Weight Annotation in Information Extraction

Johannes Doleschal, Benny Kimelfeld, Wim Martens, Liat Peterfreund

Weight Annotation in Information Extraction

Johannes Doleschal
University of Bayreuth, Germany
Hasselt University, Belgium

Benny Kimelfeld
Technion – Israel Institute of Technology, Haifa, Israel

Wim Martens
University of Bayreuth, Germany

Liat Peterfreund
CNRS, IRIF – Université de Paris, France
University of Edinburgh, United Kingdom

Abstract

The framework of document spanners abstracts the task of information extraction from text as a function that maps every document (a string) into a relation over the document's spans (intervals identified by their start and end indices). For instance, the regular spanners are the closure under the Relational Algebra (RA) of the regular expressions with capture variables, and the expressive power of the regular spanners is precisely captured by the class of vset-automata – a restricted class of transducers that mark the endpoints of selected spans.

In this work, we embark on the investigation of document spanners that can annotate extractions with auxiliary information such as confidence, support, and confidentiality measures. To this end, we adopt the abstraction of provenance semirings by Green et al., where tuples of a relation are annotated with the elements of a commutative semiring, and where the annotation propagates through the (positive) RA operators via the semiring operators. Hence, the proposed spanner extension, referred to as an *annotator*, maps every string into an annotated relation over the spans. As a specific instantiation, we explore weighted vset-automata that, similarly to weighted automata and transducers, attach semiring elements to transitions. We investigate key aspects of expressiveness, such as the closure under the positive RA, and key aspects of computational complexity, such as the enumeration of annotated answers and their ranked enumeration in the case of numeric semirings. For a number of these problems, fundamental properties of the underlying semiring, such as positivity, are crucial for establishing tractability.

2012 ACM Subject Classification Information systems → Information extraction; Theory of computation → Transducers; Theory of computation → Problems, reductions and completeness; Theory of computation → Data provenance

Keywords and phrases Information extraction, regular document spanners, weighted automata, provenance semirings, K-relations

Digital Object Identifier 10.4230/LIPIcs.ICDT.2020.8

Related Version A full version of the paper is available at `https://arxiv.org/abs/1908.11642`.

Funding This work was supported by the German-Israeli Foundation for Scientific Research and Development (GIF), grant I-1502-407.6/2019. The work of Johannes Doleschal and Wim Martens was also supported by the Deutsche Forschungsgemeinschaft (DFG), grant MA 4938/4-1. The work of Benny Kimelfeld and Liat Peterfreund was also supported by the Israel Science Foundation (ISF), grants 1295/15 and 768/19, and the DFG project 412400621 (DIP program).
Liat Peterfreund: A part of the work was donewhile the author was affiliated with the Technion.

Acknowledgements We are grateful to Matthias Niewerth for many useful discussions and his help regarding Theorem 7.1 and Shaull Almagor for many helpful comments regarding weighted automata. Furthermore, we thank the anonymous reviewers for ICDT 2020 for many helpful remarks.

1 Introduction

A plethora of paradigms have been developed over the past decades towards the challenge of extracting structured information from text – a task generally referred to as Information Extraction (IE). Common textual sources include natural language from a variety of sources such as scientific publications, customer input and social media, as well as machine-generated activity logs. Instantiations of IE are central components in text analytics and include tasks such as segmentation, named-entity recognition, relation extraction, and coreference resolution [38]. Rules and rule systems have consistently been key components in such paradigms, yet their roles have varied and evolved over time. Systems such as Xlog [42] and SystemT [4] use IE rules for materializing relations inside *relational query languages*. Machine-learning classifiers and probabilistic graphical models (e.g., Conditional Random Fields) use rules for *feature generation* [24, 44]. Rules serve as *weak constraints* (later translated into probabilistic graphical models) in Markov Logic Networks [32] and in the DeepDive system [43]. Rules are also used for generating *noisy training data* ("labeling functions") in the Snorkel system [34].

The framework of *document spanners* (*spanners* for short) provides a theoretical basis for investigating the principles of relational rule systems for IE [13]. Specifically, a spanner extracts from a document a relation over text intervals, called *spans*, using either atomic extractors or a relational query on top of the atomic extractors. More formally, by a *document* we refer to a string $\mathbf{d}$ over a finite alphabet, a *span* of $\mathbf{d}$ represents a substring of $\mathbf{d}$ by its start and end positions, and a *spanner* is a function that maps every document $\mathbf{d}$ into a relation over the spans of $\mathbf{d}$. The most studied spanner language is that of the *regular* spanners: atomic extraction is via *regex formulas*, which are regular expressions with capture variables, and relational manipulation is via the relational algebra: projection, natural join, union, and difference. Equivalently, the regular spanners are the ones expressible as *variable-set automata* (vset-automata for short), which are nondeterministic finite-state automata that can open and close variables (playing the role of the attributes of the extracted relation). Interestingly, there has been an independent recent effort to express artificial neural networks for natural language processing by means of finite-state automata [26, 27, 47].

To date, the research on spanners has focused on their expressive power [13, 17, 31], their computational complexity [2, 3, 15, 18], incompleteness [25, 30], and other system aspects such as cleaning [14] and distributed query planning [8]. That research has exclusively adopted a Boolean approach: *a tuple is either extracted or not*. Nevertheless, when applied to noisy or fuzzy domains such as natural language, modern approaches in artificial intelligence adopt a *quantitative* approach where each extracted tuple is associated with a level of confidence that the tuple coincides with the intent. When used within an end-to-end IE system, such confidence can be used as a principled way of tuning the balance between precision and recall. For instance, in probabilistic IE models (e.g., CRF), each extraction has an associated probability. In systems of weak constraints (e.g., MLN), every rule has a numerical weight, and the confidence in an extraction is an aggregation of the weights of the invoked rules that lead to the extraction. IE via artificial neural networks typically involves thresholding over a produced score or confidence value [5, 29]. Numerical scores in extraction are also used for quantifying the similarity between associated substrings, as done with sequence alignment and edit distance in the analysis of biological sequences such as DNA and RNA [45, 46].

In this work, we embark on the investigation of spanners that quantify the extracted tuples. We do so by adopting the concept of *annotated relations* from the framework of *provenance semirings* by Green et al. [20]. In essence, every tuple of the database is annotated with an

element of a commutative semiring, and the positive relational algebra manipulates both the tuples and their annotations by translating relational operators into semiring operators (e.g., product for natural join and sum for union). An annotated relation is referred to as a $\mathbb{K}$-relation, where $\mathbb{K}$ is the domain of the semiring. The conceptual extension of the spanner model is straightforward: instead of a function (i.e., spanner) that maps every document $\mathbf{d}$ into a relation over the spans of $\mathbf{d}$, we consider a function that maps every $\mathbf{d}$ into a $\mathbb{K}$-relation over the spans of $\mathbf{d}$. We refer to such a function as a $\mathbb{K}$-*annotator*. Interestingly, as in the relational case, we can vary the meaning of the annotation by varying the semiring:

- Confidence via the *probability* (a.k.a. *inside*) semiring and the *Viterbi* (best derivation) semiring [19];
- Support (i.e., number of derivations) via the *counting* semiring [19];
- Access control via the semiring of the *confidentiality policies* [16] (e.g., does the extracted tuple require reading top-secret sections? which level suffices for the tuple?);
- The traditional spanners via the *Boolean* semiring.

As a specific instantiation of $\mathbb{K}$-annotators, we study the class of $\mathbb{K}$-*weighted vset-automata*. Such automata generalize vset-automata in the same manner as weighted automata and weighted transducers (cf., e.g., the Handbook of Weighted Automata [10]): transitions are weighted by semiring elements, the cost of a run is the product of the weights along the run, and the weight (annotation) of a tuple is the sum of costs of all the runs that produce the tuple. Again, there has been recent research that studies the connection between models of artificial neural networks in natural language processing and weighted automata [39]. Our investigation answers several fundamental questions about $\mathbb{K}$-weighted vset-automata:

1. Is this class closed under the positive relational algebra (according to the semantics of provenance semirings [20])?
2. What is the computational complexity of computing the annotation of a tuple?
3. Can we enumerate the annotated tuples as efficiently as we can do so for ordinary vset-automata (i.e., regular document spanners)?
4. In cases of numerical semirings (i.e., when $\mathbb{K}$ is a set of numbers), what is the complexity of enumerating the answers in ranked order by decreasing weight?

Our answers are mostly positive, put the last question aside, and show that $\mathbb{K}$-weighted vset-automata possess appropriate expressivity and tractability properties. As for the last question, we show that ranked enumeration is intractable and inapproximable for some of the aforementioned semirings (e.g., the probability and counting semirings), but tractable for positively ordered and bipotent semirings, such as the Viterbi semiring. Due to space constraints, we sometimes omit proofs or only provide a proof sketch.

2 Preliminaries

Our annotators will read documents and produce *annotated relations* [20], which are relations in which each tuple is annotated with an element from a semiring. In this section we revisit the basic definitions and properties of annotated relations.

Semirings

A semiring $(\mathbb{K}, \oplus, \otimes, \overline{0}, \overline{1})$ is an algebraic structure consisting of a set $\mathbb{K}$, containing two distinguished elements: the *zero* element $\overline{0}$ and the *unit* element $\overline{1}$, and equipped with two binary operations, namely *addition* $\oplus$ and *multiplication* $\otimes$ such that:

- $(\mathbb{K}, \oplus)$ is a commutative monoid with identity element $\overline{0}$;

- $(\mathbb{K}, \otimes)$ is a monoid with identity element $\overline{1}$;
- multiplication distributes over addition, i.e., $(a \oplus b) \otimes c = (a \otimes c) \oplus (b \otimes c)$ and $c \otimes (a \oplus b) = (c \otimes a) \oplus (c \otimes b)$;
- $\overline{0}$ is absorbing for $\otimes$, i.e., $\overline{0} \otimes a = a \otimes \overline{0} = \overline{0}$.

A semiring is called *commutative* if $(\mathbb{K}, \otimes)$ is commutative. We follow Green et al. [20] and assume that a semiring is commutative if not stated otherwise. Furthermore, following Eilenberg [11], a semiring is *positive* if the following conditions hold:

- $\overline{0} \neq \overline{1}$,
- If $a \oplus b = \overline{0}$, then $a = \overline{0} = b$.
- If $a \otimes b = \overline{0}$, then $a = \overline{0}$ or $b = \overline{0}$.

An element $a \in \mathbb{K}$ is a *zero divisor* if $a \neq \overline{0}$ and there is an element $b \in \mathbb{K}$ with $b \neq \overline{0}$ and $a \otimes b = \overline{0}$. Furthermore, an element $a \in \mathbb{K}$ has an *additive inverse*, if there is an element $b \in \mathbb{K}$ such that $a \oplus b = \overline{0}$. In the following, we will also identify a semiring by its domain $\mathbb{K}$ if the rest is clear from the context. When we do this for numeric semirings such as $\mathbb{R}$ and $\mathbb{N}$, we always assume the usual addition and multiplication.

▶ **Example 2.1.** The following are examples for commutative semirings. It is easy to verify that all but the numeric semirings and the Łukasiewcz semiring are positive.

1. The *numeric* semirings $(\mathbb{R}, +, \cdot, 0, 1)$ and $(\mathbb{Z}, +, \cdot, 0, 1)$;
2. The *counting* semiring $(\mathbb{N}, +, \cdot, 0, 1)$;
3. The *Boolean* semiring $(\mathbb{B}, \vee, \wedge, \mathsf{false}, \mathsf{true})$ where $\mathbb{B} = \{\mathsf{true}, \mathsf{false}\}$;
4. The *probability* semiring $(\mathbb{R}^+, +, \cdot, 0, 1)$.[1] Rabin [33] and Segala [40] define probabilistic automata over this semiring, where all edge weights must be between 0 and 1 and the sum of all edge weights starting some state, labeled by the same label must be 1;
5. The *Viterbi* semiring $([0, 1], \max, \cdot, 0, 1)$ which is used in probabilistic parsing [9];
6. The *access control semiring* $\mathbb{A} = (\{P < C < S < T < 0\}, \min, \max, 0, P)$, where P is "public", C is "confidential", S is "secret", T is "top secret", and 0 is "so secret that nobody can access it" [16];
7. The *tropical semiring* $(\mathbb{N} \cup \{\infty\}, \min, +, \infty, 0)$ where min stands for the binary minimum function. This semiring is used in optimization problems of networks [9].
8. The Łukasiewcz semiring, whose domain is $[0, 1]$, with addition given by $x \oplus y = \max(x, y)$, with multiplication $x \otimes y = \max(0, x + y - 1)$, zero element 0, and unit 1. This semiring is used in multivalued logics [9].

Complexity-wise, we assume that single semiring elements can be stored in a single register and that addition and multiplication can be carried out in constant time – in similar spirit as the standard assumption for Random Access Machines. We use this assumption to simplify the analysis of algorithms.

2.1 Annotated Relations

We assume infinite and disjoint sets $\mathbf{D}$ and Vars, containing *data values* (or simply *values*) and *variables*, respectively. Let $V \subseteq \mathsf{Vars}$ be a finite set of variables. A *V-tuple* is a function $\mathbf{t} : V \to \mathbf{D}$ that assigns values to variables in V. The *arity* of $\mathbf{t}$ is the cardinality $|V|$ of V. For a subset $X \subseteq \mathsf{Vars}$, we denote the restriction of $\mathbf{t}$ to the variables in X by $\mathbf{t}\restriction X$. We denote the set of all the V-tuples by V-Tup. We sometimes leave V implicit when the precise set is

[1] One may expect the domain to be $[0, 1]$, but this is difficult to obtain while maintaining the semiring properties. For instance, defining $a \oplus b$ as $\min\{a + b, 1\}$ would violate distributivity.

not important. Let $\mathbb{K}$ be a set containing a distinguished element $\overline{0}$. A $(\mathbb{K}, \mathbf{D})$-*relation* R over V is a function $R : V\text{-}\mathsf{Tup} \to \mathbb{K}$ such that its *support* defined by $\mathsf{supp}(R) \stackrel{\text{def}}{=} \{\mathbf{t} \mid R(\mathbf{t}) \neq \overline{0}\}$ is finite. The *arity* of a $(\mathbb{K}, \mathbf{D})$-relation over V is $|V|$. When $\mathbf{D}$ is clear from the context or irrelevant, we also use $\mathbb{K}$-*relations* to refer to $(\mathbb{K}, \mathbf{D})$-relations.

▶ **Example 2.2.** The bottom left table in Figure 1 shows an example $(\mathbb{K}, \mathbf{D})$-relation, where $\mathbb{K}$ is the Viterbi semiring. The variables are x_{pers} and x_{loc}, so the V-tuples are described in the first two columns. The third column contains the element in $\mathbb{K}$ associated to each tuple.

Relational Algebra for Annotated Relations

Green et al. [20] defined a set of operators on $(\mathbb{K}, \mathbf{D})$-relations that naturally correspond to relational algebra operators and map $\mathbb{K}$-relations to $\mathbb{K}$-relations. Let $(\mathbb{K}, \oplus, \otimes, \overline{0}, \overline{1})$ be a commutative semiring. The algebraic operators[2] *union*, *projection*, and *natural join* are defined in the usual way, for all finite sets $V_1, V_2 \subseteq \mathsf{Vars}$ and for all $\mathbb{K}$-relations R_1 over V_1 and R_2 over V_2, as follows.

- **Union**: If $V_1 = V_2$ then the union $R \stackrel{\text{def}}{=} R_1 \cup R_2$ is a function $R : V_1\text{-}\mathsf{Tup} \to \mathbb{K}$ defined by $R(\mathbf{t}) \stackrel{\text{def}}{=} R_1(\mathbf{t}) \oplus R_2(\mathbf{t})$. (Otherwise, the union is not defined.)
- **Projection**: For $X \subseteq V_1$, the projection $R \stackrel{\text{def}}{=} \pi_X R_1$ is a function $R : X\text{-}\mathsf{Tup} \to \mathbb{K}$ defined by

$$R(\mathbf{t}) \stackrel{\text{def}}{=} \bigoplus_{\mathbf{t}=\mathbf{t}'\restriction X \text{ and } R_1(\mathbf{t}')\neq\overline{0}} R_1(\mathbf{t}').$$

- **Natural Join:** The natural join $R \stackrel{\text{def}}{=} R_1 \bowtie R_2$ is a function $R : (V_1 \cup V_2)\text{-}\mathsf{Tup} \to \mathbb{K}$ defined by

$$R(\mathbf{t}) \stackrel{\text{def}}{=} R_1(\mathbf{t}_1) \otimes R_2(\mathbf{t}_2)$$

  where $\mathbf{t}_1$ and $\mathbf{t}_2$ are the restrictions $\mathbf{t}\restriction V_1$ and $\mathbf{t}\restriction V_2$, respectively.
- **Selection:** If $\mathbf{P}$ is a selection predicate that maps each tuple in $V_1\text{-}\mathsf{Tup}$ to either $\overline{0}$ or $\overline{1}$ then $R \stackrel{\text{def}}{=} \sigma_{\mathbf{P}}(R_1)$ is a function $R : V_1\text{-}\mathsf{Tup} \to \mathbb{K}$ defined by

$$R(\mathbf{t}) \stackrel{\text{def}}{=} R_1(\mathbf{t}) \otimes \mathbf{P}(\mathbf{t}).$$

▶ **Proposition 2.3.** [20] *The above operators preserve the finiteness of the supports and therefore they map $\mathbb{K}$-relations into $\mathbb{K}$-relations.*

Hence, we obtain an algebra on $\mathbb{K}$-relations.

3 K-Annotators

We start by setting the basic terminology. We fix a finite alphabet Σ that is disjoint from Vars. A *document* is a finite sequence $\mathbf{d} = \sigma_1 \cdots \sigma_n$ where $\sigma_i \in \Sigma$ for each $i = 1, \ldots, n$. By Docs we denote the set of all documents. A *(k-ary) string relation* is a subset of Docs^k for some $k \in \mathbb{N}$.

[2] As in much of the work on semirings in provenance, e.g. Green et al. [20], we do not yet consider the *difference* operator (which would require additive inverses).

x_{pers}	x_{loc}	annotation
Carter	Plains,␣Georgia	0.9
Washington	Westmoreland,␣Virginia	0.9
Carter	Georgia,␣Washington	0.81
Carter	Westmoreland,␣Virginia	0.59049

x_{pers}	x_{loc}	annotation
$[1, 7\rangle$	$[13, 28\rangle$	0.9
$[30, 40\rangle$	$[46, 68\rangle$	0.9
$[1, 7\rangle$	$[21, 40\rangle$	0.81
$[1, 7\rangle$	$[46, 68\rangle$	0.59049

Figure 1 A document (top), a $(\mathbb{K}, \mathbf{D})$-relation (bottom left), and an extracted annotated span relation (bottom right).

A *span* identifies a substring of a document $\mathbf{d}$ by specifying its bounding indices, that is, a span of $\mathbf{d}$ is an expression of the form $[i, j\rangle$ where $1 \leq i \leq j \leq n + 1$. By $\mathbf{d}_{[i,j\rangle}$ we denote the substring $\sigma_i \cdots \sigma_{j-1}$. In case $i = j$ it holds that $\mathbf{d}_{[i,j\rangle}$ is the empty string, which we denote by ε. We denote by $\mathsf{Spans}(\mathbf{d})$ the set of all possible spans of a document $\mathbf{d}$ and by Spans the set of all possible spans of all possible documents. Since we will be working with relations over spans, we assume that $\mathbf{D}$ is such that $\mathsf{Spans} \subseteq \mathbf{D}$. A $(\mathbb{K}, \mathbf{d})$-*relation* over $V \subseteq \mathsf{Vars}$ is defined analogously to a $(\mathbb{K}, \mathbf{D})$-relation over V but only uses V-tuples with values from $\mathsf{Spans}(\mathbf{d})$.

▶ **Definition 3.1.** *Let $(\mathbb{K}, \oplus, \otimes, \overline{0}, \overline{1})$ be a semiring. A* $\mathbb{K}$-annotator *(or* annotator *for short), is a function S that is associated with a finite set $V \subseteq \mathsf{Vars}$ of variables and maps documents $\mathbf{d}$ into $(\mathbb{K}, \mathbf{d})$-relations over V. We denote V by $\mathsf{Vars}(S)$. We sometimes also refer to an annotator as an* annotator over $(\mathbb{K}, \oplus, \otimes, \overline{0}, \overline{1})$ *when we want to emphasize the semiring.*

Notice that $\mathbb{B}$-annotators, i.e., annotators over $(\mathbb{B}, \vee, \wedge, \mathsf{false}, \mathsf{true})$ are simply the *document spanners* as defined by Fagin et al. [13].

▶ **Example 3.2.** We provide an example document $\mathbf{d}$ in Figure 1 (top). The table at the bottom right depicts a possible $(\mathbb{K}, \mathbf{d})$-relation obtained by a spanner that extracts (person, hometown) pairs from $\mathbf{d}$. Notice that for each span $[i, j\rangle$ occurring in this table, the string $\mathbf{d}_{[i,j\rangle}$ can be found in the table to the left.

In this naïve example, which is just to illustrate the definitions, we used the Viterbi semiring and annotated each tuple with $(0.9)^k$, where k is the number of words between the spans associated to x_{pers} and x_{loc}. The annotations can therefore be interpreted as confidence scores.

Relational Algebra for K-Annotators

We now lift the relational algebra operators on $\mathbb{K}$-relations to the level of $\mathbb{K}$-annotators. For all documents $\mathbf{d}$ and for all annotators S_1 and S_2 associated with V_1 and V_2, respectively, we define the following:

- **Union:** If $V_1 = V_2$ then the union $S \stackrel{\text{def}}{=} S_1 \cup S_2$ is defined by $S(\mathbf{d}) \stackrel{\text{def}}{=} S_1(\mathbf{d}) \cup S_2(\mathbf{d})$.[3]
- **Projection:** For $X \subseteq V_1$, the projection $S \stackrel{\text{def}}{=} \pi_X S_1$ is defined by $S(\mathbf{d}) \stackrel{\text{def}}{=} \pi_X S_1(\mathbf{d})$.
- **Natural Join:** The natural join $S \stackrel{\text{def}}{=} S_1 \bowtie S_2$ is defined by $S(\mathbf{d}) \stackrel{\text{def}}{=} S_1(\mathbf{d}) \bowtie S_2(\mathbf{d})$.

[3] Here, $\cup$ stands for the union of two K-relations as was defined previously. The same is valid also for the other operators.

- **String selection:** Let R be a k-ary string relation. The string-selection operator σ^R is parametrized by k variables $x_1, \ldots, x_k$ in V_1 and may be written as $\sigma^R_{x_1,\ldots,x_k}$. Then the annotator $S \stackrel{\text{def}}{=} \sigma^R_{x_1,\ldots,x_k} S_1$ is defined as $S(\mathbf{d}) \stackrel{\text{def}}{=} \sigma_{\mathbf{P}}(S_1(\mathbf{d}))$ where $\mathbf{P}$ is a selection predicate with $\mathbf{P}(\mathbf{t}) = \overline{1}$ if $(\mathbf{d}_{\mathbf{t}(x_1)}, \ldots, \mathbf{d}_{\mathbf{t}(x_k)}) \in R$; and $\mathbf{P}(\mathbf{t}) = \overline{0}$ otherwise.

Due to Proposition 2.3 it follows that the above operators form an algebra on $\mathbb{K}$-annotators.

4 Weighted Variable-Set Automata

In this section, we define the concept of a *weighted vset-automaton* as a formalism to represent $\mathbb{K}$-annotators. This formalism is the natural generalization of vset-automata [13] and weighted automata [10]. Later in this section, we also present a formalism that is based on parametric factors, and a specification can be translated into a weighted vset-automaton (Section 4.1).

Let $V \in \mathsf{Vars}$ be a finite set of variables. Furthermore, let $\Gamma_V = \{v\vdash, \dashv v \mid v \in V\}$ be the set of *variable operations*.[4] Let $(\mathbb{K}, \oplus, \otimes, \overline{0}, \overline{1})$ be a semiring. A *weighted variable-set automaton over semiring* $\mathbb{K}$ (alternatively, a *weighted vset automaton* or a $\mathbb{K}$*-weighted vset-automaton*) is a tuple $A \stackrel{\text{def}}{=} (V, Q, I, F, \delta)$ where $V \subseteq \mathsf{Vars}$ is a finite set of variables; Q is a finite set of *states*; $I : Q \to \mathbb{K}$ is the *initial weight function*; $F : Q \to \mathbb{K}$ is the *final weight function*; and $\delta : Q \times (\Sigma \cup \{\varepsilon\} \cup \Gamma_V) \times Q \to \mathbb{K}$ is a *($\mathbb{K}$-weighted) transition* function.

We define the *transitions* of A as the set of triples (p, o, q) with $\delta(p, o, q) \neq \overline{0}$. Likewise, the *initial* (resp., *accepting*) states are those states q with $I(q) \neq \overline{0}$ (resp., $F(q) \neq \overline{0}$). A *run* ρ of A over a document $\mathbf{d} \stackrel{\text{def}}{=} d_1 \cdots d_n$ is a sequence

$$(q_0, i_0) \stackrel{o_1}{\to} \cdots (q_{m-1}, i_{m-1}) \stackrel{o_{m-1}}{\to} (q_m, i_m)$$

where

- $i_0 = 1$, $i_m = n + 1$, and $i_j \in \{1, \ldots, n\}$ for each $j \in \{1, \ldots, m-1\}$;
- each o_j is in $\Sigma \cup \{\varepsilon\} \cup \Gamma_V$;
- $i_{j+1} = i_j$ whenever $o_j \in \{\varepsilon\} \cup \Gamma_V$ and $i_{j+1} = i_j + 1$, otherwise;
- $\delta(q_j, o_j, q_{j+1}) \neq \overline{0}$ for all $j \geq 0$.

The *weight* of a run is obtained by $\otimes$-multiplying the weights of its constituent transitions. Formally, the weight w_ρ of ρ is an element in $\mathbb{K}$ given by the expression

$$I(q_0) \otimes \delta(q_0, o_1, q_1) \otimes \cdots \otimes \delta(q_{m-1}, o_{m-1}, q_m) \otimes F(q_m) \ .$$

We call ρ *nonzero* if $\mathsf{w}_\rho \neq \overline{0}$. Notice that ρ is nonzero only if q_0 and q_m are initial and final, respectively. A run is called *valid* if for every variable $v \in V$ the following hold: there is exactly one index i for which $o_i = v\vdash$ and exactly one index $j \geq i$ for which $o_j = \dashv v$.

For a nonzero and valid run ρ, we define $\mathbf{t}_\rho$ as the V-tuple that maps each variable $v \in V$ to the span $[i_j, i_{j'}\rangle$ where $o_{i_j} = v\vdash$ and $o_{i_{j'}} = \dashv v$. We denote the set of all valid and nonzero runs of A on $\mathbf{d}$ by $P(A, \mathbf{d})$. We naturally extend the notion of functionality to apply also to general (not necessarily Boolean) weighted vset-automata. A *weighted functional vset-automaton* is a weighted vset-automaton whose runs are all valid.[5]

Notice that there may be infinitely many nonzero and valid runs of a weighted vset-automaton on a given document, due to *ε-cycles*, which are sets of states $\{q_1, \ldots, q_k\}$ such that $(q_i, \varepsilon, q_{i+1})$ is a transition for every $i \in \{1, \ldots, k-1\}$. Similar to much of the standard

[4] The operation $v\vdash$ represents opening variable v and $\dashv v$ represents closing v.

[5] Notice that, while our notion of functionality indeed generalizes the notion on $\mathbb{B}$-weighted vset-automata [13], one needs positivity of $\mathbb{K}$ to ensure that a functional automaton has an output tuple for every valid run.

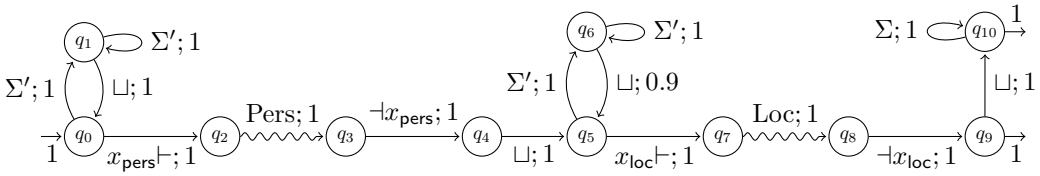

Figure 2 An example weighted vset-automaton over the Viterbi semiring with initial state q_0 and two final states q_9, q_{10}. $\Sigma' = \Sigma \setminus \{\sqcup\}$, Pers and Loc are sub-automata matching person and location names respectively. All edges, including the edges of the sub-automata, have the weight 1 besides the transition from q_6 to q_5 with weight 0.9.

literature on weighted automata (see, e.g., [12]) we will assume that weighted vset-automata do not have ε-cycles, unless mentioned otherwise. The reason for this restriction is that automata with such cycles need $\mathbb{K}$ to be closed under infinite sums for their semantics to be well-defined.[6]

As such, if A does not have ε-cycles, then the result of applying A on a document $\mathbf{d}$, denoted $[\![A]\!]^{\mathbb{K}}(\mathbf{d})$, is the $(\mathbb{K}, \mathbf{d})$-relation R for which

$$R(\mathbf{t}) \stackrel{\text{def}}{=} \bigoplus_{\rho \in P(A,\mathbf{d}) \text{ and } \mathbf{t}=\mathbf{t}_\rho} \mathsf{w}_\rho.$$

Note that we only use runs ρ that are valid and nonzero here. Observe that if $\mathbf{t}$ is a V'-tuple with $V' \neq V$ then $R(\mathbf{t}) = \overline{0}$. In addition, $[\![A]\!]^{\mathbb{K}}$ is well defined since every V-tuple in the support of $[\![A]\!]^{\mathbb{K}}(\mathbf{d})$ is a V-tuple over $\mathsf{Spans}(\mathbf{d})$. The *size* $|A|$ of a weighted vset-automaton A is its number of states plus its number of transitions.

We say that a $\mathbb{K}$-annotator S is *regular* if there exists a weighted vset-automaton A such that $S = [\![A]\!]^{\mathbb{K}}$. Similar to our terminology on annotators, we use the term $\mathbb{B}$-*weighted vset-automata* to refer to the "classical" vset-automata of Fagin et al. [13], which are indeed weighted vset-automata over the Boolean semiring.

▶ **Example 4.1.** Figure 2 shows an example weighted vset-automaton over the Viterbi semiring, which is intended to extract (person, hometown)-tuples from a document. Here, "Pers" and "Loc" should be interpreted as sub-automata that test if a string could be a person name or a location. (Such automata can be compiled from publicly available regular expressions[7] and from deterministic rules and dictionaries as illustrated in SystemT [4].)

The relation extracted by this automaton from the document in Figure 1 is exactly the annotated span relation of the same figure. The weight of a tuple $\mathbf{t}$ depends on the number of spaces occurring between the span captured by x_{pers} and the span captured by x_{loc}. More specifically the automaton assigns the weight $(0.9)^k$ to each tuple, where k is the number of words between the two variables.

4.1 Annotators via Parametric Factors

We now describe another way of introducing weights (or *softness*) in document spanners. This section can also be seen as an additional motivation for $\mathbb{K}$-annotators. Indeed, we will show that, if softness is introduced in document spanners [13] (i.e., $\mathbb{B}$-annotators) in

[6] The semirings need to fulfill additional properties as well such as distributivity, commutativity and associativity must also hold for infinite sums. Such semirings are called *complete* [28].

[7] For example, `http://regexlib.com/`.

the standard manner that we recall here, the resulting annotators can be captured in our framework.

Softness can be introduced in document spanners via the concept of *parametric factors*, which is a very common concept that is used in a wide range of contexts. Examples are the *soft keys* of Jha et al. [21], the *PrDB* model of Sen et al. [41], the *probabilistic unclean databases* of De Sa et al. [36] which can be viewed as a special case of the *Markov Logic Network* (MLN) [35]. Intuitively, a parametric factor is a succinct expression of numerical factors of a probability via weighted rules: whenever the rule *fires*, a corresponding factor (determined by the weight) is added to the product that constitutes the probability. What we want to show in this section is that, if one has rules that involve $\mathbb{B}$-annotators, and one adds uncertainty or softness to these rules in this standard way – using parametric factors – then the obtained formalism naturally leads to $\mathbb{K}$-annotators.

Next, we give the precise definition of a soft spanner and show that, when the factors are regular, a soft spanner can be translated into a weighted vset-automaton.

Formally, a *soft spanner* is a triple $Q = (P, \mathcal{S}, w)$, where:

- P is a document spanner, i.e., a $\mathbb{B}$-annotator,
- $\mathcal{S}$ is a finite set of document spanners referred to as the *factor spanners*, *and*
- $w : \mathcal{S} \to \mathbb{R}$ assigns a (positive or negative) numerical value to each factor spanner.

Given a document $\mathbf{d}$, the soft spanner Q assigns to each $\mathbf{t} \in P(\mathbf{d})$ a probability as follows:

$$\hat{Q}(\mathbf{d}, \mathbf{t}) \stackrel{\text{def}}{=} \exp\left(\sum_{S \in \mathcal{S}} \sum_{\mathbf{u} \in \{\mathbf{t}\} \bowtie S(\mathbf{d})} w(S)\right) = \prod_{S \in \mathcal{S}} e^{w(S) \cdot |\{\mathbf{t}\} \bowtie S(\mathbf{d})|},$$

$$Q(\mathbf{d}, \mathbf{t}) \stackrel{\text{def}}{=} \hat{Q}(\mathbf{d}, \mathbf{t}) / Z(\mathbf{d}),$$

where $Z(\mathbf{d})$ is a normalization factor (or the *partition function*) defined in the usual way:

$$Z(\mathbf{d}) = \sum_{\mathbf{t} \in P(\mathbf{d})} \hat{Q}(\mathbf{d}, \mathbf{t})$$

Note that $\{\mathbf{t}\} \bowtie S(\mathbf{d})$ is the join of the relation $S(\mathbf{d})$ with the relation that consists of the single tuple $\mathbf{t}$. Hence, $|\{\mathbf{t}\} \bowtie S(\mathbf{d})|$ is the number of tuples $\mathbf{t}' \in S(\mathbf{d})$ that are *compatible* (*joinable*) with $\mathbf{t}$, that is, $\mathbf{t}(x) = \mathbf{t}'(x)$ whenever x is in the domain of both $\mathbf{t}$ and $\mathbf{t}'$.

▶ **Example 4.2.** The same relation as discussed in Example 4.1 can also be extracted using a soft spanner $Q = (P, \{S\}, w)$. To this end, P is a boolean spanner extracting (person, hometown)-tuples; S is the spanner, extracting $(x_{\mathsf{pers}}, y, x_{\mathsf{loc}})$-triples of words, where y matches a word between x_{pers} and x_{loc}; and the weight function w is the function assigning $w(S) = \log(0.9)$. Note that S simply extracts words and does not test whether the words matched by x_{pers} or x_{loc} correspond to a person or location.

We therefore see that $\mathbb{K}$-annotators can also be defined by applying the standard technique of parametric factors to document spanners. In fact, as we will see next, soft spanners can be compiled into weighted vset-automata, which serves as an additional motivation for weighted vset-automata. To prove the result, we use closure properties of weighted vset-automata that we will obtain further in the paper (so the proof can be seen as a motivation for the closure- and computational properties of weighted vset-automata as well).

For the following result, we say that a $\mathbb{K}$-weighted vset-automaton A is *unambiguous* if, for every document $\mathbf{d}$ and every tuple $\mathbf{t} \in [\![A]\!]^{\mathbb{K}}(\mathbf{d})$, there exists exactly one valid and nonzero run ρ of A on $\mathbf{d}$ such that $\mathbf{t} = \mathbf{t}_\rho$.

▶ **Theorem 4.3.** *Let $Q = (P, \mathcal{S}, w)$ be a soft spanner such that P and every $S \in \mathcal{S}$ is regular. There exists an $\mathbb{R}$-weighted vset-automaton A such that $[\![A]\!]^{\mathbb{R}}(\mathbf{d})(\mathbf{t}) = \log(\hat{Q}(\mathbf{d}, \mathbf{t}))$ for all documents $\mathbf{d}$ and tuples $\mathbf{t}$; Moreover, if the spanners of Q are represented as unambiguous functional vset-automata, then A can be constructed in polynomial time in the size of Q.*

Proof sketch. Let P_u be an unambiguous version of P, interpreted as an $\mathbb{R}$-weighted vset-automaton where true is associated with 1 and false with 0 and let V_P be the variables of P. Let S_u be an unambiguous version of S. From S_u we compute a weighted vset-automaton S_u^w by interpreting it as an $\mathbb{R}$-weighted vset-automaton and assigning to each accepting state q of S_u the weight $F(q) = w(S)$. Then the automaton we need for computing $\log(\hat{Q}(\mathbf{d}, \mathbf{t}))$ is

$$A = \bigcup_{S \in \mathcal{S}} \pi_{V_P}(P_u \bowtie S_u^w) \ .$$

We show correctness, i.e., $\log(\hat{Q}(\mathbf{d}, \mathbf{t})) = [\![A]\!]^{\mathbb{R}}(\mathbf{d})(\mathbf{t})$. Due to P_u and S_u^w being unambiguous, it follows directly that $P_u \bowtie S_u^w$ has exactly one accepting run with weight $w(S)$ for every tuple $\mathbf{t} \in [\![P_u \bowtie S_u^w]\!]^{\mathbb{R}}(\mathbf{d})$. Per definition of union and projection, it follows that $[\![A]\!]^{\mathbb{R}}(\mathbf{d})(\mathbf{t}) = \sum_{S \in \mathcal{S}} \sum_{\mathbf{u} \in \{\mathbf{t}\} \bowtie S(\mathbf{d})} w(S) = \log(\hat{Q}(\mathbf{d}, \mathbf{t}))$. As we will obtain in Theorem 5.5, automaton A can be represented as an $\mathbb{R}$-weighted vset-automaton and can be constructed in PTIME, which concludes the proof. ◀

5 Fundamental Properties

We now study fundamental properties of annotators. Specifically, we will show that regular annotators are closed under union, projection, and join. Furthermore, annotators over positive semirings are closed under exactly the same string relations as document spanners. We begin the section by showing that every regular $\mathbb{K}$-annotator can be transformed into an equivalent functional regular $\mathbb{K}$-annotator without ε-transitions. We say that two vset-automata A and A' are equivalent if $[\![A]\!]^{\mathbb{K}} = [\![A']\!]^{\mathbb{K}}$.

▶ **Proposition 5.1.** *For every weighted vset-automaton A there is an equivalent weighted vset-automaton A' that has no ε-transitions. This automaton A' can be constructed from A in PTIME. Furthermore, A is functional if and only if A' is functional.*

Notice that non-functional vset-automata can be inconvenient to work with, since some of their nonzero runs are not valid and therefore do not contribute to the weight of a tuple. It is therefore desirable to be able to automatically convert weighted vset-automata into functional weighted vset-automata.

▶ **Proposition 5.2.** *Let A be a weighted vset-automaton. Then there is a functional weighted vset-automaton A_{fun} that is equivalent to A. If A has n states and uses k variables, then A_{fun} can be constructed in time polynomial in n and exponential in k.*

The exponential blow-up in Proposition 5.2 cannot be avoided, since Freydenberger [17, Proposition 3.9] showed that there is a vset-automaton A (over $\mathbb{B}$) with one state and k variables, such that every equivalent functional vset-automaton has at least 3^k states. Functionality of vset-automata can be checked efficiently, as we have the following result.

▶ **Proposition 5.3.** *Given a weighted vset-automaton A with m transitions and k variables, it can be decided whether A is functional in time $O(km)$.*

▶ **Observation 5.4** ((Similar to Freydenberger et al. [18])). Let $A \stackrel{\text{def}}{=} (V, Q, I, F, \delta)$ be a $\mathbb{K}$-weighted functional vset-automaton. Then there exists a function $C : Q \times V \mapsto \{w, o, c\}$ that maps every state to its variable configuration, i.e., $C(q, x) \in \{w, o, c\}$ depending on whether x is *waiting, open,* or *closed* in state q. More formally, the function

$$C(q, v) = \begin{cases} w & \text{there is a nonzero run where } v\vdash \text{ does not occur before reaching } q, \\ o & \text{there is a nonzero run where } v\vdash \text{ but not } \dashv v \text{ occur before reaching } q, \\ c & \text{there is a nonzero run where } v\vdash \text{ and } \dashv v \text{ occur before reaching } q. \end{cases}$$

is well-defined. Indeed, if C would not be well-defined, then two conflicting runs would contradict the functionality of A.

5.1 Closure Under Join, Union, and Projection

Here we obtain the following result.

▶ **Theorem 5.5.** *Regular annotators are closed under union, projection, and natural join.*

Whereas union and projection are fairly standard, the case of join needs some care in the case that the two automata A_1 and A_2 process variable operations in different orders. (I.e., if A_1 processes $x\vdash\, y\vdash\, a\, \dashv y\, \dashv x$ and A_2 processes $y\vdash\, x\vdash\, a\, \dashv x\, \dashv y$, then these two different sequences produce the same result. The automata construction has to deal with this.) One can also show that, if the annotators are given as functional weighted vset-automata, then the construction for a single union, projection, and join can be done in polynomial time. Furthermore, the constructions preserve functionality.

5.2 Closure under String Selection

A k-ary string relation is *recognizable* if it is a finite union of Cartesian products of regular string languages [37]. Let $\mathsf{REG}_{\mathbb{K}}$ be the set of regular $\mathbb{K}$-annotators. We say that a k-ary string relation R is *selectable by regular* $\mathbb{K}$*-annotators* if the following equivalence holds:

$$\mathsf{REG}_{\mathbb{K}} = \{\sigma^{R}_{x_1,\ldots,x_k}(S) \mid S \in \mathsf{REG}_{\mathbb{K}} \text{ and } x_i \in \mathsf{Vars}(S) \text{ for all } 1 \leq i \leq k\}\ ,$$

that is, the class of $\mathbb{K}$-annotators is closed under selection using R. If $\mathbb{K} = \mathbb{B}$, we say that R is *selectable by document spanners*. Fagin et al. [13] proved that a string relation is recognizable *if and only if* it is selectable by document spanners. Here, we generalize this result in the context of weights and annotation. Indeed, it turns out that the equivalence is maintained for all positive semirings.

▶ **Theorem 5.6.** *Let $(\mathbb{K}, \oplus, \otimes, \overline{0}, \overline{1})$ be a positive semiring and R be a string relation. The following are equivalent:*
(1) *R is recognizable.*
(2) *R is selectable by document spanners.*
(3) *R is selectable by $\mathbb{K}$-annotators.*

Proof sketch. The equivalence between (1) and (2) is known [13, Theorem 4.16]. The proof (2) ⇒ (3) is heavily based on the closure properties from Theorem 5.5 and does not use positivity of the semiring. For (3) ⇒ (2) we use semiring morphisms to turn $\mathbb{K}$-weighted vset-automata into $\mathbb{B}$-weighted vset-automata and need positivity of the semiring. ◀

Since the implication from (2) to (3) does not assume positivity of the semiring, it raises the question if the equivalence can be generalized even further. One can show that this is indeed the case, such as for the Łukasiewicz semiring, which is not positive.

6 Evaluation Problems

We consider two types of evaluation problems in this section: *answer testing* and *best weight evaluation*. The former is given an annotator, document $\mathbf{d}$, and tuple $\mathbf{t}$; and computes the annotation of $\mathbf{t}$ in $\mathbf{d}$ according to the annotator. The latter does not receive the tuple as input, but recieves a weight threshold and is asked whether there exists a tuple that is returned with a weight that is at least the threshold.

6.1 Answer Testing

It follows from Freydenberger [17, Lemma 3.1] that answer testing is NP-complete for $\mathbb{B}$-weighted vset-automata in general. Indeed, he showed that, given a $\mathbb{B}$-weighted vset-automaton A, it is NP-complete to check if A returns any output on the empty document ε, so it is even NP-complete to check if the tuple of empty spans is returned or not. However, the proof makes extensive use of non-functionality of the automaton. Indeed, we can prove that answer testing is tractable for functional weighted vset-automata.

▶ **Theorem 6.1.** *Given a functional weighted vset-automaton A, a document $\mathbf{d}$, and a tuple $\mathbf{t}$, the weight $[\![A]\!]^{\mathbb{K}}(\mathbf{d})(\mathbf{t})$ assigned to $\mathbf{t}$ by A on $\mathbf{d}$ can be computed in PTIME.*

Proof sketch. Let A, $\mathbf{d}$, and $\mathbf{t}$ be as stated. Per definition, the weight assigned to $\mathbf{t}$ by A is

$$[\![A]\!]^{\mathbb{K}}(\mathbf{d})(\mathbf{t}) \stackrel{\text{def}}{=} \bigoplus_{\rho \in P(A,\mathbf{d}) \text{ and } \mathbf{t}=\mathbf{t}_\rho} \mathsf{w}_\rho .$$

Therefore, in order to compute the weight $[\![A]\!]^{\mathbb{K}}(\mathbf{d})(\mathbf{t})$, we need to consider the weights of all runs ρ for which $\mathbf{t} = \mathbf{t}_\rho$. Furthermore, multiple runs can select the same tuple $\mathbf{t}$ but assign variables in a different order.[8]

We first define an automaton $A_{\mathbf{t}}$, such that $[\![A_{\mathbf{t}}]\!]^{\mathbb{K}}(\mathbf{d})(\mathbf{t}') = \overline{1}$ if $\mathbf{t} = \mathbf{t}'$ and $[\![A_{\mathbf{t}}]\!]^{\mathbb{K}}(\mathbf{d})(\mathbf{t}') = \overline{0}$ otherwise. Such an automaton $A_{\mathbf{t}}$ can be defined using a chain of $|\mathbf{d}| + 2|V| + 1$ states, which checks that the input document is $\mathbf{d}$ and which has exactly one nonzero run ρ, with $\mathsf{w}_\rho = \overline{1}$ and $\mathbf{t}_\rho = \mathbf{t}$.

By Theorem 5.5 there is a weighted vset-automaton A' such that $[\![A']\!]^{\mathbb{K}} = [\![A \bowtie A_{\mathbf{t}}]\!]^{\mathbb{K}}$. It follows directly from the definition of A' that $[\![A]\!]^{\mathbb{K}}(\mathbf{d})(\mathbf{t}) = [\![A']\!]^{\mathbb{K}}(\mathbf{d})(\mathbf{t})$. Furthermore, all accepting runs $\rho \in P(A', \mathbf{d})$ have length $|\mathbf{d}| + 2|V|$. Therefore, the weight $[\![A']\!]^{\mathbb{K}}(\mathbf{d})(\mathbf{t})$ can be obtained by taking the sum of the weights of all accepting runs of A'. If we assume w.l.o.g. that the states of A' are $\{1, \ldots, n\}$ for some $n \in \mathbb{N}$, then this sum can be computed as

$$[\![A']\!]^{\mathbb{K}}(\mathbf{d})(\mathbf{t}) = v_I \times (M_\delta)^{|\mathbf{d}|+2|V|} \times (v_F)^T ,$$

where

- v_I is the vector $(I(1), \ldots, I(n))$,
- M_δ is the $n \times n$ matrix with $M_\delta(i,j) = \bigoplus_{a \in \Sigma} \delta(i,a,j)$, and
- $(v_F)^T$ is the transpose of vector $v_F = (F(1), \ldots, F(n))$.

Since n is polynomial in the input, this product can be computed in polynomial time. ◀

[8] This may happen when variable operations occur consecutively, i.e., without reading a symbol in between.

6.2 Best Weight Evaluation

In many semirings, the domain is naturally ordered by some relation. For instance, the domain of the probability semiring is $\mathbb{R}^+$, which is ordered by the $\leq$-relation. This motivates evaluation problems where we are interested in some kind of optimization of the weight, which we will look into in this section.

▶ **Definition 6.2** ((Dorste and Kuich [9])). *A commutative monoid* $(\mathbb{K}, \oplus, \overline{0})$ *is* ordered *if it is equipped with a partial order* $\leqslant$ *preserved by the* $\oplus$ *operation. An ordered monoid is* positively ordered *if* $\overline{0} \leqslant a$ *for all* $a \in \mathbb{K}$. *A semiring* $(\mathbb{K}, \oplus, \otimes, \overline{0}, \overline{1})$ *is* (positively) ordered *if the additive monoid is (positively) ordered and multiplication with elements* $\overline{0} \leqslant a$ *preserves the order.*

We consider the following two problems.

Threshold	
Given:	Regular annotator A over an ordered semiring, document $\mathbf{d} \in \mathsf{Docs}$, and a weight $w \in \mathbb{K}$.
Question:	Is there a tuple $\mathbf{t}$ with $w \leqslant [\![A]\!]^{\mathbb{K}}(\mathbf{d})(\mathbf{t})$?

MaxTuple	
Given:	Regular annotator A over an ordered semiring and a document $\mathbf{d} \in \mathsf{Docs}$.
Task:	Compute a tuple with maximal weight, if it exists.

Notice that, if MaxTuple is efficiently solvable, then so is Threshold. We therefore prove upper bounds for MaxTuple and lower bounds for Threshold. The Threshold problem is sometimes also called the *emptiness problem* in the weighted automata literature. It turns out that, for positively ordered semirings that are bipotent (that is, $a \oplus b \in \{a, b\}$), both problems are tractable.

▶ **Theorem 6.3.** *Let* $(\mathbb{K}, \oplus, \otimes, \overline{0}, \overline{1})$ *be a positively ordered, bipotent semiring. Furthermore, let* A *be a functional* $\mathbb{K}$*-weighted vset-automaton and let* $\mathbf{d} \in \mathsf{Docs}$ *be a document. Then* MaxTuple *for* A *and* $\mathbf{d}$ *can be solved in PTIME.*

Proof sketch. Since $a \oplus b \in \{a, b\}$ for every $a, b \in \mathbb{K}$, the weight of a tuple $t \in [\![A]\!]^{\mathbb{K}}(\mathbf{d})$ is always equal to the weight of one of the accepting runs ρ with $\mathbf{t} = \mathbf{t}_\rho$. Thus in order to find the tuple with maximal weight, we need to find the run of A on $\mathbf{d}$ with maximal weight. This boils down to finding a maximal weight path in a DAG, which is obtained by taking a "product" between A and $\mathbf{d}$. ◀

If the semiring is not bipotent, however, the Threshold and MaxTuple problems become intractable quickly.

▶ **Theorem 6.4.** *Let* $(\mathbb{K}, \oplus, \otimes, \overline{0}, \overline{1})$ *be a semiring such that* $\bigoplus_{i=1}^{m} \overline{1}$ *is strictly monotonously increasing for increasing values of* m. *Futhermore let* A *be a functional* $\mathbb{K}$*-weighted vset-automaton, let* $\mathbf{d} \in \mathsf{Docs}$ *be a document, and* $k \in \mathbb{K}$ *be a weight threshold. Then* Threshold *for such inputs is NP-complete.*

Proof sketch. It is obvious that Threshold is in NP, as one can guess a tuple $\mathbf{t}$ and and test in PTIME whether $w \leqslant [\![A]\!]^{\mathbb{K}}(\mathbf{d})(\mathbf{t})$ using Theorem 6.1.

For the NP-hardness, we will reduce from MAX-3SAT. To this end, let $\psi = C_1 \wedge \cdots \wedge C_m$ be a boolean formula in 3CNF over variables $x_1, \ldots, x_n$ such that each clause $C_i = (\ell_{i,1} \vee \ell_{i,2} \vee \ell_{i,3})$ is a disjunction of exactly three literals $\ell_{i,j} \in \{x_c, \neg x_c \mid 1 \leq c \leq n\}$, $1 \leq i \leq k, 1 \leq j \leq 3$.

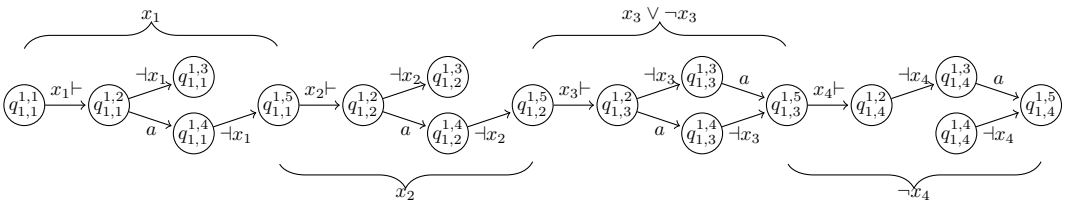

Figure 3 The sub-branch of A_ψ corresponding to C_1 and $x_1 = x_2 = 1, x_4 = 0$.

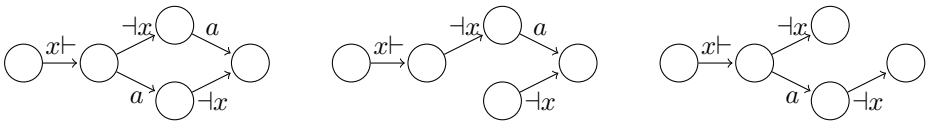

Figure 4 Example gadgets for variable x.

W.l.o.g., we can assume that no clause has two literals corresponding to the same variable. Observe that for each clause C_i there are $2^3 = 8$ assignments of the variables corresponding to the literals of C_i of which exactly 7 satisfy the clause C_i. Formally, let f_{C_i} be the function that maps a variable assignment τ to a number between 1 and 8, depending on the assignments of the literals of the clause C_i. W.l.o.g., we can assume that $f_{C_i}(\tau) = 8$ iff C_i is not satisfied by τ.

We will define a functional weighted automaton automaton A_ψ over the unary alphabet $\Sigma = \{a\}$ such that $[\![A_\psi]\!]^{\mathbb{K}}(a^n)(\mathbf{t}) = \bigoplus_{i=1}^{m} \overline{1}$ if and only if the assignment corresponding to $\mathbf{t}$ satisfies exactly m clauses in ψ and $[\![A_\psi]\!]^{\mathbb{K}}(\mathbf{d}) = \emptyset$ if $\mathbf{d} \neq a^n$.

To this end, each variable x_i of ψ is associated with a corresponding capture variable x_i of A_ψ. We associate a tuple $\mathbf{t}_\tau$ with every assignment τ such that

$$\mathbf{t}_\tau(x_i) = \begin{cases} [i, i\rangle & \text{if } \tau(x_i) = 0, \text{ and} \\ [i, i+1\rangle & \text{if } \tau(x_i) = 1. \end{cases}$$

The automaton $A_\psi \stackrel{\text{def}}{=} (V, Q, I, F, \delta)$ consists of m disjoint branches, where each branch corresponds to a clause of ψ; we call these *clause branches*. Each clause branch is divided into 7 sub-branches, such that a path in the sub-branch j corresponds to a variable assignment τ if $f_{C_i}(\tau) = j$. Thus, each clause branch has exactly one run ρ with weight $\overline{1}$ for each tuple $\mathbf{t}_\tau$ associated to a satisfying assignment τ of C_i.

More formally, the set of states $Q = \{q_{i,j}^{a,b} \mid 1 \leq i \leq m, 1 \leq j \leq n, 1 \leq a \leq 7, 1 \leq b \leq 5\}$ contains $5n$ states for every of the 7 sub-branches of each clause branch. Intuitively, A_ψ has a gadget, consisting of 5 states, for each variable and each of the 7 satisfying assignments of each clause. Figure 4 depicts the three types of gadgets we use here. Note that the weights of the drawn edges are all $\overline{1}$. We use the left gadget if x does not occur in the relevant clause and the middle (resp., right) gadget if the literal $\neg x$ (resp., x) occurs. Furthermore, within the same sub-branch of A_ψ, the last state of each gadget is the same state as the start state of the next variable, i.e., $q_{i,j}^{a,5} = q_{i,j+1}^{a,1}$ for all $1 \leq i \leq k, 1 \leq j < n, 1 \leq a \leq 7$.

We illustrate the crucial part of the construction on an example. Let $\psi = (x_1 \vee \neg x_2 \vee x_4) \wedge (x_2 \vee x_3 \vee x_4)$. The corresponding weighted vset-automaton A_ψ therefore has $14 = 2 \times 7$ disjoint branches. Figure 3 depicts the sub-branch for clause C_1 that corresponds to all assignments with $x_1 = x_2 = 1$ and $x_4 = 0$. ◀

We note that Theorem 6.3 and Theorem 6.4 give us tight bounds for all semirings we defined in Example 2.1.

Since MAX-3SAT is hard to approximate, we can turn Theorem 6.4 into an even stronger inapproximability result for semirings where approximation makes sense. To this end, we focus on semirings that contain $(\mathbb{N}, +, \cdot, 0, 1)$ (as a sub-semiring) in the following result.

▶ **Theorem 6.5.** *Let $\mathbb{K}$ be a semiring that contains $(\mathbb{N}, +, \cdot, 0, 1)$ and let A be a weighted vset-automaton over $\mathbb{K}$. Unless PTIME = NP, there is no algorithm that approximates the tuple with the best weight within a sub-exponential factor in PTIME.*

7 Enumeration Problems

In this section we consider computing the output of annotators from the perspective of enumeration problems, where we try to enumerate all tuples with nonzero weight, possibly from large to small. Such problems are highly relevant for (variants of) vset-automata, as witnessed by the recent literature on the topic [2,15]. We assume familiarity with terminology in enumeration algorithms such as *preprocessing time* and *delay*. If the order of the answers does not matter and the semiring is positive, we can guarantee a constant delay enumeration algorithm with linear preprocessing time.

▶ **Theorem 7.1.** *Given a weighted functional vset-automaton A over a positive semiring $\mathbb{K}$, and a document $\mathbf{d}$, the $\mathbb{K}$-Relation $[\![A]\!]^{\mathbb{K}}(\mathbf{d})$ can be enumerated with preprocessing linear in $|\mathbf{d}|$ and polynomial in $|A|$ and delay constant in $|\mathbf{d}|$ and polynomial in $|A|$.*

Note that the proof of the theorem essentially requires to go through the entire proof of the main result of Amarilli et al. [2, Theorem 1.1].

We now consider cases in which answers are required to arrive in a certain ordering.

Ranked Annotator Enumeration (RA-Enum)	
Given:	Regular functional annotator A over an ordered semiring $(\mathbb{K}, \oplus, \otimes, \overline{0}, \overline{1})$ and a document $\mathbf{d}$.
Task:	Enumerate all tuples $\mathbf{t} \in [\![A]\!]^{\mathbb{K}}(\mathbf{d})$ in descending order on $\mathbb{K}$.

▶ **Theorem 7.2.** *Let $(\mathbb{K}, \oplus, \otimes, \overline{0}, \overline{1})$ be an positively ordered, bipotent semiring, let A be a functional $\mathbb{K}$-weighted vset-automaton, and let $\mathbf{d} \in \mathsf{Docs}$ be a document. Then RA-Enum can be solved with polynomial delay and preprocessing.*

Proof sketch. Our algorithm is a slight adaptation of Yen's algorithm [48]. To this end, we will use the DAG we defined in the proof of Theorem 6.3, but invest a bit more preprocessing. In particular, we change the DAG so that it has a one-to-one correspondence between output tuples and some of its paths. Using this correspondence, we can then revert to Yen's algorithm for enumerating simple paths in graphs. ◀

8 Concluding Remarks

We embarked on a study that incorporates *annotations* or *weights* in information extraction and propose $\mathbb{K}$-annotators as a candidate formalism to study this problem. The $\mathbb{K}$-annotators can be instantiated with *weighted vset-automata*, thereby obtaining *regular $\mathbb{K}$-annotators*, which are powerful enough to capture the extension of the traditional spanner framework with parametric factors. Furthermore, the regular $\mathbb{K}$-annotators have favorable closure properties, such as closure under union, projection, natural join, and string selection using regular

relations. The first complexity results on evaluation problems are encouraging: answer testing is tractable and, depending on the semiring, problems such as the threshold problem, the max tuple problem, and enumeration of answers are tractable too.

We note that the addition of weights to vset-automata also introduces new challenges. For instance, some typical questions that we study in database theory are not yet fully understood for weighted automata, which are the basis of weighted vset-automata. Examples are equivalence and emptiness. Concerning equivalence, one can show that equivalence is undecidable for weighted vset-automata over the tropical semiring, using techniques from Krob [23] or Almagor et al. [1]. In general, however, it is not completely clear for which semirings equivalence is decidable or not.

The emptiness problem that is usually studied in the weighted automata literature does not ask if there exists a document $\mathbf{d}$ such that the automaton returns at least one tuple with nonzero weight on $\mathbf{d}$, but is additionally given a threshold (as in our Threshold problem) and asks if the automaton returns a tuple with at least the threshold weight (which requires an order on the semiring). It is not yet clear how much this threshold influences the complexity of the problem.

An additional challenge is that *determinization* of weighted automata is a complex matter and not always possible. It is well-known to be possible for the Boolean semiring but, for the tropical semiring, i.e., $(\mathbb{R} \cup \{-\infty\}, \max, +, -\infty, 0)$, deterministic weighted automata are strictly less expressive than unambiguous weighted automata, which are strictly less expressive than general weighted automata, cf. Klimann et al. [22].

A possible direction for further exploration could be the study of annotators which use regular cost functions (cf. Colcombet [6]) instead of weighted automata. Since regular cost functions are restructed to the domain of the natural numbers, this would probably be most interesting in the case where the semiring domain is (a subset of) the natural numbers. Indeed, in this case, it is known that regular cost functions are strictly more expressive than weighted automata over the tropical semiring (cf. Colcombet et al. [7]) and therefore could provide a useful tool to annotate document spanners. On the other hand, it is not yet clear to us how to associate regular cost functions in a natural way to annotated relations, which require semirings.

References

1 Shaull Almagor, Udi Boker, and Orna Kupferman. What's Decidable about Weighted Automata? In *Automated Technology for Verification and Analysis*, pages 482–491, 2011.

2 Antoine Amarilli, Pierre Bourhis, Stefan Mengel, and Matthias Niewerth. Constant-Delay Enumeration for Nondeterministic Document Spanners. In *ICDT*, pages 22:1–22:19, 2019.

3 Marcelo Arenas, Luis Alberto Croquevielle, Rajesh Jayaram, and Cristian Riveros. Efficient Logspace Classes for Enumeration, Counting, and Uniform Generation. In *PODS*, pages 59–73, 2019.

4 Laura Chiticariu, Rajasekar Krishnamurthy, Yunyao Li, Sriram Raghavan, Frederick Reiss, and Shivakumar Vaithyanathan. SystemT: An Algebraic Approach to Declarative Information Extraction. In *ACL*, pages 128–137, 2010. URL: `http://www.aclweb.org/anthology/P10-1014`.

5 Jason P. C. Chiu and Eric Nichols. Named Entity Recognition with Bidirectional LSTM-CNNs. *TACL*, 4:357–370, 2016.

6 T. Colcombet. Logic and regular cost functions. In *LICS*, pages 1–4, 2017. `doi:10.1109/LICS.2017.8005061`.

7 Thomas Colcombet, Denis Kuperberg, Amaldev Manuel, and Szymon Torunczyk. Cost Functions Definable by Min/Max Automata. In *STACS)*, volume 47, pages 29:1–29:13, 2016. `doi:10.4230/LIPIcs.STACS.2016.29`.

8 Johannes Doleschal, Benny Kimelfeld, Wim Martens, Yoav Nahshon, and Frank Neven. Split-Correctness in Information Extraction. In *PODS*, pages 149–163, 2019. doi:10.1145/3294052.3319684.

9 Manfred Droste and Werner Kuich. *Semirings and Formal Power Series*, pages 3–28. Springer Berlin Heidelberg, 2009. doi:10.1007/978-3-642-01492-5_1.

10 Manfred Droste, Werner Kuich, and Heiko Vogler. *Handbook of Weighted Automata*. Springer Publishing Company, Incorporated, 1st edition, 2009.

11 Samuel Eilenberg. *Automata, Languages, and Machines*. Academic Press, Inc., Orlando, FL, USA, 1974.

12 Zoltán Ésik and Werner Kuich. *Finite Automata*, pages 69–104. Springer Berlin Heidelberg, 2009. doi:10.1007/978-3-642-01492-5_3.

13 Ronald Fagin, Benny Kimelfeld, Frederick Reiss, and Stijn Vansummeren. Document Spanners: A Formal Approach to Information Extraction. *J. ACM*, 62(2):12, 2015. doi:10.1145/2699442.

14 Ronald Fagin, Benny Kimelfeld, Frederick Reiss, and Stijn Vansummeren. Declarative Cleaning of Inconsistencies in Information Extraction. *ACM Trans. Database Syst.*, 41(1):6:1–6:44, 2016. doi:10.1145/2877202.

15 Fernando Florenzano, Cristian Riveros, Martín Ugarte, Stijn Vansummeren, and Domagoj Vrgoc. Constant Delay Algorithms for Regular Document Spanners. In *PODS*, pages 165–177, 2018.

16 J. Nathan Foster, Todd J. Green, and Val Tannen. Annotated XML: queries and provenance. In *PODS*, pages 271–280, 2008.

17 Dominik D. Freydenberger. A Logic for Document Spanners. *Theory Comput. Syst.*, 63(7):1679–1754, 2019.

18 Dominik D. Freydenberger, Benny Kimelfeld, and Liat Peterfreund. Joining Extractions of Regular Expressions. In *PODS*, pages 137–149, 2018.

19 Joshua Goodman. Semiring Parsing. *Computational Linguistics*, 25(4):573–605, 1999.

20 Todd J. Green, Gregory Karvounarakis, and Val Tannen. Provenance semirings. In *Proceedings of the Twenty-Sixth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, June 11-13, 2007, Beijing, China*, pages 31–40, 2007. doi:10.1145/1265530.1265535.

21 Abhay Kumar Jha, Vibhor Rastogi, and Dan Suciu. Query evaluation with soft-key constraints. In *PODS*, pages 119–128, 2008.

22 Ines Klimann, Sylvain Lombardy, Jean Mairesse, and Christophe Prieur. Deciding unambiguity and sequentiality from a finitely ambiguous max-plus automaton. *Theoretical Computer Science*, 327(3):349–373, 2004. doi:10.1016/j.tcs.2004.02.049.

23 Daniel Krob. The Equality Problem for Rational Series with Multiplicities in the tropical Semiring is Undecidable. *IJAC*, 4(3):405–426, 1994. doi:10.1142/S0218196794000063.

24 Yaoyong Li, Kalina Bontcheva, and Hamish Cunningham. SVM based learning system for information extraction. In *Deterministic and Statistical Methods in Machine Learning*, volume 3635 of *Lecture Notes in Computer Science*, pages 319–339, 2004.

25 Francisco Maturana, Cristian Riveros, and Domagoj Vrgoc. Document Spanners for Extracting Incomplete Information: Expressiveness and Complexity. In *PODS*, pages 125–136, 2018.

26 Franz Mayr and Sergio Yovine. Regular Inference on Artificial Neural Networks. In *CD-MAKE*, volume 11015 of *Lecture Notes in Computer Science*, pages 350–369, 2018.

27 Joshua J. Michalenko, Ameesh Shah, Abhinav Verma, Richard G. Baraniuk, Swarat Chaudhuri, and Ankit B. Patel. Representing Formal Languages: A Comparison Between Finite Automata and Recurrent Neural Networks. In *ICLR (Poster)*, 2019.

28 Mehryar Mohri. *Weighted Automata Algorithms*, pages 213–254. Springer Berlin Heidelberg, 2009. doi:10.1007/978-3-642-01492-5_6.

29 Nanyun Peng, Hoifung Poon, Chris Quirk, Kristina Toutanova, and Wen-tau Yih. Cross-Sentence N-ary Relation Extraction with Graph LSTMs. *TACL*, 5:101–115, 2017.

30 Liat Peterfreund, Dominik D. Freydenberger, Benny Kimelfeld, and Markus Kröll. Complexity Bounds for Relational Algebra over Document Spanners. In *PODS*, pages 320–334. ACM, 2019.

31 Liat Peterfreund, Balder ten Cate, Ronald Fagin, and Benny Kimelfeld. Recursive Programs for Document Spanners. In *ICDT*, volume 127, pages 13:1–13:18, 2019.

32 Hoifung Poon and Pedro M. Domingos. Joint Inference in Information Extraction. In *AAAI*, pages 913–918, 2007.

33 Michael O. Rabin. Probabilistic automata. *Information and Control*, 6(3):230–245, 1963. doi:10.1016/S0019-9958(63)90290-0.

34 Alexander Ratner, Stephen H. Bach, Henry R. Ehrenberg, Jason Alan Fries, Sen Wu, and Christopher Ré. Snorkel: Rapid Training Data Creation with Weak Supervision. *PVLDB*, 11(3):269–282, 2017.

35 Matthew Richardson and Pedro Domingos. Markov Logic Networks. *Mach. Learn.*, 62(1-2):107–136, 2006. doi:10.1007/s10994-006-5833-1.

36 Christopher De Sa, Ihab F. Ilyas, Benny Kimelfeld, Christopher Ré, and Theodoros Rekatsinas. A Formal Framework for Probabilistic Unclean Databases. In *ICDT*, volume 127, pages 6:1–6:18, 2019.

37 Jacques Sakarovitch. *Elements of Automata Theory*. Cambridge University Press, 2009. doi:10.1017/CBO9781139195218.

38 Sunita Sarawagi. Information Extraction. *Foundations and Trends in Databases*, 1(3):261–377, 2008. doi:10.1561/1900000003.

39 Roy Schwartz, Sam Thomson, and Noah A. Smith. Bridging CNNs, RNNs, and Weighted Finite-State Machines. In *ACL*, pages 295–305, 2018.

40 Roberto Segala. Probability and Nondeterminism in Operational Models of Concurrency. In *CONCUR*, pages 64–78, 2006.

41 Prithviraj Sen, Amol Deshpande, and Lise Getoor. PrDB: managing and exploiting rich correlations in probabilistic databases. *VLDB J.*, 18(5):1065–1090, 2009.

42 Warren Shen, AnHai Doan, Jeffrey F. Naughton, and Raghu Ramakrishnan. Declarative Information Extraction Using Datalog with Embedded Extraction Predicates. In *VLDB*, pages 1033–1044, 2007. URL: http://www.vldb.org/conf/2007/papers/research/p1033-shen.pdf.

43 Jaeho Shin, Sen Wu, Feiran Wang, Christopher De Sa, Ce Zhang, and Christopher Ré. Incremental Knowledge Base Construction Using DeepDive. *PVLDB*, 8(11):1310–1321, 2015. URL: http://www.vldb.org/pvldb/vol8/p1310-shin.pdf.

44 Charles A. Sutton and Andrew McCallum. An Introduction to Conditional Random Fields. *Foundations and Trends in Machine Learning*, 4(4):267–373, 2012.

45 David Torrents, Mikita Suyama, Evgeny Zdobnov, and Peer Bork. A genome-wide survey of human pseudogenes. *Genome research*, 13(12):2559–2567, 2003.

46 Lusheng Wang and Tao Jiang. On the Complexity of Multiple Sequence Alignment. *Journal of Computational Biology*, 1(4):337–348, 1994.

47 Gail Weiss, Yoav Goldberg, and Eran Yahav. Extracting Automata from Recurrent Neural Networks Using Queries and Counterexamples. In *ICML*, volume 80, pages 5244–5253, 2018.

48 Jin Y. Yen. Finding the k Shortest Loopless Paths in a Network. *Management Science*, 17(11):712–716, 1971. URL: http://www.jstor.org/stable/2629312.