutative $\mathbb{S}$ and effective $\bar{\Phi}$. There exists two effectively constructible sw-VPA $A_1 \oplus A_2$ and $A_1 \otimes A_2$, such that for all $s \in \Delta^*$, $(A_1 \oplus A_2)(s) = A_1(s) \oplus A_2(s)$ and $(A_1 \otimes A_2)(s) = A_1(s) \otimes A_2(s)$.*

Proof. We do a classical product construction, see Appendix C for details. ◀

We present now a procedure for searching, for a sw-VPA A, a word of minimal weight for A.

▶ **Proposition 16.** *For a sw-VPA A over Δ, $\mathbb{S}$ commutative, bounded, total and complete, and $\bar{\Phi}$ effective, one can construct in PTIME a word $t \in \Delta^*$ such that $A(t)$ is minimal wrt the natural ordering $\leq_\oplus$ for $\mathbb{S}$.*

Let $A = \langle Q, P, \mathsf{in}, \bar{\mathsf{w}}, \mathsf{out}\rangle$. We propose a Dijkstra algorithm computing, for every $q, q' \in Q$, the minimum, *wrt* $\leq_\oplus$, of the function $\beta_{q,q'} : t \mapsto \mathsf{weight}_A(q[\bot], t, q'[\bot])$. Let us denote by $b_\bot(q, q')$ this minimum. By definition of $\leq_\oplus$, and since $\mathbb{S}$ is total, it holds that:

$$b_\bot(q, q') = \bigoplus_{t \in \Delta^*} \mathsf{weight}_A\big(q[\bot], t, q'[\bot]\big). \tag{7}$$

The infinite sum in (7) is well defined since $\mathbb{S}$ is complete. Following (6), and the associativity, commutativity and distributivity for $\otimes$ and $\oplus$, the minimum of $A(t)$ is:

$$\bigoplus_{t \in \Delta^*} A(t) = \bigoplus_{t \in \Delta^*} \bigoplus_{q,q' \in Q} \mathsf{in}(q) \otimes \beta_{q,q'}(t) \otimes \mathsf{out}(q') = \bigoplus_{q,q' \in Q} \mathsf{in}(q) \otimes b_\bot(q, q') \otimes \mathsf{out}(q') \tag{8}$$

In order to compute the above function $b_\bot : Q \times Q \to \mathbb{S}$, we shall consider an auxiliary function $b_\top : Q \times P \times Q \to \Phi_{\mathsf{c}}$. Intuitively, $b_\top(q, p, q')$ is a function of Φ_{c}, mapping every $c \in \Delta_{\mathsf{c}}$ to the minimum weight of a computation of A starting in state q with a non-empty stack $\gamma' = \langle c, p\rangle\,\gamma \in \Gamma^+$, and ending in state q' with the same stack γ', such that moreover, the computation does not pop the pair $\langle c, p\rangle$ at the top of γ' (*i.e.* γ' is left untouched during the computation). However, the computation can read $\langle c, p\rangle$ at the top of γ', and can also push another pair $\langle c', p'\rangle \in \Gamma$ on γ', following the third case of in the definition (5) of weight_A (call symbol). The pair $\langle c', p'\rangle$ can be pop later, during the computation from q to q', following the fifth case of (5) (return symbol). Formally, in order to define $b_\top$, we consider a fresh stack symbol $\top \notin \Gamma$, representing the above untouched stack, and let:

$$b_\top(q, p, q') : c \mapsto \bigoplus_{s \in \Delta^*} \mathsf{weight}_A\Big(q \begin{bmatrix} \langle c, p\rangle \\ \top \end{bmatrix}, s, q' \begin{bmatrix} \langle c, p\rangle \\ \top \end{bmatrix}\Big) \quad \text{for all } c \in \Delta_{\mathsf{c}} \tag{9}$$

By definition of weight_A in (5), using the symbol $\top$ for the part of the stack below $\langle c, p\rangle$ (*i.e.* the substack γ in the above $\gamma' = \langle c, p\rangle\,\gamma$) ensures that this part is not touched during the computation. This ensures in particular that the subword read during the computation is well parenthesized (every symbol in Δ_{c} has a matching symbol in Δ_{r}).

Algorithm 1 constructs iteratively, using a priority queue $\mathcal{Q}$, two markings $d_\bot : Q \times Q \to \mathbb{S}$ and $d_\top : Q \times P \times Q \to \Phi_{\mathsf{c}}$, that converges eventually to $b_\top$ and $b_\bot$. It terminates in

Algorithm 1 Best search for sw-VPA

initially let $\mathcal{Q} = (Q \times Q) \cup (Q \times P \times Q)$, and let $d_\perp(q_1, q_2) = d_\top(q_1, p, q_2) = \mathbb{1}$ if $q_1 = q_2$ and $d_\perp(q_1, q_2) = d_\top(q_1, p, q_2) = \mathbb{0}$ otherwise

while $\mathcal{Q} \neq \emptyset$ **do**

  extract $\langle q_1, q_2\rangle$ or $\langle q_1, p, q_2\rangle$ from $\mathcal{Q}$ such that $d_\perp(q_1, q_2)$, resp. $\bigoplus_{\Delta_c} d_\top(q_1, p, q_2)$, is minimal in $\mathbb{S}$ *wrt* $\leq_\oplus$

  update $d_\perp$ with $\langle q_1, q_2\rangle$ or $d_\top$ with $\langle q_1, p, q_2\rangle$ (Figure 3).

For all $q_0, q_3 \in Q$,

$$\begin{array}{lcl}
d_\top(q_1, p, q_3) & \oplus= & d_\top(q_1, p, q_2) \otimes \bigoplus_{\Delta_i} \mathsf{w}_i(q_2, p, q_3) \\
d_\perp(q_1, p, q_3) & \oplus= & d_\perp(q_1, q_2) \otimes \bigoplus_{\Delta_i} \mathsf{w}_i^e(q_2, q_3) \\
d_\top(q_0, p, q_3) & \oplus= & \bigoplus_{\Delta_c}^{2} \big[\big(\mathsf{w}_c(q_0, p, q_1, p') \otimes_2 d_\top(q_1, p', q_2)\big) \otimes_2 \bigoplus_{\Delta_r} \mathsf{w}_r(q_2, p', q_3)\big] \\
d_\perp(q_0, q_3) & \oplus= & \bigoplus_{\Delta_c} \big(\mathsf{w}_c^e(q_0, p, q_1) \otimes d_\top(q_1, p, q_2) \otimes \bigoplus_{\Delta_r} \mathsf{w}_r(q_2, p, q_3)\big) \\
d_\perp(q_1, q_3) & \oplus= & d_\perp(q_1, q_2) \otimes \bigoplus_{\Delta_r} \mathsf{w}_r^e(q_2, q_3) \\
d_\top(q_1, p, q_3) & \oplus= & d_\top(q_1, p, q_2) \otimes d_\top(q_2, p, q_3), \text{if } \langle q_2, p, q_3\rangle \notin \mathcal{Q} \\
d_\perp(q_1, q_3) & \oplus= & d_\perp(q_1, q_2) \otimes d_\perp(q_2, q_3), \text{if } \langle q_2, q_3\rangle \notin \mathcal{Q}
\end{array}$$

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

PTIME and at termination (when $\mathcal{Q}$ is empty), and its correctness is ensured by (8) and the following invariants: $\langle q_1, q_2\rangle \notin \mathcal{Q}$ iff $d_\perp(q_1, q_2) = b_\perp(q_1, q_2)$, and $\langle q_1, p, q_2\rangle \notin \mathcal{Q}$ iff $\bigoplus_{\Delta_c} d_\top(q_1, p, q_2) = \bigoplus_{\Delta_c} b_\top(q_1, p, q_2)$.

Thanks to the hypothesis that $\bar{\Phi}$ is effective, it is possible to construct during the iteration a witness word for Proposition 16, *i.e.* a word $t \in \Delta^*$ with a minimal weight $A(t)$ *wrt* $\leq_\oplus$.

5 Symbolic Weighted Parsing

Let us now apply the models and results of the previous sections to the problem of parsing over an infinite alphabet. Let Σ and $\Delta = \Delta_i \uplus \Delta_c \uplus \Delta_r$ be countable input and output alphabets, let $\langle \mathbb{S}, \oplus, \mathbb{0}, \otimes, \mathbb{1}\rangle$ be a commutative, bounded, total and complete semiring and let $\bar{\Phi}$ be an effective label theory over $\mathbb{S}$, containing Φ_Σ, Φ_{Σ,Δ_i}, as well as Φ_i, Φ_c, Φ_r, Φ_{cr} (following the notations of Section 4). We assume given the following input:

- a swT T over Σ, Δ_i, $\mathbb{S}$, and $\bar{\Phi}$, defining a measure $T : \Sigma^* \times {\Delta_i}^* \to \mathbb{S}$,
- a sw-VPA A over Δ, $\mathbb{S}$, and $\bar{\Phi}$, defining a measure $A : \Delta^* \to \mathbb{S}$,
- an input word $s \in \Sigma^*$.

For all $u \in \Sigma^*$ and $t \in \Delta^*$, let $d(u, t) = T\big(u, t|_{\Delta_i}\big)$, where $t|_{\Delta_i} \in {\Delta_i}^*$ is the projection of t onto Δ_i, obtained from t by removing all symbols in $\Delta \setminus \Delta_i$. *Symbolic weighted parsing* is the problem, given the above input, to find $t \in \Delta^*$ minimizing $d(s, t) \otimes A(t)$ *wrt* $\leq_\oplus$, *i.e.* s.t.

$$d(s, t) \otimes A(t) = \bigoplus_{u \in \Delta^*} d(s, u) \otimes A(u) \tag{10}$$

Following the terminology of [22], sw-parsing is the problem of computing the distance (10) between the input s and the output weighted language of A, and returning a witness t.

▶ **Example 17** (Symbolic Weighted Parsing and the transcription problem)**.** Applied to the music transcription problem, the above formalism is interpreted as follows:

- The input word is I of Example 1.
- The swT T evaluates a "fitness measure" that expresses a correspondance between a performance and a nested representation of a music score. See Example 8.
- The sw-VPA A expresses a cost related to the music notation.

As seen in Example 14, , will be favored on  when the weight a second time division with $\langle_2$ is less than the difference of weight between 'C♯5' and C♯5. The SW-parsing framework, applied to the transcription problem, allows to find an optimal solution that considers both the fitness of the result, and its structural complexity. ◁

The application to music transcription suggested briefly in the examples has been implemented in a C++ tool [1], following the principles of the present SW-parsing framework, although it differs in several points. In particular, the automata constructions are performed on the on-the-fly during the search of a best AST, for efficiency reasons.

▶ **Proposition 18.** *The problem of Symbolic Weighted Parsing can be solved in PTIME in the size of the input swT T, sw-VPA A and input word s, and the computation time of the functions and operators of the label theory.*

Proof. (sketch) We follow a *Bar-Hillel* construction for parsing by intersection. We first extend the swT T over Σ, Δ_{i} into a swT T' over Σ and Δ (and the same semiring and label theory $\mathbb{S}$ and $\bar{\Phi}$), such that for all $u \in \Sigma^*$, and $t \in \Delta^*$, $T'(u,t) = T(u, t|_{\Delta_{\mathsf{i}}})$. T' simply skips every symbol $b \in \Delta \setminus \Delta_{\mathsf{i}}$ by the addition to T, of new transitions of the form $\mathsf{w}_{01}(q, \varepsilon, b, q')$. Then, using Corolary 11, we construct from $s \in \Sigma^*$ and T' a swA $B_{s,T'}$, such that for all $t \in \Delta^*$, $B_{s,T'}(t) = d(s,t)$. Next, we compute the sw-VPA $B_{s,T'} \otimes A$, using Proposition 15. It remains to compute a best nested word $t \in \Delta^*$ using the procedure of Proposition 16. ◀

The sw-parsing generalizes the problem of searching the best derivation (AST) of a weighted CF-grammar G that yields a given input word w, called *weighted parsing*, (see [14] and the more general framework [24]), with infinite input alphabet instead of a finite one and transducer-defined distances instead of equality. See Appendix D for more details on the correspondence between nested words $t \in \Delta^*$ and AST and CF grammars and sw-VPA.

Conclusion

We introduced Symbolic Weighted language models and applied them to the problem of parsing with infinitely many possible input symbols (typically timed events). Our approach extends conventional parsing and weighted parsing by computing a derivation tree modulo a distance between words defined by a SW transducer given in input. This allows to consider finer word relationships than strict equality.

This work can be extended in several directions. First, the the best search algorithm could be generalized from 1-best to n-best [18], and to *k-closed* semirings [21] (instead of *bounded*, which corresponds to 0-*closed*). Second, the complexity bounds of the algorithms could be more precisely characterized, as well as expressiveness of swM compared to the automata of *e.g.* [27, 19, 26, 3]. Finally, the best search algorithm presented here offline, whereas an on-the-fly automata construction would allow for online parsing, a suitable feature in the context of applications such as, *e.g.* automatic music transcription.

References

1. qparse, a library for automated rhythm transcription. URL: `https://qparse.gitlabpages.inria.fr`.
2. Rajeev Alur and Parthasarathy Madhusudan. Adding nesting structure to words. *Journal of the ACM (JACM)*, 56(3):1–43, 2009.
3. Mikołaj Bojańczyk, Claire David, Anca Muscholl, Thomas Schwentick, and Luc Segoufin. Two-variable logic on data words. *ACM Transactions on Computational Logic (TOCL)*, 12(4):1–26, 2011.
4. Patricia Bouyer, Antoine Petit, and Denis Thérien. An algebraic approach to data languages and timed languages. *Information and Computation*, 182(2):137–162, 2003.
5. Mathieu Caralp, Pierre-Alain Reynier, and Jean-Marc Talbot. Visibly pushdown automata with multiplicities: finiteness and k-boundedness. In *International Conference on Developments in Language Theory*, pages 226–238. Springer, 2012.
6. Hubert Comon, Max Dauchet, Rémi Gilleron, Florent Jacquemard, Christoph Löding, Denis Lugiez, Sophie Tison, and Marc Tommasi. *Tree Automata Techniques and Applications*. `http://tata.gforge.inria.fr`, 2007.
7. Loris D'Antoni and Rajeev Alur. Symbolic visibly pushdown automata. In *International Conference on Computer Aided Verification*, pages 209–225. Springer, 2014.
8. Loris D'Antoni and Margus Veanes. The power of symbolic automata and transducers. In *International Conference on Computer Aided Verification*, pages 47–67. Springer, 2017.
9. Loris D'Antoni and Margus Veanes. Automata modulo theories. *Communications of the ACM*, 64(5):86–95, 2021. URL: `seealsoseealsohttps://pages.cs.wisc.edu/~loris/symbolicautomata.html`.
10. E. W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1(1):269–271, 1959.
11. Manfred Droste and Werner Kuich. Semirings and formal power series. In *Handbook of Weighted Automata*, pages 3–28. Springer, 2009.
12. Manfred Droste, Werner Kuich, and Heiko Vogler. *Handbook of weighted automata*. Springer Science & Business Media, 2009.
13. Francesco Foscarin, Florent Jacquemard, Philippe Rigaux, and Masahiko Sakai. A Parse-based Framework for Coupled Rhythm Quantization and Score Structuring. In *Mathematics and Computation in Music (MCM)*, volume 11502 of *Lecture Notes in Artificial Intelligence*, Madrid, Spain, 2019. Springer. URL: `https://hal.inria.fr/hal-01988990`, `doi:10.1007/978-3-030-21392-3\_20`.
14. Joshua Goodman. Semiring parsing. *Computational Linguistics*, 25(4):573–606, 1999.
15. Elaine Gould. *Behind Bars: The Definitive Guide to Music Notation*. Faber Music, 2011.
16. Dick Grune and Ceriel J.H. Jacobs. *Parsing Techniques*. Number 2nd edition in Monographs in Computer Science. Springer, 2008.
17. Liang Huang. Advanced dynamic programming in semiring and hypergraph frameworks. In *In COLING*, 2008.
18. Liang Huang and David 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. URL: `http://dl.acm.org/citation.cfm?id=1654494.1654500`.
19. Michael Kaminski and Nissim Francez. Finite-memory automata. *Theor. Comput. Sci.*, 134:329–363, November 1994. URL: `http://dx.doi.org/10.1016/0304-3975(94)90242-9`, `doi:http://dx.doi.org/10.1016/0304-3975(94)90242-9`.
20. Sylvain Lombardy and Jacques Sakarovitch. The removal of weighted ε-transitions. In *International Conference on Implementation and Application of Automata*, pages 345–352. Springer, 2012.
21. Mehryar Mohri. Semiring frameworks and algorithms for shortest-distance problems. *Journal of Automata, Languages and Combinatorics*, 7(3):321–350, 2002.

22 Mehryar Mohri. Edit-distance of weighted automata: General definitions and algorithms. *International Journal of Foundations of Computer Science*, 14(06):957–982, 2003. URL: `https://www.worldscientific.com/doi/abs/10.1142/S0129054103002114`, `arXiv:https://www.worldscientific.com/doi/pdf/10.1142/S0129054103002114`, `doi:10.1142/S0129054103002114`.

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

24 Richard Mörbitz and Heiko 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, September 2019. Association for Computational Linguistics. URL: `https://www.aclweb.org/anthology/W19-3108`, `doi:10.18653/v1/W19-3108`.

25 Mark-Jan Nederhof. Weighted deductive parsing and Knuth's algorithm. *Computational Linguistics*, 29(1):135–143, 2003. URL: `https://doi.org/10.1162/089120103321337467`.

26 Frank Neven, Thomas Schwentick, and Victor Vianu. Finite state machines for strings over infinite alphabets. *ACM Trans. Comput. Logic*, 5(3):403–435, July 2004. URL: `http://doi.acm.org/10.1145/1013560.1013562`, `doi:10.1145/1013560.1013562`.

27 Luc Segoufin. Automata and logics for words and trees over an infinite alphabet. In *Computer Science Logic*, volume 4207 of *LNCS*. Springer, 2006.

28 Eleanor Selfridge-Field, editor. *Beyond MIDI: the handbook of musical codes*. MIT press, 1997. URL: `http://beyondmidi.ccarh.org/beyondmidi-600dpi.pdf`.

29 Moshe 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 Properties of Label Theory Operators

The following facts are immediate consequences of the definitions of the operators on the functions of labels theories in Section 2.

▶ **Lemma 19.** *For a complete label theory $\bar{\Phi}$, and for all $\alpha \in \mathbb{S}$, $\phi, \phi' \in \Phi_\Sigma$, $\psi \in \Phi_\Delta$, and $\eta \in \Phi_{\Sigma,\Delta}$, it holds that:*

i. $\bigoplus_\Sigma \bigoplus^2_\Delta \eta = \bigoplus_\Delta \bigoplus^1_\Sigma \eta$

ii. $\alpha \otimes \bigoplus_\Sigma \phi = \bigoplus_\Sigma (\alpha \otimes \phi)$ *and* $\big(\bigoplus_\Sigma \phi\big) \otimes \alpha = \bigoplus_\Sigma (\phi \otimes \alpha)$*, and similarly for* $\oplus$

iii. $\big(\bigoplus_\Sigma \phi\big) \oplus \big(\bigoplus_\Sigma \phi'\big) = \bigoplus_\Sigma (\phi \oplus \phi')$ *and* $\big(\bigoplus_\Sigma \phi\big) \otimes \big(\bigoplus_\Sigma \phi'\big) = \bigoplus_\Sigma (\phi \otimes \phi')$

iv. $\big(\bigoplus^2_\Delta \eta\big) \oplus \big(\bigoplus^2_\Delta \eta'\big) = \bigoplus^2_\Delta (\eta \oplus \eta')$*, and* $\big(\bigoplus^2_\Delta \eta\big) \otimes \big(\bigoplus^2_\Delta \eta'\big) = \bigoplus^2_\Delta (\eta \otimes \eta')$

v. $\phi \otimes \big(\bigoplus^2_\Delta \eta\big) = \bigoplus_\Delta (\phi \otimes_1 \eta)$*, and* $\big(\bigoplus^2_\Delta \eta\big) \otimes \phi = \bigoplus_\Delta (\eta \otimes_1 \phi)$*, and similarly for* $\oplus$

vi. $\psi \otimes \big(\bigoplus^1_\Sigma \eta\big) = \bigoplus_\Sigma (\psi \otimes_2 \eta)$*, and* $\big(\bigoplus^1_\Sigma \eta\big) \otimes \psi = \bigoplus_\Sigma (\eta \otimes_2 \psi)$*, and similarly for* $\oplus$

B Proof of Proposition 10

Let $T = \langle Q, \mathsf{in}_T, \bar{\mathsf{w}}, \mathsf{out}_T\rangle$, where $\bar{\mathsf{w}}$ contains w_{10}, w_{01}, and w_{11}, from $Q \times Q$ into respectively Φ_Σ, Φ_Δ, and $\Phi_{\Sigma,\Delta}$, and let $A = \langle P, \mathsf{in}_A, \mathsf{w}_1, \mathsf{out}_A\rangle$ with $\mathsf{w}_1 : Q \times Q \to \Phi_\Sigma$. The state set of $B_{A,T}$ will be $Q' = P \times Q$.

The entering, leaving and transition functions of $B_{A,T}$ will simulate synchronized computations of A and T, while reading an output word of $t \in \Delta^*$, and some input word $s \in \Sigma^*$. Its state entering functions is defined for all $\langle p_1, q_1\rangle \in Q'$, by:

$$\mathsf{in}'\big(\langle p_1, q_1\rangle\big) = \mathsf{in}_A(p_1) \otimes \mathsf{in}_T(q_1). \tag{11}$$

The transition function w'_1 will roughly perform a synchronized product of transitions defined by w_1, w_{01} (T reading in output word and not in input word) and w_{11} (T reading both in input word and in output word). Moreover, w'_1 also needs to simulate transitions defined by w_{10}: T reading in input word and not in output word. Since $B_{A,T}$ will read only in the output word, such a transition would correspond to an ε-transition of swA. But swA have been defined without ε-transitions. Therefore, in order to take care of this case, we perform an on-the-fly elimination of ε-transitions during the construction of the swA, following Algorithm 1 of [20]. The transition function w'_1 is constructed iteratively.
Initially, for all $p_1, p_2 \in P$, and $q_1, q_2 \in Q$, let

$$\mathsf{w}'_1\big(\langle p_1, q_1\rangle, \langle p_2, q_2\rangle\big) = \Big(\bigoplus_{p_1 = p_2} \mathsf{w}_{01}(q_1, q_2) \Big) \oplus \bigoplus^1_\Sigma \big(\mathsf{w}_1(p_1, p_2) \otimes_1 \mathsf{w}_{11}(q_1, q_2)\big) \tag{12}$$

$$\mathsf{out}'\big(\langle p_1, q_1\rangle\big) = \mathsf{out}_A(p_1) \otimes \mathsf{out}_T(q_1) \tag{13}$$

We recall that by convention, $\bigoplus\limits_{p_1 = p_2} \mathsf{w}_{01}(q_1, q_2)$ is equal to $\mathbb{0}$ if $p_1 \neq p_2$.
Then, we iterate the following updates for all $p_1, p_2, p_3 \in P$ and $q_1, q_2, q_3 \in Q$:

$$\mathsf{w}'_1\big(\langle p_1, q_1\rangle, \langle p_3, q_3\rangle\big) \mathrel{\oplus=} \bigoplus_\Sigma \big(\mathsf{w}_1(p_1, p_2) \otimes \mathsf{w}_{10}(q_1, q_2)\big) \otimes \mathsf{w}'_1\big(\langle p_2, q_2\rangle, \langle p_3, q_3\rangle\big) \tag{14}$$

$$\mathsf{out}'\big(\langle p_2, q_2\rangle\big) \mathrel{\oplus=} \bigoplus_\Sigma \big(\mathsf{w}_1(p_1, p_2) \otimes \mathsf{w}_{10}(q_1, q_2)\big) \otimes \mathsf{out}'(p_1, q_1) \tag{15}$$

In both cases of updates of w'_1 (14) and out' (15) during the iteration, $\mathsf{w}_1(p_1, p_2) \otimes \mathsf{w}_{10}(q_1, q_2)$ is the weight of an ε-transition. It corresponds to the reading, by A and T, of a symbol a in the input word s without moving in the output word, *i.e.* the synchronization of a transition $\mathsf{w}_1(p_1, a, p_2)$ of A and a transition $\mathsf{w}_{10}(q_1, a, \varepsilon, q_2)$ of T.

The iteration stops if it does not change the value of w_1' and out'. By hypothesis and Lemma 4, $\mathbb{S}$ is idempotent. Therefore, the construction of $B_{A,T}$ will stop after at most $|P|^2.|Q|^2$ iterations.

Let us now show that $B_{A,T}(t) = \bigoplus_{s\in\Sigma^*} A(s) \otimes T(s,t)$ for all $t \in \Delta^+$.

We call *path* from $\langle p_0, q_0\rangle$ to $\langle p_n, q_n\rangle$ a finite sequence of the form: $\pi = \langle p_0, q_0\rangle, a_1, \langle p_1, q_1\rangle, \ldots, a_n, \langle p_n, q_n\rangle$ where $\langle p_i, q_i\rangle \in Q'$ for all $0 \leq i \leq n$ and $a_j \in \Sigma$ for all $1 \leq j \leq n$. The state $\langle p_0, q_0\rangle$ is called source of the path, denoted $\mathit{src}(\pi)$ and $\langle p_n, q_n\rangle$ is called target of the path, denoted $\mathit{trg}(\pi)$; the set of paths with source $\langle p, q\rangle$ and target $\langle p', q'\rangle$ is denoted $\Pi(\langle p, q\rangle, \langle p', q'\rangle)$. The word $a_1 \ldots a_n \in \Sigma^*$ is called word of the path π and denoted $\mathit{word}(\pi)$. Moreover, we associate a weight value in $\mathbb{S}$ to every path, defined by:

$$\mathsf{weight}(\pi) = \bigotimes_{i=1}^{n} \mathsf{w}_1(p_{i-1}, a_i, p_i) \otimes \mathsf{w}_{10}(q_{i-1}, a_i, \varepsilon, q_i) \tag{16}$$

By definition of the weight functions, and associativity, commutativity, and distributivity of $\oplus$, $\otimes$, it holds that:

$$\mathsf{weight}_A(p, s, p') \otimes \mathsf{weight}_T(q, s, \varepsilon, q') = \bigoplus_{\substack{\pi\in\Pi(\langle p,q\rangle,\langle p',q'\rangle)\\ \mathit{word}(\pi)=s}} \mathsf{weight}(\pi) \tag{17}$$

Using (16) and Lemma 19 repetively, (17) implies that:

$$\begin{aligned}&\bigoplus_{s\in\Sigma^*} \mathsf{weight}_A(p, s, p') \otimes \mathsf{weight}_T(q, s, \varepsilon, q') = \\ &\bigoplus_{\substack{\pi\in\Pi(\langle p,q\rangle,\langle p',q'\rangle)\\ \pi=\langle p_0,q_0\rangle,a_1,\langle p_1,q_1\rangle,\ldots,a_n,\langle p_n,q_n\rangle}} \bigotimes_{i=1}^{n} \bigoplus_{\Sigma} \big(\mathsf{w}_1(p_{i-1}, p_i) \otimes \mathsf{w}_{10}(q_{i-1}, q_i)\big)\end{aligned} \tag{18}$$

Note that the symbols $a_1, \ldots, a_n \in \Sigma$ in the path π are not significant in (18). Using a pumping argument, we can show that (18) still holds when restricting π to the set $\Pi_0(\langle p, q\rangle, \langle p', q'\rangle)$ of paths without repetition in the state symbols. Indeed, assume that in $\pi = \langle p_0, q_0\rangle, a_1, \langle p_1, q_1\rangle, \ldots, a_n, \langle p_n, q_n\rangle$, $\langle p_{i_1}, q_{i_1}\rangle = \langle p_{i_2}, q_{i_2}\rangle$ for $0 \leq i_1 < i_2 \leq n$. Then $\pi' = \langle p_0, q_0\rangle, \ldots, a_{i_1-1}, \langle p_{i_1-1}, q_{i_1-1}\rangle, a_{i_2}, \langle p_{i_2}, q_{i_2}\rangle, \ldots, a_n, \langle p_n, q_n\rangle$ also belongs to $\Pi\big(\langle p_0, q_0\rangle, \langle p_n, q_n\rangle\big)$ and yields a smaller expression (*wrt* $\leq_\oplus$) in the right-hand-side of (18) than π. It follows, by (12) and (14), that for all $b \in \Delta$,

$$\mathsf{w}_1'(\langle p, q\rangle, b, \langle p', q'\rangle) = \bigoplus_{s\in\Sigma^*} \bigoplus_{\substack{p''\in P\\ q''\in Q}} \mathsf{weight}_A(p, s, p'') \otimes \mathsf{weight}_T(q, s, \varepsilon, q'') \otimes \psi_1(b) \tag{19}$$

where $\psi_1 = \mathsf{w}_{01}(q'', q') \oplus \bigoplus_{\Sigma}^{1} \big(\mathsf{w}_1(p'', p') \otimes_1 \mathsf{w}_{11}(q'', q')\big)$.

We show now by induction on the length of $t \in \Delta^+$, that

$$\mathsf{weight}_{B_{A,T}}(\langle p, q\rangle, t, \langle p', q'\rangle) = \bigoplus_{s\in\Sigma^*} \mathsf{weight}_A(p, s, p') \otimes \mathsf{weight}_A(q, s, t, q')$$

This permits to conclude, using the definition of in' in (11), and the definition of out' in (13), and (15).

The base case $t \in \Delta$ follows from (19) and the distributivity of $\otimes$.
For $t = bu$, with $b \in \Delta$ and $u \in \Delta^*$, by definition of weight_A and weight_T, it holds that for all $s \in \Sigma^*$:

$$\mathsf{weight}_A(p, s, p') \otimes \mathsf{weight}_T(q, s, t, q') = \bigoplus_{s = s_1 s_2} \bigoplus_{\substack{p'', p''' \in P \\ q'', q''' \in Q}} \mathsf{weight}_A(p, s_1, p''') \otimes \mathsf{weight}_A(p''', s_2, p'') \otimes$$
$$\mathsf{weight}_T(q, s_1, \varepsilon, q'') \otimes \left(\begin{array}{l} \bigoplus_{q''' \in Q} \mathsf{w}_{01}(q'', \varepsilon, b, q''') \otimes \mathsf{weight}_T(q''', s_2, u, q') \oplus \\ \bigoplus_{q''' \in Q} \bigoplus_{s_2 = a s'_2} \mathsf{w}_{11}(q'', a, b, q''') \otimes \mathsf{weight}_T(q''', s'_2, u, q') \end{array} \right)$$

Using (19), it follows that:

$$\begin{aligned} &\bigoplus_{s \in \Sigma^*} \mathsf{weight}_A(p, s, p') \otimes \mathsf{weight}_T(q, s, t, q') = \\ &\quad \bigoplus_{\substack{p'', p''' \in P \\ q'', q''' \in Q}} \bigoplus_{s_1 \in \Sigma^*} \mathsf{weight}_A(p, s_1, p'') \otimes \mathsf{weight}_T(q, s_1, \varepsilon, q'') \otimes \psi_1(b) \\ &\quad \otimes \bigoplus_{s_2 \in \Sigma^*} \mathsf{weight}_A(p''', s_2, p') \otimes \mathsf{weight}_T(q''', s_2, u, q') \end{aligned}$$

with $\psi_1 = \mathsf{w}_{01}(q'', q''') \oplus \bigoplus_{\Sigma}^{1} \big(\mathsf{w}_1(p'', p''') \otimes_1 \mathsf{w}_{11}(q'', q''')\big)$.
The first term in the right-hand-side is $\mathsf{w}'_1(\langle p, q\rangle, b, \langle p''', q'''\rangle)$ by (19), and the second term is $\mathsf{weight}_{B_{A,T}}(\langle p''', q'''\rangle, u, \langle p', q'\rangle)$ by induction hypothesis. Hence, by definition,

$$\begin{aligned} \bigoplus_{s \in \Sigma^*} \mathsf{weight}_A(p, s, p') \otimes \mathsf{weight}_T(q, s, t, q') &= \\ \bigoplus_{\substack{p''' \in P \\ q''' \in Q}} \mathsf{w}'_1(\langle p, q\rangle, b, \langle p''', q'''\rangle) \otimes \mathsf{weight}_{B_{A,T}}(\langle p''', q'''\rangle, u, \langle p', q'\rangle) & \\ &= \mathsf{weight}_{B_{A,T}}(\langle p, q\rangle, t, \langle p', q'\rangle). \end{aligned}$$

C Proof of Proposition 15

We prove the closure under $\otimes$. The proof of the closure under $\oplus$ is similar.

Let $A_1 = \langle Q_1, P_1, \mathsf{in}_1, \bar{\mathsf{w}}_1, \mathsf{out}_1\rangle$ and $A_2 = \langle Q_2, P_2, \mathsf{in}_2, \bar{\mathsf{w}}_2, \mathsf{out}_2\rangle$. The sw-VPA $A_1 \otimes A_2$ is built by a classical product construction. It has a state set $Q = Q_1 \times Q_2$ and a auxiliary set of stack symbols $P = P_1 \times P_2$: $A_1 \otimes A_2 = \langle Q, P, \mathsf{in}_{1,\otimes}, \bar{\mathsf{w}}_\otimes, \mathsf{out}_\otimes\rangle$. The weight entering and leaving functions $\mathsf{in}_\otimes$, $\mathsf{out}_\otimes$ and the sextuplet of transition functions $\bar{\mathsf{w}}_\otimes$ are defined using the label-theory operators of Section 2. They will simulate the synchronized behaviour of A_1 and A_2. For all $\langle q_1, q_2\rangle, \langle q'_1, q'_2\rangle \in Q$ and $\langle p_1, p_2\rangle, \langle p'_1, p'_2\rangle \in P$:

$$\begin{array}{lcl} \mathsf{in}_\otimes\big(\langle q_1, q_2\rangle\big) & = & \mathsf{in}_1(q_1) \otimes \mathsf{in}_2(q_2) \\ \mathsf{out}_\otimes\big(\langle q_1, q_2\rangle\big) & = & \mathsf{out}_1(q_1) \otimes \mathsf{out}_2(q_2) \\ \mathsf{w}_{\mathsf{i},\otimes}\big(\langle q_1, q_2\rangle, \langle p_1, p_2\rangle, \langle q'_1, q'_2\rangle\big) & = & \mathsf{w}_{\mathsf{i},1}(q_1, p_1, q'_1) \otimes \mathsf{w}_{\mathsf{i},2}(q_2, p_2, q'_2) \\ \mathsf{w}^{\mathsf{e}}_{\mathsf{i},\otimes}\big(\langle q_1, q_2\rangle, \langle q'_1, q'_2\rangle\big) & = & \mathsf{w}^{\mathsf{e}}_{\mathsf{i},1}(q_1, q'_1) \otimes \mathsf{w}^{\mathsf{e}}_{\mathsf{i},2}(q_2, q'_2) \\ \mathsf{w}_{\mathsf{c},\otimes}\big(\langle q_1, q_2\rangle, \langle p_1, p_2\rangle, \langle q'_1, q'_2\rangle, \langle p'_1, p'_2\rangle\big) & = & \mathsf{w}_{\mathsf{c},1}(q_1, p_1, q'_1, p'_1) \otimes \mathsf{w}_{\mathsf{c},2}(q_2, p_2, q'_2, p'_2) \\ \mathsf{w}^{\mathsf{e}}_{\mathsf{c},\otimes}\big(\langle q_1, q_2\rangle, \langle p_1, p_2\rangle, \langle q'_1, q'_2\rangle\big) & = & \mathsf{w}^{\mathsf{e}}_{\mathsf{c},1}(q_1, p_1, q'_1) \otimes \mathsf{w}^{\mathsf{e}}_{\mathsf{i},2}(q_2, p_2, q'_2) \\ \mathsf{w}_{\mathsf{r},\otimes}\big(\langle q_1, q_2\rangle, \langle p_1, p_2\rangle, \langle q'_1, q'_2\rangle\big) & = & \mathsf{w}_{\mathsf{r},1}(q_1, p_1, q'_1) \otimes \mathsf{w}_{\mathsf{r},2}(q_2, p_2, q'_2) \\ \mathsf{w}^{\mathsf{e}}_{\mathsf{r},\otimes}\big(\langle q_1, q_2\rangle, \langle q'_1, q'_2\rangle\big) & = & \mathsf{w}^{\mathsf{e}}_{\mathsf{r},1}(q_1, q'_1) \otimes \mathsf{w}^{\mathsf{e}}_{\mathsf{i},2}(q_2, q'_2) \end{array}$$

D Nested Words and Parse Trees

The hierarchical structure of nested words, defined with the *call* and *return* markup symbols suggest a correspondence with trees. The lifting of this correspondence to languages, of tree automata and VPA, has been discussed in [2], and [5] for the weighted case. In this section, we describe a correspondence between the symbolic-weighted extensions of tree automata and VPA.

Let Ω be a countable ranked alphabet, such that every symbol $a \in \Omega$ has a rank $\mathsf{rk}(a) \in [0..M]$ where M is a fixed natural number. We denote by Ω_k the subset of all symbols a of Ω with $\mathsf{rk}(a) = k$, where $0 \leq k \leq M$, and $\Omega_{>0} = \Omega \setminus \Omega_0$. The free Ω-algebra of finite, ordered, Ω-labeled trees is denoted by $\mathcal{T}_\Omega$. It is the smallest set such that $\Omega_0 \subset \mathcal{T}_\Omega$ and for all $1 \leq k \leq M$, all $a \in \Omega_k$, and all $t_1, \ldots, t_k \in \mathcal{T}_\Omega$, $a(t_1, \ldots, t_k) \in \mathcal{T}_\Omega$. Let us assume a commutative semiring $\mathbb{S}$ and a label theory $\bar{\Phi}$ over $\mathbb{S}$ containing one set Φ_{Ω_k} for each $k \in [0..M]$.

▶ **Definition 20.** *A symbolic-weighted tree automaton (swTA) over Ω, $\mathbb{S}$, and $\bar{\Phi}$ is a triplet $A = \langle Q, \mathsf{in}, \bar{\mathsf{w}} \rangle$ where Q is a finite set of states, $\mathsf{in} : Q \to \Phi_\Omega$ is the starting weight function, and $\bar{\mathsf{w}}$ is a tuplet of transition functions containing, for each $k \in [0..M]$, the functions $\mathsf{w}_k : Q \times Q^k \to \Phi_{\Omega_{>0}, \Omega_k}$ and $\mathsf{w}^{\mathsf{e}}_k : Q \times Q^k \to \Phi_{\Omega_k}$.*

We define a transition function $\mathsf{w} : Q \times (\Omega_{>0} \cup \{\varepsilon\}) \times \Omega \times \bigcup_{k=0}^{M} Q^k \to \mathbb{S}$ by:

$$\begin{array}{rcll} \mathsf{w}(q_0, a, b, q_1 \ldots q_k) & = & \eta(a, b) & \text{where } \eta = \mathsf{w}_k(q_0, q_1 \ldots q_k) \\ \mathsf{w}(q_0, \varepsilon, b, q_1 \ldots q_k) & = & \phi(b) & \text{where } \phi = \mathsf{w}^{\mathsf{e}}_k(q_0, q_1 \ldots q_k). \end{array}$$

where $q_1 \ldots q_k$ is ε if $k = 0$. The first case deals with a strict subtree, with a parent node labeled by a, and the second case is for a root tree.
Every swTA defines a mapping from trees of $\mathcal{T}_\Omega$ into $\mathbb{S}$, based on the following intermediate function $\mathsf{weight}_A : Q \times (\Omega \cup \{\varepsilon\}) \times \mathcal{T}_\Omega \to \mathbb{S}$

$$\mathsf{weight}_A(q_0, a, t) = \bigoplus_{q_1 \ldots q_k \in Q^k} \mathsf{w}(q_0, a, b, q_1 \ldots q_k) \otimes \bigotimes_{i=1}^{k} \mathsf{weight}_A(q_i, b, t_i) \tag{20}$$

where $q_0 \in Q$, $a \in \Omega_{>0} \cup \{\varepsilon\}$ and $t = b(t_1, \ldots, t_k) \in \mathcal{T}_\Omega$, $0 \leq k \leq M$.

Finally, the weight associated by A to $t \in \mathcal{T}_\Omega$ is

$$A(t) = \bigoplus_{q \in Q} \mathsf{in}(q) \otimes \mathsf{weight}_A(q, \varepsilon, t) \tag{21}$$

Intuitively, $\mathsf{w}(q_0, a, b, q_1 \ldots q_k)$ can be seen as the weight of a production rule $q_0 \to b(q_1, \ldots, q_k)$ of a regular tree grammar [6], that replaces the non-terminal symbol q_0 by $b(q_1, \ldots, q_k)$, provided that the parent of q_0 is labeled by a (or q_0 is the root node if $a = \varepsilon$). The above production rule can also be seen as a rule of a weighted CF grammar, of the form $[a, b]\, q_0 := q_1 \ldots q_k$ if $k > 0$, and $[a]\, q_0 := b$ if $k = 0$. In the first case, b is a label of the rule, and in the second case, it is a terminal symbol. And in both cases, a is a constraint on the label of rule applied on the parent node in the derivation tree. This features of observing the parent's label are useful in the case of infinite alphabet, where it is not possible to memorize a label with the states. The weight of a labeled derivation tree t of the weighted CF grammar associated to A as above, is $\mathsf{weight}_A(q, t)$, when q is the start non-terminal. We shall now establish a correspondence between such a derivation tree t and some word describing a linearization of t, in a way that $\mathsf{weight}_A(q, t)$ can be computed by a sw-VPA.

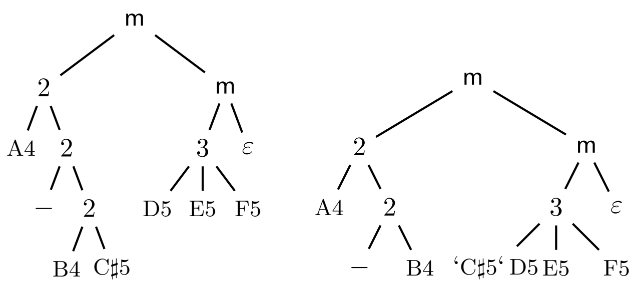

Figure 4 Tree representation of scores of Examples 1,12, linearized respectively into O and O'.

Let $\hat{\Omega}$ be the countable (unranked) alphabet obtained from Ω by: $\hat{\Omega} = \Delta_{\mathsf{i}} \uplus \Delta_{\mathsf{c}} \uplus \Delta_{\mathsf{r}}$, with $\Delta_{\mathsf{i}} = \Omega_0$, $\Delta_{\mathsf{c}} = \{\, \langle_a |\; a \in \Omega_{>0}\}$, $\Delta_{\mathsf{r}} = \{\, {}_a\rangle \mid a \in \Omega_{>0}\}$.
We associate to $\hat{\Omega}$ a label theory $\hat{\Phi}$ like in Section 4, and 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 $b \in \Omega_k$ for $1 \leq k \leq M$.

▶ **Example 21.** The trees in Figure 4 represent the two scores in Examples 1,12, and their linearization are respectively O and O' inn the same examples.

▶ **Proposition 22.** *For all swTA A over Ω, $\mathbb{S}$ commutative, and $\bar{\Phi}$, there exists an effectively constructible sw-VPA A' over $\hat{\Omega}$, $\mathbb{S}$ and $\hat{\Phi}$ such that for all $t \in \mathcal{T}_\Omega$, $A'\big(\mathsf{lin}(t)\big) = A(t)$.*

Proof. Let $A = \langle Q, \mathsf{in}, \bar{\mathsf{w}}\rangle$ where $\bar{\mathsf{w}}$ is presented as above by a function We build $A' = \langle Q', P', \mathsf{in}', \bar{\mathsf{w}}', \mathsf{out}'\rangle$, where $Q' = \bigcup_{k=0}^{M} Q^k$ is the set of sequences of state symbols of A, of length at most M, including the empty sequence denoted by ε, and where $P' = Q'$ and $\bar{\mathsf{w}}$ is defined by:

$$
\begin{array}{lcll}
\mathsf{w}_{\mathsf{i}}(q_0\, \bar{u}, \langle_c, \bar{p}, a, \bar{u}) & = & \mathsf{w}(q_0, c, a, \varepsilon) & \text{for all } c \in \Omega_{>0}, a \in \Omega_0 \\
\mathsf{w}^{\mathsf{e}}_{\mathsf{i}}(q_0\, \bar{u}, a, \bar{u}) & = & \mathsf{w}(q_0, \varepsilon, a, \varepsilon) & \text{for all } a \in \Omega_0 \\
\mathsf{w}_{\mathsf{c}}(q_0\, \bar{u}, \langle_c, \bar{p}, \langle_d, \bar{u}, \bar{q}) & = & \mathsf{w}(q_0, c, d, \bar{q}) & \text{for all } c, d \in \Omega_{>0} \\
\mathsf{w}^{\mathsf{e}}_{\mathsf{c}}(q_0\, \bar{u}, \langle_c, \bar{u}, \bar{q}) & = & \mathsf{w}(q_0, \varepsilon, c, \bar{q}) & \text{for all } c \in \Omega_{>0} \\
\mathsf{w}_{\mathsf{r}}(\varepsilon, \langle_c, \bar{p}, {}_c\rangle, \bar{p}) & = & \mathbb{1} & \text{for all } c \in \Omega_{>0} \\
\mathsf{w}^{\mathsf{e}}_{\mathsf{r}}(\bar{u}, {}_c\rangle, \bar{q}) & = & \mathbb{0} & \text{for all } c \in \Omega_{>0}
\end{array}
$$

All cases not matched by one of the above equations have a weight $\mathbb{0}$, for instance $\mathsf{w}_{\mathsf{r}}(\bar{u}, \langle_c, \bar{p}, {}_d\rangle, \bar{q}) = \mathbb{0}$ if $c \neq d$ or $\bar{u} \neq \varepsilon$ or $\bar{q} \neq \bar{p}$. ◀