

Symbolic Weighted Language Models and Parsing over Infinite Alphabets

Florent Jacquemard @ H 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 analyse (*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. In this problem, a music performance, represented symbolically in the form of a sequence of timed musical events, is converted into a score in Common Western Music Notation [14], structured according to nested grouping and metric strength of events. 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. 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

I think that we can make the case for a larger set of situations: given a linear music notation input, for instance elements in an XML file, we want to structure the input according to a hierarchical rhythmic space. I can elaborate...

register: skip refs and details, add Mikolaj recent



■ **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, $q \xrightarrow{a} q'$ is a transition between states q and q' .

44 formula or Binary Decision Diagrams. Following a similar idea, in symbolic automata
 45 (sA) [7, 8], the transitions are guarded by predicates over infinite alphabet domains. With
 46 appropriate closure conditions on the sets of such predicates, all the good properties enjoyed
 47 by automata over finite alphabets are preserved.

48 Other extensions of language models help in dealing with non-determinism, by the
 49 computation of weight values. With an ambiguous grammar, there may exist several
 50 derivations (*abstract syntax trees* – AST) yielding one input word. The association of one
 51 weight value to each AST permits to select a best one (or n bests). This is roughly the
 52 principle of *weighted parsing* approaches [13, 24, 23]. In *weighted language models*, like *e.g.*
 53 probabilistic context-free grammars and weighted automata (wA) [11], a weight value is
 54 associated to each transition rule, and the rule’s weights can be combined with a associative
 55 product operator \otimes into the weight of an AST. A second operator \oplus , associative and
 56 commutative, is moreover used to handle the ambiguity of the model, by summing the
 57 weights of the possibly several (in general exponentially many) AST associated to a given
 58 input word. Typically, \oplus will select the best of two weight values. The weight domain,
 59 equipped with these two operators shall be, at minima, a *semiring* where \oplus can be extended
 60 to infinite sums, such as the Viterbi semiring and the tropical min-plus algebra, see Figure 2.

61 In this paper, we present a uniform framework for weighted parsing over infinite input
 62 alphabets. It is based on *symbolic weighted* finite states language models (swM), gener-
 63 alizing sA, with functions into an arbitrary semiring instead of Boolean guards, and wA,
 64 by handling infinite alphabets, see Figure 1. In the transition rules of swM models, input
 65 symbols appear as variables, and the weight associated to a transition rule is a function of
 66 these variables. The models presented here are finite automata called symbolic-weighted
 67 (swA), transducers (swT), and pushdown automata with a visibly restriction [1] (sw-VPA).
 68 The latter model of automata operates on *nested words* [1], a structured form of words
 69 parenthesized with markup symbols, corresponding to a linearization of trees. In the context
 70 of parsing, they can represent (weighted) AST of CF grammars. More precisely, a sw-VPA A
 71 associates a weight value $A(t)$ to a given nested word t , which is the linearization of an AST.
 72 On the other hand, a swT can define a distance $T(s, t)$ between finite words s and t over
 73 infinite alphabets. Then, the *SW-parsing* problem aims at finding t minimizing $T(s, t) \otimes A(t)$
 74 (*wrt* the ranking defined by \oplus), given an input word s . The latter value is called the distance
 75 between s and A in [21]. Like weighted-parsing methods [13, 24, 23], our approach proceeds

66 This sentence (sym-
 67 bols as variables)
 68 is not immediatly
 clear to me. Maybe
 a short example or
 intuition?

72 The weight is a label
 73 on edges in the de-
 74 rivation tree, right?
 Not sure I under-
 stand the sentence

75 You mean the dis-
 76 tance between s and
 t ? "The latter value"
 is a bit ambiguous

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].

chap. intersection in [15]

The notation $A_{T,s}$ has not been introduced so far. It is not clear why T is a parameter there

OK, à quelques détails (pour moi), tout ça est très bien expliqué

expressiveness: VPA have restricted equality test. comparable to pebble automata? → conclusion

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.

2 Preliminary Notions

Semirings

We shall consider semirings for the weight values of our language models. . 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 $\mathbb{1}$, and such that:

The results are established for a general class of semirings. They can be instantiated for concrete cases

- \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 : $\forall x \in \mathbb{S}$, $\mathbb{0} \otimes x = x \otimes \mathbb{0} = \mathbb{0}$.

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

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.

A semiring \mathbb{S} is *commutative* if \otimes is commutative. It is *idempotent* if for each $x \in \text{dom}(\mathbb{S})$, $x \oplus x = x$. Every idempotent semiring \mathbb{S} induces a partial ordering \leq_{\oplus} called the *natural ordering* of \mathbb{S} [20] and defined, by: for all x and y , $x \leq_{\oplus} y$ 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 $x \oplus y = y$.

► **Lemma 1** (Monotony, [20]). *Let $\langle \mathbb{S}, \oplus, \otimes, \mathbb{0}, \mathbb{1} \rangle$ be an idempotent semiring. For all $x, y, z \in \mathbb{S}$, if $x \leq_{\oplus} y$ then $x \oplus z \leq_{\oplus} y \oplus z$, $x \otimes z \leq_{\oplus} y \otimes z$ and $z \otimes x \leq_{\oplus} z \otimes y$.*

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

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

XX:4 Symbolic Weighted Language Models and Parsing over Infinite Alphabets

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

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

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

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 2 that it implies that $\mathbf{1} \leq_{\oplus} x$ for all $x \in \mathbb{S}$. Similarly, by the first inequality of (i) with $y = \mathbf{0}$, $x \leq_{\oplus} x \otimes \mathbf{0} = \mathbf{0}$. Hence, in a superior semiring, it holds that for all $x \in \mathbb{S}$, $\mathbf{1} \leq_{\oplus} x \leq_{\oplus} \mathbf{0}$. Intuitively, from an optimization point of view, it means that $\mathbf{1}$ is the best value, and $\mathbf{0}$ the worst. In [20], \mathbb{S} with the property (ii) of Lemma 2 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 $n \geq 1$, $x \oplus x^n = x \otimes (\mathbf{1} \oplus x^{n-1}) = x$).

► **Lemma 3.** *Every bounded semiring is idempotent.*

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

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:*

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

ii. *associativity and commutativity:*

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

iii. *distributivity of product over infinite sum:*

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

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 $(\mathbb{S}, \oplus, \otimes, \mathbf{0}, \mathbf{1})$, 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} ,

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 $j \subset \mathbb{N}$

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

OK, donc c'est là que les fonctions d'étiquettes prennent en argument l'input de la règle. Je ne sais pas dans quelle mesure il faut donner un peu d'explications pour faciliter la compréhension du formalisme.

- 152 – for all $\alpha \in \mathbb{S}$ and $\phi \in \Phi_\Sigma$, $\alpha \otimes \phi : x \mapsto \alpha \otimes \phi(x)$, and $\phi \otimes \alpha : x \mapsto \phi(x) \otimes \alpha$
- 153 belong to Φ_Σ , and similarly for \oplus and for $\Phi_{\Sigma,\Delta}$
- 154 – for all $\phi, \phi' \in \Phi_\Sigma$, $\phi \otimes \phi' : x \mapsto \phi(x) \otimes \phi'(x)$ belongs to Φ_Σ
- 155 – for all $\eta, \eta' \in \Phi_{\Sigma,\Delta}$, $\eta \otimes \eta' : x, y \mapsto \eta(x, y) \otimes \eta'(x, y)$ belongs to $\Phi_{\Sigma,\Delta}$
- 156 – 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
- 157 $\eta \otimes_1 \phi : x, y \mapsto \eta(x, y) \otimes \phi(x)$ belong to $\Phi_{\Sigma,\Delta}$
- 158 – 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
- 159 $\eta \otimes_2 \psi : x, y \mapsto \eta(x, y) \otimes \psi(y)$ belong to $\Phi_{\Sigma,\Delta}$
- 160 – similar closures hold for \oplus .

161

162 In what follows, we might omit the subscripts in \otimes_1 , \otimes_2 , \oplus_1 , \oplus_2 when there is no ambiguity,
 163 and keep them only for the special case $\Sigma = \Delta$, i.e. $\eta \in \Phi_{\Sigma,\Sigma}$. When the semiring \mathbb{S} is
 164 complete, let us consider the following operators on the functions of a label theory.

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

166 Similarly as for the above product and sum of functions, the superscripts in \bigoplus_Σ^1 and \bigoplus_Σ^2
 167 shall be reserved to the ambiguous case of $\Phi_{\Sigma,\Sigma}$, in order to distinguish between the first
 168 and the second argument.

169 ► **Definition 4.** A label theory $\bar{\Phi}$ is complete when its underlying semiring \mathbb{S} is complete,
 170 and for all $\Phi_{\Sigma,\Delta} \in \bar{\Phi}$ and all $\eta \in \Phi_{\Sigma,\Delta}$, $\bigoplus_\Sigma \eta \in \Phi_\Delta$ and $\bigoplus_\Delta \eta \in \Phi_\Sigma$.

171

172 The following facts are immediate.

173 ► **Lemma 5.** For $\bar{\Phi}$ complete $\alpha \in \mathbb{S}$, $\phi, \phi' \in \Phi_\Sigma$, $\psi \in \Phi_\Delta$, and $\eta \in \Phi_{\Sigma,\Delta}$:

- 174 i. $\bigoplus_\Sigma \bigoplus_\Delta^2 \eta = \bigoplus_\Delta \bigoplus_\Sigma^1 \eta$
- 175 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
- 176 iii. $(\bigoplus_\Sigma \phi) \oplus (\bigoplus_\Sigma \phi') = \bigoplus_\Sigma (\phi \oplus \phi')$ and $(\bigoplus_\Sigma \phi) \otimes (\bigoplus_\Sigma \phi') = \bigoplus_\Sigma (\phi \otimes \phi')$
- 177 iv. $(\bigoplus_\Delta^2 \eta) \oplus (\bigoplus_\Delta^2 \eta') = \bigoplus_\Delta^2 (\eta \oplus \eta')$, and $(\bigoplus_\Delta^2 \eta) \otimes (\bigoplus_\Delta^2 \eta') = \bigoplus_\Delta^2 (\eta \otimes \eta')$
- 178 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
- 179 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

180 Intuitively, the operators \bigoplus_Σ return global minimum, wrt \leq_\oplus , of functions of $\bar{\Phi}$. A label
 181 theory is called *effective* when for all $\phi \in \Phi_\Sigma$ and $\eta \in \Phi_{\Sigma,\Delta}$, $\bigoplus_\Sigma \phi$, $\bigoplus_\Sigma \eta$, and $\bigoplus_\Delta \eta$ can be
 182 effectively computed from ϕ and η .

183 3 SW Automata and Transducers

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

191 Let \mathbb{S} be a commutative semiring, Σ and Δ be alphabets called respectively *input* and *output*,
 192 and $\bar{\Phi}$ be a label theory over \mathbb{S} containing Φ_Σ , Φ_Δ , $\Phi_{\Sigma,\Delta}$.

partial application is
needed?

notion of diagram of
functions akin BDD
for transitions in
practice

mv appendix?

Je trouve qu'il y a
beaucoup de notions
à retenir (complete,
effective) et ça devient
difficile pour un
lecteur non spécial-
iste. Est-ce que tout
est nécessaire (je ne
sais plus qui m'avait
dit: un concept en
plus, un point en
moins.

precise/restrict com-
plexity

XX:6 Symbolic Weighted Language Models and Parsing over Infinite Alphabets

► **Definition 6.** A symbolic-weighted transducer (*swT*) over Σ , Δ , \mathbb{S} and $\bar{\Phi}$ is a tuple $T = \langle Q, \text{in}, \bar{w}, \text{out} \rangle$, where Q is a finite set of states, $\text{in} : Q \rightarrow \mathbb{S}$ (respectively $\text{out} : Q \rightarrow \mathbb{S}$) 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 $w_{11} : Q \times Q \rightarrow \Phi_{\Sigma, \Delta}$.

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

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

The *swT* T computes on pairs 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 $\langle s, t \rangle \in \Sigma^* \times \Delta^* \setminus \{\langle \varepsilon, \varepsilon \rangle\}$, each considered as the concatenation of the symbol $a \in \Sigma$ (resp. $b \in \Delta$) with a word $u \in \Sigma^*$ (resp. $v \in \Delta^*$), by :

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

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:

I missed sth: what is this ε ? Intuitively clear but not defined?

added u and v def

OK tout ça se lit bien :-)

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

230

231 ► **Example 7.** In Common Western Music Notation [14], several symbols may be used to
 232 represent one single sounding event. For instance, several notes can be combined with a
 233 tie, like in $\text{♪} \text{---} \text{♪}$, and one note can be augmented by half its duration with a dot like in $\text{♪}.$
 234 These notations are perceived equivalent when played, as their duration is equal, yet the
 235 notation is different. We thus want to be able to compare a music score with music played
 236 by a performer. We propose a small weighted transducer model that calculates the distance
 237 between an input sequence of sounding events (music "performance") to an output sequence
 238 of written events (music "score"). Let us consider the tropical (*min-plus*) semiring \mathbb{S} of
 239 Figure 2 and let $\Sigma = \mathbb{R}_+$ be an input alphabet of event dates and $\Delta = \{\mathbf{e}, -\} \times \mathbb{R}_+$ be
 240 an output alphabet of symbols with timestamps. A symbol $\langle \mathbf{e}, d \rangle \in \Delta$ represents an event
 241 starting at date d , and $\langle -, d \rangle$ is a continuation of the previous event.

242 We consider a **swT** with two states q_0 and q_1 whose purpose is to compare a recorded
 243 performance $s \in \Sigma^*$ with a notated music sheet $t \in \Delta^*$. One timestamp $d_i \in \Sigma$ may
 244 correspond to one notated event $\langle \mathbf{e}, d'_i \rangle \in \Delta$, in which case the weight value computed by the
 245 **swT** is the time distance between both (see transitions w_{11} below). If $\langle \mathbf{e}, d'_i \rangle$ is followed by
 246 continuations $\langle -, d'_{i+1} \rangle \dots$, they are just skipped with no cost (transitions w_{01} or weight 1).

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

248 We also must be able to take performing errors into account, while still being able to compare
 249 with the score, since a performer could, for example, play an unwritten extra note. This is
 250 modelled by the transition w_{10} with an arbitrary weight value $\alpha \in \mathbb{S}$, switching from state q_0
 251 (normal) to q_1 (error). The transitions in the second column below switch back to the normal
 252 state q_0 . At last, we let q_0 be the only initial and final state, with $\text{in}(q_0) = \text{out}(q_0) = 1$, and
 253 $\text{in}(q_1) = \text{out}(q_1) = 0$.

254 That way, an **swT** is capable of evaluating the differences between a score and a perform-
 255 ance, all the while ensuring that performance errors are plausible.

256

257 The *Symbolic Weighted Automata* are defined similarly as the transducers of Definition 6, by
 258 simply omitting the output symbols.

259 ► **Definition 8.** A symbolic-weighted automaton (*swA*) over Σ , \mathbb{S} and $\bar{\Phi}$ is a tuple $A =$
 260 $\langle Q, \text{in}, w_1, \text{out} \rangle$, where Q is a finite set of states, $\text{in} : Q \rightarrow \mathbb{S}$ (respectively $\text{out} : Q \rightarrow \mathbb{S}$) are
 261 functions defining the weight for entering (respectively leaving) computation in a state, and
 262 w_1 is a transition function from $Q \times Q$ into Φ_Σ .

263 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 $\phi(a)$.
 264 The computation of A on words $s \in \Sigma^*$ is defined with an intermediate function weight_A ,
 265 defined as follows for $q, q' \in Q$, $a \in \Sigma$, $u \in \Sigma^*$,

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

Je crois qu'il faudrait numéroter les exemples indépendamment des définitions. Cet exemple est le premier qui donne des détails sur l'application visée. Il arrive peut-être un peu tard et est long. On pourrait introduire la motivation dans l'intro, et développer des petits exemples au fur et à mesure.

unique → similar

similar → single

modif.

changed end

reformulated this sentence

ccl to the ex

269

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

$$271 \quad A(s) = \bigoplus_{q, q' \in Q} \text{in}(q) \otimes \text{weight}_A(q, s, q') \otimes \text{out}(q') \quad (4)$$

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

274 ► **Proposition 9.** *Given a swT T over $\Sigma, \Delta, \mathbb{S}$ commutative, bounded and complete, and $\bar{\Phi}$
275 effective, and a swA A over Σ, \mathbb{S} and $\bar{\Phi}$, there exists an effectively constructible swA $B_{T,A}$
276 over Δ, \mathbb{S} and $\bar{\Phi}$, such that for all $t \in \Delta^*$, $B_{T,A}(t) = \bigoplus_{s \in \Sigma^*} A(s) \otimes T(s, t)$.*

277 **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
278 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
279 state set of $B_{T,A}$ will be $Q' = P \times Q$. The entering, leaving and transition functions of $B_{T,A}$
280 will simulate synchronized computations of A and T , while reading an output word of Δ^* .
281 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
282 transition function w'_1 will roughly perform a synchronized product of transitions defined by
283 w_1, w_{01} (T reading in output word and not an input word) and w_{11} (T reading both an input
284 word and an output word). Moreover, w'_1 also needs to simulate transitions defined by w_{10} :
285 T reading in input word and not an output word. Since $B_{T,A}$ will read only in the output
286 word, such a transition corresponds to an ε -transition of swA , but swA have been defined
287 without ε -transitions. Therefore, in order to take care of this case, we perform an on-the-fly
288 suppression of ε -transition in the swA in construction, following the algorithm of [19].
289 Initially, for all $p_1, p_2 \in P$, and $q_1, q_2 \in Q$, let

$$290 \quad w'_1(\langle p_1, q_1 \rangle, \langle p_2, q_2 \rangle) = w_1(p_1, p_2) \otimes [w_{01}(q_1, q_2) \oplus \bigoplus_{\Sigma} w_{11}(q_1, q_2)].$$

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

$$292 \quad w'_1(\langle p_1, q_1 \rangle, \langle p_2, q_3 \rangle) \oplus = \bigoplus_{\Sigma} w_{10}(q_1, q_2) \otimes w'_1(\langle p_1, q_2 \rangle, \langle p_2, q_3 \rangle)$$

proof correctness

$$293 \quad \text{and } \text{out}'(p_1, q_1) \oplus = \bigoplus_{\Sigma} w_{10}(q_1, q_2) \otimes \text{out}'(p_1, q_2) \quad \blacktriangleleft$$

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

revise with nb of tr.
and states

296 ► **Corollary 10.** *Given a swT T over $\Sigma, \Delta, \mathbb{S}$ commutative, bounded and complete, and $\bar{\Phi}$
297 effective, and $s \in \Sigma^+$, there exists an effectively constructible swA $B_{T,s}$ over Δ, \mathbb{S} and $\bar{\Phi}$,
298 such that for all $t \in \Delta^*$, $B_{T,s}(t) = T(s, t)$.*

299 4 SW Visibly Pushdown Automata

300 The model presented in this section generalizes Symbolic VPA [6] from Boolean semirings to
301 arbitrary semiring weight domains. It will compute on nested words over infinite alphabets,
302 associating to every such word a weight value. Nested words are able to describe structures
303 of labeled trees, and in the context of parsing, they will be useful to represent AST.

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 $\langle \mathbb{S}, \oplus, \emptyset, \otimes, \mathbb{1} \rangle$ 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 11.** 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:

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

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

Formally, the transitions of the automaton A are defined in term of an intermediate function weight_A , like in Section 3. A configuration, denoted by $q[\gamma]$, is here composed of a state $q \in Q$ and a stack content $\gamma \in \Gamma^*$, where $\Gamma = \Omega_c \times P$. Hence, weight_A is a function from $[Q \times \Gamma^*] \times \Omega^* \times [Q \times \Gamma^*]$ into \mathbb{S} . The empty stack is denoted by \perp , and the upmost 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 $u \in \Omega^*$.

$$\begin{aligned}
 \text{weight}_A(q[\perp], \varepsilon, q'[\perp]) &= \mathbb{1} \text{ if } q = q' \text{ and } \emptyset \text{ otherwise} \\
 \text{weight}_A\left(q \left[\begin{array}{c} \langle c, p \rangle \\ \gamma \end{array} \right], a, u, q'[\gamma']\right) &= \bigoplus_{q'' \in Q} w_i(q, c, p, a, q'') \otimes \text{weight}_A\left(q'' \left[\begin{array}{c} \langle c, p \rangle \\ \gamma \end{array} \right], u, q'[\gamma']\right) \\
 \text{weight}_A(q[\perp], a, u, q'[\gamma']) &= \bigoplus_{q'' \in Q} w_i^e(q, a, q'') \otimes \text{weight}_A(q''[\perp], u, q'[\gamma']) \\
 \text{weight}_A\left(q \left[\begin{array}{c} \langle c, p \rangle \\ \gamma \end{array} \right], c', u, q'[\gamma']\right) &= \bigoplus_{\substack{q'' \in Q \\ p' \in P}} w_c(q, c, p, c', p', q'') \otimes \text{weight}_A\left(q'' \left[\begin{array}{c} \langle c', p' \rangle \\ \langle c, p \rangle \\ \gamma \end{array} \right], u, q'[\gamma']\right)
 \end{aligned} \tag{5}$$

Là je crois qu'il faudrait expliquer ces Omega, je commence à fatiguer et je suis un peu largué par toutes ces définitions. J'intuie qu'il s'agit des symboles, parenthèses ouvrantes et fermantes? Pourquoi il faut un alphabet pour les parenthèses?

Est-ce que tout le monde sait ce qu'est un pushdown automata? Je suppose que c'est lié à la pile.

moved this to the beginning

intro to func

introduced the 6 cases

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

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

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

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

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

c p to <c, p>

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 12.** 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 0.

Like VPA and symbolic VPA, the class of sw-VPA is closed under the binary operators of the underlying semiring.

► **Proposition 13.** 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 $(A_1 \otimes A_2)(s) = A_1(s) \otimes A_2(s)$.

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

total?

Let us assume that the semiring \mathbb{S} is commutative, bounded, and complete, and that $\bar{\Phi}$ is an effective label theory. We propose a Dijkstra algorithm computing, for a sw-VPA A over Ω, \mathbb{S} and $\bar{\Phi}$, the minimal weight for a word in Ω^* . We distinguish two cases : when the stack is empty, and when it is not. In the case of an empty stack, let $b_\perp : Q \times Q \rightarrow \mathbb{S}$ be such that :

introduced 2 cases for b

$$b_\perp(q, q') = \bigoplus_{s \in \Omega^*} \text{weight}_A(q[\perp], s, q'[\perp]). \quad (7)$$

Since \mathbb{S} is complete, the infinite sum in (7) is well defined, and, providing that \mathbb{S} is total, it is the minimum in Ω^* , wrt \leq_\oplus , of the function $s \mapsto \text{weight}_A(q[\sigma], s, q'[\sigma])$. The term $q[\perp], s, q'[\perp]$ of this sum is the central expression in the definition (6) of $A(s_0)$, for the minimum s_0 of the function weight_A .

so ?

b_\top : mot bien par-
enthésé c/r

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

$$b_\top(q, p, q') : c \mapsto \bigoplus_{s \in \Omega^*} \text{weight}_A\left(q \left[\begin{array}{c} \langle c, p \rangle \\ \top \end{array} \right], s, q' \left[\begin{array}{c} \langle c, p \rangle \\ \top \end{array} \right]\right) \quad (8)$$

Intuitively, the function defined in (8) associates to $c \in \Omega_c$ the minimum weight of a computation of A starting in state q with a stack $\langle c, p \rangle \cdot \gamma \in \Gamma^+$ and ending in state q' with the same stack, such that the computation can not pop the pair made of c and p at the top of this stack, but may only read these symbols. Moreover, A may push another pair $\langle c', p' \rangle$

■ **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 $d_{\perp}(q_1, q_2) = d_{\top}(q_1, p, q_2) = 0$ 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 $q_0, q_3 \in Q$,

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

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

371 on the top of $\langle c, p \rangle \cdot \gamma$, following the third case of in the definition (5) of weight_A , and may
 372 pop $\langle c', p' \rangle$ later, following the fifth case of (5) (return symbol).

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

375 The infinite sums in the updates of d in Algorithm 1, Figure 3 are well defined since \mathbb{S}
 376 is complete. ** effectively computable by hypothesis that the label theory is effective**

377 The algorithm performs $2 \cdot |Q|^2$ iterations until P is empty, and each iteration has a time
 378 complexity $O(|Q|^2 \cdot |P|)$. That gives a time complexity $O(|Q|^4 \cdot |P|)$. It can be reduced by
 379 implementing P as a priority queue, prioritized by the value returned by d .

380 The correctness of Algorithm 1 is ensured by the invariant expressed in the following
 381 lemma.

382 ► **Lemma 14.** For all $\langle q_1, q_2 \rangle \notin \mathcal{Q}$, $d_{\perp}(q_1, q_2) = b_{\perp}(q_1, q_2)/$

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

385 ► **Lemma 15.** For all $\langle q_1, p, q_2 \rangle \notin \mathcal{Q}$, $d_{\top}(q_1, p, q_2) = b_{\top}(q_1, p, q_2)$,

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

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

explication Fig. 3
suivant cas de (5)

complete **

detail with nb tr.
and states

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

5 Symbolic Weighted Parsing

Let us now apply the models and results of the previous sections to the problem of parsing over infinite alphabet. Let Σ be a countable input alphabet, and $\Omega = \Omega_i \uplus \Omega_c \uplus \Omega_r$ be a countable output alphabet. Let $(\mathbb{S}, \oplus, \emptyset, \otimes, \mathbb{1})$ be a commutative, bounded, and complete semiring and let $\bar{\Phi}$ 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 the notations of Section 4). We assume given the following input:

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

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

$$d(s, t) \otimes A(t) = \bigoplus_{t' \in \mathcal{T}(\Omega)} d(s, t') \otimes A(t') \quad (9)$$

Following the terminology of [21], *sw-parsing* is the problem of computing the distance (9) between the input s and the output weighted language of A , and returning a witness t . Every labeled tree t can be linearized into a nested word $\text{lin}(t) \in \Omega^*$, assuming *e.g.* that Ω_i contain the symbols labelling the leaves (symbols of rank 0) and Ω_c and Ω_r contain respectively one left and right parenthesis $\langle b$ and $\rangle b$ for each symbol b labelling inner nodes (symbols of rank > 0). With this representation, the projection $\text{lin}(t)|_{\Omega_i}$ is then the sequence of leaves of t , enumerated in a *dfs*-traversal. We show in Appendix A how to convert a (sw) tree automaton A into a sw-VPA computing $A(\text{lin}(t))$ for every tree t . That also holds, for the set of ASTs of a weighted CF-grammar. Therefore, *sw-parsing* generalizes the problem of searching, the best derivation of a weighted CF-grammar that yields a given input, sometimes referred as *weighted parsing*, see *e.g.* [13] and [23] for a more general weighted parsing framework. The latter indeed corresponds to the particular case where the alphabet is finite, T computes identity and A is obtained from the weighted CF grammar.

► **Proposition 17.** *The problem of Symbolic Weighted parsing can be solved in PTIME in the size of the input swT T , sw-VPA A and input word s , and the computation time of the functions of the label theory.*

Proof. (sketch) We follow a *Bar-Hillel* construction, also called parsing by intersection. We first extend the swT T over Σ , Ω_i , \mathbb{S} , and $\bar{\Phi}$, into a swT T' over Σ and Ω (and the same semiring and label theory), such that for all $s \in \Sigma^*$, and $u \in \Omega^*$, $T'(s, u) = T(s, u|_{\Omega_i})$. The transducer T' simply skips every symbol $b \in \Omega \setminus \Omega_i$, by the addition to the transition of T , of new transitions of the form $w_{01}(q, \varepsilon, b, q')$. Then, given an input word $s \in \Sigma^*$, using Corolary 10, we compute the swA $B_{T', s}$, such that for all $t \in \Omega^*$, $B_{T', s}(t) = d(s, t)$. Next, we compute the sw-VPA $B_{T', s} \otimes A$, using Proposition 13. It remains to compute a best nested-word $w \in \Omega^*$ using the best-search procedure of Proposition 16. ◀

total?

Ah oui, ça aurait pu être dit avant.

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

Conclusion

We have introduced weighted language models (SW transducers and visibly pushdown automata) computing over infinite alphabets, and applied them to the problem of parsing with infinitely many possible input symbols (typically timed events). This approach extends conventional parsing and weighted parsing by computing a derivation tree modulo a generic distance between words, defined by a SW transducer given in input. This enables to consider finer word relationships than strict equality, opening possibilities of quantitative analysis via this method.

Ongoing and future work include

- The study of other theoretical properties of SW models, such as the extension of the best search algorithm from 1-best to n -best [17], and to k -closed semirings [20] (instead of *bounded*, which corresponds to 0-closed).
- ...there is room to improve the complexity bounds for the algorithms ... modular approach with oracles ...
- present here an offline algorithm for best search, semi-online implementation for AMT (bar-by-bar approach) with an on-the-fly automata construction.

TODO future work

References

- 1 Rajeev Alur and Parthasarathy Madhusudan. Adding nesting structure to words. *Journal of the ACM (JACM)*, 56(3):1–43, 2009.
- 2 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.
- 3 Patricia Bouyer, Antoine Petit, and Denis Thérien. An algebraic approach to data languages and timed languages. *Information and Computation*, 182(2):137–162, 2003.
- 4 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.
- 5 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.
- 6 Loris D’Antoni and Rajeev Alur. Symbolic visibly pushdown automata. In *International Conference on Computer Aided Verification*, pages 209–225. Springer, 2014.
- 7 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.
- 8 Loris D’Antoni and Margus Veanes. Automata modulo theories. *Communications of the ACM*, 64(5):86–95, 2021. URL: [seealsohttps://pages.cs.wisc.edu/~loris/symbolicautomata.html](https://pages.cs.wisc.edu/~loris/symbolicautomata.html).
- 9 E. W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1(1):269–271, 1959.
- 10 Manfred Droste and Werner Kuich. Semirings and formal power series. In *Handbook of Weighted Automata*, pages 3–28. Springer, 2009.
- 11 Manfred Droste, Werner Kuich, and Heiko Vogler. *Handbook of weighted automata*. Springer Science & Business Media, 2009.
- 12 Francesco Foscarin, Florent Jacquemard, Philippe Rigaux, and Masahiko Sakai. A Parse-based Framework for Coupled Rhythm Quantization and Score Structuring. In *Mathematics and Computation in Music (MCM)*, volume 11502 of *Lecture Notes in Artificial Intelligence*, Madrid, Spain, 2019. Springer. URL: <https://hal.inria.fr/hal-01988990>, doi:10.1007/978-3-030-21392-3_20.

- 479 13 Joshua Goodman. Semiring parsing. *Computational Linguistics*, 25(4):573–606, 1999.
- 480 14 Elaine Gould. *Behind Bars: The Definitive Guide to Music Notation*. Faber Music, 2011.
- 481 15 Dick Grune and Criel J.H. Jacobs. *Parsing Techniques*. Number 2nd edition in Monographs
482 in Computer Science. Springer, 2008.
- 483 16 Liang Huang. Advanced dynamic programming in semiring and hypergraph frameworks. In
484 *In COLING*, 2008.
- 485 17 Liang Huang and David Chiang. Better k-best parsing. In *Proceedings of the Ninth International*
486 *Workshop on Parsing Technology*, Parsing '05, pages 53–64, Stroudsburg, PA, USA, 2005.
487 Association for Computational Linguistics. URL: [http://dl.acm.org/citation.cfm?id=](http://dl.acm.org/citation.cfm?id=1654494.1654500)
488 [1654494.1654500](http://dl.acm.org/citation.cfm?id=1654494.1654500).
- 489 18 Michael Kaminski and Nissim Francez. Finite-memory automata. *Theor. Comput. Sci.*,
490 134:329–363, 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),
491 doi:[http://dx.doi.org/10.1016/0304-3975\(94\)90242-9](http://dx.doi.org/10.1016/0304-3975(94)90242-9).
- 492 19 Sylvain Lombardy and Jacques Sakarovitch. The removal of weighted ε -transitions. In
493 *International Conference on Implementation and Application of Automata*, pages 345–352.
494 Springer, 2012.
- 495 20 Mehryar Mohri. Semiring frameworks and algorithms for shortest-distance problems. *Journal*
496 *of Automata, Languages and Combinatorics*, 7(3):321–350, 2002.
- 497 21 Mehryar Mohri. Edit-distance of weighted automata: General definitions and al-
498 gorithms. *International Journal of Foundations of Computer Science*, 14(06):957–982,
499 2003. URL: <https://www.worldscientific.com/doi/abs/10.1142/S0129054103002114>,
500 arXiv:<https://www.worldscientific.com/doi/pdf/10.1142/S0129054103002114>, doi:10.
501 1142/S0129054103002114.
- 502 22 Mehryar Mohri. Edit-distance of weighted automata: General definitions and algorithms.
503 *International Journal of Foundations of Computer Science*, 14(06):957–982, 2003.
- 504 23 Richard Mörbitz and Heiko Vogler. Weighted parsing for grammar-based language models.
505 In *Proceedings of the 14th International Conference on Finite-State Methods and Natural*
506 *Language Processing*, pages 46–55, Dresden, Germany, September 2019. Association for
507 Computational Linguistics. URL: <https://www.aclweb.org/anthology/W19-3108>, doi:10.
508 18653/v1/W19-3108.
- 509 24 Mark-Jan Nederhof. Weighted deductive parsing and Knuth’s algorithm. *Computational*
510 *Linguistics*, 29(1):135–143, 2003. URL: <https://doi.org/10.1162/089120103321337467>.
- 511 25 Frank Neven, Thomas Schwentick, and Victor Vianu. Finite state machines for strings
512 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.
- 513 26 Luc Segoufin. Automata and logics for words and trees over an infinite alphabet. In *Computer*
514 *Science Logic*, volume 4207 of *LNCS*. Springer, 2006.
- 515 27 Moshe Y Vardi. Linear-time model checking: automata theory in practice. In *International*
516 *Conference on Implementation and Application of Automata*, pages 5–10. Springer, 2007.
- 517

A

 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 [1], and [4] 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 $\mathcal{T}(\Omega)$. It is the smallest set such that $\Omega_0 \subset \mathcal{T}(\Omega)$ and for all $1 \leq k \leq M$, all $a \in \Omega_k$, and all $t_1, \dots, t_k \in \mathcal{T}(\Omega)$, $a(t_1, \dots, t_k) \in \mathcal{T}(\Omega)$. 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 18.** 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 $w_k^e : Q \times Q^k \rightarrow \Phi_{\Omega_k}$.

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:

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

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

$$\text{weight}_A(q_0, a, t) = \bigoplus_{q_1 \dots q_k \in Q^k} w(q_0, a, b, q_1 \dots q_k) \otimes \bigotimes_{i=1}^k \text{weight}_A(q_i, b, t_i) \quad (10)$$

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

$$A(t) = \bigoplus_{q \in Q} \text{in}(q) \otimes \text{weight}_A(q, \varepsilon, t) \quad (11)$$

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 [5], 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 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*.

XX:16 Symbolic Weighted Language Models and Parsing over Infinite Alphabets

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

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

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

► **Proposition 19.** *For all swTA A over Ω , \mathbb{S} commutative, and $\bar{\Phi}$, there exists an effectively constructible sw-VPA A' over $\hat{\Omega}$, \mathbb{S} and $\hat{\Phi}$ such that for all $t \in \mathcal{T}(\Omega)$, $A'(\text{lin}(t)) = A(t)$.*

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:

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

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

572 **Todo list**

573	■ I think that we can make the case for a larger set of situations: given a linear music	
574	notation input, for instance elements in an XML file, we want to structure the	
575	input according to a hierarchical rhythmic space. I can elaborate...	1
576	■ register: skip refs and details, add Mikolaj recent	1
577	■ This sentence (symbols as variables) is not immediatly clear to me. Maybe a short	
578	example or intuition?	2
579	■ The weight is a label on edges in the derivation tree, right? Not sure I understand	
580	the sentence	2
581	■ You mean the distance between s and t ? "The latter value" is a bit ambiguous	2
582	■ chap. intersection in [15]	3
583	■ The notation $A_{T,s}$ has not been introduced so far. It is not clear why T is a	
584	parameter there	3
585	■ OK, à quelques détails (pour moi), tout ça est très bien expliqué	3
586	■ expressiveness: VPA have restricted equality test. comparable to pebble automata?	
587	→ conclusion	3
588	■ The results are established for a general class of semirings. They can be instantiated	
589	for concrete cases	3
590	■ There is sometimes a confusion in the text between the struture and the domain \mathbb{S} .	
591	Not essential	3
592	■ Here the difference between \mathbb{S} as a structure and as a domain is blurred.	4
593	■ $j \in \mathbb{N}$: j is en element of \mathbb{N} , not the same $s \in \mathbb{N}$	4
594	■ results of this paper: for semirings commutative, bounded, total and complete	4
595	■ OK, donc c'est là que les fonctions d'étiquettes prennent en argument l'input de la	
596	règle. Je ne sais pas dans quelle mesure il faut donner un peu d'explications pour	
597	faciliter la compréhension du formalisme.	4
598	■ partial application is needed?	5
599	■ notion of diagram of functions akin BDD for transitions in practice	5
600	■ mv appendix?	5
601	■ Je trouve qu'il y a beaucoup de notions à retenir (complete, effective) et ça devient	
602	difficile pour un lecteur non spécialiste. Est-ce que tout est nécessaire (je ne sais	
603	plus qui m'avait dit: un concept en plus, un point en moins.	5
604	■ precise/restrict complexity	5
605	■ I missed sth: what is this ε ? Intuitively clear but not defined?	6
606	■ added u and v def	6
607	■ OK tout ça se lit bien :-)	6
608	■ Je crois qu'il faudrait numéroter les exemples indépendamment des définitions. Cet	
609	exemple est le premier qui donne des détails sur l'application visée. Il arrive	
610	peut-être un peu tard et est long. On pourrait introduire la motivation dans	
611	l'intro, et développer des petits exemples au fur et à mesure.	7
612	■ unique → similar	7
613	■ similar → single	7
614	■ modif.	7
615	■ changed end	7
616	■ reformulated this sentence	7
617	■ ccl to the ex	7
618	■ proof correctness	8
619	■ revise with nb of tr. and states	8

XX:18 Symbolic Weighted Language Models and Parsing over Infinite Alphabets

620	■	Là je crois qu'il faudrait expliquer ces Omega, je commence à fatiguer et je suis un peu	
621		largué par toutes ces définitions. J'intuite qu'il s'agit des symboles, parenthèses	
622		ouvrantes et fermantes? Pourquoi il faut un alphabet pour les parenthèses? . . .	9
623	■	Est-ce que tout le monde sait ce qu'est un pushdown automata? Je suppose que	
624		c'est lié à la pile.	9
625	■	moved this to the beginning	9
626	■	intro to func	9
627	■	introduced the 6 cases	9
628	■	notation cp for $\langle c, p \rangle$?	9
629	■	c p to $\langle c, p \rangle$	10
630	■	todo example VPA	10
631	■	total?	10
632	■	introduced 2 cases for b	10
633	■	so ?	10
634	■	b_{\top} : mot bien parenthésé c/r	10
635	■	explication Fig. 3 suivant cas de (5)	11
636	■	complete **	11
637	■	detail with nb tr. and states	11
638	■	total?	12
639	■	2 lines Application to Automated Music Transcription: implementation \neq but same	
640		principle, on-the-fly automata construction during best search, for efficiency. . . .	12
641	■	TODO future work	13