

Symbolic Weighted Language Models and Quantitative Parsing over Infinite Alphabets

Florent Jacquemard @ [ORCID](#)

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 is unlike **sA** whose transitions are guarded by boolean predicates over symbols in an infinite alphabet and also unlike **wA** whose transitions are labeled by constant weight values, and who deal only with finite automata. We present some properties of **swA**, **swT** and **sw-VPA** models, that we use to define and solve a variant of parsing over infinite alphabets. We also briefly describe the application that motivated the introduction of these models: a parse-based approach to automated music transcription.

2012 ACM Subject Classification Theory of computation → Quantitative automata

Keywords and phrases Weighted Automata, Symbolic Automata, Visibly Pushdown, Parsing

Digital Object Identifier 10.4230/LIPIcs...

Funding Florent Jacquemard: Inria AEx Codex, ANR Collabscore, EU H2020 Polifonia

Acknowledgements I want to thank ...

1 Introduction

Parsing is the problem of structuring a linear representation on input (a finite word), according to a language model. Most of the context-free parsing approaches [15] assume a finite and reasonably small input alphabet. Such a restriction makes perfect sense in the context of NLP tasks such as constituency parsing, or of programming languages compilers or interpreters. Considering large or infinite alphabets can however be of practical interest, for instance, when dealing with large characters encodings such as UTF-16, *e.g.* for vulnerability detection in Web-applications [8], for the analysis (*e.g.* validation or filtering) of data streams or serialization of structured documents (with textual or numerical attributes) [26], or for processing timed execution traces [3].

The latter case is related to a study that motivated the present work: automated music transcription. Most representations of music are essentially linear. This is true for audio files, but also for widely used symbolic representations like MIDI. 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 music notation [14]: 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 [12], over an infinite alphabet of timed events.

Various extensions of language models for handling infinite alphabets have been studied.



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

register: skip refs
and details, add
Mikolaj recent

For instance, some automata with memory extensions allow restricted storage and comparison of input symbols, (see [26] for a survey), with pebbles for marking positions [25], registers [18], or the possibility to compute on subsequences with the same attribute values [2]. The automata at the core of model checkers compute on input symbols represented by large bitvectors [27] (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) [7, 8], the transitions are guarded by predicates over infinite alphabet domains. With appropriate closure conditions on the sets of such predicates, all the good properties enjoyed by automata over finite alphabets are preserved.

Other extensions of language models help in dealing with non-determinism, by the computation of weight values. With an ambiguous grammar, there may 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). This is roughly the principle of *weighted parsing* approaches [13, 24, 23]. In *weighted language models*, like *e.g.* probabilistic context-free grammars and weighted automata (wA) [11], a weight value is associated to each transition rule, and the rule's weights can be combined with an associative product operator \otimes into the weight of an AST. A second operator \oplus , associative and commutative, 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 will select the best of two weight values. The weight domain, equipped with these two operators shall be, 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 uniform 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 into functions into an arbitrary semiring, and generalizing also wA, by handling infinite alphabets, see Figure 1.

In short, a transition rule $q \xrightarrow{\phi} q'$ from state q to q' of a swM, is labeled by a function ϕ associating to every input symbol a a weight value $\phi(a)$ in a semiring domain. The framework relies on several language models: finite automata called symbolic-weighted (swA), transducers (swT), and pushdown automata with a visibly restriction [1] (sw-VPA). The latter model of automata operates on *nested words* [1], a structured form of words parenthesized

with markup symbols, corresponding to a linearization of trees. In the context of parsing, they can represent (weighted) AST of CF grammars. More precisely, a **sw-VPA** A associates a weight value $A(t)$ to a given nested word t , which is the linearization of an AST. On the other hand, a **swT** can define a distance $T(s, t)$ between finite words s and t over infinite alphabets. Then, the *SW-parsing* problem aims at finding t minimizing $T(s, t) \otimes A(t)$ (wrt the ranking defined by \oplus), given an input word s . The latter value is called the distance between s and A in [21].

Like weighted-parsing methods [13, 24, 23], our approach proceeds in two steps, based on properties of the **swM**. The first step is an intersection (Bar-Hillel construction [15]) where, given a **swT** T , a **sw-VPA** A , and an input word s , a **sw-VPA** $A_{T,s}$ is built, such that for all t , $A_{T,s}(t) = T(s, t) \otimes A(t)$. In the second step, a best AST t is found by applying to $A_{T,s}$ a best search algorithm similar to the shortest distance in graphs [20, 17].

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** (Running example). Throughout the paper we illustrate our framework with music transcription examples: Given a *timeline* of musical events with arbitrary timestamps as input, parse it into a structured music score. In our example, input events are pairs $\langle \eta, \tau \rangle$ made of a symbol $\eta \in \Sigma$, where Σ stands for the set of MIDI message symbols [?] and $\tau \in \mathbb{Q}$ is a timestamp. The output of parsing is a representation of the sequence in Common Western Music Notation (CWMN) [14] where event symbols belong to the domain Δ of *pitch*s (e.g., A4, G5, etc.), temporal information is encoded as *durations* (whole ♩ , quarter ♪ , eighth ♪ , etc), and notes are grouped in high-level structures (beams, measures, tuplets). The following inputs will be used:

1. $I_1 = [\langle e_1, 0.07 \rangle, \langle e_2, 0.72 \rangle, \langle e_3, 0.91 \rangle]$, over interval $[0, 1[$
2. $I_2 = [\langle e_3, 1.05 \rangle, \langle e_4, 1.36 \rangle, \langle e_5, 1.71 \rangle]$, over interval $[1, 2[$

There exists many possible parsings of $I_1 \cup I_2$ in music notation, among which $\text{♩} \text{♪} \text{♪} \text{♪}$ and $\text{♩} \text{♪} \text{♪} \text{♪}$. Weighted parsing associates a cost to each solution, and our framework aims at selecting the best one with respect to this cost. ◇

2 Preliminary Notions

Semirings

We shall consider semirings for the weight values of our language models. . A *semiring* $\langle \mathbb{S}, \oplus, \otimes, 0, 1 \rangle$ is a structure with a domain \mathbb{S} , equipped with two associative binary operators \oplus and \otimes , with respective neutral elements 0 and 1 , and such that:

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

Intuitively, in the models presented in this paper, \oplus selects an optimal value from two given values, in order to handle non-determinism, and \otimes combines two values into a single value, in a chaining of transitions.

chap. intersection
in [15]

expressiveness: VPA
have restricted
equality test. com-
parable to pebble
automata? → con-
clusion

The results are es-
tablished for a gen-
eral class of semir-
ings. They can be
instantiated for con-
crete cases

There is sometimes a
confusion in the text
between the struture
and the domain \mathbb{S} .
Not essential

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} [20] 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 [10]; 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 $x \oplus y = y$.

is total necessary?

► **Lemma 2** (Monotony, [20]). *Let $\langle \mathbb{S}, \oplus, 0, \otimes, 1 \rangle$ 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 $z \otimes x \leq_{\oplus} z \otimes y$.*

To express the property of Lemma 2, we call \mathbb{S} *monotonic wrt \leq_{\oplus}* . Another important semiring property in the context of optimization is superiority [16], which corresponds to the *non-negative weights* condition in shortest-path algorithms [9]. 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 $\langle \mathbb{S}, \oplus, 0, \otimes, 1 \rangle$ 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 $x \in \mathbb{S}$, $1 \oplus x = 1$.

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

In [16], when the property (i) holds, \mathbb{S} is called *superior wrt the ordering \leq_{\oplus}* . We have seen in the proof of Lemma 3 that it implies that $1 \leq_{\oplus} x$ for all $x \in \mathbb{S}$. Similarly, by the first inequality of (i) with $y = 0$, $x \leq_{\oplus} x \otimes 0 = 0$. Hence, in a superior semiring, it holds that for all $x \in \mathbb{S}$, $1 \leq_{\oplus} x \leq_{\oplus} 0$. Intuitively, from an optimization point of view, it means that 1 is the best value, and 0 the worst. In [20], \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 $n \geq 1$, $x \oplus x^n = x \otimes (1 \oplus x^{n-1}) = x$.

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

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

Here the difference between \mathbb{S} as a structure and as a domain is blurred.

$j \in \mathbb{N}$: j is an element of \mathbb{N} , not the same as $j \subset \mathbb{N}$

We shall need below infinite sums with \oplus . A semiring \mathbb{S} is called *complete* [11] 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 $I \subset \mathbb{N}$, such that:

i. *infinite sums extend finite sums:*

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

ii. *associativity and commutativity:*

$$\text{for all } I \subseteq \mathbb{N} \text{ and all partition } (I_j)_{j \in J} \text{ of } I, \quad \bigoplus_{j \in J} \bigoplus_{i \in I_j} x_i = \bigoplus_{i \in I} x_i,$$

iii. *distributivity of product over infinite sum:*

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

results of this paper: for semirings commutative, bounded, total and complete

	domain	\oplus	\otimes	\emptyset	$\mathbb{1}$
Boolean	$\{\perp, \top\}$	\vee	\wedge	\perp	\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.

Label Theory

We shall now define the functions labeling the transitions of SW automata and transducers, generalizing the Boolean algebras of [7] from Boolean to other semiring domains. We consider *alphabets*, which are countable sets of symbols denoted Σ, Δ, \dots . Given a semiring $\langle \mathbb{S}, \oplus, 0, \otimes, \mathbb{1} \rangle$, a *label theory* over \mathbb{S} is a set $\bar{\Phi}$ of recursively enumerable sets denoted Φ_Σ , containing unary functions of type $\Sigma \rightarrow \mathbb{S}$, 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 $\Phi_\Delta \in \bar{\Phi}$
- 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 .

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

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

In what follows, we might omit the sub- and superscripts in $\otimes_1, \bigoplus_\Sigma^1, \dots$, when there is no ambiguity. We shall keep them only for the special case $\Sigma = \Delta$, i.e. $\eta \in \Phi_{\Sigma, \Sigma}$, in order to be able to distinguish between the first and the second argument.

► **Definition 5.** A label theory $\bar{\Phi}$ is complete when the underlying semiring \mathbb{S} is complete, and for all $\Phi_{\Sigma, \Delta} \in \bar{\Phi}$ and all $\eta \in \Phi_{\Sigma, \Delta}$, $\bigoplus_\Sigma^1 \eta \in \Phi_\Delta$ and $\bigoplus_\Delta^2 \eta \in \Phi_\Sigma$.

The following facts are immediate.

► **Lemma 6.** For $\bar{\Phi}$ complete $\alpha \in \mathbb{S}$, $\phi, \phi' \in \Phi_\Sigma$, $\psi \in \Phi_\Delta$, and $\eta \in \Phi_{\Sigma, \Delta}$:

- 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 $(\bigoplus_\Sigma \phi) \otimes (\bigoplus_\Sigma \phi') = \bigoplus_\Sigma (\phi \otimes \phi')$

partial application is needed?

notion of diagram of functions akin BDD for transitions in practice

mv appendix?

- 195 *iv.* $(\bigoplus_{\Delta}^2 \eta) \oplus (\bigoplus_{\Delta}^2 \eta') = \bigoplus_{\Delta}^2 (\eta \oplus \eta')$, and $(\bigoplus_{\Delta}^2 \eta) \otimes (\bigoplus_{\Delta}^2 \eta') = \bigoplus_{\Delta}^2 (\eta \otimes \eta')$
 196 *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
 197 *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

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 .

► **Example 7.** Consider the music transcription problem, with an input representing a music performance. In order to align the input with a music score, we must take into consideration the expressive timing of human performance that results in small time shifts between an input event and the corresponding notation event. These shifts can be weighted as the time distance between both, computed in the tropical semiring with a base function based on a given $\delta \in \Phi_{\Sigma, \Delta}$.

$$\delta(\langle e_1, t1 \rangle, \langle e_2, t2 \rangle) = \begin{cases} |t_1 - t_2| & \text{if } e_1 = e_2 \\ 0 & \text{otherwise} \end{cases}$$

For the sake of concreteness, consider our example with $s = I_1 \text{ cup } I_2 = [\langle e_1, 0.07 \rangle, \langle e_2, 0.72 \rangle, \langle e_3, 0.91 \rangle] \cup [\langle e_3, 1.05 \rangle, \langle e_4, 1.36 \rangle, \langle e_5, 1.71 \rangle]$ and the music notation $t = \text{♩} \text{♩} \text{♩} \text{♩} \text{♩}$.

204 **3** SW Automata and Transducers

We follow the approach of [21] for the computation of distances, between words and languages, using weighted transducers, and extend it to infinite alphabets. The models introduced in this section generalize weighted automata and transducers [11] by labeling each transition with a weight function (instead of a simple weight value), that takes the input and output symbols as parameters. These functions are similar to the guards of symbolic automata [7, 8], but they can return values in a generic semiring, whereas 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 $\Phi_\Sigma, \Phi_\Delta, \Phi_{\Sigma, \Delta}$.

214 **Definition 8.** A symbolic-weighted transducer (*swT*) over Σ , Δ , \mathbb{S} and $\bar{\Phi}$ is a tuple
215 $T = \langle Q, \text{in}, \bar{\mathbf{w}}, \text{out} \rangle$, where Q is a finite set of states, $\text{in} : Q \rightarrow \mathbb{S}$ (respectively $\text{out} : Q \rightarrow \mathbb{S}$)
216 are functions defining the weight for entering (respectively leaving) computation in a state,
217 and $\bar{\mathbf{w}}$ is a triplet of transition functions $\mathbf{w}_{10} : Q \times Q \rightarrow \Phi_{\Sigma}$, $\mathbf{w}_{01} : Q \times Q \rightarrow \Phi_{\Delta}$, and
218 $\mathbf{w}_{11} : Q \times Q \rightarrow \Phi_{\Sigma, \Delta}$.

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

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

The **swT** T computes on pairs of words $\langle s, t \rangle \in \Sigma^* \times \Delta^*$, s and t , being respectively called *input* and *output* word. More precisely, T defines a mapping from $\Sigma^* \times \Delta^*$ into \mathbb{S} , based on

an intermediate function weight_T defined recursively, for every states $q, q' \in Q$, and every pairs of strings $\langle s, t \rangle \in \Sigma^* \times \Delta^*$, where au , and bv , denote the concatenation of the symbol $a \in \Sigma$ (resp. $b \in \Delta$) with a word $u \in \Sigma^*$ (resp. $v \in \Delta^*$).

added u and v def



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

We recall that, by convention (Section 2), an empty sum with \bigoplus is equal to $\mathbb{0}$. Intuitively, using a transition $w_{ij}(q, a, b, q')$ means for T : when reading respectively a and b at the current positions 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 $w_{ij}(q, a, b, q'')$ 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 [22]. 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, 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:

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

► **Example 9.** Let us develop the example of comparison between music played by a performer, represented as a sequence $s \in \Sigma^*$ of events in the MIDI alphabet Σ , and a music score represented as a sequence $t \in \Delta^*$ in the CWMN alphabet Δ . We build a small weighted transducer model with two states q_0 and q_1 that calculates the distance between s and t .

If one performed event s_i corresponds to one notated event t_1 (for instance MIDI code 61 and pitch A4), the weight value computed by the **swT** is the time distance between both, as in Example 7, and is modeled by transitions w_{11} below. If we meet the music notation symbol '-' that represents continuation (such as instance in *ties* , or *dots* , it is skipped with no cost (transitions w_{01} or weight $\mathbb{1}$).

$$\begin{aligned}
w_{11}(q_0, d, \langle e, d' \rangle, q_0) &= |d' - d| & w_{11}(q_1, d, \langle e, d' \rangle, q_0) &= |d' - d| \\
w_{01}(q_0, \varepsilon, \langle -, d' \rangle, q_0) &= \mathbb{1} & w_{01}(q_1, \varepsilon, \langle -, d' \rangle, q_0) &= \mathbb{1} \\
w_{10}(q_0, d, \varepsilon, q_1) &= \alpha
\end{aligned}$$

We also must be able to take performing errors into account, while still being able to compare with the score, since a performer could, for example, play an unwritten extra note. This is modelled by the transition w_{10} with an arbitrary weight value $\alpha \in \mathbb{S}$, switching from state q_0 (normal) to q_1 (error). 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 $\text{in}(q_1) = \text{out}(q_1) = \mathbb{0}$.

That way, an swT is capable of evaluating the differences between a score and a performance, all the while ensuring that performance errors are plausible. \diamond

The *Symbolic Weighted Automata* are defined similarly as the transducers of Definition 8, by simply omitting the output symbols.

► **Definition 10.** 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 .

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

$$\text{weight}_A(q, \varepsilon, q) = \mathbb{1} \quad (3)$$

$$\text{weight}_A(q, \varepsilon, q') = \mathbb{0} \quad \text{if } q \neq q'$$

$$\text{weight}_A(q, au, q') = \bigoplus_{q'' \in Q} w_1(q, a, q'') \otimes \text{weight}_A(q'', u, q')$$

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

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

The following property will be useful to the approach on symbolic weighted parsing presented in Section 5.

► **Proposition 11.** 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 an effectively constructible swA $B_{A,T}$ over Δ , \mathbb{S} and $\bar{\Phi}$, such that for all $t \in \Delta^*$, $B_{A,T}(t) = \bigoplus_{s \in \Sigma^*} A(s) \otimes T(s, t)$.

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 $w_1 : Q \times Q \rightarrow \Phi_\Sigma$. 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 Δ^* . Its state entering functions is defined for all $p \in P$, $q \in Q$ by $\text{in}'(p, q) = \text{in}_A(p) \otimes \text{in}_T(q)$. 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 an input word) and w_{11} (T reading both an input word and an output word). Moreover, w'_1 also needs to simulate transitions defined by w_{10} : T reading in input word and not an output word. Since $B_{A,T}$ will read only in the output word, such a transition corresponds 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 suppression of ε -transition in the swA in construction, following the algorithm of [19].

Comprends pas cette phrase

ccl to the ex

reformulated this sentence

Il me manque une explication: on construit un automate qui, étant donnée une partition t , renvoie la distance minimale avec n'importe quelle performance (distance donnée par un transducer)? Quel est le rôle de $A(s)$?

Initially, for all $p_1, p_2 \in P$, and $q_1, q_2 \in Q$, let

$$w'_1(\langle p_1, q_1 \rangle, \langle p_2, q_2 \rangle) = w_1(p_1, p_2) \otimes [w_{01}(q_1, q_2) \oplus \bigoplus_{\Sigma} w_{11}(q_1, q_2)].$$

Iterate the following for all $p_1 \in P$ and $q_1, q_2 \in Q$: for all $p_2 \in P$ and $q_3 \in Q$,

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

$$\text{and } \text{out}'(p_1, q_1) \oplus = \bigoplus_{\Sigma} w_{10}(q_1, q_2) \otimes \text{out}'(p_1, q_2)$$

◀

proof correctness

The construction time and size for $B_{A,T}$ are $O(\|T\|^3 \cdot \|A\|^2)$, where the sizes $\|T\|$ and $\|A\|$ are their number of states.

revise with nb of tr. and states

► **Corollary 12.** *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 in this section generalizes symbolic VPA (sVPA [6], generalizing themselves VPA [1] to infinite alphabets) from Boolean semirings to arbitrary semiring weight domains. It will compute on nested words over infinite alphabets, associating to every such word a weight value. Nested words are able to describe structures of labeled trees, and in the context of parsing, they will be useful to represent AST.

see §5 and App.A

Let Ω be a countable alphabet that we assume partitioned into three subsets $\Omega_i, \Omega_c, \Omega_r$, whose elements are respectively called *internal*, *call* and *return* symbols. Let $\langle \mathbb{S}, \oplus, 0, \otimes, 1 \rangle$ 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} (with $x \in \{i, c, r\}$) stand respectively for $\Phi_{\Omega_i}, \Phi_{\Omega_c}, \Phi_{\Omega_r}$ and $\Phi_{\Omega_c, \Omega_x}$.

► **Definition 13.** A Symbolic Weighted Visibly Pushdown Automata (sw-VPA) over $\Omega = \Omega_i \uplus \Omega_c \uplus \Omega_r$, \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 : $w_i : Q \times P \times Q \rightarrow \Phi_{ci}$, $w_i^e : Q \times Q \rightarrow \Phi_i$, $w_c : Q \times P \times Q \times P \rightarrow \Phi_{cc}$, $w_c^e : Q \times P \times Q \rightarrow \Phi_c$, $w_r : Q \times P \times Q \rightarrow \Phi_{cr}$, $w_r^e : Q \times Q \rightarrow \Phi_r$.

Similarly as in Section 3, we extend the above transition functions as follows for all $q, q' \in Q$, $p \in P$, $a \in \Omega_i$, $c \in \Omega_c$, $r \in \Omega_r$, overloading their names:

$w_i : Q \times [\Omega_c \times P] \times \Omega_i \times Q \rightarrow \mathbb{S}$	$w_i(q, c, p, a, q') = \eta_{ci}(c, a)$	where $\eta_{ci} = w_i(q, p, q')$,
$w_i^e : Q \times \Omega_i \times Q \rightarrow \mathbb{S}$	$w_i^e(q, a, q') = \phi_i(a)$	where $\phi_i = w_i^e(q, q')$.
$w_c : Q \times [\Omega_c \times P] \times [\Omega_c \times P] \times Q \rightarrow \mathbb{S}$	$w_c(q, c, p, c', p', q') = \eta_{cc}(c, c')$	where $\eta_{cc} = w_c(q, p, p', q')$,
$w_c^e : Q \times [\Omega_c \times P] \times Q \rightarrow \mathbb{S}$	$w_c^e(q, c, p, q') = \phi_c(c)$	where $\phi_c = w_c^e(q, p, q')$.
$w_r : Q \times [\Omega_c \times P] \times \Omega_r \times Q \rightarrow \mathbb{S}$	$w_r(q, c, p, r, q') = \eta_{cr}(c, r)$	where $\eta_{cr} = w_r(q, p, q')$,
$w_r^e : Q \times \Omega_r \times Q \rightarrow \mathbb{S}$	$w_r^e(q, r, q') = \phi_r(r)$	where $\phi_r = w_r^e(q, q')$.

The intuition is the following for the above transitions. w_i^e , w_c^e , and w_r^e describe the cases where the stack is empty. w_i and w_i^e 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 $c \in \Omega_c$ and

moved this to the beginning

XX:10 Symbolic Weighted Language Models and Parsing over Infinite Alphabets

334 $p \in P$ on the top of the stack (c is compared to a by the weight function $\eta_{ci} \in \Phi_{ci}$). w_c and
 335 w_c^e read the input call symbol c' , push it to the stack along with p' , and change state from q
 336 to to q' . Moreover, w_c reads c and p at the top of the stack (c is compared to c'). w_r and w_r^e
 337 read the input return symbol r , and change state from q to to q' . Moreover, w_r reads and
 338 pop from stack a pair made of c and p , (c is compared to r).

339 Formally, the transitions of the automaton A are defined in term of an intermediate
 340 function weight_A , like in Section 3. A configuration, denoted by $q[\gamma]$, is here composed of a
 341 state $q \in Q$ and a stack content $\gamma \in \Gamma^*$, where $\Gamma = \Omega_c \times P$. Hence, weight_A is a function
 342 from $[Q \times \Gamma^*] \times \Omega^* \times [Q \times \Gamma^*]$ into \mathbb{S} . The empty stack is denoted by \perp , and the upmost
 343 symbol is the last pushed content. The following functions illustrate each of the possible
 cases, being : reading $a \in \Omega_i$, or $c \in \Omega_c$, or $r \in \Omega_r$ for each possible state of the stack (empty
 or not), to add to $u \in \Omega^*$.

intro to func

introduced the 6
cases

notation cp for
 $\langle c, p \rangle$?

$$\text{weight}_A(q[\perp], \varepsilon, q'[\perp]) = \mathbb{1} \text{ if } q = q' \text{ and } \emptyset \text{ otherwise} \quad (5)$$

$$\text{weight}_A\left(q \left[\begin{array}{c} \langle c, p \rangle \\ \gamma \end{array} \right], a u, q'[\gamma']\right) = \bigoplus_{q'' \in Q} w_i(q, c, p, a, q'') \otimes \text{weight}_A\left(q'' \left[\begin{array}{c} \langle c, p \rangle \\ \gamma \end{array} \right], u, q'[\gamma']\right)$$

$$\text{weight}_A(q[\perp], a u, q'[\gamma']) = \bigoplus_{q'' \in Q} w_i^e(q, a, q'') \otimes \text{weight}_A(q''[\perp], u, q'[\gamma'])$$

$$\text{weight}_A\left(q \left[\begin{array}{c} \langle c, p \rangle \\ \gamma \end{array} \right], c' u, q'[\gamma']\right) = \bigoplus_{\substack{q'' \in Q \\ p' \in P}} w_c(q, c, p, c', p', q'') \otimes \text{weight}_A\left(q'' \left[\begin{array}{c} \langle c', p' \rangle \\ \langle c, p \rangle \\ \gamma \end{array} \right], u, q'[\gamma']\right)$$

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

$$\text{weight}_A\left(q \left[\begin{array}{c} \langle c, p \rangle \\ \gamma \end{array} \right], r u, q'[\gamma']\right) = \bigoplus_{q'' \in Q} w_r(q, c, p, r, q'') \otimes \text{weight}_A(q''[\gamma], u, q'[\gamma'])$$

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

$c p$ to $\langle c, p \rangle$

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

$$A(s) = \bigoplus_{q, q' \in Q} \text{in}(q) \otimes \text{weight}_A(q[\perp], s, q'[\perp]) \otimes \text{out}(q'). \quad (6)$$

todo example VPA

► **Example 14.** structured words with timed symbols... intro language of music notation?
 (markup = time division, leaves = events etc)

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

Similarly to VPA [1] and sVPA [6], 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 Ω, \mathbb{S} and $\bar{\Phi}$. There
 exists two effectively constructible sw-VPA $A_1 \oplus A_2$ and $A_1 \otimes A_2$, such that for all $s \in \Omega^*$,
 $(A_1 \oplus A_2)(s) = A_1(s) \oplus A_2(s)$ and $(A_1 \otimes A_2)(s) = A_1(s) \otimes A_2(s)$.

367 **Proof.** The construction is essentially the same as in the case of the Boolean semiring [6].

368

complete proof

369 We shall now present a procedure for searching, for a sw-VPA A , a word of minimal weight
370 for A , as stated in the following proposition.

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

374 Let $A = \langle Q, P, \text{in}, \bar{w}, \text{out} \rangle$. We propose a Dijkstra algorithm computing, for every $q, q' \in Q$,
375 the minimum, wrt \leq_{\oplus} , of the function $\beta_{q,q'} : t \mapsto \text{weight}_A(q[\perp], t, q'[\perp])$. Let us denote by
376 $b_{\perp}(q, q')$ this minimum. By definition of \leq_{\oplus} it holds that:

total?

$$377 \quad b_{\perp}(q, q') = \bigoplus_{t \in \Omega^*} \text{weight}_A(q[\perp], t, q'[\perp]). \quad (7)$$

378 Hence, following (6), and the associativity and commutativity and distributivity for \otimes and \oplus ,
379 the minimum of $A(t)$ is $\bigoplus_{t \in \Omega^*} \bigoplus_{q, q' \in Q} \text{in}(q) \otimes \beta_{q,q'}(t) \otimes \text{out}(q') = \bigoplus_{q, q' \in Q} \text{in}(q) \otimes b_{\perp}(q, q') \otimes \text{out}(q')$.

380 In order to compute the above function $b_{\perp} : Q \times Q \rightarrow \mathbb{S}$, we shall use the auxiliary function
381 $b_{\top} : Q \times P \times Q \rightarrow \Phi_c$. Intuitively, the function defined in (9) associates to $c \in \Omega_c$ the
382 minimum weight of a computation of A starting in state q with a stack $\langle c, p \rangle \cdot \gamma \in \Gamma^+$ and
383 ending in state q' with the same stack, such that the computation can not pop the pair made
384 of c and p at the top of this stack, but may only read these symbols. Moreover, A may push
385 another pair $\langle c', p' \rangle$ on the top of $\langle c, p \rangle \cdot \gamma$, following the third case of in the definition (5) of
386 weight_A , and may pop $\langle c', p' \rangle$ later, following the fifth case of (5) (return symbol).

387 over Ω , \mathbb{S} and $\bar{\Phi}$, the minimal weight for a word in Ω^* .

388 We distinguish two cases : when the stack is empty, and when it is not. In the case of an
389 empty stack, let $b_{\perp} : Q \times Q \rightarrow \mathbb{S}$ be such that :

introduced 2 cases
for b

$$390 \quad b_{\perp}(q, q') = \bigoplus_{s \in \Omega^*} \text{weight}_A(q[\perp], s, q'[\perp]). \quad (8)$$

391 Since \mathbb{S} is complete, the infinite sum in (8) is well defined, and, providing that \mathbb{S} is total,
392 it is the minimum in Ω^* , wrt \leq_{\oplus} , of the function $s \mapsto \text{weight}_A(q[\sigma], s, q'[\sigma])$. The term
393 $q[\perp], s, q'[\perp]$ of this sum is the central expression in the definition (??) of $A(s_0)$, for the
394 minimum s_0 of the function weight_A .

so ?

395 If the stack is not empty, let \top be a fresh stack symbol which does not belong to Γ , and let
396 $b_{\top} : Q \times P \times Q \rightarrow \Phi_c$ be such that, for every two states $q, q' \in Q$ and stack symbol $p \in P$:

b_{\top} : mot bien par-
enthesé c/r

$$397 \quad b_{\top}(q, p, q') : c \mapsto \bigoplus_{s \in \Omega^*} \text{weight}_A\left(q \left[\begin{array}{c} \langle c, p \rangle \\ \top \end{array} \right], s, q' \left[\begin{array}{c} \langle c, p \rangle \\ \top \end{array} \right] \right) \quad (9)$$

398 Algorithm 1 constructs iteratively markings $d_{\perp} : Q \times Q \rightarrow \mathbb{S}$ and $d_{\top} : Q \times P \times Q \rightarrow \Phi_c$
399 that converges eventually to b_{\top} and b_{\perp} .

400 The infinite sums in the updates of d in Algorithm 1, Figure 3 are well defined since \mathbb{S}
401 is complete. ** effectively computable by hypothesis that the label theory is effective**
402 The algorithm performs $2 \cdot |Q|^2$ iterations until P is empty, and each iteration has a time
403 complexity $O(|Q|^2 \cdot |P|)$. That gives a time complexity $O(|Q|^4 \cdot |P|)$. It can be reduced by
404 implementing P as a priority queue, prioritized by the value returned by d .

explication Fig. 3
suivant cas de (5)

complete **

detail with nb tr.
and states

405 The correctness of Algorithm 1 is ensured by the invariant expressed in the following
406 lemma.

■ **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.
 $\bigoplus_{c \in \Omega_c} d_{\top}(q_1, p, q_2)(c)$, 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 $q_0, q_3 \in Q$,

$$\begin{aligned}
 d_{\top}(q_1, p, q_3) &\oplus= d_{\top}(q_1, p, q_2) \otimes \bigoplus_{\Omega_i} w_i(q_2, p, q_3) \\
 d_{\perp}(q_1, p, q_3) &\oplus= d_{\perp}(q_1, q_2) \otimes \bigoplus_{\Omega_i} w_i^e(q_2, q_3) \\
 d_{\top}(q_0, p, q_3) &\oplus= \bigoplus_{\Omega_c}^2 [(w_c(q_0, p, p', q_1) \otimes_2 d_{\top}(q_1, p', q_2)) \otimes_2 \bigoplus_{\Omega_r} w_r(q_2, p', q_3)] \\
 d_{\perp}(q_0, q_3) &\oplus= \bigoplus_{\Omega_c} (w_c^e(q_0, p, q_1) \otimes d_{\top}(q_1, p, q_2) \otimes \bigoplus_{\Omega_r} w_r(q_2, p, q_3)) \\
 d_{\perp}(q_1, q_3) &\oplus= d_{\perp}(q_1, q_2) \otimes \bigoplus_{\Omega_r} w_r^e(q_2, q_3) \\
 d_{\top}(q_1, p, q_3) &\oplus= d_{\top}(q_1, p, q_2) \otimes d_{\top}(q_2, p, q_3), \text{ if } \langle q_2, \top, q_3 \rangle \notin P \\
 d_{\perp}(q_1, q_3) &\oplus= d_{\perp}(q_1, q_2) \otimes d_{\perp}(q_2, q_3), \text{ if } \langle q_2, \perp, q_3 \rangle \notin P
 \end{aligned}$$

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

407 ► **Lemma 17.** For all $\langle q_1, q_2 \rangle \notin \mathcal{Q}$, $d_{\perp}(q_1, q_2) = b_{\perp}(q_1, q_2)/$

408 The proof is by contradiction, assuming a counter-example minimal in the length of the
 409 witness word.

410 ► **Lemma 18.** For all $\langle q_1, p, q_2 \rangle \notin \mathcal{Q}$, $d_{\top}(q_1, p, q_2) = b_{\top}(q_1, p, q_2)$,

411 For computing the minimal weight of a computation of A , we use the fact that, at the
 412 termination of Algorithm 1, $\bigoplus_{s \in \Omega^*} A(s) = \bigoplus_{q, q' \in Q} \text{in}(q) \otimes d_{\perp}(q, q') \otimes \text{out}(q')$.

413 In order to obtain effectively a witness (word of Ω^* with a computation of A of minimal
 414 weight), we require the additional property of convexity of weight functions.

415 ► **Proposition 19.** For a sw-VPA A over Ω , \mathbb{S} commutative, bounded, total and complete,
 416 and $\bar{\Phi}$ effective, one can construct in PTIME a word $t \in \Omega^*$ such that $A(t)$ is minimal wrt
 417 the natural ordering for \mathbb{S} .

418 5 Symbolic Weighted Parsing

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

total?

- 424 – a **swT** T over Σ , Ω_i , \mathbb{S} , and $\bar{\Phi}$, defining a measure $T : \Sigma^* \times \Omega_i^* \rightarrow \mathbb{S}$,
 - 425 – a **sw-VPA** A over Ω , \mathbb{S} , and $\bar{\Phi}$, defining a measure $A : \Omega^* \rightarrow \mathbb{S}$,
 - 426 – an input word $s \in \Sigma^*$.
- 427 For all $u \in \Sigma^*$ and $t \in \Omega^*$, let $d(u, t) = T(u, t|_{\Omega_i})$, where $t|_{\Omega_i} \in \Omega_i^*$ is the projection of t
- 428 onto Ω_i , obtained from t by removing all symbols in $\Omega \setminus \Omega_i$. *Symbolic weighted parsing* is the
- 429 problem, given the above input, to find $t \in \Omega^*$ minimizing $d(s, t) \otimes A(t)$ wrt \leq_{\oplus} , i.e. s.t.

$$430 \quad d(s, t) \otimes A(t) = \bigoplus_{t' \in \mathcal{T}(\Omega)} d(s, t') \otimes A(t') \quad (10)$$

431 Following the terminology of [21], **sw-parsing** is the problem of computing the distance (10)

432 between the input s and the output weighted language of A , and returning a witness t .

433 ► **Proposition 20.** *The problem of Symbolic Weighted parsing can be solved in PTIME in*

434 *the size of the input **swT** T , **sw-VPA** A and input word s , and the computation time of the*

435 *functions and operators of the label theory.*

436 **Proof.** (sketch) We follow a *Bar-Hillel* construction, for parsing by intersection. Let us first

437 extend the **swT** T over Σ , Ω_i into a **swT** T' over Σ and Ω (and the same semiring and label

438 theory \mathbb{S} and $\bar{\Phi}$), such that for all $u \in \Sigma^*$, and $t \in \Omega^*$, $T'(u, u) = T(u, t|_{\Omega_i})$. The transducer

439 T' simply skips every symbol $b \in \Omega \setminus \Omega_i$, by the addition to T , of new transitions of the

440 form $w_{01}(q, \varepsilon, b, q')$. Then, using Corolary 12, we construct from the input word $s \in \Sigma^*$ and

441 T' a **swA** $B_{s, T'}$, such that for all $t \in \Omega^*$, $B_{s, T'}(t) = d(s, t)$. Next, we compute the **sw-VPA**

442 $B_{s, T'} \otimes A$, using Proposition 15. It remains to compute a best nested-word $t \in \Omega^*$ using the

443 best-search procedure of Proposition 19. ◀

444 The **sw-parsing** generalizes the problem of searching the best derivation (AST) of a weighted

445 CF-grammar G that yields a given input word w . The latter problem, sometimes called *weighted*

446 *parsing*, (see e.g. [13] and [23] for general weighted parsing frameworks) corresponds to **sw-**

447 **parsing** in the case of finite alphabets, a transducer T computing the identity and some

448 **sw-VPA** A obtained from G . Indeed, the *depth-first* traversal of an AST τ yields a well-

449 parenthesised word $\text{lin}(\tau)$ over an alphabet $\Omega = \Omega_i \uplus \Omega_c \uplus \Omega_r$, assuming e.g. that Ω_i contains

450 the symbols labelling the leaves of τ (symbols of rank 0), and Ω_c and Ω_r contain respectively

451 one left and right parenthesis \langle_b and \rangle_b for each symbol b labelling inner nodes of τ (symbols

452 of rank > 0). We show in Appendix A how to construct a **sw-VPA** A such that $A(\text{lin}(\tau))$ is

453 the weight the AST τ of G .

454

455 Conclusion

456 We have introduced weighted language models (SW transducers and visibly pushdown

457 automata) computing over infinite alphabets, and applied them to the problem of parsing

458 with infinitely many possible input symbols (typically timed events). This approach extends

459 conventional parsing and weighted parsing by computing a derivation tree modulo a generic

460 distance between words, defined by a SW transducer given in input. This enables to consider

461 finer word relationships than strict equality, opening possibilities of quantitative analysis via

462 this method.

463 Ongoing and future work include

- 464 – The study of other theoretical properties of SW models, such as the extension of the best
- 465 search algorithm from 1-best to n -best [17], and to k -closed semirings [20] (instead of *bounded*,
- 466 which corresponds to 0-closed).

2 lines Application
to Automated Mu-
sic Transcription:
implementation \neq
but same principle,
on-the-fly automata
construction during
best search, for effi-
ciency.

TODO future work

467 – ...there is room to improve the complexity bounds for the algorithms ... modular approach
 468 with oracles ...
 469 – present here an offline algorithm for best search, semi-online implementation for AMT
 470 (bar-by-bar approach) with an on-the-fly automata construction.

471 — References —

- 472 1 Rajeev Alur and Parthasarathy Madhusudan. Adding nesting structure to words. *Journal of*
 473 *the ACM (JACM)*, 56(3):1–43, 2009.
- 474 2 Mikołaj Bojańczyk, Claire David, Anca Muscholl, Thomas Schwentick, and Luc Segoufin.
 475 Two-variable logic on data words. *ACM Transactions on Computational Logic (TOCL)*,
 476 12(4):1–26, 2011.
- 477 3 Patricia Bouyer, Antoine Petit, and Denis Thérien. An algebraic approach to data languages
 478 and timed languages. *Information and Computation*, 182(2):137–162, 2003.
- 479 4 Mathieu Caralp, Pierre-Alain Reynier, and Jean-Marc Talbot. Visibly pushdown automata
 480 with multiplicities: finiteness and k-boundedness. In *International Conference on Developments*
 481 *in Language Theory*, pages 226–238. Springer, 2012.
- 482 5 Hubert Comon, Max Dauchet, Rémi Gilleron, Florent Jacquemard, Christoph Löding, Denis
 483 Lugiez, Sophie Tison, and Marc Tommasi. *Tree Automata Techniques and Applications*.
 484 <http://tata.gforge.inria.fr>, 2007.
- 485 6 Loris D’Antoni and Rajeev Alur. Symbolic visibly pushdown automata. In *International*
 486 *Conference on Computer Aided Verification*, pages 209–225. Springer, 2014.
- 487 7 Loris D’Antoni and Margus Veanes. The power of symbolic automata and transducers. In
 488 *International Conference on Computer Aided Verification*, pages 47–67. Springer, 2017.
- 489 8 Loris D’Antoni and Margus Veanes. Automata modulo theories. *Communications of*
 490 *the ACM*, 64(5):86–95, 2021. URL: [seealsohttps://pages.cs.wisc.edu/~loris/](https://pages.cs.wisc.edu/~loris/symbolicautomata.html)
 491 [symbolicautomata.html](https://pages.cs.wisc.edu/~loris/symbolicautomata.html).
- 492 9 E. W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*,
 493 1(1):269–271, 1959.
- 494 10 Manfred Droste and Werner Kuich. Semirings and formal power series. In *Handbook of*
 495 *Weighted Automata*, pages 3–28. Springer, 2009.
- 496 11 Manfred Droste, Werner Kuich, and Heiko Vogler. *Handbook of weighted automata*. Springer
 497 Science & Business Media, 2009.
- 498 12 Francesco Foscari, Florent Jacquemard, Philippe Rigaux, and Masahiko Sakai. A Parse-
 499 based Framework for Coupled Rhythm Quantization and Score Structuring. In *Mathematics*
 500 *and Computation in Music (MCM)*, volume 11502 of *Lecture Notes in Artificial Intelligence*,
 501 Madrid, Spain, 2019. Springer. URL: <https://hal.inria.fr/hal-01988990>, doi:10.1007/
 502 978-3-030-21392-3_20.
- 503 13 Joshua Goodman. Semiring parsing. *Computational Linguistics*, 25(4):573–606, 1999.
- 504 14 Elaine Gould. *Behind Bars: The Definitive Guide to Music Notation*. Faber Music, 2011.
- 505 15 Dick Grune and Criel J.H. Jacobs. *Parsing Techniques*. Number 2nd edition in Monographs
 506 in Computer Science. Springer, 2008.
- 507 16 Liang Huang. Advanced dynamic programming in semiring and hypergraph frameworks. In
 508 *In COLING*, 2008.
- 509 17 Liang Huang and David Chiang. Better k-best parsing. In *Proceedings of the Ninth International*
 510 *Workshop on Parsing Technology*, Parsing ’05, pages 53–64, Stroudsburg, PA, USA, 2005.
 511 Association for Computational Linguistics. URL: [http://dl.acm.org/citation.cfm?id=](http://dl.acm.org/citation.cfm?id=1654494.1654500)
 512 [1654494.1654500](http://dl.acm.org/citation.cfm?id=1654494.1654500).
- 513 18 Michael Kaminski and Nissim Francez. Finite-memory automata. *Theor. Comput. Sci.*,
 514 134:329–363, November 1994. URL: [http://dx.doi.org/10.1016/0304-3975\(94\)90242-9](http://dx.doi.org/10.1016/0304-3975(94)90242-9),
 515 doi:[http://dx.doi.org/10.1016/0304-3975\(94\)90242-9](http://dx.doi.org/10.1016/0304-3975(94)90242-9).

- 516 19 Sylvain Lombardy and Jacques Sakarovitch. The removal of weighted ε -transitions. In
 517 *International Conference on Implementation and Application of Automata*, pages 345–352.
 518 Springer, 2012.
- 519 20 Mehryar Mohri. Semiring frameworks and algorithms for shortest-distance problems. *Journal*
 520 *of Automata, Languages and Combinatorics*, 7(3):321–350, 2002.
- 521 21 Mehryar Mohri. Edit-distance of weighted automata: General definitions and al-
 522 gorithms. *International Journal of Foundations of Computer Science*, 14(06):957–982,
 523 2003. URL: <https://www.worldscientific.com/doi/abs/10.1142/S0129054103002114>,
 524 arXiv:<https://www.worldscientific.com/doi/pdf/10.1142/S0129054103002114>, doi:10.
 525 1142/S0129054103002114.
- 526 22 Mehryar Mohri. Edit-distance of weighted automata: General definitions and algorithms.
 527 *International Journal of Foundations of Computer Science*, 14(06):957–982, 2003.
- 528 23 Richard Mörbitz and Heiko Vogler. Weighted parsing for grammar-based language models.
 529 In *Proceedings of the 14th International Conference on Finite-State Methods and Natural*
 530 *Language Processing*, pages 46–55, Dresden, Germany, September 2019. Association for
 531 Computational Linguistics. URL: <https://www.aclweb.org/anthology/W19-3108>, doi:10.
 532 18653/v1/W19-3108.
- 533 24 Mark-Jan Nederhof. Weighted deductive parsing and Knuth’s algorithm. *Computational*
 534 *Linguistics*, 29(1):135–143, 2003. URL: <https://doi.org/10.1162/089120103321337467>.
- 535 25 Frank Neven, Thomas Schwentick, and Victor Vianu. Finite state machines for strings
 536 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.
- 537 26 Luc Segoufin. Automata and logics for words and trees over an infinite alphabet. In *Computer*
 538 *Science Logic*, volume 4207 of *LNCS*. Springer, 2006.
- 539 27 Moshe Y Vardi. Linear-time model checking: automata theory in practice. In *International*
 540 *Conference on Implementation and Application of Automata*, pages 5–10. Springer, 2007.
- 541

542 A Nested-Words and Parse-Trees

543 The hierarchical structure of nested-words, defined with the *call* and *return* markup symbols
544 suggest a correspondence with trees. The lifting of this correspondence to languages, of tree
545 automata and VPA, has been discussed in [1], and [4] for the weighted case. In this section,
546 we describe a correspondence between the symbolic-weighted extensions of tree automata
547 and VPA.

548 Let Ω be a countable ranked alphabet, such that every symbol $a \in \Omega$ has a rank
549 $\text{rk}(a) \in [0..M]$ where M is a fixed natural number. We denote by Ω_k the subset of all symbols
550 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,
551 ordered, Ω -labeled trees is denoted by $\mathcal{T}(\Omega)$. It is the smallest set such that $\Omega_0 \subset \mathcal{T}(\Omega)$
552 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
553 assume a commutative semiring \mathbb{S} and a label theory $\bar{\Phi}$ over \mathbb{S} containing one set Φ_{Ω_k} for
554 each $k \in [0..M]$.

555 ► **Definition 21.** A symbolic-weighted tree automaton (*swTA*) over Ω , \mathbb{S} , and $\bar{\Phi}$ is a triplet
556 $A = \langle Q, \text{in}, \bar{w} \rangle$ where Q is a finite set of states, $\text{in} : Q \rightarrow \Phi_{\Omega}$ is the starting weight function,
557 and \bar{w} is a tuple of transition functions containing, for each $k \in [0..M]$, the functions
558 $w_k : Q \times Q^k \rightarrow \Phi_{\Omega_{>0}, \Omega_k}$ and $w_k^e : Q \times Q^k \rightarrow \Phi_{\Omega_k}$.

559 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:

$$\begin{aligned} 560 \quad w(q_0, a, b, q_1 \dots q_k) &= \eta(a, b) && \text{where } \eta = w_k(q_0, q_1 \dots q_k) \\ w(q_0, \varepsilon, b, q_1 \dots q_k) &= \phi(b) && \text{where } \phi = w_k^e(q_0, q_1 \dots q_k). \end{aligned}$$

561 where $q_1 \dots q_k$ is ε if $k = 0$. The first case deals with a strict subtree, with a parent node
562 labeled by a , and the second case is for a root tree.

563 Every swTA defines a mapping from trees of $\mathcal{T}(\Omega)$ into \mathbb{S} , based on the following intermediate
564 function $\text{weight}_A : Q \times (\Omega \cup \{\varepsilon\}) \times \mathcal{T}(\Omega) \rightarrow \mathbb{S}$

$$565 \quad \text{weight}_A(q_0, a, t) = \bigoplus_{q_1 \dots q_k \in Q^k} w(q_0, a, b, q_1 \dots q_k) \otimes \bigotimes_{i=1}^k \text{weight}_A(q_i, b, t_i) \quad (11)$$

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

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

$$568 \quad A(t) = \bigoplus_{q \in Q} \text{in}(q) \otimes \text{weight}_A(q, \varepsilon, t) \quad (12)$$

569 Intuitively, $w(q_0, a, b, q_1 \dots q_k)$ can be seen as the weight of a production rule $q_0 \rightarrow b(q_1, \dots, q_k)$
570 of a regular tree grammar [5], that replaces the non-terminal symbol q_0 by $b(q_1, \dots, q_k)$,
571 provided that the parent of q_0 is labeled by a (or q_0 is the root node if $a = \varepsilon$). The
572 above production rule can also be seen as a rule of a weighted CF grammar, of the form
573 $[a, b] q_0 := q_1 \dots q_k$ if $k > 0$, and $[a] q_0 := b$ if $k = 0$. In the first case, b is a label of the rule,
574 and in the second case, it is a terminal symbol. And in both cases, a is a constraint on the
575 label of rule applied on the parent node in the derivation tree. This features of observing
576 the parent's label are useful in the case of infinite alphabet, where it is not possible to
577 memorize a label with the states. The weight of a labeled derivation tree t of the weighted
578 CF grammar associated to A as above, is $\text{weight}_A(q, t)$, when q is the start non-terminal. We
579 shall now establish a correspondence between such derivation tree t and some word describing
580 a linearization of t , in a way that $\text{weight}_A(q, t)$ can be computed by a sw-VPA.

581 Let $\hat{\Omega}$ be the countable (unranked) alphabet obtained from Ω by: $\hat{\Omega} = \Omega_i \uplus \Omega_c \uplus \Omega_r$, with
 582 $\Omega_i = \Omega_0$, $\Omega_c = \{ \langle a \mid a \in \Omega_{>0} \rangle \}$, $\Omega_r = \{ a \mid a \in \Omega_{>0} \}$.

583 We associate to $\hat{\Omega}$ a label theory $\hat{\Phi}$ like in Section 4, and we define a linearization of trees of
 584 $\mathcal{T}(\Omega)$ into words of $\hat{\Omega}^*$ as follows:

585 $\text{lin}(a) = a$ for all $a \in \Omega_0$,

586 $\text{lin}(b(t_1, \dots, t_k)) = \langle_b \text{lin}(t_1) \dots \text{lin}(t_k)_b \rangle$ when $b \in \Omega_k$ for $1 \leq k \leq M$.

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

589 **Proof.** Let $A = \langle Q, \text{in}, \bar{w} \rangle$ where \bar{w} is presented as above by a function We build $A' =$
 590 $\langle Q', P', \text{in}', \bar{w}', \text{out}' \rangle$, where $Q' = \bigcup_{k=0}^M Q^k$ is the set of sequences of state symbols of A , of
 591 length at most M , including the empty sequence denoted by ε , and where $P' = Q'$ and \bar{w} is
 592 defined by:

$$\begin{array}{lll}
 \mathbf{w}_i(q_0 \bar{u}, \langle_c \bar{p}, a, \bar{u} \rangle) & = & \mathbf{w}(q_0, c, a, \varepsilon) \quad \text{for all } c \in \Omega_{>0}, a \in \Omega_0 \\
 \mathbf{w}_i^e(q_0 \bar{u}, a, \bar{u}) & = & \mathbf{w}(q_0, \varepsilon, a, \varepsilon) \quad \text{for all } a \in \Omega_0 \\
 \mathbf{w}_c(q_0 \bar{u}, \langle_c \bar{p}, \langle_d \bar{u}, \bar{q} \rangle) & = & \mathbf{w}(q_0, c, d, \bar{q}) \quad \text{for all } c, d \in \Omega_{>0} \\
 593 \quad \mathbf{w}_c^e(q_0 \bar{u}, \langle_c \bar{u}, \bar{q} \rangle) & = & \mathbf{w}(q_0, \varepsilon, c, \bar{q}) \quad \text{for all } c \in \Omega_{>0} \\
 \mathbf{w}_r(\varepsilon, \langle_c \bar{p}, c \rangle, \bar{p}) & = & \mathbb{1} \quad \text{for all } c \in \Omega_{>0} \\
 \mathbf{w}_r^e(\bar{u}, c, \bar{q}) & = & \mathbb{0} \quad \text{for all } c \in \Omega_{>0}
 \end{array}$$

594 All cases not matched by one of the above equations have a weight $\mathbb{0}$, for instance $\mathbf{w}_r(\bar{u}, \langle_c \bar{p}, d \rangle, \bar{q}) =$
 595 $\mathbb{0}$ if $c \neq d$ or $\bar{u} \neq \varepsilon$ or $\bar{q} \neq \bar{p}$. ◀

596 **Todo list**

597	register: skip refs and details, add Mikolaj recent	2
598	chap. intersection in [15]	3
599	expressiveness: VPA have restricted equality test. comparable to pebble automata?	
600	→ conclusion	3
601	The results are established for a general class of semirings. They can be instantiated	
602	for concrete cases	3
603	There is sometimes a confusion in the text between the struture and the domain \mathbb{S} .	
604	Not essential	3
605	is total necessary?	4
606	Here the difference between \mathbb{S} as a structure and as a domain is blurred.	4
607	$j \in \mathbb{N}$: j is en element of \mathbb{N} , not the same s $j \subset \mathbb{N}$	4
608	results of this paper: for semirings commutative, bounded, total and complete . . .	4
609	partial application is needed?	5
610	notion of diagram of functions akin BDD for transitions in practice	5
611	mv appendix?	5
612	Je trouve qu'il y a beaucoup de notions à retenir (complete, effective) et ça devient	
613	difficile pour un lecteur non spécialiste. Est-ce que tout est nécessaire (je ne sais	
614	plus qui m'avait dit: un concept en plus, un point en moins.	6
615	\exists oracle returning ... in worst time complexity T	6
616	added u and v def	7
617	reformulated this sentence	7
618	Comprends pas cette phrase	8
619	ccl to the ex	8
620	Il me manque une explication: on construit un automate qui, étant donnée une	
621	partition t , renvoie la distance minimale avec n'importe quelle performance	
622	(distance donnée par un transducer)? Quel est le rôle de $A(s)$?	8
623	proof correctness	9
624	revise with nb of tr. and states	9
625	see §5 and App.A	9
626	moved this to the beginning	9
627	intro to func	10
628	introduced the 6 cases	10
629	notation cp for $\langle c, p \rangle$?	10
630	$c p$ to $\langle c, p \rangle$	10
631	todo example VPA	10
632	complete proof	10
633	total?	11
634	introduced 2 cases for b	11
635	so ?	11
636	b_{\top} : mot bien parenthésé c/r	11
637	explication Fig. 3 suivant cas de (5)	12
638	complete **	12
639	detail with nb tr. and states	12
640	total?	12
641	2 lines Application to Automated Music Transcription: implementation \neq but same	
642	principle, on-the-fly automata construction during best search, for efficiency. . . .	13
643	TODO future work	13