The Formal Language Theory Column

by

Giovanni Pighizzini

Dipartimento di Informatica
Università degli Studi di Milano
20133 Milano, Italy
pighizzini@di.unimi.it

A Short Note on the Mutually Beneficial Relationship Between Information Extraction in Database Theory and Classical Formal Language Theory

Markus L. Schmid
Humboldt Universität zu Berlin, Berlin, Germany
MLSchmid@MLSchmid.de

Abstract

This short note highlights the relationship between a recent research area in database theory, namely the information extraction framework of document spanners, and classical formal language theory.

1 Formal Languages in Other Areas of TCS

Formal language theory is one of the main research areas of theoretical computer science, but its importance and relevance has changed (although not declined) over time. For example, Hopcroft, Motwani and Ullman state in their famous textbook [25] that "in 1979, automata and language theory was still an area of active research", while "today, there is little direct research in automata theory", where *today* refers to 2007. This assessment by Hopcroft, Motwani and Ullman, which they gave mainly as a justification for changing the focus and content of their textbook, has offended some of the purists among the formal languages researchers.

However, for young researchers in formal language theory who try to build scientific careers in the current landscape of theoretical computer science, it is perhaps a good idea to be less emotional and more pragmatic about these things. Neither abandoning formal languages altogether nor pretending that we are still in the 60s or 70s seems the right way to go. In this regard, it is interesting to note that Hopcroft, Motwani and Ullman also say in their preface that there is little direct research in automata theory "*as opposed to its applications*". In general, the preface of their book stresses the fact that the field of formal languages has grown

from something quite special into standard content of undergraduate computer science courses, which is a good thing.

I do not want to take sides in the controversy of whether formal languages have become less important nowadays in research and teaching. But compared to roughly 60 years ago, it seems that formal language theory exists now less as a coherent and well-defined research area, but instead is scattered throughout the various sub-areas of theoretical computer science. This can be a great opportunity, since it means that various TCS-communities are interested in formal languages, whether they realise it or not. For researchers in formal languages, it is therefore beneficial to discover those "hidden gems" of formal language theory that can be found in seemingly unconnected areas of theoretical computer science.

One such area is information extraction in database theory, which shall be discussed in the next section. Note that it is not the purpose of this article to provide a survey of the field of information extraction, but rather to explain the connections between this topic and the area of formal languages.

2 A Formal Framework for Information Extraction

The concept of information extraction in database theory and the framework of document spanners is surveyed in more detail in [37, 2, 34]. Here, we focus on the relation between this area and classical formal language theory.

Document spanners (or simply spanners) have been introduced in the seminal paper [13], and they constitute a framework for extracting information from texts (i. e., strings, sequences or words, or, as is the common term in the data management community, documents); it is therefore called an *information extraction framework*. Since its introduction, many papers in database theory have been published that are concerned with document spanners (see [13, 18, 31, 3, 27, 14, 32, 15, 16, 19, 20, 29, 30, 12, 35, 36, 38, 28, 10, 8, 11, 4]). From a formal languages point of view, this framework is appealing, since it is a query mechanism that solely works on textual data.

A document spanner (over a set $\mathcal{X}$ of variables)[1] is a function that maps a word w to a finite table with a column for each variable from $\mathcal{X}$ and the entries of the cells of the table are just pairs of positions of w. This can be illustrated as follows:

$w = \mathsf{abbabccabc} \quad \Longrightarrow$

x	y	z
$(2,5)$	$(4,7)$	$(1,10)$
$(3,5)$	$(5,8)$	$(4,7)$
$(1,3)$	$(3,10)$	$(2,4)$
$\vdots$	$\vdots$	$\vdots$

[1] Let us fix $\mathcal{X} = \{\mathsf{x}, \mathsf{y}, \mathsf{z}\}$ in our examples.

The pairs (i, j) are called *spans* and are interpreted as pointers to factors of w, e. g., $(4, 7)$ refers to abcc, since this is the factor that starts at position 4 and ends at position 7. A row of the table is called a *span tuple*, the whole table is called a *span relation* and, as already mentioned above, a function S that maps each $w \in \Sigma^*$ to a span relation $S(w)$ is called a *document spanner*.[2] Regarding the question why spanners are a good formalisation of relevant information extraction tasks, the reader is referred to the surveys [37, 2, 34], or the introductions of the papers mentioned above.

A spanner S over variables $\mathcal{X}$ is therefore a function from Σ^* to the set of span relations over $\mathcal{X}$, but it can conveniently be represented as a formal language over the alphabet $\Sigma \cup \{{}^{x}\triangleright, \triangleleft^{x} \mid x \in \mathcal{X}\}$: We simply encode every span tuple t of the span relation $S(w)$ by enclosing the factor of w that corresponds to the span of $x \in \mathcal{X}$ by the special marker symbols ${}^{x}\triangleright$ and $\triangleleft^{x}$ (we may think of these markers as pairs of brackets). For example, marking in abbabccabc the factor from position 2 to position 5 with ${}^{x}\triangleright$ and $\triangleleft^{x}$, the factor from position 4 to position 7 with ${}^{y}\triangleright$ and $\triangleleft^{y}$ and so on yields the string ${}^{z}\triangleright\,a\,{}^{x}\triangleright\,bb\,{}^{y}\triangleright\,ab\,\triangleleft^{x}\,cc\,\triangleleft^{y}\,abc\triangleleft^{z}$, which encodes the span tuple $((2, 5), (4, 7), (1, 10))$ of the span relation $S(\mathtt{abbabccabc})$. We shall denote such marked strings as *subword-marked words*.[3]

In this way, any span relation $S(w)$ yields a finite set of subword-marked words and joining all these sets of subword-marked words yields a (usually infinite) language of subword-marked words that represents the spanner. Conversely, any language L of subword-marked words uniquely describes a spanner $[\![L]\!]$: For any word $w \in \Sigma^*$, the span relation $[\![L]\!](w)$ simply contains all span tuples described by some subword-marked word from L whose Σ-part equals w. For example, having ${}^{x}\triangleright\ {}^{y}\triangleright\,a\,\triangleleft^{y}\,b\,\triangleleft^{x}\,ab\,{}^{z}\triangleright\,c\triangleleft^{z}$ in L simply means that $[\![L]\!](\mathtt{ababc})$ contains the span tuple $((1, 2), (1, 1), (5, 5))$.[4]

A consequence of this relation between spanners and subword-marked languages is that spanners can be represented by formal language description mechanisms, like automata, expressions, grammars, or just any decision algorithm that accepts a subword-marked language. In particular, we can use those language description mechanisms as tools for specifying spanners as well as algorithmic tools for dealing with them computationally. For example, a spanner that produces a table of all pairs of a non-empty unary factor over the symbol a and a non-empty

[2] Here and henceforth in this article, we use Σ as our underlying terminal alphabet.

[3] The term *ref-word* is also common in the literature on spanners.

[4] We ignore the fact here that the representation of a spanner by means of a subword-marked language is not unique, since consecutive occurrences of marker symbols can be reordered without changing the described spanner. For example, ${}^{x}\triangleright\ {}^{y}\triangleright\,a\,\triangleleft^{y}\,b\,\triangleleft^{x}\,ab\,{}^{z}\triangleright\,c\triangleleft^{z}$ and ${}^{y}\triangleright\ {}^{x}\triangleright\,a\,\triangleleft^{y}\,b\,\triangleleft^{x}\,ab\,{}^{z}\triangleright\,c\triangleleft^{z}$ are different subword-marked words that nevertheless encode the same string and span tuple. This is a bit annoying and can cause some difficulties, but this aspect is thoroughly discussed and addressed in the literature on document spanners.

unary factor over the symbol b can be specified as the subword-marked language $\{u \vartriangleright^x c^n \vartriangleleft^x v \vartriangleright^y d^m \vartriangleleft^y w \mid u, v, w \in \Sigma^*, n, m \geq 1, \{c, d\} = \{\mathsf{a}, \mathsf{b}\}\}$. The subword-marked language $\{u \vartriangleright^x c^n \vartriangleleft^x v \vartriangleright^y d^n \vartriangleleft^y w \mid u, v, w \in \Sigma^*, n \geq 1, \{c, d\} = \{\mathsf{a}, \mathsf{b}\}\}$ describes the variant of the spanner, where the two factors must have the same length.

It is probably not surprising that *regular* spanners, i. e., those spanners that can be represented by regular subword-marked languages, play a central role in the area of document spanners. Such spanners can be represented by regular expressions like $\mathsf{a} \vartriangleright^x (\mathsf{b} + \mathsf{c})^* \, \mathsf{a} \vartriangleleft^x \mathsf{b}^* \vartriangleright^y \mathsf{c}^* \vartriangleleft^y \mathsf{c}^*$ (which is very convenient on the specification level) and as finite automata (which is useful for algorithms). In fact, regular spanners exhibit many desirable algorithmic properties, which follow from the good algorithmic properties of regular languages. On the other hand, the spanner mentioned above that extracts pairs of unary factors of the same size is obviously not a regular spanner (by the typical pumping argument). Obviously, we can consider arbitrary classes of subword-marked languages, which then describe classes of document spanners. An obvious choice that comes to mind are context-free document spanners, which have been investigated in the literature on information extraction (see [30, 29, 4]).

2.1 Research Tasks Motivated by Document Spanners

From a purely formal language theoretical point of view, it is interesting that document spanners – a recent hot topic in foundational database research – can be phrased in terms of classical (regular) languages. This means that we can use the known classical results about regular languages for regular spanners (and for their possible implementations). Probably more interesting is the fact that this information extraction perspective towards regular languages has also lead to new questions and computational problems about finite automata and formal languages in general that have not yet been addressed by the formal languages research of the last six decades. Let us now discuss some of these research questions.

Enumerating the Span Tuples: In the context of information extraction, it is an important task to enumerate for a given word w and a regular spanner S (given as finite automaton) all span tuples of $S(w)$ without repetition. Ideally, such an algorithm has a preprocessing phase that is linear in $|w|$ and then provides an enumeration with a delay (i. e., the time between two consecutive output elements) that is independent from $|w|$ (although it can still depend on the automaton for S). These requirements are justified by the practically relevant assumption that w represents data and is therefore very large, while S is a human-readable query and is therefore in comparison rather small. So these time bounds can be considered to be the optimum (assuming that we have to read the data at least once).

This is a problem on classical finite automata, but it has not yet been consid-

ered in the research of automata theory. Note that this problem is quite different from enumerating the words of a regular language, which has been investigated in formal language theory (see, e. g., [1]). It is also different from enumerating all accepting runs of an automaton on a given input word (i. e., paths between two nodes in a DAG), since such runs have length $|w|$, which would result in a prohibitively large delay. What we actually have to do is the following: For a given word $w \in \Sigma^*$, we have to consider all marked version of w (i. e., subword-marked words with a Σ-portion that equals w) that are accepted by the automaton, but for these marked versions, we only want to produce the marked positions as output (in the right format of course). Another way of phrasing this is in terms of finite transducers: We get a transducer that maps a string to a marked version of it (i. e., we have a finite set Γ of markers and the transducer has the choice of either marking an input symbol $\mathtt{a} \in \Sigma$ as $\mathtt{a}_\gamma$ for some $\gamma \in \Gamma$, or to leave it unmarked). Then, for a given input, we want to enumerate all possible marked outputs, but these outputs are to be represented only by the marked symbols along with their positions in the input string.

In any case, we are dealing with an algorithmic problem about classical finite automata, that is highly relevant in the context of document spanners, but does not seem to be covered by the classical studies on finite automata. An important particularity is the requirement that after the preprocessing we cannot spend time between two outputs that depends on the size of the input or the size of the accepting run.

As a main result in the field of document spanners, it has been shown that we can solve this enumeration problem with preprocessing linear in $|w|$ and delay independent of $|w|$ (see [3, 2, 14]).

An obvious next step is the question whether we can achieve the same for larger classes of spanners. Recall the spanner from above that is represented by the subword-marked language $\{u \vdash^{x} c^n \dashv^{x} v \vdash^{y} d^n \dashv^{y} w \mid u, v, w \in \Sigma^*, n \geq 1, \{c, d\} = \{\mathtt{a}, \mathtt{b}\}\}$. It is not difficult to see that this language is context-free. So can we also enumerate the span tuples of span relations extracted by context-free spanners (i. e., the class of spanners that can be described by context-free subword-marked languages)? It has been shown in [4] that for unambiguous grammars this is indeed possible with preprocessing that is cubic in $|w|$ and a delay that is independent of $|w|$ (observe that context-free parsing reduces to this problem, so the cubic bound seems to be necessary at least for combinatorial algorithms).

String Equality Selection and Angluin's Pattern Languages: In the literature on document spanners, a string equality selection operator has been introduced that simply removes those span tuples from a span relation for which certain spans *do not* refer to equal factors. It is used on top of a regular spanner to describe spanners that are non-regular due to equality checking of factors. For example,

in this way we can describe spanners of the form $\{u\,\mathtt{a}\,{}^{x}\!\triangleright\, v \triangleleft^{x} u'\,\mathtt{b}\,{}^{y}\!\triangleright\, v \triangleleft^{y} u'' \mid u, u', u'', v \in \Sigma^*\}$, so a spanner that extracts two different occurrences of the same factor, the first one directly preceded by symbol a and the second one directly preceded by symbol b.

This string equality selection operator does not serve the mere purpose of playing around with the concept of document spanners, it is in fact already included in the original paper [13] and vital for covering the core of IBM's SystemT, which is the practical counterpart of the theoretical concept of document spanners.

What is interesting from a formal languages point of view is that the string equality selection operator turns spanners into a model that covers Angluin's pattern languages [6] and that is also related to regular expressions with backreferences [9, 33, 17]. Recall that Angluin's patterns are strings of the form xabxaaay, where x and y are variables that can be substituted with arbitrary strings over Σ such that this pattern describes the pattern language $\{u\mathtt{ab}u\mathtt{aaa}v \mid u, v \in \Sigma^*\}$. Regular expressions with backreferences are regular expressions that can enclose a subexpression with special brackets ${}^{x}\!\triangleright\, r \triangleleft^{x}$ and then use variable x as a repetition of whatever was matched to r, e. g., $\mathtt{a}^* {}^{x}\!\triangleright\, (\mathtt{a} + \mathtt{b})^* \triangleleft^{x} \mathtt{c}^*(\mathtt{x}\,\mathtt{a} + (\mathtt{b}\,\mathtt{x})^*)$ is a regular expression with backreferences.

As a result of these connections to established classes of formal languages, many lower bounds for regular spanners extended with string equality selection can be obtained directly from known results, or by extending techniques used for pattern languages and regular expressions with backreferences (see [16]). Moreover, techniques previously used for regular expressions with backreferences have been applied to define a subclass of regular spanners with string equality selection that exhibits better decidability and complexity (see [35]).

Document Spanners on Compressed Input Strings: When dealing with strings, it is a classical and practically highly relevant question whether we can also handle the case where the input strings are compressed by the lossless compression scheme of so-called *straight-line programs* or SLPs for short (i. e., strings compressed by a context-free grammar which can derive only this string).

It is a comparatively simple observation that checking $w \in L(M)$ for an NFA M and a word w that is represented by an SLP G can be done in time linear in $|G|$ (and cubic in $|M|$). However, in terms of document spanners, this problems presents itself in a new light, i. e., as the extension to the more difficult (and arguably more interesting) problem of how to enumerate all span tuples of $S(w)$ for some regular spanner S in the case that w is given as an SLP G. It turns out that we can do this with a preprocessing that is only linear in $|G|$ and a delay that still only depends on $|M|$ (see [36, 28]). On the one hand, this result is a contribution to the field of SLP-compressed algorithmics (further demonstrating its usefulness), while, on the other hand, it further demonstrates the practical relevance of regular spanners.

In query evaluation, the dynamic setting is quite important: If we have enumerated the answers to a query over some data and then update our data by only changing a small portion of it, can we exploit this redundancy somehow such that when we want to enumerate our query again, we do not have to completely re-run the preprocessing, but can perform substantially faster? In the SLP-compressed setting this leads to the question of how to handle updates of SLP-compressed strings, which is an aspect mostly neglected in the classical literature on algorithmics on SLP-compressed strings. It has been shown in [38] that enumeration of regular spanners in the SLP-compressed setting extends well to the dynamic case.

Weighted Document Spanners: Another classical field of formal languages that also has a natural equivalent in document spanners is that of weighted automata. In the weighted automata setting, transitions are equipped with weights from an algebraic structure, which then extends automata from decision procedures to algorithms that produce quantitative outputs.

In the context of information extraction, representing spanners by weighted automata allows to allocate weights to the single span tuples of the span relation that is extracted by the spanner from an input word. In a database context, this is quite helpful, since it can be assumed that not all of the (potentially exponentially many) span tuples are equally relevant. Instead, it is likely that some span tuples are more interesting than others and it is desirable if the more important tuples appear first in an enumeration. In fact, an important motivation of the whole enumeration point of view is that the enumeration starts rather fast and can then be stopped by the user as soon as enough relevant outputs have been received. This point of view makes even more sense if the outputs of the enumeration are guaranteed to be produced in decreasing order with respect to their importance.

Such weighted settings of regular spanners and respective enumeration algorithms have been investigated in [12, 10, 8].

Other Areas in Database Theory Connected to Formal Languages: Despite the fact that many papers about document spanners have been published over the last decade (mostly in conferences ICDT and PODS), information extraction is still an active research area within the field of database theory.

Regular expressions (and therefore finite automata) are also a central tool for query classes for graph databases (see, e. g., [7, 5, 39]).

Typical approaches in the field of complex event processing (see [24, 26] for surveys) are often automaton-based (see, e. g., [23, 22, 21]).

3 Conclusions

Information extraction in database theory is a good example of how classical results of formal language theory can play a central role in new and emerging research topics. In particular, one can observe that document spanners is not just an application of established results, but rather poses new research questions about formal languages that are not covered by the existing literature.

References

[1] Margareta Ackerman and Jeffrey Shallit. Efficient enumeration of words in regular languages. *Theoretical Computer Science*, 410(37):3461–3470, 2009.

[2] Antoine Amarilli, Pierre Bourhis, Stefan Mengel, and Matthias Niewerth. Constant-delay enumeration for nondeterministic document spanners. *SIGMOD Rec.*, 49(1):25–32, 2020.

[3] Antoine Amarilli, Pierre Bourhis, Stefan Mengel, and Matthias Niewerth. Constant-delay enumeration for nondeterministic document spanners. *ACM Trans. Database Syst.*, 46(1):2:1–2:30, 2021.

[4] Antoine Amarilli, Louis Jachiet, Martin Muñoz, and Cristian Riveros. Efficient enumeration for annotated grammars. In *PODS '22: International Conference on Management of Data, Philadelphia, PA, USA, June 12 - 17, 2022*, pages 291–300, 2022.

[5] Renzo Angles, Angela Bonifati, Stefania Dumbrava, George Fletcher, Alastair Green, Jan Hidders, Bei Li, Leonid Libkin, Victor Marsault, Wim Martens, Filip Murlak, Stefan Plantikow, Ognjen Savkovic, Michael Schmidt, Juan Sequeda, Slawek Staworko, Dominik Tomaszuk, Hannes Voigt, Domagoj Vrgoc, Mingxi Wu, and Dusan Zivkovic. Pg-schema: Schemas for property graphs. *Proc. ACM Manag. Data*, 1(2):198:1–198:25, 2023.

[6] Dana Angluin. Finding patterns common to a set of strings. *J. Comput. Syst. Sci.*, 21(1):46–62, 1980.

[7] Pablo Barceló. Querying graph databases. In *Proceedings of the 32nd ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, PODS 2013, New York, NY, USA - June 22 - 27, 2013*, pages 175–188, 2013.

[8] Pierre Bourhis, Alejandro Grez, Louis Jachiet, and Cristian Riveros. Ranked enumeration of MSO logic on words. In *24th International Conference on Database Theory, ICDT 2021, March 23-26, 2021, Nicosia, Cyprus*, pages 20:1–20:19, 2021.

[9] Cezar Câmpeanu, Kai Salomaa, and Sheng Yu. Regex and extended regex. In *Implementation and Application of Automata, 7th International Conference, CIAA 2002, Tours, France, July 3-5, 2002, Revised Papers*, pages 77–84, 2002.

[10] Johannes Doleschal, Benny Kimelfeld, and Wim Martens. The complexity of aggregates over extractions by regular expressions. *Log. Methods Comput. Sci.*, 19(3), 2023.

[11] Johannes Doleschal, Benny Kimelfeld, Wim Martens, Yoav Nahshon, and Frank Neven. Split-correctness in information extraction. In *Proceedings of the 38th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems, PODS 2019, Amsterdam, The Netherlands, June 30 - July 5, 2019*, pages 149–163, 2019.

[12] Johannes Doleschal, Benny Kimelfeld, Wim Martens, and Liat Peterfreund. Weight annotation in information extraction. In *23rd International Conference on Database Theory, ICDT 2020, March 30-April 2, 2020, Copenhagen, Denmark*, pages 8:1–8:18, 2020.

[13] R. Fagin, B. Kimelfeld, F. Reiss, and S. Vansummeren. Document spanners: A formal approach to information extraction. *J. ACM*, 62(2):12:1–12:51, 2015.

[14] Fernando Florenzano, Cristian Riveros, Martín Ugarte, Stijn Vansummeren, and Domagoj Vrgoc. Efficient enumeration algorithms for regular document spanners. *ACM Trans. Database Syst.*, 45(1):3:1–3:42, 2020.

[15] D. Freydenberger. A logic for document spanners. *Theory Comput. Syst.*, 63(7):1679–1754, 2019.

[16] D. Freydenberger and M. Holldack. Document spanners: From expressive power to decision problems. *Theory Comput. Syst.*, 62(4):854–898, 2018.

[17] Dominik D. Freydenberger. Extended regular expressions: Succinctness and decidability. *Theory of Computing Systems (ToCS)*, 53(2):159–193, 2013.

[18] Dominik D. Freydenberger, Benny Kimelfeld, and Liat Peterfreund. Joining extractions of regular expressions. In *Proceedings of the 37th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems, Houston, TX, USA, June 10-15, 2018*, pages 137–149, 2018.

[19] Dominik D. Freydenberger and Sam M. Thompson. Dynamic complexity of document spanners. In *23rd International Conference on Database Theory, ICDT 2020, March 30-April 2, 2020, Copenhagen, Denmark*, pages 11:1–11:21, 2020.

[20] Dominik D. Freydenberger and Sam M. Thompson. Splitting spanner atoms: A tool for acyclic core spanners. In *25th International Conference on Database Theory, ICDT 2022, March 29 to April 1, 2022, Edinburgh, UK (Virtual Conference)*, pages 10:1–10:18, 2022.

[21] Alejandro Grez and Cristian Riveros. Towards streaming evaluation of queries with correlation in complex event processing. In *23rd International Conference on Database Theory, ICDT 2020, March 30-April 2, 2020, Copenhagen, Denmark*, pages 14:1–14:17, 2020.

[22] Alejandro Grez, Cristian Riveros, Martín Ugarte, and Stijn Vansummeren. On the expressiveness of languages for complex event recognition. In *23rd International Conference on Database Theory, ICDT 2020, March 30-April 2, 2020, Copenhagen, Denmark*, pages 15:1–15:17, 2020.

[23] Alejandro Grez, Cristian Riveros, Martín Ugarte, and Stijn Vansummeren. A formal framework for complex event recognition. *ACM Trans. Database Syst.*, 46(4):16:1–16:49, 2021.

[24] Martin Hirzel, Guillaume Baudart, Angela Bonifati, Emanuele Della Valle, Sherif Sakr, and Akrivi Vlachou. Stream processing languages in the big data era. *SIGMOD Rec.*, 47(2):29–40, 2018.

[25] John E. Hopcroft, Rajeev Motwani, and Jeffrey D. Ullman. *Introduction to automata theory, languages, and computation, 3rd Edition*. Pearson international edition. Addison-Wesley, 2007.

[26] Alessandro Margara and Gianpaolo Cugola. Processing flows of information: from data stream to complex event processing. In *Proceedings of the Fifth ACM International Conference on Distributed Event-Based Systems, DEBS 2011, New York, NY, USA, July 11-15, 2011*, pages 359–360, 2011.

[27] Francisco Maturana, Cristian Riveros, and Domagoj Vrgoc. Document spanners for extracting incomplete information: Expressiveness and complexity. In *Proceedings of the 37th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems, Houston, TX, USA, June 10-15, 2018*, pages 125–136, 2018.

[28] Martin Muñoz and Cristian Riveros. Constant-delay enumeration for slp-compressed documents. In *26th International Conference on Database Theory, ICDT 2023, March 28-31, 2023, Ioannina, Greece*, pages 7:1–7:17, 2023.

[29] L. Peterfreund. *The Complexity of Relational Queries over Extractions from Text*. PhD thesis, 2019.

[30] Liat Peterfreund. Grammars for document spanners. In *24th International Conference on Database Theory, ICDT 2021, March 23-26, 2021, Nicosia, Cyprus*, pages 7:1–7:18, 2021.

[31] Liat Peterfreund, Dominik D. Freydenberger, Benny Kimelfeld, and Markus Kröll. Complexity bounds for relational algebra over document spanners. In *Proceedings of the 38th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems, PODS 2019, Amsterdam, The Netherlands, June 30 - July 5, 2019.*, pages 320–334, 2019.

[32] Liat Peterfreund, Balder ten Cate, Ronald Fagin, and Benny Kimelfeld. Recursive programs for document spanners. In *22nd International Conference on Database Theory, ICDT 2019, March 26-28, 2019, Lisbon, Portugal*, pages 13:1–13:18, 2019.

[33] Markus L. Schmid. Characterising REGEX languages by regular languages equipped with factor-referencing. *Information and Computation (I&C)*, 249:1–17, 2016.

[34] Markus L. Schmid. The information extraction framework of document spanners - A very informal survey. In *SOFSEM 2024: Theory and Practice of Computer Science - 49th International Conference on Current Trends in Theory and Practice of Computer Science, SOFSEM 2024, Cochem, Germany, February 19-23, 2024, Proceedings*, pages 3–22, 2024.

[35] Markus L. Schmid and Nicole Schweikardt. A purely regular approach to non-regular core spanners. In *24th International Conference on Database Theory, ICDT 2021, March 23-26, 2021, Nicosia, Cyprus*, pages 4:1–4:19, 2021.

[36] Markus L. Schmid and Nicole Schweikardt. Spanner evaluation over slp-compressed documents. In *PODS'21: Proceedings of the 40th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems, Virtual Event, China, June 20-25, 2021*, pages 153–165, 2021.

[37] Markus L. Schmid and Nicole Schweikardt. Document spanners - A brief overview of concepts, results, and recent developments. In *PODS '22: International Conference on Management of Data, Philadelphia, PA, USA, June 12 - 17, 2022*, pages 139–150, 2022.

[38] Markus L. Schmid and Nicole Schweikardt. Query evaluation over slp-represented document databases with complex document editing. In *PODS '22: International Conference on Management of Data, Philadelphia, PA, USA, June 12 - 17, 2022*, pages 79–89, 2022.

[39] Domagoj Vrgoc, Carlos Rojas, Renzo Angles, Marcelo Arenas, Diego Arroyuelo, Carlos Buil-Aranda, Aidan Hogan, Gonzalo Navarro, Cristian Riveros, and Juan Romero. Millenniumdb: An open-source graph database system. *Data Intell.*, 5(3):560–610, 2023.