



HAL
open science

Symbolic Weighted Language Models, Quantitative Parsing and Verification over Infinite Alphabets

Florent Jacquemard, Philippe Rigaux, Lydia Rodriguez de La Nava

► **To cite this version:**

Florent Jacquemard, Philippe Rigaux, Lydia Rodriguez de La Nava. Symbolic Weighted Language Models, Quantitative Parsing and Verification over Infinite Alphabets. 2021. hal-03380268

HAL Id: hal-03380268

<https://hal.inria.fr/hal-03380268>

Preprint submitted on 15 Oct 2021

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Symbolic Weighted Language Models, Quantitative Parsing and Verification over Infinite Alphabets

Florent Jacquemard^{1,2}, Philippe Rigaux², and Lydia Rodriguez de la Nava^{1,2}

¹ INRIA & CNAM, Paris, France

² CNAM, Paris, France

Abstract. We study properties and relationship between three classes of quantitative language models computing over infinite input alphabets: 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 over symbols in an infinite alphabet, and also **wA**, whose transitions are labeled by constant weight values, and which deal only with finite alphabets. We present a Bar-Hillel Perles Shamir construction of a **swA** computing a **swT**-defined distance between a **swA** input language and a word, some closure results and a polynomial best-search algorithm for **sw-VPA**. These results are applied to solve a variant of parsing over infinite alphabets.

1 Introduction

Various extensions of language models have been proposed for handling infinite alphabets. Some automata with memory extensions allow restricted storage and comparison of input symbols (see [33] for a survey). They use pebbles for marking positions [32], registers [22], or separate computation on subsequences with the same attribute values [4]. Moreover, automata at the core of model checkers compute on input symbols represented by large bitvectors [35] and, in practice, each transition accepts a set of such symbols (instead of an individual symbol), represented by Boolean Formulas or Binary Decision Diagrams. Following this idea, in symbolic automata (**sA**) [36,9,10], transitions are guarded by predicates over large or infinite domains. With appropriate closure conditions on the sets of such predicates, all the good properties enjoyed by automata over finite alphabets are preserved. The ability of **sA** to compare input symbols is restricted compared to the former automata with memory. It can however be extended with the addition of a stack [8].

Other extensions of language models assign one weight value to every input [13]. This is useful for the quantitative modelling of *e.g.* probabilistic or

stochastic recursive programs, quantitative database queries, or semi-structured data, as well as for the verification of quantitative properties of systems related to quality measures, resource-consumption, distance metrics, probabilistic guarantees, *etc*. In the context of parsing, when grammars return weight values, it is possible to rank derivations (and *abstract syntax trees*) in order to select a best one (or n bests), *e.g.* in case of ambiguity [15,30,29]. In *weighted language models* like *e.g.* probabilistic context-free grammars and weighted automata (wA) [13], a weight is associated to each transition rule, and the weights of the rules involved in a computation are combined with an associative product operator \otimes . A second operator \oplus is moreover used to resolve the ambiguity raised by the existence of several (in general exponentially many) computations on a given input. Typically, \oplus selects the best of two weight values. The weight domain, equipped with these two operators is typically 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 some results for *Symbolic Weighted* finite states language models generalizing the Boolean guards of sA to functions into an arbitrary semiring, and also the wA, by handling infinite alphabets (Figure 1). In short, a transition rule $q \xrightarrow{\phi} q'$ of a Symbolic Weighted Automaton, 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} . These models are also particular cases of the very general class of Weighted Symbolic Automata with Data Storage [19,18], for appropriate storage types.

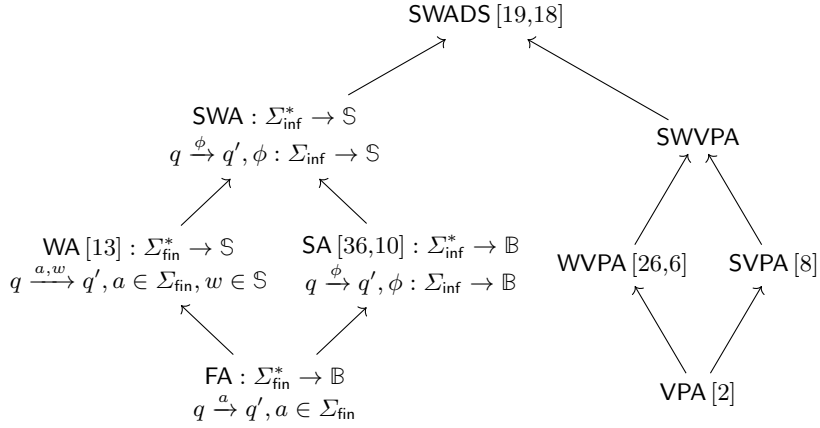


Fig. 1. Classes of Symbolic/Weighted Automata. Here, Σ_{fin} and Σ_{inf} denote finite/countable alphabets, \mathbb{B} the Boolean algebra, \mathbb{S} a commutative semiring. $q \xrightarrow{\phi} q'$ is a transition between states q and q' .

The models studied are: symbolic-weighted automata (swA), defining series over infinite alphabets, transducers (swT), defining distances between finite words

over infinite alphabets, and pushdown automata with a visibility restriction [2] (sw-VPA), operating sequentially on words structured with markup symbols (parentheses) and describing linearizations of trees. The main contributions of this paper are:

- (i) a construction à la Bar-Hillel Perles Shamir of a swA computing a swT-defined distance between a swA input language and a word (Proposition 1),
- (ii) closure results for sw-VPA (Proposition 2),
- (iii) a polynomial best-search algorithm for sw-VPA (Proposition th:best-search).

Moreover, we present in Section 5 an

- (iv) application to the problem of weighted parsing over infinite input alphabets, called *SW-parsing*.

The goal of the latter problem is, given an input word s , to find t minimizing the distance, in the sense of [28], $T(s, t) \otimes A(t)$, where T is a swT and A a sw-VPA. The notion of transducer-based distances allows to consider different infinite alphabets for the input s and output t . Moreover, the use of sw-VPA permits to search for an output t in the form of a (nested) word, instead of a tree. *SW-parsing* is solved with a Bar-Hillel construction [31] of a sw-VPA B such that, for all t , $B(t) = T(s, t) \otimes A(t)$ and the application of a best-search procedure to this automaton B .


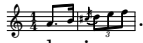
Context-free parsing approaches [17] generally assume a finite and reasonably small input alphabet. Considering large or infinite alphabets can however be of practical interest when dealing with large characters encodings such as UTF-16 [10]. It is also true in the context of automata-based quantitative verification techniques [24], in particular when dealing with data streams, serialization of structured documents [33,32], or timed execution traces [5].

The latter case of timed traces is related to a problem that motivated the present work: automated music transcription, that we shall use as a running example thorough the paper, in order to illustrate our results. Representations capturing music performances are essentially linear [34]; they ignore the hierarchical structures that frame the conception of music, at least in the Western area. Hierarchical structures, on the other hand, are present, either explicitly or implicitly, in Common Western Music Notation [16]: Music scores are partitioned in measures, measures in beats, and beats can be further recursively divided. It follows that written 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 performance (in audio or MIDI format) and aims at re-constructing structured notation, by mapping input events to this hierarchical rhythmic space. It can therefore be stated as a parsing problem over an infinite alphabet of timed events [14].

Example 1. We consider a very simplified example of *music transcription*: a given input *timeline* of musical events from an infinite alphabet Σ , is parsed into a structured music score. Input events of Σ are pairs $\mu:\tau$, where μ is a MIDI

key number [34], 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 3), 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 (♩), eighths (♩), *etc.* For instance, the output score , corresponds to a hierarchical structure that can be linearized as the sequence $O = [{}_m: [0,1], [{}_2: [0,1], A4: 0, [{}_2: [\frac{1}{2},1], -, \frac{1}{2}, [{}_2: [\frac{3}{4},1], B4: \frac{3}{4}, C\sharp 5: \frac{7}{8},]_2: 1,]_2: 1,]_2: 1,]_m: 1, [{}_m: [1,2], [{}_3: [1,2], D5: 1, E5: \frac{4}{3}, F5: \frac{5}{3},]_3: 2,]_m: 2$. The opening markups $[{}_m$ delimit *measures*, which are time intervals of duration 1 in this example, while the subsequences of O between markups $[{}_d$ and $]_d$, for some natural number d , represent a division of the time interval attached to $[{}_d$, of duration ℓ , into d sub-intervals of equal duration $\frac{\ell}{d}$. 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 approach aims at selecting the best one with respect to this weight. \diamond

2 Preliminary Notions

2.1 Semirings

A *semiring* $\langle \mathbb{S}, \oplus, \otimes, \mathbb{0}, \mathbb{1} \rangle$ 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 : $\forall x \in \mathbb{S}, \mathbb{0} \otimes x = x \otimes \mathbb{0} = \mathbb{0}$.

A semiring \mathbb{S} is *commutative* if \otimes is commutative. It is *idempotent* if for every $x \in \mathbb{S}, x \oplus x = x$. Every idempotent semiring \mathbb{S} induces a partial ordering \leq_{\oplus} which is called the *natural ordering* of \mathbb{S} [27], and is defined by: for every $x, y \in \mathbb{S}, x \leq_{\oplus} y$ iff $x \oplus y = x$. The natural ordering is sometimes defined in the opposite direction [12]; 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 \leq_{\oplus} is total, *i.e.* when for every $x, y \in \mathbb{S},$ either $x \oplus y = x$ or $x \oplus y = y$.

Lemma 1 (Monotony, [27]). *Let $\langle \mathbb{S}, \oplus, \otimes, \mathbb{0}, \mathbb{1} \rangle$ be an idempotent semiring. For every $x, y, z \in \mathbb{S},$ if $x \leq_{\oplus} y$ then $x \oplus z \leq_{\oplus} y \oplus z, x \otimes z \leq_{\oplus} y \otimes z$ and $z \otimes x \leq_{\oplus} z \otimes y$.*

We thus say the \mathbb{S} is *monotonic wrt* \leq_{\oplus} . Another important semiring property in the context of optimization is superiority [23,20], which generalizes the *non-negative weights* condition in shortest-path algorithms [11]. Intuitively, it means that combining elements with \otimes always increases their weight. Formally, \mathbb{S} is called *superior wrt* \leq_{\oplus} when property (i) of Lemma 2 below holds.

Lemma 2 (Superiority, Boundedness). *Let $\langle \mathbb{S}, \oplus, \mathbb{0}, \otimes, \mathbb{1} \rangle$ be an idempotent semiring. The two following statements are equivalent:*

- i. for every $x, y \in \mathbb{S}$, $x \leq_{\oplus} x \otimes y$ and $y \leq_{\oplus} x \otimes y$
- ii. for every $x \in \mathbb{S}$, $\mathbb{1} \oplus x = \mathbb{1}$.

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} , $\mathbb{1} \oplus x = \mathbb{1}$.

The property (i) of superiority implies that $\mathbb{1} \leq_{\oplus} x$ for every $x \in \mathbb{S}$. 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 every $x \in \mathbb{S}$, $\mathbb{1} \leq_{\oplus} x \leq_{\oplus} \mathbb{0}$. From an optimization point of view, it means that $\mathbb{1}$ is the best value, and $\mathbb{0}$ the worst. In [27], \mathbb{S} with the property (ii) of Lemma 2 is called *bounded* – we shall use this term in the rest of the paper. Boundedness 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 every $x \in \mathbb{S}$ and $n \geq 1$, $x \oplus x^n = x \otimes (\mathbb{1} \oplus x^{n-1}) = x$.

Lemma 3 ([27], Lemma 3). *Every bounded semiring is idempotent.*

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

We need to extend \oplus to infinitely many operands. A semiring \mathbb{S} is called *complete* [13] if it has an operation $\bigoplus_{i \in I} x_i$ for every family $(x_i)_{i \in I}$ of elements of $\text{dom}(\mathbb{S})$ over an index set $I \subseteq \mathbb{N}$, such that:

- i. *infinite sums extend finite sums:* $\forall j, k \in \mathbb{N}, j \neq k$,

$$\bigoplus_{i \in \emptyset} x_i = \mathbb{0}, \quad \bigoplus_{i \in \{j\}} x_i = x_j, \quad \bigoplus_{i \in \{j, k\}} x_i = x_j \oplus x_k,$$

- ii. *associativity and commutativity:* for all partitions $(I_j)_{j \in J}$ of I ,

$$\bigoplus_{j \in J} \bigoplus_{i \in I_j} x_i = \bigoplus_{i \in I} x_i,$$

- iii. *distributivity of products over infinite sums:* for all $I \subseteq \mathbb{N}$, $\forall x, y \in \mathbb{S}$,

$$\bigoplus_{i \in I} (x \otimes y_i) = x \otimes \bigoplus_{i \in I} y_i, \quad \text{and} \quad \bigoplus_{i \in I} (x_i \otimes y) = \left(\bigoplus_{i \in I} x_i \right) \otimes y.$$

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

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

2.2 Label Theories

We define the functions labeling the transitions of SW automata and transducers, generalizing the Boolean algebras of [36,9]. We consider *alphabets*, which are non-empty countable sets of symbols denoted by Σ, Δ, \dots . Moreover, 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 $(\mathbb{S}, \oplus, \mathbf{0}, \otimes, \mathbf{1})$, a *label theory* $\bar{\Phi}$ over \mathbb{S} is an indexed family of sets denoted by Φ_Σ , containing unary functions of type $\Sigma \rightarrow \mathbb{S}$, or $\Phi_{\Sigma, \Delta}$, containing binary functions $\Sigma \times \Delta \rightarrow \mathbb{S}$, and such that:

- for all $\Phi_{\Sigma, \Delta} \in \bar{\Phi}$, we have $\Phi_\Sigma \in \bar{\Phi}$ and $\Phi_\Delta \in \bar{\Phi}$.
- all $\Phi_\Sigma, \Phi_{\Sigma, \Delta} \in \bar{\Phi}$ contain all the constant functions of $\Sigma \rightarrow \mathbb{S}$, resp. $\Sigma \times \Delta \rightarrow \mathbb{S}$.
- for all $\Phi_\Sigma \in \bar{\Phi}$, for all $\phi \in \Phi_\Sigma$, and $\alpha \in \mathbb{S}$, $\alpha \otimes \phi : x \mapsto \alpha \otimes \phi(x)$, 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 $\Phi_\Sigma \in \bar{\Phi}$, for all $\phi, \phi' \in \Phi_\Sigma$, $\phi \otimes \phi' : x \mapsto \phi(x) \otimes \phi'(x)$ belongs to Φ_Σ .
- for all $\Phi_{\Sigma, \Delta} \in \bar{\Phi}$, 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 $\Phi_\Sigma, \Phi_{\Sigma, \Delta} \in \bar{\Phi}$, 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 $\Phi_\Delta, \Phi_{\Sigma, \Delta} \in \bar{\Phi}$, 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 moreover the following operators on the functions of $\bar{\Phi}$.

$$\begin{aligned} \bigoplus_\Sigma : \Phi_\Sigma &\rightarrow \mathbb{S}, \quad \phi \mapsto \bigoplus_{a \in \Sigma} \phi(a) \\ \bigoplus_\Sigma^1 : \Phi_{\Sigma, \Delta} &\rightarrow \Phi_\Delta, \quad \eta \mapsto (y \mapsto \bigoplus_{a \in \Sigma} \eta(a, y)) \\ \bigoplus_\Delta^2 : \Phi_{\Sigma, \Delta} &\rightarrow \Phi_\Sigma, \quad \eta \mapsto (x \mapsto \bigoplus_{b \in \Delta} \eta(x, b)) \end{aligned}$$

Intuitively, \bigoplus_Σ returns the global minimum, *wrt* \leq_\oplus , of a function ϕ of Φ_Σ , and $\bigoplus_\Sigma^1, \bigoplus_\Delta^2$ return partial minimums of a function ϕ of $\Phi_{\Sigma, \Delta}$.

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 2. 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 . These shifts can be weighted as the time distance between both, computed in the tropical semiring by $\delta \in \Phi_{\Sigma, \Delta}$, defined as follows:

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

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

The following facts are immediate consequences of the definitions of the operators on the functions of labels theories and properties of complete semirings.

Lemma 4. *For a label theory $\bar{\Phi}$ over a complete semiring \mathbb{S} , for all $\Phi_{\Sigma}, \Phi_{\Delta}, \Phi_{\Sigma, \Delta} \in \bar{\Phi}$, $\alpha \in \mathbb{S}$, $\phi, \phi' \in \Phi_{\Sigma}$, $\psi \in \Phi_{\Delta}$, and $\eta \in \Phi_{\Sigma, \Delta}$, it holds that:*

- i. $\bigoplus_{\Sigma} \bigoplus_{\Delta}^2 \eta = \bigoplus_{\Delta} \bigoplus_{\Sigma}^1 \eta$.
- ii. $\alpha \otimes \bigoplus_{\Sigma} \phi = \bigoplus_{\Sigma} (\alpha \otimes \phi)$ and $(\bigoplus_{\Sigma} \phi) \otimes \alpha = \bigoplus_{\Sigma} (\phi \otimes \alpha)$, and similarly for \oplus .
- iii. $(\bigoplus_{\Sigma} \phi) \oplus (\bigoplus_{\Sigma} \phi') = \bigoplus_{\Sigma} (\phi \oplus \phi')$ and $(\bigoplus_{\Sigma} \phi) \otimes (\bigoplus_{\Sigma} \phi') = \bigoplus_{\Sigma} (\phi \otimes \phi')$.
- iv. $(\bigoplus_{\Delta}^2 \eta) \oplus (\bigoplus_{\Delta}^2 \eta') = \bigoplus_{\Delta}^2 (\eta \oplus \eta')$, and $(\bigoplus_{\Delta}^2 \eta) \otimes (\bigoplus_{\Delta}^2 \eta') = \bigoplus_{\Delta}^2 (\eta \otimes \eta')$.
- v. $\phi \otimes (\bigoplus_{\Delta}^2 \eta) = \bigoplus_{\Delta} (\phi \otimes_1 \eta)$, and $(\bigoplus_{\Delta}^2 \eta) \otimes \phi = \bigoplus_{\Delta} (\eta \otimes_1 \phi)$, and similarly for \oplus .
- vi. $\psi \otimes (\bigoplus_{\Sigma}^1 \eta) = \bigoplus_{\Sigma} (\psi \otimes_2 \eta)$, and $(\bigoplus_{\Sigma}^1 \eta) \otimes \psi = \bigoplus_{\Sigma} (\eta \otimes_2 \psi)$, and similarly for \oplus .

The following property of label theories will be useful in the following results, in order to ensure the computability of the above infinite sum operators.

Definition 1. *A function $\phi : \Sigma \rightarrow \mathbb{S}$, resp. $\eta : \Sigma \times \Delta \rightarrow \mathbb{S}$, is called effective when the operations of \mathbb{S} are computable and $\bigoplus_{\Sigma} \phi$, resp. $\bigoplus_{\Delta} \bigoplus_{\Sigma}^1 \eta$ and $\bigoplus_{\Sigma} \bigoplus_{\Delta}^2 \eta$, can be effectively computed from ϕ , resp. η , as well as one symbol of Σ , resp. symbols of Σ and Δ , reaching this bound. A label theory $\bar{\Phi}$ is effective when all its functions are.*

The effectiveness is a strong restriction of label theories, but it is not unrealistic in the context of the problems for languages models considered below, namely the combination of automata, best-search and symbolic weighted parsing. In fact, every instance of such problems comes with a finite number of automata, each one containing a finite number of functions in a label theory, in their transitions. We may assume that the global minimums $\bigoplus_{\Sigma} \phi$, $\bigoplus_{\Delta} \bigoplus_{\Sigma}^1 \eta$, and $\bigoplus_{\Sigma} \bigoplus_{\Delta}^2 \eta$ of all these functions are known. Then, the other functions considered when solving the problems are obtained by combination with the above operators, and are effective by Lemma 4. In practice, the combinations may be represented by structures like Algebraic Decision Diagrams [3].

3 SW Automata and Transducers

We follow the approach of [28] for the computation of distances between words and languages and extend it to infinite alphabets. The models presented in this section are weighted automata and transducers [13] with transitions labeled by weight functions that take the input and output symbols as parameters. These functions generalize the guards of symbolic automata [36,9,10], from Boolean domains to commutative semirings. These models are also particular cases of the very general model of Weighted Symbolic Automata with Data Storage [19]. Let \mathbb{S} be a commutative semiring, Σ and Δ be alphabets called *input* and *output* respectively, and $\bar{\Phi}$ be a label theory over \mathbb{S} containing Φ_Σ , Φ_Δ , $\Phi_{\Sigma,\Delta}$.

Definition 2. A symbolic-weighted transducer (*swT*) over Σ , Δ , \mathbb{S} and $\bar{\Phi}$ is a tuple $T = (Q, \text{in}, \bar{w}, \text{out})$, where Q is a finite set of states, $\text{in}: Q \rightarrow \mathbb{S}$ (respectively $\text{out}: Q \rightarrow \mathbb{S}$) are functions defining the weight for entering (respectively leaving) computation in a state, and \bar{w} is a triplet of transition functions $w_{10}: Q \times Q \rightarrow \Phi_\Sigma$, $w_{01}: Q \times Q \rightarrow \Phi_\Delta$, and $w_{11}: Q \times Q \rightarrow \Phi_{\Sigma,\Delta}$.

Note that the SW-transducers of Definition 2 are actually classical weighted transducers where the alphabets are permitted to be infinite. A tuple of $Q \times (\Sigma \cup \{\varepsilon\}) \times (\Delta \cup \{\varepsilon\}) \times Q \rightarrow \mathbb{S}$ is called a *transition* of T . For convenience, we shall sometimes present the above w_{10} , w_{01} , w_{11} as mappings associating to every transition a weight value in \mathbb{S} as follows, for every $q, q' \in Q$, $a \in \Sigma$, $b \in \Delta$:

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

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



$$\begin{aligned} \text{weight}_T(q, \varepsilon, \varepsilon, q') &= \mathbb{1} \quad \text{if } q = q' \text{ and } \mathbb{0} \text{ otherwise} & (1) \\ \text{weight}_T(q, s, t, q') &= \bigoplus_{\substack{q'' \in Q \\ s=ua, a \in \Sigma}} \text{weight}_T(q, u, t, q'') \otimes w_{10}(q'', a, \varepsilon, q') \\ &\oplus \bigoplus_{\substack{q'' \in Q \\ t=vb, b \in \Delta}} \text{weight}_T(q, s, v, q'') \otimes w_{01}(q'', \varepsilon, b, q') \\ &\oplus \bigoplus_{\substack{q'' \in Q \\ s=ua, t=vb}} \text{weight}_T(q, u, v, q'') \otimes w_{11}(q'', a, b, q') \end{aligned}$$

We recall that, by convention (Section 2), an empty sum with \bigoplus is equal to $\mathbb{0}$. Intuitively, a transition $w_{ij}(q, a, b, q')$ 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 , and neutral for \oplus in \mathbb{S} , if $w_{ij}(q, a, b, q'')$ is equal to $\mathbb{0}$ (meaning that there is no possible transition from state q into state q' while reading a and b), then the entire term containing this expression can be ignored in the sum. This is analogous to the case of a transition's guard not satisfied by $\langle a, b \rangle$ for symbolic transducers [37].

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 [28]. Intuitively, $w_{10}(q, a, \varepsilon, r)$ is the cost of the deletion of the symbol $a \in \Sigma$ in s , $w_{01}(q, \varepsilon, b, r)$ is the cost of the insertion of $b \in \Delta$ in t , and $w_{11}(q, a, b, r)$ 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 in terms of \otimes of the individual costs of the operations involved; and the distance between s and t is the sum in terms of \oplus of all possible products. Formally, the weight associated by T to $\langle s, t \rangle \in \Sigma^* \times \Delta^*$ is:

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

Example 3. We build a small **swT** over the alphabets Σ and Δ_i of Ex. 1 and 2, 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}).

$$\begin{aligned} w_{11}(q_0, \mu: \tau, \nu: \tau', q_0) &= \delta(\mu: \tau, \nu: \tau') \text{ if } \nu \neq - \\ w_{11}(q_1, \mu: \tau, \nu: \tau', q_0) &= \delta(\mu: \tau, \nu: \tau') \text{ if } \nu \neq - \\ w_{01}(q_0, \varepsilon, -: \tau', q_0) &= \mathbb{1} & w_{01}(q_1, \varepsilon, -: \tau', q_0) &= \mathbb{1} \\ w_{10}(q_0, \mu: \tau, \varepsilon, q_1) &= \alpha \end{aligned}$$

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 a 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 $\text{in}(q_0) = \text{out}(q_0) = \mathbb{1}$, and $\text{in}(q_1) = \text{out}(q_1) = \mathbb{0}$. \diamond

Symbolic Weighted Automata are defined as the transducers of Definition 2, by simply omitting the output symbols.

Definition 3. A symbolic-weighted automaton (*swA*) over Σ , \mathbb{S} and $\bar{\Phi}$ is a tuple $A = \langle Q, \text{in}, w_1, \text{out} \rangle$, where Q is a finite set of states, $\text{in} : Q \rightarrow \mathbb{S}$, respectively $\text{out} : Q \rightarrow \mathbb{S}$, 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 .

A *swA transition* is a triplet of $Q \times \Sigma \times Q$, and, as in the case of *swT*, we may write $w_1(q, a, q')$ for $\phi(a)$ when $w_1(q, q') = \phi \in \Phi_\Sigma$. The computation of A on words $s \in \Sigma^*$ is based on an intermediate function weight_A , defined as follows for $q, q' \in Q, a \in \Sigma, u \in \Sigma^*$

$$\begin{aligned} \text{weight}_A(q, \varepsilon, q') &= \mathbb{1} \text{ if } q = q' \text{ and } 0 \text{ otherwise} \\ \text{weight}_A(q, ua, q') &= \bigoplus_{q'' \in Q} \text{weight}_A(q, u, q'') \otimes w_1(q'', a, q') \end{aligned} \quad (3)$$

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

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

Proposition 1. *Given a swT T over $\Sigma, \Delta, \mathbb{S}$ 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 every $t \in \Delta^+$, $B_{A,T}(t) = \bigoplus_{s \in \Sigma^*} A(s) \otimes T(s, t)$.*

Proof. Let $T = \langle Q, \text{in}_T, \bar{w}, \text{out}_T \rangle$, where \bar{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 $A = \langle P, \text{in}_A, w_1, \text{out}_A \rangle$ with $w_1 : P \times P \rightarrow \Phi_\Sigma$. The state set of $B_{A,T}$ will be $Q' = P \times Q$, and its entering, leaving and transition functions will simulate, while reading an output word $t \in \Delta^*$, the synchronized computations of A and T , when they read both the output word t and an input word $s \in \Sigma^*$.

The main difficulty comes from the transitions of T of the form w_{10} , which read in input s and ignore the output t . Since the automaton $B_{A,T}$ only reads the output word t , such transitions would correspond to ε -transitions in $B_{A,T}$. But ε -transitions are not defined for swA. Therefore, we shall perform on-the-fly elimination of the ε -transitions during the construction of $B_{A,T}$, following an approach of [25], Algorithm 1.

The state entering function of $B_{A,T}$ is defined for all $\langle p_1, q_1 \rangle \in Q'$, by:

$$\text{in}'(\langle p_1, q_1 \rangle) = \text{in}_A(p_1) \otimes \text{in}_T(q_1). \quad (5)$$

The transition function w'_1 of $B_{A,T}$ will roughly perform a synchronized product of transitions defined by w_1 (A reading in input word s), w_{01} (T reading in output word t and not in input word s), and w_{11} (T reading both in input word s and in output word t). Moreover, the ε -transitions to eliminate come from the simulation, by w'_1 , of the transitions defined by w_1 and w_{10} (A and T reading in input word s and not in output word t). Let us construct the transition function w'_1 iteratively. Initially, for all $p_1, p_2 \in P$, and $q_1, q_2 \in Q$, let

$$w'_1(\langle p_1, q_1 \rangle, \langle p_2, q_2 \rangle) = \left(\bigoplus_{p_1=p_2} w_{01}(q_1, q_2) \right) \oplus \bigoplus_{\Sigma}^1 (w_1(p_1, p_2) \otimes_1 w_{11}(q_1, q_2)) \quad (6)$$

$$\text{out}'(\langle p_1, q_1 \rangle) = \text{out}_A(p_1) \otimes \text{out}_T(q_1) \quad (7)$$

We recall that by convention, $\bigoplus_{p_1=p_2} w_{01}(q_1, q_2)$ 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 $q_1, q_2, q_3 \in Q$:

$$w'_1(\langle p_1, q_1 \rangle, \langle p_3, q_3 \rangle) \oplus = \bigoplus_{\Sigma} (w_1(p_1, p_2) \otimes w_{10}(q_1, q_2)) \otimes w'_1(\langle p_2, q_2 \rangle, \langle p_3, q_3 \rangle) \quad (8)$$

$$\text{out}'(\langle p_2, q_2 \rangle) \oplus = \bigoplus_{\Sigma} (w_1(p_1, p_2) \otimes w_{10}(q_1, q_2)) \otimes \text{out}'(\langle p_1, q_1 \rangle) \quad (9)$$

In both cases of updates of w'_1 (8) and out' (9) during the iteration, $w_1(p_1, p_2) \otimes w_{10}(q_1, q_2)$ 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 $w_1(p_1, a, p_2)$ of A and a transition $w_{10}(q_1, a, \varepsilon, q_2)$ of T . The iteration stops if it does not change the value of w'_1 and out' . By hypothesis and Lemma 3, \mathbb{S} is idempotent. Therefore, the construction of $B_{A,T}$ will stop after at most $|P|^2 \cdot |Q|^2$ iterations. The correctness of this construction of $B_{A,T}$ is proved in Appendix A. \square


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

Corollary 1. *Given a swTT over $\Sigma, \Delta, \mathbb{S}$ 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 every $t \in \Delta^+$, $B_{s,T}(t) = T(s, t)$.*

4 SW Visibly Pushdown Automata and Best-Search

We consider now a language model generalizing symbolic VPA (sVPA [8], themselves generalizing VPA [2] to infinite alphabets) from Boolean semirings to arbitrary semiring domains. It is also a particular case of Weighted Symbolic Automata with Data Storage [19] for a special type of data storage, and 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 C).

Let Δ be a countable alphabet, partitioned into three pairwise disjoint subsets $\Delta_i, \Delta_c, \Delta_r$, whose elements are respectively called *internal*, *call* and *return* symbols [2]. Let $(\mathbb{S}, \oplus, \mathbb{0}, \otimes, \mathbb{1})$ be a commutative and complete semiring and let $\bar{\Phi} = \langle \Phi_i, \Phi_c, \Phi_r, \Phi_{ci}, \Phi_{cc}, \Phi_{cr} \rangle$ be a label theory over \mathbb{S} where Φ_i, Φ_c, Φ_r and Φ_{cx} (where x is either i, c , or r) stand respectively for $\Phi_{\Delta_i}, \Phi_{\Delta_c}, \Phi_{\Delta_r}$ and $\Phi_{\Delta_c, \Delta_x}$.

Example 4. 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 $O' = [m: [0,1], [2: [0,1], A4: 0, [2: [\frac{1}{2}, 1], -: \frac{1}{2}$,

B4: $\frac{3}{4}$,]₂:1,]₂:1,]_m:1, [̄_m: [1,2], [̄₃: [1,2], ‘C#5’: 1, D5: 1, E5: $\frac{4}{3}$, F5: $\frac{5}{3}$,]₃:2,]_m:2 (see also Fig. 3). The symbol between quotes ‘C#5’ represents an *appoggiatura*, i.e. an ornamental note with theoretical duration 0. \diamond

Definition 4. A Symbolic Weighted Visibly Pushdown Automata (*sw-VPA*) over $\Delta = \Delta_i \uplus \Delta_c \uplus \Delta_r$, \mathbb{S} and $\bar{\Phi}$ is a tuple $A = \langle Q, P, \text{in}, \bar{w}, \text{out} \rangle$, where Q is a finite set of states, P is a finite set of stack symbols, $\text{in} : Q \rightarrow \mathbb{S}$ (respectively $\text{out} : Q \rightarrow \mathbb{S}$) are functions defining the weight for entering (respectively leaving) a state, and \bar{w} is a sextuplet composed of the transition functions : $w_i : Q \times P \times Q \rightarrow \Phi_{ci}$, $w_i^e : Q \times Q \rightarrow \Phi_i$, $w_c : Q \times P \times Q \times P \rightarrow \Phi_{cc}$, $w_c^e : Q \times P \times Q \rightarrow \Phi_c$, $w_r : Q \times P \times Q \rightarrow \Phi_{cr}$, $w_r^e : Q \times Q \rightarrow \Phi_r$.

We extend the above transition functions as follows for every $q, q' \in Q$, $p \in P$, $a \in \Delta_i$, $c \in \Delta_c$, $r \in \Delta_r$, overloading their names:

$$\begin{array}{ll}
w_i : Q \times [\Delta_c \times P] \times \Delta_i \times Q \rightarrow \mathbb{S} & w_i(q, c, p, a, q') = \eta_{ci}(c, a) \\
& \text{where } \eta_{ci} = w_i(q, p, q'), \\
w_i^e : Q \times \Delta_i \times Q \rightarrow \mathbb{S} & w_i^e(q, a, q') = \phi_i(a) \\
& \text{where } \phi_i = w_i^e(q, q'), \\
w_c : Q \times [\Delta_c \times P] \times [\Delta_c \times P] \times Q \rightarrow \mathbb{S} & w_c(q, c, p, c', p', q') = \eta_{cc}(c, c') \\
& \text{where } \eta_{cc} = w_c(q, p, p', q'), \\
w_c^e : Q \times [\Delta_c \times P] \times Q \rightarrow \mathbb{S} & w_c^e(q, c, p', q') = \phi_c(c) \\
& \text{where } \phi_c = w_c^e(q, p, q'), \\
w_r : Q \times [\Delta_c \times P] \times \Delta_r \times Q \rightarrow \mathbb{S} & w_r(q, c, p, r, q') = \eta_{cr}(c, r) \\
& \text{where } \eta_{cr} = w_r(q, p, q'), \\
w_r^e : Q \times \Delta_r \times Q \rightarrow \mathbb{S} & w_r^e(q, r, q') = \phi_r(r) \\
& \text{where } \phi_r = w_r^e(q, q').
\end{array}$$

The tuples in the definition domains of the above functions are also called *transitions* of the *sw-VPA*. We denote by $\text{src}(\tau)$ (respectively $\text{snd}(\tau)$) the first (respectively second) state component of a transition τ . Intuitively, w_i^e , w_c^e , and w_r^e describe the cases where the stack is empty. The transitions of w_i and w_i^e both read an internal symbol a and change state from q to q' , without changing the stack. Moreover, w_i reads a pair made of $c \in \Delta_c$ and $p \in P$ on the top of the stack (c is compared to a by the weight function $\eta_{ci} \in \Phi_{ci}$). The transitions of w_c and w_c^e read the call symbol c' , push it to the stack along with p' , and change state from q to q' . Moreover, w_c reads c and p at the top of the stack (c is compared to c'). Finally, w_r and w_r^e read the return symbol r , and change state from q to q' . Moreover, w_r reads and pops from stack a pair made of c and p (c is compared to r).

Formally, the computations 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 $\Gamma = \Delta_c \times P$. 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 topmost symbol is the last pushed pair. The recursive definition of weight_A enumerates each of the six possible cases: reading $a \in \Delta_i$, or $c \in \Delta_c$, or $r \in \Delta_r$, for each possible state of the stack (empty or not).

$$\begin{aligned}
\text{weight}_A(q[\gamma], \varepsilon, q'[\gamma']) &= \mathbb{1} \text{ if } q = q', \gamma = \gamma' = \perp \text{ and } 0 \text{ otherwise} \quad (10) \\
\text{weight}_A(q[\gamma], u a, q' \left[\begin{array}{c} \langle c, p \rangle \\ \gamma' \end{array} \right]) &= \bigoplus_{q'' \in Q} \text{weight}_A(q[\gamma], u, q'' \left[\begin{array}{c} \langle c, p \rangle \\ \gamma' \end{array} \right]) \otimes w_i(q'', c, p, a, q') \\
\text{weight}_A(q[\gamma], u a, q'[\perp]) &= \bigoplus_{q'' \in Q} \text{weight}_A(q[\gamma], u, q''[\perp]) \otimes w_i^e(q'', a, q') \\
\text{weight}_A(q[\gamma], u c', q' \left[\begin{array}{c} \langle c', p' \rangle \\ \langle c, p \rangle \\ \gamma' \end{array} \right]) &= \bigoplus_{\substack{q'' \in Q \\ p' \in P}} \text{weight}_A(q[\gamma], u, q'' \left[\begin{array}{c} \langle c, p \rangle \\ \gamma' \end{array} \right]) \otimes w_c(q'', c, p, c', p', q') \\
\text{weight}_A(q[\gamma], u c, q'[\langle c, p \rangle]) &= \bigoplus_{\substack{q'' \in Q \\ p \in P}} \text{weight}_A(q[\gamma], u, q''[\perp]) \otimes w_c^e(q'', c, p, q') \\
\text{weight}_A(q[\gamma], u r, q'[\gamma']) &= \bigoplus_{q'' \in Q} \text{weight}_A(q[\gamma], u, q'' \left[\begin{array}{c} \langle c, p \rangle \\ \gamma' \end{array} \right]) \otimes w_r(q'', c, p, r, q') \\
\text{weight}_A(q[\gamma], u r, q'[\perp]) &= \bigoplus_{q'' \in Q} \text{weight}_A(q[\gamma], u, q''[\perp]) \otimes w_r^e(q'', r, q')
\end{aligned}$$

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

$$A(t) = \bigoplus_{q, q' \in Q} \text{in}(q) \otimes \text{weight}_A(q[\perp], t, q'[\perp]) \otimes \text{out}(q') \quad (11)$$

Every $\text{sw}A$ $A = \langle Q, \text{in}, w_1, \text{out} \rangle$, over Σ, \mathbb{S} and $\bar{\Phi}$ is a special case of sw-VPA $\langle Q, \emptyset, \text{in}, \bar{w}, \text{out} \rangle$ over Δ, \mathbb{S} and $\bar{\Phi}$ with $\Delta_i = \Sigma$ and $\Delta_c = \Delta_r = \emptyset$, and computing with an always empty stack: $w_i^e = w_1$ and all the other functions of \bar{w} are the constant \emptyset .

Example 5. We consider a sw-VPA over the alphabet of Example 4 and with $P = Q$, 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 simplest one, *i.e.* the one with the smallest weight.

Let us assume that the top of the stack is a tuple made of the call symbol $\lceil_n: [\tau, \tau + \ell]$ and the state q . Let us now consider a new call transition starting in state $q_{\frac{j}{n}}$, that is the j -th state of the n sub-intervals of duration $\frac{\ell}{n}$. We thus read a new time-division symbol \lceil_d and compute the weight of the transition from $q_{\frac{j}{n}}$ to $q_{\frac{1}{d}}$, the first state after the new \lceil_d , reading on the top of the stack the pair $\langle \lceil_n: [\tau, \tau + \ell], q \rangle$, and pushing $\langle \lceil_d: \iota, q_{\frac{j+1}{n}} \rangle$ on top, or: $w_c(q_{\frac{j}{n}}, \lceil_n: [\tau, \tau + \ell], q, \lceil_d: \iota, q_{\frac{j+1}{n}}, q_{\frac{1}{d}}) = \alpha_d$. Reading the k -th musical event μ in this current sub-interval is computed with: $w_i(q_{\frac{k}{d}}, \lceil_d: [\tau, \tau + \ell], q_{\frac{j}{n}}, \mu: \tau + \frac{j\ell}{d}, q_{\frac{k+1}{d}}) = \alpha_\mu$. Finally, this sub-interval will end after reading a return symbol \lceil_d and compute: $w_r(q_{\frac{j}{d}}, \lceil_d: [\tau, \tau + \ell], q_{\frac{j+1}{n}}, \lceil_d: \tau + \ell, q_{\frac{j+1}{n}}) = \mathbb{1}$.

We described earlier the special cases where the stack is empty, which applies for example when reading the first measure for which we only push $\langle \lceil m: [0,1], q_{\perp} \rangle$ in the stack: $w_c^e(q_{\perp}, \lceil m: [0,1], q_{1/1}, q_{1/1} \rangle) = \mathbb{1}$. In comparison, any other new measure would compute instead: $w_c(q_{\perp}, \lceil m: [\tau-1, \tau], q_{\perp}, \lceil m: [\tau, \tau+1], q_{\perp}, q_{\perp} \rangle) = \mathbb{1}$.

Each transition thus has a weight computing an overall weight for one representation of music notation. When setting the weights, we decide which notation is preferred if possible. For example, let's say we want to prioritize tuplets over triplets, then we set α_2 and α_3 such as $\alpha_2 > \alpha_3$. In conclusion, a sw-VPA is capable of computing several representations of the same piece of music, allowing to choose the best one. \diamond

Similarly to VPA [2] and sVPA [8], the class of sw-VPA is closed under the binary operators of the underlying semiring.

Proposition 2. *Let A_1 and A_2 be two sw-VPA over the same Δ , commutative \mathbb{S} and effective $\bar{\Phi}$. There exist two effectively constructible sw-VPA $A_1 \oplus A_2$ and $A_1 \otimes A_2$, such that for every $s \in \Delta^*$, $(A_1 \oplus A_2)(s) = A_1(s) \oplus A_2(s)$ and $(A_1 \otimes A_2)(s) = A_1(s) \otimes A_2(s)$.*

Proof. We prove the closure under \otimes - the case of \oplus is similar. Let $A_1 = \langle Q_1, P_1, in_1, \bar{w}_1, out_1 \rangle$ and $A_2 = \langle Q_2, P_2, in_2, \bar{w}_2, out_2 \rangle$. 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 $P = P_1 \times P_2$: $A_1 \otimes A_2 = \langle Q, P, in_{\otimes}, \bar{w}_{\otimes}, out_{\otimes} \rangle$. The weight entering and leaving functions in_{\otimes} , out_{\otimes} and the sextuplet of transition functions \bar{w}_{\otimes} are defined using the label-theory operators of Section 2.1 as follows, for all $\langle q_1, q_2 \rangle, \langle q'_1, q'_2 \rangle \in Q$ and $\langle p_1, p_2 \rangle, \langle p'_1, p'_2 \rangle \in P$:

$$\begin{aligned} in_{\otimes}(\langle q_1, q_2 \rangle) &= in_1(q_1) \otimes in_2(q_2) & out_{\otimes}(\langle q_1, q_2 \rangle) &= out_1(q_1) \otimes out_2(q_2) \\ w_{i,\otimes}(\langle q_1, q_2 \rangle, \langle p_1, p_2 \rangle, \langle q'_1, q'_2 \rangle) &= w_{i,1}(q_1, p_1, q'_1) \otimes w_{i,2}(q_2, p_2, q'_2) \\ w_{i,\otimes}^e(\langle q_1, q_2 \rangle, \langle q'_1, q'_2 \rangle) &= w_{i,1}^e(q_1, q'_1) \otimes w_{i,2}^e(q_2, q'_2) \\ w_{c,\otimes}(\langle q_1, q_2 \rangle, \langle p_1, p_2 \rangle, \langle q'_1, q'_2 \rangle, \langle p'_1, p'_2 \rangle) &= w_{c,1}(q_1, p_1, q'_1, p'_1) \otimes w_{c,2}(q_2, p_2, q'_2, p'_2) \\ w_{c,\otimes}^e(\langle q_1, q_2 \rangle, \langle p_1, p_2 \rangle, \langle q'_1, q'_2 \rangle) &= w_{c,1}^e(q_1, p_1, q'_1) \otimes w_{c,2}^e(q_2, p_2, q'_2) \\ w_{r,\otimes}(\langle q_1, q_2 \rangle, \langle p_1, p_2 \rangle, \langle q'_1, q'_2 \rangle) &= w_{r,1}(q_1, p_1, q'_1) \otimes w_{r,2}(q_2, p_2, q'_2) \\ w_{r,\otimes}^e(\langle q_1, q_2 \rangle, \langle q'_1, q'_2 \rangle) &= w_{r,1}^e(q_1, q'_1) \otimes w_{r,2}^e(q_2, q'_2) \end{aligned}$$

With these functions, A simulate the synchronized behaviour of A_1 and A_2 . \square

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

Proposition 3. *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 $A = \langle Q, P, in, \bar{w}, out \rangle$. We propose first a method for computing, for every $q, q' \in Q$, the minimum, wrt \leq_{\oplus} , of the function $\beta_{q,q'} : t \mapsto \mathbf{weight}_A(q[\perp], t, q'[\perp])$. Let us denote by $b_{\perp}(q, q')$ this minimum. By definition of \leq_{\oplus} , and since \mathbb{S} is

total, it holds that (the infinite sum in (12) is well defined since \mathbb{S} is complete):

$$b_{\perp}(q, q') = \bigoplus_{t \in \Delta^*} \text{weight}_A(q[\perp], t, q'[\perp]). \quad (12)$$

Following (11), and the associativity, commutativity and distributivity for \otimes and \oplus , the minimum of $A(t)$ is:

$$\bigoplus_{t \in \Delta^*} A(t) = \bigoplus_{t \in \Delta^*} \bigoplus_{q, q' \in Q} \text{in}(q) \otimes \beta_{q, q'}(t) \otimes \text{out}(q') = \bigoplus_{q, q' \in Q} \text{in}(q) \otimes b_{\perp}(q, q') \otimes \text{out}(q') \quad (13)$$

Hence, in order to prove Proposition 3, it is sufficient to show that we can compute $b_{\perp}(q, q')$ for all $q, q' \in Q$. For this purpose, we shall consider an auxiliary function $b_{\top} : Q \times P \times Q \rightarrow \Phi_c$ that corresponds to the cases where the stack of A is not empty. Intuitively, $b_{\top}(q, p, q')$ is a function of Φ_c , mapping every $c \in \Delta_c$ 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 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 top of γ' , following the third case in the definition (10) of weight_A (call symbol). The pair $\langle c', p' \rangle$ must be popped later, during the computation from q to q' , following the fifth case of (10) (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:

$$b_{\top}(q, p, q') : c \mapsto \bigoplus_{s \in \Delta^*} \text{weight}_A\left(q \left[\begin{array}{c} \langle c, p \rangle \\ \top \end{array} \right], s, q' \left[\begin{array}{c} \langle c, p \rangle \\ \top \end{array} \right] \right) \quad \text{for all } c \in \Delta_c \quad (14)$$

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).

We shall compute the values of the functions b_{\perp} and b_{\top} by search of shortest paths in a \mathbb{S} -labeled graph \mathcal{G}_A associated to the sw-VPA A . This graph is defined by $\mathcal{G}_A = \langle V_A, E_A, \eta_A \rangle$, with set of vertices $V_A = (Q \times Q) \cup (Q \times P \times Q)$, set of edges $E_A = V_A \times V_A$ (an edge $\langle v_1, v_2 \rangle$, with $v_1, v_2 \in V_A$ is denoted by $v_1 \rightarrow v_2$), and with an edge labelling $\eta_A : E_A \rightarrow \mathbb{S}$ defined by, for $q_0, q_1, q_2, q_3 \in Q, p, p' \in P$:

$$\begin{aligned} \langle q_0, q_1 \rangle \rightarrow \langle q_0, q_2 \rangle &\mapsto \bigoplus_{\Delta_i} w_i^e(q_1, q_2) \oplus \bigoplus_{\Delta_r} w_r^e(q_1, q_2) \\ \langle q_1, p, q_2 \rangle \rightarrow \langle q_0, q_3 \rangle &\mapsto \bigoplus_{\Delta_c} [w_c^e(q_0, p, q_1) \otimes \bigoplus_{\Delta_r}^2 w_r(q_2, p, q_3)] \\ \langle q_1, p', q_2 \rangle \rightarrow \langle q_0, p, q_3 \rangle &\mapsto \bigoplus_{q_0=q_1 p=p'} \bigoplus_{\Delta_i} w_i(q_2, p, q_3) \\ &\quad \oplus \bigoplus_{\Delta_c}^2 [w_c(q_0, p, p', q_1) \otimes_2 \bigoplus_{\Delta_r} w_r(q_2, p', q_3)] \end{aligned}$$

A *path* of \mathcal{G}_A is a sequence $\pi = v_0, \dots, v_n \in V_A^*$, where v_0 and v_n are respectively called *fst*(π) and *last*(π). The path π is called *safe* when *fst*(π) has the form $\langle q, q \rangle$ for some $q \in Q$. A path π is assigned a weight value $\text{weight}(\pi) \in \mathbb{S}$ which is the product by \otimes of the weights of the edges involved in the path. More

precisely, for all $q, q' \in Q$ and $p \in P$, we let $\text{weight}(\langle q, q' \rangle) = \mathbb{1}$ if $q = q'$, and $\mathbb{0}$ otherwise, $\text{weight}(\langle q, p, q' \rangle) = \mathbb{0}$ and for all $n \geq 1$, $\text{weight}(v_0, \dots, v_n) = \text{weight}(v_0, \dots, v_{n-1}) \otimes \eta_A(v_{n-1} \rightarrow v_n)$. For $v, v' \in V_A$, let $\Pi_{v, v'}$ be the set of pathes π such that $\text{fst}(\pi) = v$ and $\text{last}(\pi) = v'$, and let $\text{short}(v, v')$ be the weight of the shortest path from v to v' , more precisely, $\text{short}(v, v') = \bigoplus_{\pi \in \Pi_{v, v'}} \text{weight}(\pi)$.

Proposition 4. For all $q, q' \in Q$, $b_{\perp}(q, q') = \bigoplus_{q_0 \in Q} \text{short}(\langle q_0, q_0 \rangle, \langle q, q' \rangle)$.

This proposition is a consequence of Lemmata 5 and 6 found in Appendix B.

Following (13) and Proposition 4, in order to compute the minimum of $t \mapsto A(t)$ wrt \leq_{\oplus} , it is sufficient to compute one shortest paths in \mathcal{G}_A of source $\langle q_0, q_0 \rangle$ and target $\langle q, q' \rangle$ for all $q_0, q, q' \in Q$. This can be done [27] with an overall worst case time complexity in $O(|Q|^2 \cdot |P|^3)$. Moreover, a witness $t \in \Delta^*$ for this minimum can be associated to the appropriate shortest path, with no additional cost, like in the proof of Lemma 5.

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 to be given the following input:

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

For every $u \in \Sigma^*$ and $t \in \Delta^*$, let $d(u, t) = T(u, t|_{\Delta_i})$, 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.

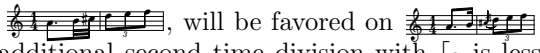
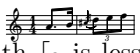
$$d(s, t) \otimes A(t) = \bigoplus_{u \in \Delta^*} d(s, u) \otimes A(u) \quad (15)$$

Following the terminology of [28], sw-parsing is the problem of computing the distance (15) between the input string s and the weighted language over the output alphabet defined by A , and returning a witness t .

Example 6 (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 $\text{swT } T$ evaluates a “fitness measure” expressing a correspondence between a performance and a nested representation of a music score - Ex. 3.
- The $\text{sw-VPA } A$ expresses a cost related to the complexity of music notation.

As seen in Example 5, , will be favored on  when the weight assigned to an additional second time division with $\frac{1}{2}$ is less than the difference of weight between the appoggiatura ‘C#5’ and the standard note 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. \diamond

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 5. *The problem of Symbolic Weighted Parsing can be solved in PTIME in the size of the input $\text{swT } T$, $\text{sw-VPA } A$ and input word s , and the computation time of the functions and operators of the label theory.*

Proof. We follow a *Bar-Hillel* construction for parsing by intersection. We first extend the $\text{swT } T$ over Σ , Δ_i into a $\text{swT } T'$ over Σ and Δ (and the same semiring and label theory \mathbb{S} and $\bar{\Phi}$), such that for every $u \in \Sigma^*$, and $t \in \Delta^*$, $T'(u, t) = T(u, t|_{\Delta_i})$. T' simply skips every symbol $b \in \Delta \setminus \Delta_i$ by the addition of new transitions of the form $w_{01}(q, \varepsilon, b, q')$ to T . Then, using Corollary 1, we construct from $s \in \Sigma^*$ and T' a $\text{swA } B_{s, T'}$, such that for every $t \in \Delta^*$, $B_{s, T'}(t) = d(s, t)$. Next, we compute the $\text{sw-VPA } B_{s, T'} \otimes A$, using Proposition 2. It remains to compute a best nested word $t \in \Delta^*$ for this sw-VPA , using the procedure of Proposition 3. \square

The sw -parsing problem generalizes the problem of searching for the best derivation (AST) of a weighted CF-grammar G that yields a given input word w , with an infinite input alphabet instead of a finite one and transducer-defined distances instead of equality (it is however uncomparable to related problem called *semiring parsing*, [15], called *weighted parsing* [29]). The interested reader might find in Appendix C more details on the correspondence between nested words $t \in \Delta^*$, AST, CF grammars and sw-VPA .

Conclusion

We presented three properties of three Symbolic Weighted language models (automata, transducers and visibly pushdown automata), and applied them to the problem of parsing with infinitely many possible input symbols (typically timed events). One originality of this approach is the comparison of words using a distance between words defined by a given SW transducer. This allows to consider finer word relationships than strict equality.

This work can be extended in several directions. The best-search algorithm for sw-VPA could be generalized from 1-best to n -best [21], and to k -closed semirings [27] (instead of *bounded*, which corresponds to 0-closed). Its complexity upper bound could also certainly be improved. Another possible study could be the existence of best-search algorithms for the more general models of [19]. Finally, the best-search algorithm presented here works offline, whereas an on-the-fly approach coupling automata construction and best-search would be interesting *e.g.* for online XML validation or filtering, or program monitoring [8].

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. R Iris Bahar, Erica A Frohm, Charles M Gaona, Gary D Hachtel, Enrico Macii, Abelardo Pardo, and Fabio Somenzi. Algebraic decision diagrams and their applications. *Formal methods in system design*, 10(2-3):171–206, 1997.
4. 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.
5. Patricia Bouyer, Antoine Petit, and Denis Thérien. An algebraic approach to data languages and timed languages. *Information and Computation*, 182(2):137–162, 2003.
6. 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.
7. 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.
8. Loris D’Antoni and Rajeev Alur. Symbolic visibly pushdown automata. In *International Conference on Computer Aided Verification*, pages 209–225. Springer, 2014.
9. 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.
10. Loris D’Antoni and Margus Veanes. Automata modulo theories. *Communications of the ACM*, 64(5):86–95, 2021. URL: <https://pages.cs.wisc.edu/~loris/symbolicautomata.html>.
11. E. W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1(1):269–271, 1959.
12. Manfred Droste and Werner Kuich. Semirings and formal power series. In *Handbook of Weighted Automata*, pages 3–28. Springer, 2009.
13. Manfred Droste, Werner Kuich, and Heiko Vogler. *Handbook of weighted automata*. Springer Science & Business Media, 2009.
14. 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 *LNAI*. Springer, 2019.

15. Joshua Goodman. Semiring parsing. *Computational Linguistics*, 25(4):573–606, 1999.
16. Elaine Gould. *Behind Bars: The Definitive Guide to Music Notation*. Faber Music, 2011.
17. Dick Grune and Criel J.H. Jacobs. *Parsing Techniques*. Number 2nd edition in Monographs in Computer Science. Springer, 2008.
18. Luisa Herrmann. *Weighted Automata with Storage*. PhD thesis, Technische Universität Dresden, Dresden, 2020.
19. Luisa Herrmann and Heiko Vogler. Weighted symbolic automata with data storage. In *International Conference on Developments in Language Theory*, pages 203–215. Springer, 2016.
20. Liang Huang. Advanced dynamic programming in semiring and hypergraph frameworks. In *In COLING*, 2008.
21. 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.
22. Michael Kaminski and Nissim Francez. Finite-memory automata. *Theoretical Computer Science*, 134:329–363, November 1994.
23. Donald Knuth. A generalization of Dijkstra’s algorithm. *Inform. Process. Lett.*, 6(1), 1977.
24. Marta Kwiatkowska. Quantitative verification: models techniques and tools. In *Proceedings of the the 6th joint meeting of the European software engineering conference and the ACM SIGSOFT symposium on The foundations of software engineering*, pages 449–458, 2007.
25. 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.
26. Christian Mathissen. Weighted logics for nested words and algebraic formal power series. In *International Colloquium on Automata, Languages, and Programming*, pages 221–232. Springer, 2008.
27. Mehryar Mohri. Semiring frameworks and algorithms for shortest-distance problems. *Journal of Automata, Languages and Combinatorics*, 7(3):321–350, 2002.
28. Mehryar Mohri. Edit-distance of weighted automata: General definitions and algorithms. *International Journal of Foundations of Computer Science*, 14(06):957–982, 2003.
29. 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. Association for Computational Linguistics, 2019.
30. Mark-Jan Nederhof. Weighted deductive parsing and Knuth’s algorithm. *Computational Linguistics*, 29(1):135–143, 2003.
31. Mark-Jan Nederhof and Giorgio Satta. Probabilistic parsing as intersection. In *Proceedings of the Eighth International Conference on Parsing Technologies*, pages 137–148, 2003.
32. 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.
33. Luc Segoufin. Automata and logics for words and trees over an infinite alphabet. In *Computer Science Logic*, volume 4207 of *LNCS*. Springer, 2006.
34. Eleanor Selfridge-Field, editor. *Beyond MIDI: the handbook of musical codes*. MIT press, 1997. URL: <http://beyondmidi.ccarh.org/beyondmidi-600dpi.pdf>.

35. 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.
36. Margus Veanes, Pieter Hooimeijer, Benjamin Livshits, David Molnar, and Nikolaj Bjorner. Symbolic finite state transducers: Algorithms and applications. In *Proceedings of the 39th annual ACM SIGPLAN-SIGACT symposium on Principles of programming languages*, pages 137–150, 2012.
37. Margus Veanes, Pieter Hooimeijer, Benjamin Livshits, David Molnar, and Nikolaj Bjorner. Symbolic finite state transducers: Algorithms and applications. In *ACM SIGPLAN Notices*, volume 47, pages 137–150. ACM, 2012.

A End of proof of Proposition 1

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

We call *run* from $\langle p_0, q_0 \rangle$ to $\langle p_n, q_n \rangle$ a finite sequence of the form: $\rho = \langle p_0, q_0 \rangle, a_1, \langle p_1, q_1 \rangle, \dots, a_n, \langle p_n, q_n \rangle$ 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 run, denoted $src(\rho)$ and $\langle p_n, q_n \rangle$ is called target of the run, denoted $trg(\rho)$; the set of runs with source $\langle p, q \rangle$ and target $\langle p', q' \rangle$ is denoted $R(\langle p, q \rangle, \langle p', q' \rangle)$. The word $a_1 \dots a_n \in \Sigma^*$ is called word of the run ρ and denoted $word(\rho)$. Moreover, we associate a weight value in \mathbb{S} to every run, defined by:

$$\text{weight}(\rho) = \bigotimes_{i=1}^n w_1(p_{i-1}, a_i, p_i) \otimes w_{10}(q_{i-1}, a_i, \varepsilon, q_i) \quad (16)$$

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

$$\text{weight}_A(p, s, p') \otimes \text{weight}_T(q, s, \varepsilon, q') = \bigoplus_{\substack{\rho \in R(\langle p, q \rangle, \langle p', q' \rangle) \\ word(\rho) = s}} \text{weight}(\rho) \quad (17)$$

Using (16) and Lemma 4 repetively, (17) implies that:

$$\begin{aligned} \bigoplus_{s \in \Sigma^*} \text{weight}_A(p, s, p') \otimes \text{weight}_T(q, s, \varepsilon, q') = \\ \bigoplus_{\substack{\rho \in R(\langle p, q \rangle, \langle p', q' \rangle) \\ \rho = \langle p_0, q_0 \rangle, a_1, \langle p_1, q_1 \rangle, \dots, a_n, \langle p_n, q_n \rangle}} \bigoplus_{i=1}^n \bigoplus_{\Sigma} (w_1(p_{i-1}, p_i) \otimes w_{10}(q_{i-1}, q_i)) \end{aligned} \quad (18)$$

Note that the symbols $a_1, \dots, a_n \in \Sigma$ in the run ρ are not significant in (18). Using a pumping argument, we can show that (18) still holds when restricting ρ to the set $R_0(\langle p, q \rangle, \langle p', q' \rangle)$ of runs without repetition in the state symbols. Indeed, assume that in $\rho = \langle p_0, q_0 \rangle, a_1, \langle p_1, q_1 \rangle, \dots, a_n, \langle p_n, q_n \rangle, \langle p_{i_1}, q_{i_1} \rangle = \langle p_{i_2}, q_{i_2} \rangle$ for $0 \leq i_1 < i_2 \leq n$. Then $\rho' = \langle p_0, q_0 \rangle, \dots, a_{i_1-1}, \langle p_{i_1-1}, q_{i_1-1} \rangle, a_{i_2}, \langle p_{i_2}, q_{i_2} \rangle, \dots, a_n, \langle p_n, q_n \rangle$ also belongs to $R(\langle p_0, q_0 \rangle, \langle p_n, q_n \rangle)$ and yields a smaller expression ($wrt \leq_{\oplus}$) in the right-hand-side of (18) than ρ . It follows, by (6) and (8), that for all $b \in \Delta$,

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

where $\psi_1 = w_{01}(q'', q') \oplus \bigoplus_{\Sigma}^1 (w_1(p'', p') \otimes w_{11}(q'', q'))$.

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

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

This permits to conclude, using the definition of in' in (5), and the definition of out' in (7), and (9).

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 $s \in \Sigma^*$:

$$\begin{aligned} \text{weight}_A(p, s, p') \otimes \text{weight}_T(q, s, t, q') &= \bigoplus_{s=s_1 s_2} \bigoplus_{\substack{p'', p''' \in P \\ q'', q''' \in Q}} \text{weight}_A(p, s_1, p'') \otimes \text{weight}_A(p''', s_2, p') \otimes \\ &\quad \text{weight}_T(q, s_1, \varepsilon, q'') \otimes \left(\bigoplus_{q''' \in Q} w_{01}(q'', \varepsilon, b, q''') \otimes \text{weight}_T(q''', s_2, u, q') \oplus \right. \\ &\quad \left. \bigoplus_{q''' \in Q} \bigoplus_{s_2=as'_2} w_{11}(q'', a, b, q''') \otimes \text{weight}_T(q''', s'_2, u, q') \right) \end{aligned}$$

Using (19), it follows that:

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

with $\psi_1 = w_{01}(q'', q''') \oplus \bigoplus_{\Sigma}^1 (w_1(p'', p''') \otimes_1 w_{11}(q'', q'''))$.

The first term in the right-hand-side is $w'_1(\langle p, q \rangle, b, \langle p''', q''' \rangle)$ by (19), and the second term is $\text{weight}_{B_{A,T}}(\langle p''', q''' \rangle, u, \langle p', q' \rangle)$ by induction hypothesis. Hence, by definition,

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

B Proof of Proposition 4

Proposition 4 states the correctness of the construction of the graph G_A for the computation of b_{\perp} and, by extension, the minimum of A . Its proof is a consequence of the two following Lemmata 5 and 6.

Lemma 5 (Correctness). *For all safe path π of \mathcal{G}_A , (i) if $\text{last}(\pi) = \langle q, q' \rangle$, then there exists $u \in \Delta^*$ such that $\text{weight}_A(q[\perp], u, q'[\perp]) = \text{weight}(\pi)$, (ii) if $\text{last}(\pi) = \langle q, p, q' \rangle$, then there exists $v \in \Delta^*$ such that*

$$\bigoplus_{c \in \Delta_c} \text{weight}_A \left(q \begin{bmatrix} \langle c, p \rangle \\ \top \end{bmatrix}, v, q' \begin{bmatrix} \langle c, p \rangle \\ \top \end{bmatrix} \right) = \text{weight}(\pi).$$

Proof. We prove (i) and (ii) simultaneously by induction on the length $|\pi|$ of the path π .

The base case $|\pi| = 1$ is a direct consequence of the definition of weight of paths of length 1 and the first line of (10).

If $\pi = v_0, \dots, v_n$ with $n \geq 1$, we assume that Lemma 5 holds for a safe path $\pi' = v_0, \dots, v_{n-1}$ and a word $u' \in \Delta^*$, and do a case analysis on the edge $v_{n-1} \rightarrow v_n$.

Firstly, let us consider the case where $v_{n-1} = \langle q, q'' \rangle$ and $v_n = \langle q, q' \rangle$. By definition, the edge $\langle q, q'' \rangle \rightarrow \langle q, q' \rangle$ is possible only if an internal symbol or a return symbol is read, keeping the stack empty:

- if we read $a \in \Delta_i$ and have an empty stack, then the new weight_A is computed with the third case in equation (10). We have

$$\begin{aligned} \text{weight}_A(q[\gamma], u' a, q'[\perp]) &= \text{weight}_A(q[\gamma], u', q''[\perp]) \otimes w_i^e(q'', a, q') \\ &= \text{weight}(\pi') \otimes w_i^e(q'', a, q') \\ &= \text{weight}(\pi) \end{aligned}$$

since we know that $\text{weight}(v_0, \dots, v_{n-1}) \otimes \eta_A(v_{n-1} \rightarrow v_n) = \text{weight}(v_0, \dots, v_n)$ and $w_i^e(q'', a, q')$ is indeed a possible label η_A for the edge $\langle q, q'' \rangle \rightarrow \langle q, q' \rangle$ (first line of label definition).

- Following the same reasoning as above, we find that the Lemma is also true when $a \in \Delta_r$ by using the last equation in (10) which corresponds to the case of an unmatched return symbol.

let us now consider the case where $v_{n-1} = \langle q_1, p, q_2 \rangle$ and $v_n = \langle q_0, q_3 \rangle$. The edge $\langle q_1, p, q_2 \rangle \rightarrow \langle q_0, q_3 \rangle$ exists if we have a return symbol matching a call symbol on the top of the stack, which is pop, leaving the stack empty. With the 6th equation in (10), we can compute the weight_A for $\langle q_0, q_3 \rangle$ and then replace the first term with the 5th equation in (10), such as:

$$\begin{aligned} \text{weight}_A(q_0[\perp], u r, q_3[\perp]) &= \text{weight}_A(q_0[\perp], u, q_2 \left[\begin{array}{c} \langle c, p \rangle \\ \perp \end{array} \right]) \otimes w_r(q_2, c, p, r, q_3) \\ &= \text{weight}_A(q_1 \left[\begin{array}{c} \langle c, p \rangle \\ \perp \end{array} \right], u, q_2 \left[\begin{array}{c} \langle c, p \rangle \\ \perp \end{array} \right]) \otimes w_c^e(q_0, c, p, q_1) \otimes w_r(q_2, c, p, r, q_3) \\ &= \text{weight}_A(\pi') \otimes w_c^e(q_0, c, p, q_1) \otimes w_r(q_2, c, p, r, q_3) \\ &= \text{weight}_A(\pi) \end{aligned}$$

Here the Lemma holds since $w_r(q_2, c, p, r, q_3)$ is also part of the edge labels (second line).

Finally, for the case where $v_{n-1} = \langle q_1, p, q_2 \rangle$ and $v_n = \langle q_0, p', q_3 \rangle$ for $p, p' \in Q$, we can following the same reasoning as the two previous cases, and complete the proof of Lemma algo-correct. \square

Lemma 6 (Completeness). *For all $q, q' \in Q$, $p \in P$, (i) there exists a safe path π of \mathcal{G}_A such that $\text{last}(\pi) = \langle q, q' \rangle$, and $\text{weight}(\pi) = b_\perp(q, q')$, (ii) there exists a safe path π' of \mathcal{G}_A such that $\text{last}(\pi') = \langle q, p, q' \rangle$, and $\text{weight}(\pi') = \bigoplus_{\Delta_c} b_\top(q, p, q')$.*

Proof. By associativity, commutativity and distributivity for \mathbb{S} , (12) can be rewritten into the form, unfolding (10):

$$b_{\perp}(q, q') = \bigoplus_{t \in \Delta^*} \bigoplus_{\substack{q_0, \dots, q_n \in Q \\ p_i, \dots, p_k \in P}} \bigotimes_{i=1}^n w_i(\tau_i) \quad (20)$$

where n is the length of t , $k \leq n$, $q_0 = q$, $q_n = q'$, for all $1 \leq i < n$, w_i is one of the functions of \bar{w} , τ_i is a transition of A and $\text{src}(\tau_i) = q_{i-1}$, $\text{snd}(\tau_i) = q_i$. Since \mathbb{S} is total, there exists finite sequences as above such that $b_{\perp}(q, q') = \bigotimes_{i=1}^n w_i(\tau_i)$. There might exist, for q and q' , several finite sequences $\bar{\tau}$ and \bar{w} as above, let us choose arbitrarily one of minimal length n . This integer n will be denoted $n_{q, q'}$ in the following.

Similarly, following (14) and (10), $\bigoplus_{\Delta_c} b_{\top}(q, p, q')$ can be put in the form:

$$\bigoplus_{\Delta_c} b_{\top}(q, p, q') = \bigoplus_{\Delta_c} \bigoplus_{\substack{t \in \Delta^* \\ p_i, \dots, p_k \in P}} \bigoplus_{q_0, \dots, q_n \in Q} \bigotimes_{i=1}^n w_i(\tau_i) \quad (21)$$

and hence $\bigoplus_{\Delta_c} b_{\top}(q, p, q') = \bigotimes_{i=1}^n w_i(\tau_i)$, using $c \in \Delta_c$ that minimize the function in rhs of (21). We denote the smallest n as above by $n_{q, p, q'}$.

We show now the existence of a path π as expected, by simultaneous induction on $n_{q, q'}$ and $n_{q, p, q'}$.

The base case, $n = 0$ corresponds to $t = \varepsilon$. In this case, by (10), $b_{\top}(q, p, q') = 0$, $b_{\perp}(q, q) = \mathbb{1}$ and $b_{\perp}(q, q') = 0$ if $q \neq q'$.

For $n = n_{q, q'} > 0$, let $\bar{\tau}$ and \bar{w} be the sequences associated to q and q' as above. We can perform a case analysis of w_n . Computing $b_{\perp}(q, q')$ comes down to running A on a word $u \in \Delta^*$, such that $|u| = n$, and u is well-matched except for unmatched return symbols read on empty stack. Let us consider the case where $w_n = w_i^e$. Then using Equation 10, we can decompose $b_{\perp}(q, q')$ as follows:

$$\begin{aligned} b_{\perp}(q, q') &= \text{weight}_A(q[\perp], u', q''[\perp]) \otimes w_i^e(q'', a, q') \\ &= b_{\perp}(q, q'') \otimes w_i^e(q'', a, q') \end{aligned}$$

Since $u = u'a$ is minimal in length, then so is u' , and $\text{weight}_A(q[\perp], u', q''[\perp]) = b_{\perp}(q, q'')$. Consequently, for this case the Lemma holds. The other cases can be proven similarly, but only for the w_n that can form a well-matched word. For example, w_n cannot be w_c since this call would not have a matching return. \square

C 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 [6] 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 $\text{rk}(a) \in [0..M]$ where M is a fixed natural number. We denote by Ω_k the subset of all symbols a of Ω with $\text{rk}(a) = k$, 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}_Ω . 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 $t_1, \dots, t_k \in \mathcal{T}_\Omega$, $a(t_1, \dots, t_k) \in \mathcal{T}_\Omega$. 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 5. A symbolic-weighted tree automaton (swTA) over Ω , \mathbb{S} , and $\bar{\Phi}$ is a triplet $A = \langle Q, \text{in}, \bar{w} \rangle$ where Q is a finite set of states, $\text{in} : Q \rightarrow \Phi_\Omega$ is the starting weight function, and \bar{w} is a tuple of transition functions containing, for each $k \in [0..M]$, the functions $w_k : Q \times Q^k \rightarrow \Phi_{\Omega_{>0}, \Omega_k}$ and $w_k^\varepsilon : Q \times Q^k \rightarrow \Phi_{\Omega_k}$.

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

$$\begin{aligned} w(q_0, a, b, q_1 \dots q_k) &= \eta(a, b) & \text{where } \eta &= w_k(q_0, q_1 \dots q_k) \\ w(q_0, \varepsilon, b, q_1 \dots q_k) &= \phi(b) & \text{where } \phi &= w_k^\varepsilon(q_0, q_1 \dots q_k). \end{aligned}$$

where $q_1 \dots q_k$ 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}_Ω into \mathbb{S} , based on the following intermediate function $\text{weight}_A : Q \times (\Omega \cup \{\varepsilon\}) \times \mathcal{T}_\Omega \rightarrow \mathbb{S}$

$$\text{weight}_A(q_0, a, t) = \bigoplus_{q_1 \dots q_k \in Q^k} w(q_0, a, b, q_1 \dots q_k) \otimes \bigotimes_{i=1}^k \text{weight}_A(q_i, b, t_i) \quad (22)$$

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

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

$$A(t) = \bigoplus_{q \in Q} \text{in}(q) \otimes \text{weight}_A(q, \varepsilon, t) \quad (23)$$

Intuitively, $w(q_0, a, b, q_1 \dots q_k)$ can be seen as the weight of a production rule $q_0 \rightarrow b(q_1, \dots, q_k)$ of a regular tree grammar [7], that replaces the non-terminal symbol q_0 by $b(q_1, \dots, q_k)$, 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 $[a, b] q_0 := q_1 \dots q_k$ 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 $\text{weight}_A(q, t)$, 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 $\text{weight}_A(q, t)$ can be computed by a sw-VPA.

Let $\hat{\Omega}$ be the countable (unranked) alphabet obtained from Ω by: $\hat{\Omega} = \Delta_i \uplus \Delta_c \uplus \Delta_r$, with $\Delta_i = \Omega_0$, $\Delta_c = \{ \langle a \mid a \in \Omega_{>0} \rangle \}$, $\Delta_r = \{ a \mid a \in \Omega_{>0} \}$. We associate to $\hat{\Omega}$ a label theory $\hat{\Phi}$ like in Section 4, and we define a linearization of trees of $\mathcal{T}_{\hat{\Omega}}$ into words of $\hat{\Omega}^*$ as follows:

$$\begin{aligned} \text{lin}(a) &= a \text{ for all } a \in \Omega_0, \\ \text{lin}(b(t_1, \dots, t_k)) &= \langle_b \text{lin}(t_1) \dots \text{lin}(t_k)_b \rangle \text{ when } b \in \Omega_k \text{ for } 1 \leq k \leq M. \end{aligned}$$

Example 7. The trees in Figure 3 represent the two scores in Examples 1,4, and their linearization are respectively O and O' in the same examples.

Proposition 6. *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 $t \in \mathcal{T}_{\hat{\Omega}}$, $A'(\text{lin}(t)) = A(t)$.*

Proof. Let $A = \langle Q, \text{in}, \bar{w} \rangle$ where \bar{w} is presented as above by a function. We build $A' = \langle Q', P', \text{in}', \bar{w}', \text{out}' \rangle$, 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{w}' is defined by:

$$\begin{aligned} w_i(q_0 \bar{u}, \langle_c, \bar{p}, a, \bar{u} \rangle) &= w(q_0, c, a, \varepsilon) \text{ for all } c \in \Omega_{>0}, a \in \Omega_0 \\ w_i^e(q_0 \bar{u}, a, \bar{u}) &= w(q_0, \varepsilon, a, \varepsilon) \text{ for all } a \in \Omega_0 \\ w_c(q_0 \bar{u}, \langle_c, \bar{p}, \langle_d, \bar{u}, \bar{q} \rangle \rangle) &= w(q_0, c, d, \bar{q}) \text{ for all } c, d \in \Omega_{>0} \\ w_c^e(q_0 \bar{u}, \langle_c, \bar{u}, \bar{q} \rangle) &= w(q_0, \varepsilon, c, \bar{q}) \text{ for all } c \in \Omega_{>0} \\ w_r(\varepsilon, \langle_c, \bar{p}, c \rangle, \bar{p}) &= \mathbb{1} \text{ for all } c \in \Omega_{>0} \\ w_r^e(\bar{u}, c, \bar{q}) &= 0 \text{ for all } c \in \Omega_{>0} \end{aligned}$$

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

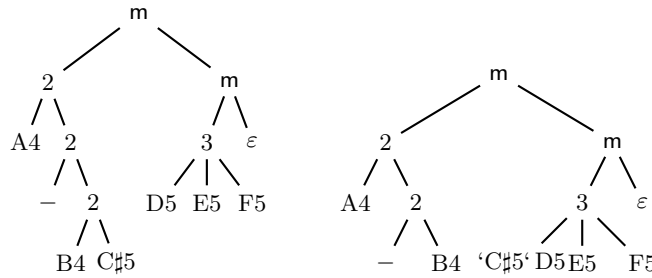


Fig. 3. Tree representation of the scores of Ex 1,4, linearized respectively into O and O' .