A Purely Regular Approach to Non-Regular Core Spanners

Markus L. Schmid
Humboldt-Universität zu Berlin, Germany

Nicole Schweikardt
Humboldt-Universität zu Berlin, Germany

Abstract

The regular spanners (characterised by vset-automata) are closed under the algebraic operations of union, join and projection, and have desirable algorithmic properties. The core spanners (introduced by Fagin, Kimelfeld, Reiss, and Vansummeren (PODS 2013, JACM 2015) as a formalisation of the core functionality of the query language AQL used in IBM's SystemT) additionally need string equality selections and it has been shown by Freydenberger and Holldack (ICDT 2016, Theory of Computing Systems 2018) that this leads to high complexity and even undecidability of the typical problems in static analysis and query evaluation. We propose an alternative approach to core spanners: by incorporating the string-equality selections directly into the regular language that represents the underlying regular spanner (instead of treating it as an algebraic operation on the table extracted by the regular spanner), we obtain a fragment of core spanners that, while having slightly weaker expressive power than the full class of core spanners, arguably still covers the intuitive applications of string equality selections for information extraction and has much better upper complexity bounds of the typical problems in static analysis and query evaluation.

2012 ACM Subject Classification Information systems → Information retrieval; Theory of computation → Automata extensions; Theory of computation → Regular languages; Theory of computation → Design and analysis of algorithms; Theory of computation → Database query languages (principles)

Keywords and phrases Document spanners, regular expressions with backreferences

Digital Object Identifier 10.4230/LIPIcs.ICDT.2021.4

Related Version *Full Version*: `https://arxiv.org/abs/2010.13442`

Funding *Markus L. Schmid*: Supported by the German Research Foundation (Deutsche Forschungsgemeinschaft, DFG) – project number 416776735 (gefördert durch die Deutsche Forschungsgemeinschaft (DFG) – Projektnummer 416776735).
Nicole Schweikardt: Partially supported by the ANR project EQUUS ANR-19-CE48-0019; funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) – project number 431183758 (gefördert durch die Deutsche Forschungsgemeinschaft (DFG) – Projektnummer 431183758).

Acknowledgements We wish to thank the anonymous reviewers for their valuable feedback that improved the readability of this paper.

1 Introduction

The information extraction framework of *document spanners* has been introduced by Fagin, Kimelfeld, Reiss, and Vansummeren [4] as a formalisation of the query language AQL, which is used in IBM's information extraction engine SystemT. A document spanner performs information extraction by mapping a *document*, formalised as a word w over a finite alphabet Σ, to a relation over so-called *spans* of w, which are intervals $[i, j\rangle$ with $0 \leqslant i < j \leqslant |w| + 1$.

The document spanners (or simply *spanners*, for short) introduced in [4] follow a two-stage approach: *Primitive* spanners extract relations directly from the input document, which are then further manipulated by using some relational algebra. As primitive spanners, [4] introduces *vset-automata* and *regex-formulas*, which are variants of nondeterministic finite automata and regular expressions, respectively, that can use meta-symbols ${}^{\mathsf{x}}\triangleright$ and $\triangleleft^{\mathsf{x}}$, where x is a *variable* from a set $\mathcal{X}$ of variables, in order to bind those variables to start and end positions of spans, therefore extracting an $|\mathcal{X}|$-ary span-relation, or a table with columns labelled by the variables $\mathcal{X}$. For example, $\alpha = (\,{}^{\mathsf{x}}\triangleright\, (\mathtt{a} \vee \mathtt{b})^* \triangleleft^{\mathsf{x}}) \cdot (\,{}^{\mathsf{y}}\triangleright\, (\mathtt{a}^* \vee \mathtt{b}^*) \triangleleft^{\mathsf{y}})\mathtt{c}^*$ is a regex-formula and it describes a spanner $[\![\alpha]\!]$ by considering for a given word w all possibilities of how w can be generated by α and for each such generation of w, the variables x and y extract the spans that correspond to those subwords of w that are generated by the subexpressions ${}^{\mathsf{x}}\triangleright\, (\mathtt{a} \vee \mathtt{b})^* \triangleleft^{\mathsf{x}}$ and ${}^{\mathsf{y}}\triangleright\, (\mathtt{a}^* \vee \mathtt{b}^*) \triangleleft^{\mathsf{y}}$, respectively. For example, on input $w = \mathtt{abaac}$, we have $[\![\alpha]\!](w) = \{([1,3\rangle, [3,5\rangle), ([1,4\rangle, [4,5\rangle), ([1,5\rangle, [5,5\rangle)\}$, since α can generate ${}^{\mathsf{x}}\triangleright\, \mathtt{ab} \triangleleft^{\mathsf{x}}\, {}^{\mathsf{y}}\triangleright\, \mathtt{aa} \triangleleft^{\mathsf{y}}\, \mathtt{c}$, ${}^{\mathsf{x}}\triangleright\, \mathtt{aba} \triangleleft^{\mathsf{x}}\, {}^{\mathsf{y}}\triangleright\, \mathtt{a} \triangleleft^{\mathsf{y}}\, \mathtt{c}$ and ${}^{\mathsf{x}}\triangleright\, \mathtt{abaa} \triangleleft^{\mathsf{x}}\, {}^{\mathsf{y}}\triangleright\, \triangleleft^{\mathsf{y}}\, \mathtt{c}$. The vset-automata follow the same principle, but take the form of nondeterministic finite automata. Since these primitive spanners are based on formal language description mechanisms, they are also called *regular spanners* and, for the sake of presentation, we denote this class of regular spanners by $\mathsf{reg}\text{-}\mathfrak{S}$ for the remainder of this introduction (there are different ways of characterising *regular spanners* and also different semantics (see [12, 4]); these aspects shall be discussed in more detail below).

The considered algebraic operations are union $\cup$, natural join $\bowtie$, projection π (with the obvious meaning) and string-equality selection $\varsigma^{=}_{\mathcal{Z}}$, which is a unary operator parameterised by a set $\mathcal{Z} \subseteq \mathcal{X}$ of variables, and it selects exactly those rows of the table for which all spans of columns $\mathcal{Z}$ refer to (potentially different occurrences of) the same subwords of w.

The *core spanners* (capturing the *core* of SystemT's query language AQL) introduced in [4] are defined as $\mathsf{reg}\text{-}\mathfrak{S}^{\{\cup,\bowtie,\pi,\varsigma^{=}\}}$, i. e., the closure of regular spanners under the operations $\cup$, $\bowtie$, π and $\varsigma^{=}$ (these relational operations are interpreted as operations on spanners in the natural way). A central result of [4] is that the operations $\cup$, $\bowtie$ and π can be directly incorporated into the regular spanners, i. e., $\mathsf{reg}\text{-}\mathfrak{S}^{\{\cup,\bowtie,\pi\}} = \mathsf{reg}\text{-}\mathfrak{S}$. This is due to the fact that regular spanners are represented by finite automata and therefore the closure properties for regular languages carry over to regular spanners by similar automaton constructions. This also holds in the case of so-called *schemaless semantics* (see [12]). However, as soon as we also consider the operator of string-equality selection, the picture changes considerably.

In terms of expressive power, it can be easily seen that not all core spanners are regular spanners, simply because for all regular spanners S the language $\{w \in \Sigma^* \mid S(w) \neq \emptyset\}$ is regular, which is not necessarily the case for core spanners. As shown in [4], we can nevertheless represent any core spanner $S \in \mathsf{reg}\text{-}\mathfrak{S}^{\{\cup,\bowtie,\pi,\varsigma^{=}\}}$ in the form $\pi_{\mathcal{Y}}\varsigma^{=}_{\mathcal{Z}_1}\varsigma^{=}_{\mathcal{Z}_2}\ldots\varsigma^{=}_{\mathcal{Z}_k}(S')$ for a regular spanner S' (this is called the *core-simplification lemma* in [4]).

Regular spanners have excellent algorithmic properties: enumerating $S(w)$ can be done with linear preprocessing and constant delay, even if the spanner is given as vset-automaton (see [1, 6]), while spanner containment or inclusion can be decided efficiently if the spanner is represented by a certain deterministic vset-automaton (see [3]). However, in terms of complexity, we have to pay a substantial price for adding string-equality selections to regular spanners. It has been shown in [8] that for core spanners the typical problems of query evaluation and static analysis are NP- or PSpace-hard, or even undecidable (see Table 1).

The results from [8] identify features that are, from an intuitive point of view, sources of complexity for core spanners. Thus, the question arises whether tractability can be achieved by restricting core spanners respectively. We shall illustrate this with some examples.

Table 1 Comparison of decision problems of regular spanners, core spanners and refl-spanners. A formal definition of the problems can be found in Section 5. In the case of regular spanners and refl-spanners, the input spanner is represented by an NFA M. The abbreviation "str. ref." means *strongly reference extracting*, a restriction for refl-spanners to be formally defined in Section 5. The authors are not aware of a (non-trivial) upper bound for ModelChecking for regular spanners (note that since regular spanners are covered by refl-spanners, the upper bound for refl-spanners applies).

	Problem	Regular sp.	Refl-sp.		Core sp. [8]
Evaluation	ModelChecking	?	$\mathrm{poly}(\lvert M\rvert)(\lvert w\rvert + (2\lvert\mathcal{X}\rvert)!)$	[T. 5.1]	NP-c
problems	NonEmptiness	$O(\lvert M\rvert\lvert w\rvert)$	NP-c	[T. 5.2]	NP-h
Static	Satisfiability	$O(\lvert M\rvert)$	$O(\lvert M\rvert)$	[T. 5.3]	PSpace-c
analysis	Containment	PSpace-c [12]	ExpSpace (for str. ref.)	[T. 5.10]	undec.
problems	Equivalence	PSpace-c [12]	ExpSpace (for str. ref.)	[T. 5.10]	undec.
	Hierarchicality	$O(\lvert M\rvert\lvert\mathcal{X}\rvert^3)$	$O(\lvert M\rvert\lvert\mathcal{X}\rvert^3)$	[T. 5.3]	PSpace-c

Consider a regex formula $\alpha = {}^{\mathsf{x}_1}\!\triangleright \Sigma^* \triangleleft^{\mathsf{x}_1}\ {}^{\mathsf{x}_2}\!\triangleright \Sigma^* \triangleleft^{\mathsf{x}_2} \ldots {}^{\mathsf{x}_n}\!\triangleright \Sigma^* \triangleleft^{\mathsf{x}_n}$. Then checking, for some $\mathcal{Z}_1, \mathcal{Z}_2, \ldots, \mathcal{Z}_k \subseteq \{\mathsf{x}_1, \mathsf{x}_2, \ldots, \mathsf{x}_n\}$, whether the empty tuple is in $(\pi_{\emptyset}\varsigma^{=}_{\mathcal{Z}_1}\varsigma^{=}_{\mathcal{Z}_2}\ldots\varsigma^{=}_{\mathcal{Z}_k}(\llbracket\alpha\rrbracket))(w)$, is identical to checking whether w can be factorised into n factors such that for each $\mathcal{Z}_i$ all factors that correspond to the variables in $\mathcal{Z}_i$ are the same. This is the pattern matching problem with variables (or the membership problem for pattern languages), a well-known NP-complete problem (see, e. g., [11]). However, checking for a (non-empty) span-tuple t whether it is in $(\varsigma^{=}_{\mathcal{Z}_1}\varsigma^{=}_{\mathcal{Z}_2}\ldots\varsigma^{=}_{\mathcal{Z}_k}(\llbracket\alpha\rrbracket))(w)$ can be easily done in polynomial time, since the task of checking the existence of a suitable factorisation boils down to the task of evaluating a factorisation that is implicitly given by t. Hence, instead of blaming the string-equality selections for intractability, we could as well blame the projection operator. Can we achieve tractability by restricting projections instead of string-equality selections?

Another feature that yields intractability is that we can use string-equality selections in order to concisely express the intersection non-emptiness of regular languages (a well-known PSpace-complete problem). For example, let $r_1, r_2, \ldots, r_n$ be some regular expressions, and let $\alpha = {}^{\mathsf{x}_1}\!\triangleright r_1 \triangleleft^{\mathsf{x}_1}\ {}^{\mathsf{x}_2}\!\triangleright r_2 \triangleleft^{\mathsf{x}_2} \ldots {}^{\mathsf{x}_n}\!\triangleright r_n \triangleleft^{\mathsf{x}_n}$. Then there is a word w with $(\varsigma^{=}_{\{\mathsf{x}_1,\mathsf{x}_2,\ldots,\mathsf{x}_n\}}(\llbracket\alpha\rrbracket))(w) \neq \emptyset$ if and only if $\bigcap_{i=1}^{n} \mathcal{L}(r_i) \neq \emptyset$. So string-equality selections do not only check whether the same subword has several occurrences, but also, as a "side-effect", check membership of this repeated subword in the intersection of several regular languages. Can we achieve tractability by somehow limiting the power of string-equality selections to the former task?

A third observation is that by using string-equality selections on *overlapping* spans, we can use core spanners to express rather complex word-combinatorial properties. In fact, we can even express word equations as core spanners (see [8, Proposition 3.7, Example 3.8, Theorem 3.13] for details). Can we achieve tractability by requiring all variables that are subject to string-equality selections to extract only pairwise non-overlapping spans?

1.1 Our Contribution

We introduce *refl-spanners* (based on *regular ref-languages*), a new formalism for spanners that properly extends regular spanners, describes a large class of core spanners, and has better upper complexity bounds than core spanners. Moreover, the formalism is purely based on regular language description mechanisms. The main idea is a paradigm shift in the two-stage approach of core spanners: instead of extracting a span-relation with a regular spanner and then applying string-equality selections on it, we handle string-equality selections directly with the finite automaton (or regular expression) that describes the regular spanner. However, checking the equality of unbounded factors in strings is a task that, in most formalisms, can be considered highly “non-regular” (the well-known *copy-language* $\{ww \mid w \in \Sigma^*\}$ is a textbook example for demonstrating the limits of regular languages in this regard). We deal with this obstacle by representing the factors that are subject to string-equality selections as *variables* in the regular language. For example, while $L = \{\mathtt{a}^n\mathtt{b}\mathtt{a}^n \mid n \geqslant 0\}$ is non-regular, the language $L' = \{\,{}^{\mathsf{x}}\!\triangleright \mathtt{a}^n \lhd^{\mathsf{x}} \mathtt{b}\mathsf{x} \mid n \geqslant 0\}$ can be interpreted as a regular description of L by means of meta-symbols ${}^{\mathsf{x}}\!\triangleright$ and $\lhd^{\mathsf{x}}$ to *capture* a factor, and a meta-symbol x to *copy* or *reference* the captured factor. In particular, all words of L can be easily obtained from the words of L' by simply replacing the occurrence of x with the factor it refers to. As long as core spanners use string equality selections in a not too complicated way, this simple formalism seems also to be suited for describing core spanners, e. g., the core spanner $\pi_{\{\mathsf{x},\mathsf{y}\}}\varsigma^{=}_{\{\mathsf{x},\mathsf{x}'\}}\varsigma^{=}_{\{\mathsf{y},\mathsf{y}'\}}([\![\alpha]\!])$ with $\alpha = {}^{\mathsf{x}}\!\triangleright \mathtt{a}^*\mathtt{b}\,{}^{\mathsf{y}}\!\triangleright \mathtt{c} \lhd^{\mathsf{x}} \mathtt{b}^*\,{}^{\mathsf{x}'}\!\triangleright \mathtt{a}^*\mathtt{bc} \lhd^{\mathsf{x}'} \lhd^{\mathsf{y}}\,{}^{\mathsf{y}'}\!\triangleright \mathtt{cb}^*\mathtt{a}^*\mathtt{bc} \lhd^{\mathsf{y}'}$ could be represented as $[\![{}^{\mathsf{x}}\!\triangleright \mathtt{a}^*\mathtt{b}\,{}^{\mathsf{y}}\!\triangleright \mathtt{c} \lhd^{\mathsf{x}} \mathtt{b}^*\mathsf{x} \lhd^{\mathsf{y}} \mathsf{y}]\!]$.

The class of refl-spanners can now informally be described as the class of all spanners that can be represented by a regular language over the alphabet $\Sigma \cup \mathcal{X} \cup \{{}^{\mathsf{x}}\!\triangleright, \lhd^{\mathsf{x}} \mid \mathsf{x} \in \mathcal{X}\}$ that has the additional property that the meta-symbols $\mathcal{X} \cup \{{}^{\mathsf{x}}\!\triangleright, \lhd^{\mathsf{x}} \mid \mathsf{x} \in \mathcal{X}\}$ are “well-behaved” in the sense that each word describes a valid span-tuple (one of our main conceptional contributions is to formalise this idea in a sound way).

The refl-spanner formalism automatically avoids exactly the features of core spanners that we claimed above to be sources of complexity. More precisely, refl-spanners cannot project out variables, which means that they cannot describe the task of checking the existence of some complicated factorisation. Furthermore, it can be easily seen that in the refl-spanner formalism, we cannot describe intersection non-emptiness of regular languages in a concise way, as is possible by core spanners. Finally, we can only have overlaps with respect to the spans captured by ${}^{\mathsf{x}}\!\triangleright \ldots \lhd^{\mathsf{x}}$, but all references x represent pairwise non-overlapping factors, which immediately shows that we cannot express word equations as core spanners can. This indicates that refl-spanners are restricted in terms of expressive power, but it also gives hope that for refl-spanners we can achieve better upper complexity bounds of the typical decision problems compared to core spanners, and, in fact, this is the case (see Table 1).

It is obvious that not all core spanners can be represented as refl-spanners, but we can nevertheless show that a surprisingly large class of core spanners can be handled by the refl-spanner formalism. Recall that the core simplification lemma from [4] states that in every core spanner $S \in \textsf{reg-}\mathfrak{S}^{\{\cup,\pi,\bowtie,\varsigma^{=}\}}$, we can “push” all applications of $\cup$ and $\bowtie$ into the automaton that represents the regular spanner, leaving us with an expression $\pi_{\mathcal{Y}}\varsigma^{=}_{\mathcal{Z}_1}\varsigma^{=}_{\mathcal{Z}_2}\ldots\varsigma^{=}_{\mathcal{Z}_k}(M)$ for an automaton M that represents a regular spanner. We can show that if the string-equality selections $\varsigma^{=}_{\mathcal{Z}_1}\varsigma^{=}_{\mathcal{Z}_2}\ldots\varsigma^{=}_{\mathcal{Z}_k}$ apply to a set of variables that never capture overlapping spans, then we can further “push” even all string-equality selections into M, turning it into a representation of a refl-spanner that “almost” describes S: in order to get S, we only have to merge certain columns into a single one by creating the fusion of the corresponding spans.

1.2 Related Work

Spanners have recently received a lot of attention [4, 9, 15, 1, 12, 6, 16, 7, 8, 10, 13]. However, as it seems, most of the recent progress on document spanners concerns regular spanners. For example, it has recently been shown that results of regular spanners can be enumerated with linear preprocessing and constant delay [1, 6], the paper [12] is concerned with different semantics of regular spanners and their expressive power, and [15] investigates the evaluation of algebraic expressions over regular spanners.

Papers that are concerned with string-equality selection are [8] in which many negative results for core spanner evaluation are shown, [7] which, by presenting a logic that exactly covers core spanners, answers question on the expressive power of core spanners, [16] that shows that datalog over regular spanners covers the whole class of core spanners, and [9] which investigates conjunctive queries on top of regular spanners and, among mostly negative results, also contains the positive result that such queries with equality-selections can be evaluated efficiently if the number of string equalities is bounded by a constant. The paper [10] investigates the dynamic descriptive complexity of regular spanners and core spanners. While all these papers contribute deep insights with respect to document spanners, positive algorithmic results for the original core spanners from [4] seem scarce and the huge gap in terms of tractability between regular and core spanners seems insufficiently bridged by tractable fragments of core spanners.

A rather recent paper that also deals with non-regular document spanners is [14]. However, the non-regular aspect of [14] does not consist in string-equality selections, but rather that spanners are represented by context-free language descriptors (in particular grammars) instead of regular ones.

1.3 Organisation

The rest of the paper is structured as follows. Section 2 fixes the basic notation concerning spanners and lifts the core-simplification lemma of [4] to the schemaless case. In Section 3, we develop a simple declarative approach to spanners by establishing a natural one-to-one correspondence between spanners and so-called subword-marked languages. In Section 4 we extend the concept of subword-marked languages in order to describe spanners with string-equality selections which we call refl-spanners. Section 5 is devoted to the complexity of evaluation and static analysis problems for refl-spanners. Section 6 studies the expressive power of refl-spanners. We conclude the paper in Section 7. Due to space constraints, most proof details are omitted, although we give proof sketches for some results; detailed proofs can be found in the preliminary full version of this paper [18].

2 Preliminaries

Let $\mathbb{N} = \{1, 2, 3, \ldots\}$ and $[n] = \{1, 2, \ldots, n\}$ for $n \in \mathbb{N}$. For a (partial) mapping $f : X \to Y$, we write $f(x) = \perp$ for some $x \in X$ to denote that $f(x)$ is not defined; we also set $\text{dom}(f) = \{x \mid f(x) \neq \perp\}$. By $\mathcal{P}(A)$ we denote the power set of a set A, and A^+ denotes the set of non-empty words over A, and $A^* = A^+ \cup \{\varepsilon\}$, where ε is the empty word. For a word $w \in A^*$, $|w|$ denotes its length (in particular, $|\varepsilon| = 0$), and for every $b \in A$, $|w|_b$ denotes the number of occurrences of b in w. Let A and B be alphabets with $B \subseteq A$, and let $w \in A^*$. Then $\mathsf{e}_B : A \to A \cup \{\varepsilon\}$ is a mapping with $\mathsf{e}_B(b) = \varepsilon$ if $b \in B$ and $\mathsf{e}_B(b) = b$ if $b \in A \setminus B$; we also use e_B to denote the natural extension of e_B to the morphism $A^* \to A^*$. Technically, e_B depends on the alphabet A, but whenever we use $\mathsf{e}_B(w)$ we always assume that $\mathsf{e}_B : A \to A \cup \{\varepsilon\}$ for some alphabet A with $w \in A^*$.

2.1 Regular Language Descriptors

For an alphabet Σ, the set RE_Σ of *regular expressions* (*over* Σ) is defined as usual: every $a \in \Sigma \cup \{\varepsilon\}$ is in RE_Σ with $\mathcal{L}(a) = \{a\}$, and, for $r, s \in \mathsf{RE}_\Sigma$, $(r \cdot s), (r \vee s), (r)^+ \in \mathsf{RE}_\Sigma$ with $\mathcal{L}((r \cdot s)) = \mathcal{L}(r) \cdot \mathcal{L}(s)$, $\mathcal{L}((r \vee s)) = \mathcal{L}(r) \cup \mathcal{L}(s)$, $\mathcal{L}((r)^+) = (\mathcal{L}(r))^+$. For $r \in \mathsf{RE}_\Sigma$, we use r^* as a shorthand form for $(r)^+ \vee \varepsilon$, and we usually omit the operator "$\cdot$", i. e., we use juxtaposition. For the sake of readability, we often omit parentheses, if this does not cause ambiguities.

A *nondeterministic finite automaton* (NFA for short) is a tuple $M = (Q, \Sigma, \delta, q_0, F)$ with a set Q of states, a finite alphabet Σ, a start state q_0, a set F of accepting states and a transition function $\delta : Q \times (\Sigma \cup \{\varepsilon\}) \to \mathcal{P}(Q)$. We also interpret NFA as directed, edge-labelled graphs in the obvious way. A word $w \in \Sigma^*$ is accepted by M if there is a path from q_0 to some $q_f \in F$ that is labelled by w; $\mathcal{L}(M)$ is the accepted language, i. e., the set of all accepted words. The size $|M|$ of an NFA is measured in $|Q| + |\delta|$. However, we will mostly consider NFA with constant out-degree, which means that $|M| = \mathrm{O}(|Q|)$. For a language descriptor D (e. g., an NFA or a regular expression), we denote by $\mathcal{L}(D)$ the language defined by D. The class of languages described by NFA or regular expressions is the class of regular languages, denoted by reg-$\mathfrak{L}$.

2.2 Spans and Spanners

For a word $w \in \Sigma^*$ and for every $i, j \in [|w|+1]$ with $i \leqslant j$, $[i, j\rangle$ is a *span of* w and its *value*, denoted by $w[i, j\rangle$, is the substring of w from symbol i to symbol $j-1$. In particular, $w[i, i\rangle = \varepsilon$ (this is called *empty span*) and $w[1, |w|+1\rangle = w$. By $\mathsf{Spans}(w)$, we denote the set of spans of w, and by Spans we denote the set of spans for any word (elements from Spans shall simply be called *spans*). A span $[i, j\rangle$ can also be interpreted as the set $\{i, i+1, \ldots, j-1\}$ and therefore we can use set-theoretical notions for spans. Two spans $s = [i, j\rangle$ and $s' = [i', j'\rangle$ are *equal* if $s = s'$, they are *disjoint* if $j \leqslant i'$ or $j' \leqslant i$ and they are *quasi-disjoint* if they are equal or disjoint. Note that s and s' being disjoint is sufficient, but not necessary for $s \cap s' = \emptyset$, e. g., $[3, 6\rangle$ and $[5, 5\rangle$ are not disjoint, but $[3, 6\rangle \cap [5, 5\rangle = \emptyset$.

For a finite set of variables $\mathcal{X}$, an $(\mathcal{X}, w)$*-tuple* (also simply called *span-tuple*) is a partial function $\mathcal{X} \to \mathsf{Spans}(w)$, and a $(\mathcal{X}, w)$*-relation* is a set of $(\mathcal{X}, w)$-tuples. For simplicity, we usually denote $(\mathcal{X}, w)$-tuples in tuple-notation, for which we assume an order on $\mathcal{X}$ and use the symbol "$\perp$" for undefined variables, e. g., $([1, 5\rangle, \perp, [5, 7\rangle)$ describes a $(\{\mathsf{x}_1, \mathsf{x}_2, \mathsf{x}_3\}, w)$-tuple that maps x_1 to $[1, 5\rangle$, x_3 to $[5, 7\rangle$, and is undefined for x_2.

An $(\mathcal{X}, w)$-tuple t is *functional* if it is a total function, t is *hierarchical* if, for every $\mathsf{x}, \mathsf{y} \in \mathrm{dom}(t)$, $t(\mathsf{x}) \subseteq t(\mathsf{y})$ or $t(\mathsf{y}) \subseteq t(\mathsf{x})$ or $t(\mathsf{x}) \cap t(\mathsf{y}) = \emptyset$, and t is *quasi-disjoint* if, for every $\mathsf{x}, \mathsf{y} \in \mathrm{dom}(t)$, $t(\mathsf{x})$ and $t(\mathsf{y})$ are quasi-disjoint. An $(\mathcal{X}, w)$-relation is *functional*, *hierarchical* or *quasi-disjoint*, if all its elements are functional, hierarchical or quasi-disjoint, respectively.

A *spanner* (*over terminal alphabet* Σ *and variables* $\mathcal{X}$) is a function that maps every $w \in \Sigma^*$ to an $(\mathcal{X}, w)$-relation (note that the empty relation $\emptyset$ is also a valid image of a spanner). Since the dependency on the word w is often negligible, we also use the term $\mathcal{X}$*-tuple* or $\mathcal{X}$*-relation* to denote an $(\mathcal{X}, w)$-tuple or $(\mathcal{X}, w)$-relation, respectively.

▶ **Example 2.1.** Let $\Sigma = \{\mathsf{a}, \mathsf{b}\}$ and let $\mathcal{X} = \{\mathsf{x}, \mathsf{y}, \mathsf{z}\}$. Then the function S that maps words $w \in \Sigma^*$ to the $(\mathcal{X}, w)$-relation $\{([1, i\rangle, [i, i+1\rangle, [i+1, |w|+1\rangle) \mid 1 \leqslant i < |w|, w[i, i+1\rangle = \mathsf{b}\}$ is a spanner. For example, $S(\mathsf{ababbab}) = \{t_1, t_2, t_3, t_4\}$ with $t_1 = ([1, 2\rangle, [2, 3\rangle, [3, 8\rangle)$, $t_2 = ([1, 4\rangle, [4, 5\rangle, [5, 8\rangle)$, $t_3 = ([1, 5\rangle, [5, 6\rangle, [6, 8\rangle)$ and $t_4 = ([1, 7\rangle, [7, 8\rangle, [8, 8\rangle)$.

Let S_1 and S_2 be spanners over Σ and $\mathcal{X}$. Then S_1 and S_2 are said to be *equal* if, for every $w \in \Sigma^*$, $S_1(w) = S_2(w)$ (this coincides with the usual equality of functions and shall also be denoted by $S_1 = S_2$). We say that S_2 *contains* S_1, written as $S_1 \subseteq S_2$, if, for every $w \in \Sigma^*$, $S_1(w) \subseteq S_2(w)$. A spanner S over Σ and $\mathcal{X}$ is *functional*, *hierarchical* or *quasi-disjoint* if, for every w, $S(w)$ is functional, hierarchical or quasi-disjoint, respectively. Note that, for span-tuples, span-relations and spanners, quasi-disjointness implies hierarchicality.

Next, we define operations on spanners. The *union* $S_1 \cup S_2$ of two spanners S_1 and S_2 over Σ and $\mathcal{X}$ is defined via $(S_1 \cup S_2)(w) = S_1(w) \cup S_2(w)$ for all $w \in \Sigma^*$.

The *natural join* $S_1 \bowtie S_2$ is defined via $(S_1 \bowtie S_2)(w) = S_1(w) \bowtie S_2(w)$ for all $w \in \Sigma^*$. Here, for two $(\mathcal{X}, w)$-relations R_1, R_2 we let $R_1 \bowtie R_2 = \{t_1 \bowtie t_2 \mid t_1 \in R_1, t_2 \in R_2, t_1 \sim t_2\}$. Two $(\mathcal{X}, w)$-tuples t_1 and t_2 are *compatible* (denoted by $t_1 \sim t_2$) if $t_1(\mathsf{x}) = t_2(\mathsf{x})$ for every $\mathsf{x} \in \mathrm{dom}(t_1) \cap \mathrm{dom}(t_2)$, and for compatible $(\mathcal{X}, w)$-tuples t_1 and t_2, the $(\mathcal{X}, w)$-tuple $t_1 \bowtie t_2$ is defined by $(t_1 \bowtie t_2)(\mathsf{x}) = t_i(\mathsf{x})$ if $\mathsf{x} \in \mathrm{dom}(t_i)$ for $i \in \{1, 2\}$.

The *projection* $\pi_{\mathcal{Y}}(S_1)$ for a set $\mathcal{Y} \subseteq \mathcal{X}$ is defined by letting $(\pi_{\mathcal{Y}}(S_1))(w) = \{t_{|\mathcal{Y}} : t \in S_1(w)\}$, where $t_{|\mathcal{Y}}$ is the restriction of t to domain $\mathrm{dom}(t) \cap \mathcal{Y}$.

The *string-equality selection* $\varsigma^{=}_{\mathcal{Y}}(S_1)$ for a set $\mathcal{Y} \subseteq \mathcal{X}$ is defined by letting $(\varsigma^{=}_{\mathcal{Y}}(S_1))(w)$ contain all $t \in S_1(w)$ such that, for every $\mathsf{x}, \mathsf{y} \in \mathcal{Y} \cap \mathrm{dom}(t)$, if $t(\mathsf{x}) = [i, j\rangle$ and $t(\mathsf{y}) = [i', j'\rangle$, then $w[i, j\rangle = w[i', j'\rangle$. Further below, we will discuss why we only require $w[i, j\rangle = w[i', j'\rangle$ for $\mathsf{x}, \mathsf{y} \in \mathcal{Y} \cap \mathrm{dom}(t)$ instead of $\mathsf{x}, \mathsf{y} \in \mathcal{Y}$.

For convenience, we omit the parentheses if we apply sequences of unary operations of spanners, e. g., we write $\pi_{\mathcal{Z}} \varsigma^{=}_{\mathcal{Y}_1} \varsigma^{=}_{\mathcal{Z}_1}(S)$ instead of $\pi_{\mathcal{Z}}(\varsigma^{=}_{\mathcal{Y}_1}(\varsigma^{=}_{\mathcal{Z}_1}(S)))$. For any $E = \{\mathcal{Y}_1, \mathcal{Y}_2, \ldots, \mathcal{Y}_\ell\} \subseteq \mathcal{P}(\mathcal{X})$, we also write $\varsigma^{=}_{E}(S)$ instead of $\varsigma^{=}_{\mathcal{Y}_1} \varsigma^{=}_{\mathcal{Y}_2} \ldots \varsigma^{=}_{\mathcal{Y}_\ell}(S)$, and in this case we will call $\varsigma^{=}_{E}$ a *generalised* string-equality selection, or also just string-equality selection if it is clear from the context that $E \subseteq \mathcal{P}(\mathcal{X})$. For generalised string-equality selections $\varsigma^{=}_{E}$, we will always assume that all sets of E are pairwise disjoint. For a class $\mathfrak{S}$ of spanners and set P of spanner operations, $\mathfrak{S}^P$ (or $\mathfrak{S}^p$ if $P = \{p\}$) denotes the closure of $\mathfrak{S}$ under the operations from P.

Whenever formulating complexity bounds, we consider the terminal alphabet Σ to be constant, but we always explicitly state any dependency on $|\mathcal{X}|$.

2.3 Regular Spanners and Core Spanners

In [4], the class of *regular spanners*, denoted by $\mathsf{reg}\text{-}\mathfrak{S}$, is defined as the class of spanners represented by *vset-automata*, and the class of *core spanners* is defined as $\mathsf{core}\text{-}\mathfrak{S} = [\![\mathsf{RGX}]\!]^{\{\cup, \pi, \bowtie, \varsigma^=\}}$, where RGX is the class of so-called *regex-formulas* (we refer to [4] for a formal definition of vset-automata and regex-formulas). A crucial result from [4] is the *core-simplification lemma*: every $S \in \mathsf{core}\text{-}\mathfrak{S}$ can be represented as $\pi_{\mathcal{Y}} \varsigma^{=}_{E}(S')$, where S' is a regular spanner. The setting in [4] uses a *function semantics* for spanners, i. e., $(\mathcal{X}, w)$-tuples are always functional. In our definitions above, we allow variables in span-tuples and spanners to be undefined, i. e., we use partial mappings as introduced in [12], and in the terminology of [15], we consider the *schemaless* semantics.

In [12], it is shown that the classical framework for *regular spanners* with function semantics introduced in [4] can be extended to the schemaless case, i. e., vset-automata and regex-formula are extended to the case of schemaless semantics, and it is shown that the basic results still hold (e. g., vset-automata are equally powerful as regex-formulas (or vstack-automata) equipped with union, natural join and projection). However, the string-equality selection operator – which turns regular spanners into the more powerful core spanners – is not treated in [12]. Our definition of the string-equality selection operator given above extends the definition from [4] from the functional to the schemaless case by interpreting $\varsigma^{=}_{\mathcal{Y}}$ to apply only to those variables from $\mathcal{Y}$ that are in the domain of the span-tuple. This way of treating undefined variables is natural and also corresponds to how the join operator is extended to the schemaless case in [12]. Due to [12], we can also in the schemaless case define $\mathsf{reg}\text{-}\mathfrak{S}$ as the class of spanners defined by vset-automata (with schemaless semantics), and we can also define the class of core spanners with schemaless semantics as $\mathsf{core}\text{-}\mathfrak{S} = [\![\mathsf{RGX}]\!]^{\{\cup, \pi, \bowtie, \varsigma^=\}}$. However, to the knowledge of the authors, the core-simplification lemma from [4] has so far not been extended to the schemaless semantics. Since we wish to apply the core-simplification lemma in the context of our results (for schemaless semantics), and since this seems to be a worthwhile task in its own right, we show that the core-simplification lemma from [4] holds verbatim for the schemaless case. For those parts of the proof's argument that are not concerned with string-equality selections, we heavily rely on the results from [12].

▶ **Lemma 2.2** (Core Simplification Lemma). *For every $S \in$ core-$\mathfrak{S}$ over $\mathcal{X}$ there are $S' \in$ reg-$\mathfrak{S}$, $\mathcal{Y} \subseteq \mathcal{X}$ and $E \subseteq \mathcal{P}(\mathcal{X})$ such that $S = \pi_{\mathcal{Y}} \varsigma^{=}_{E}(S')$.*

3 A Declarative Approach to Spanners

In this section, we develop a simple declarative approach to spanners by establishing a natural one-to-one correspondence between spanners over Σ and so-called *subword-marked languages* over Σ.[1] This approach conveniently allows to investigate or define non-algorithmic properties of spanners completely independently from any machine model or other description mechanisms (e. g., types of regular expressions, automata, etc.), while at the same time we can use the existing algorithmic toolbox for formal languages whenever required (instead of inventing special-purpose variants of automata or regular expressions to this end).

In particular, this declarative approach is rather versatile and provides some modularity in the sense that we could replace "regular languages" by any kind of language class (e. g., (subclasses of) context-free languages, context-sensitive languages, etc.) to directly obtain (i. e., without any need to adopt our definitions) a formally sound class of document spanners and also have the full technical machinery that exists for this language class at our disposal. Note that the idea of using non-regular languages from the Chomsky hierarchy to define more powerful classes of document spanners has been recently used in [14].

In the context of this paper, however, the main benefit is that this approach provides a suitable angle to treat string-equality selections in a regular way.

3.1 Subword-Marked Words

For any set $\mathcal{X}$ of variables, we shall use the set $\Gamma_{\mathcal{X}} = \{{}^{\mathsf{x}}\!\triangleright, \triangleleft^{\mathsf{x}} \mid \mathsf{x} \in \mathcal{X}\}$ as an alphabet of meta-symbols. In particular, for every $\mathsf{x} \in \mathcal{X}$, we interpret the pair of symbols ${}^{\mathsf{x}}\!\triangleright$ and $\triangleleft^{\mathsf{x}}$ as a pair of opening and closing parentheses.

▶ **Definition 3.1** (Subword-Marked Words). *A* subword-marked *word (*over terminal alphabet Σ and variables $\mathcal{X}$*) is a word $w \in (\Sigma \cup \Gamma_{\mathcal{X}})^*$ such that, for every $\mathsf{x} \in \mathcal{X}$, $\mathsf{e}_{\Sigma \cup \Gamma_{\mathcal{X} \setminus \{\mathsf{x}\}}}(w) \in \{\varepsilon,\ {}^{\mathsf{x}}\!\triangleright \triangleleft^{\mathsf{x}}\}$. A subword-marked word is* functional*, if $|w|_{{}^{\mathsf{x}}\triangleright} = 1$ for every $\mathsf{x} \in \mathcal{X}$. For a subword-marked word w over Σ and $\mathcal{X}$, we set $\mathfrak{e}(w) = \mathsf{e}_{\Gamma_{\mathcal{X}}}(w)$.*

A subword-marked word w can be interpreted as a word over Σ, i. e., the word $\mathfrak{e}(w)$, in which some subwords are marked by means of the parentheses ${}^{\mathsf{x}}\!\triangleright$ and $\triangleleft^{\mathsf{x}}$. In this way, it represents an $(\mathcal{X}, \mathfrak{e}(w))$-tuple, i. e., every $\mathsf{x} \in \mathcal{X}$ is mapped to $[i, j\rangle \in \mathsf{Spans}(\mathfrak{e}(w))$, where $w = w_1 \, {}^{\mathsf{x}}\!\triangleright \, w_2 \triangleleft^{\mathsf{x}} w_3$ with $i = |\mathfrak{e}(w_1)| + 1$ and $j = |\mathfrak{e}(w_1 w_2)| + 1$. In the following, the $(\mathcal{X}, \mathfrak{e}(w))$-tuple defined by a subword-marked word w is denoted by $\mathsf{st}(w)$. We note that $\mathsf{st}(w)$ is a total function if and only if w is functional. Moreover, we say that a subword-marked word w is *hierarchical* or *quasi-disjoint*, if $\mathsf{st}(w)$ is hierarchical or quasi-disjoint, respectively.

[1] In the literature on spanners, subword-marked words have previously been used as a tool to define the semantics of regex-formulas or vset-automata (see, e. g., [3, 7, 9, 10]). However, in these papers, the term *ref-word* is used instead of subword-marked word, which is a bit of a misnomer due to the following reasons. Ref-words have originally been used in [17] (in a different context) as words that contain *references* to some of their subwords, which are explicitly marked. In the context of spanners, only ref-words with marked subwords, but *without* any references have been used so far. Since in this work we wish to use ref-words in the sense of [17], i. e., with actual references, but also the variants without references, we introduce the term subword-marked word for the latter.

▶ **Example 3.2.** Let $\mathcal{X} = \{\mathsf{x},\mathsf{y},\mathsf{z}\}$ and $\Sigma = \{\mathsf{a},\mathsf{b},\mathsf{c}\}$. Then ${}^{\mathsf{x}}\!\triangleright \mathsf{aa} \triangleleft^{\mathsf{x}} \mathsf{ab}\ {}^{\mathsf{y}}\!\triangleright\ {}^{\mathsf{z}}\!\triangleright \mathsf{ca} \triangleleft^{\mathsf{z}} \mathsf{a} \triangleleft^{\mathsf{y}}$ is a functional and hierarchical subword-marked word. The subword-marked word $u = \mathsf{b}\ {}^{\mathsf{x}}\!\triangleright \mathsf{a}\ {}^{\mathsf{y}}\!\triangleright \mathsf{aba}\ {}^{\mathsf{z}}\!\triangleright \mathsf{a} \triangleleft^{\mathsf{z}} \mathsf{c} \triangleleft^{\mathsf{x}} \mathsf{ab} \triangleleft^{\mathsf{y}} \mathsf{c}$ is functional, but not hierarchical, while $v = {}^{\mathsf{x}}\!\triangleright \mathsf{a}\ {}^{\mathsf{y}}\!\triangleright \mathsf{ba} \triangleleft^{\mathsf{y}} \mathsf{cab} \triangleleft^{\mathsf{x}} \mathsf{caa}$ is a non-functional, but hierarchical subword-marked word. Moreover, $\mathsf{st}(u) = ([2,8\rangle, [3,10\rangle, [6,7\rangle)$ and $\mathsf{st}(v) = ([1,7\rangle, [2,4\rangle, \bot)$. On the other hand, neither ${}^{\mathsf{x}}\!\triangleright \mathsf{aa} \triangleleft^{\mathsf{x}} \mathsf{ab}\ {}^{\mathsf{y}}\!\triangleright\ {}^{\mathsf{x}}\!\triangleright \mathsf{ca} \triangleleft^{\mathsf{x}} \mathsf{a} \triangleleft^{\mathsf{y}}$ nor ${}^{\mathsf{x}}\!\triangleright \mathsf{a}\ {}^{\mathsf{y}}\!\triangleright \mathsf{ba} \triangleleft^{\mathsf{x}} \mathsf{c}$ are valid subword-marked words.

3.2 Subword-Marked Languages and Spanners

A set L of subword-marked words (over Σ and $\mathcal{X}$) is a *subword-marked language* (*over* Σ *and* $\mathcal{X}$); a subword-marked language L is called *functional*, *hierarchical* or *quasi-disjoint* if all $w \in L$ are functional, hierarchical or quasi-disjoint, respectively. Since every subword-marked word w over Σ and $\mathcal{X}$ describes a $(\mathcal{X}, \mathfrak{e}(w))$-tuple, subword-marked languages can be interpreted as spanners as follows.

▶ **Definition 3.3.** *Let L be a subword-marked language (over Σ and $\mathcal{X}$). Then the spanner $[\![L]\!]$ (over Σ and $\mathcal{X}$) is defined as follows: for every $w \in \Sigma^*$, $[\![L]\!](w) = \{\mathsf{st}(v) \mid v \in L, \mathfrak{e}(v) = w\}$. For a class $\mathcal{L}$ of subword-marked languages, we set $[\![\mathcal{L}]\!] = \{[\![L]\!] \mid L \in \mathcal{L}\}$.*

▶ **Example 3.4.** Let $\Sigma = \{\mathsf{a},\mathsf{b}\}$ and $\mathcal{X} = \{\mathsf{x}_1,\mathsf{x}_2,\mathsf{x}_3\}$. Let $\alpha = {}^{\mathsf{x}_1}\!\triangleright (\mathsf{a} \vee \mathsf{b})^* \triangleleft^{\mathsf{x}_1}\ {}^{\mathsf{x}_2}\!\triangleright \mathsf{b} \triangleleft^{\mathsf{x}_2}\ {}^{\mathsf{x}_3}\!\triangleright (\mathsf{a} \vee \mathsf{b})^* \triangleleft^{\mathsf{x}_3}$ be a regular expression over the alphabet $\Sigma \cup \Gamma_{\mathcal{X}}$. We can note that $\mathcal{L}(\alpha)$ is a subword-marked language (over terminal alphabet Σ and variables $\mathcal{X}$) and therefore $[\![\mathcal{L}(\alpha)]\!]$ is a spanner over $\mathcal{X}$. In fact, $[\![\mathcal{L}(\alpha)]\!]$ is exactly the spanner described by the function S in Example 2.1.

In this way, every subword-marked language L over Σ and $\mathcal{X}$ describes a spanner $[\![L]\!]$ over Σ and $\mathcal{X}$, and since it is also easy to transform any $(\mathcal{X}, w)$-tuple t into a subword-marked word v with $\mathfrak{e}(v) = w$ and $\mathsf{st}(v) = t$, also every spanner S over Σ and $\mathcal{X}$ can be represented by a subword-marked language over Σ and $\mathcal{X}$. Moreover, for a subword-marked language L over Σ and $\mathcal{X}$, $[\![L]\!]$ is a functional, hierarchical or quasi-disjoint spanner if and only if L is functional, hierarchical or quasi-disjoint, respectively. This justifies that we can use the concepts of spanners (over Σ and $\mathcal{X}$) and the concept of subword-marked languages (over Σ and $\mathcal{X}$) completely interchangeably. By considering only *regular* subword-marked languages, we automatically obtain the class of regular spanners (usually defined as the class of spanners that can be described by vset-automata [4, 12]). More formally, let $\mathsf{reg\text{-}swm\text{-}}\mathfrak{L}_{\Sigma,\mathcal{X}}$ be the class of regular subword-marked languages over Σ and $\mathcal{X}$ and let $\mathsf{reg\text{-}swm\text{-}}\mathfrak{L} = \bigcup_{\Sigma,\mathcal{X}} \mathsf{reg\text{-}swm\text{-}}\mathfrak{L}_{\Sigma,\mathcal{X}}$.

▶ **Proposition 3.5.** $\mathsf{reg\text{-}}\mathfrak{S} = [\![\mathsf{reg\text{-}swm\text{-}}\mathfrak{L}]\!]$.

It is a straightforward, but important observation that for any given NFA over $\Sigma \cup \Gamma_{\mathcal{X}}$, we can efficiently check whether $\mathcal{L}(M)$ is a subword-marked language. Since most description mechanisms for regular languages (e. g., expressions, grammars, logics, etc.) easily translate into NFA, they can potentially all be used for defining regular spanners.

▶ **Proposition 3.6.** *Given an* NFA *M over alphabet $\Sigma \cup \Gamma_{\mathcal{X}}$, we can decide in time $\mathrm{O}(|M||\mathcal{X}|^2)$ if $\mathcal{L}(M)$ is a subword-marked language, and, if so, whether $\mathcal{L}(M)$ is functional in time $\mathrm{O}(|M||\mathcal{X}|^2)$, and whether it is hierarchical or quasi-disjoint in time $\mathrm{O}(|M||\mathcal{X}|^3)$.*

4 Refl-Spanners: Spanners with Built-In String-Equality Selections

In this section, we extend the concept of subword-marked words and languages in order to describe spanners with string-equality selections.

4.1 Ref-Words and Ref-Languages

We consider subword-marked words with extended terminal alphabet $\Sigma \cup \mathcal{X}$, i. e., in addition to symbols from Σ, also variables from $\mathcal{X}$ can appear as terminal symbols (the marking of subwords with symbols $\Gamma_{\mathcal{X}}$ remains unchanged).

▶ **Definition 4.1** (Ref-Words). *A* ref-word over Σ and $\mathcal{X}$ *is a subword-marked word over terminal alphabet $\Sigma \cup \mathcal{X}$ and variables $\mathcal{X}$, such that, for every $\mathsf{x} \in \mathcal{X}$, if $w = w_1 \mathsf{x} w_2$, then there exist words v_1, v_2, v_3 such that $w_1 = v_1 \, {}^{\mathsf{x}}\!\triangleright \, v_2 \lhd^{\mathsf{x}} v_3$.*

Since ref-words are subword-marked words, the properties “functional”, “hierarchical” and “quasi-disjoint” are well-defined.

▶ **Example 4.2.** Let $\Sigma = \{\mathtt{a}, \mathtt{b}, \mathtt{c}\}$ and $\mathcal{X} = \{\mathsf{x}, \mathsf{y}\}$. The subword-marked word $u = \mathtt{ab} \, {}^{\mathsf{x}}\!\triangleright \, \mathtt{ab} \lhd^{\mathsf{x}} \mathtt{c} \, {}^{\mathsf{y}}\!\triangleright \, \mathtt{xaa} \lhd^{\mathsf{y}} \mathtt{y}$ and $v = \mathtt{a} \, {}^{\mathsf{x}}\!\triangleright \, \mathtt{ab} \, {}^{\mathsf{y}}\!\triangleright \, \mathtt{ab} \lhd^{\mathsf{x}} \mathtt{a} \lhd^{\mathsf{y}} \mathtt{xy}$ (over terminal alphabet $\Sigma \cup \mathcal{X}$ and variables $\mathcal{X}$) are valid ref-words (over terminal alphabet Σ and variables $\mathcal{X}$). Note that both u and v are functional, and u is also hierarchical, while v is not. On the other hand, $\mathtt{axb} \, {}^{\mathsf{x}}\!\triangleright \, \mathtt{ab} \lhd^{\mathsf{x}} \mathtt{c} \, {}^{\mathsf{y}}\!\triangleright \, \mathtt{xaa} \lhd^{\mathsf{y}} \mathtt{y}$ and $\mathtt{aa} \, {}^{\mathsf{x}}\!\triangleright \, \mathtt{ab} \lhd^{\mathsf{x}} \mathtt{c} \, {}^{\mathsf{y}}\!\triangleright \, \mathtt{ya} \lhd^{\mathsf{y}}$ are subword-marked words (over terminal alphabet $\Sigma \cup \mathcal{X}$ and variables $\mathcal{X}$), but not ref-words.

The idea of ref-words is that occurrences of $\mathsf{x} \in \mathcal{X}$ are interpreted as *references* to the subword ${}^{\mathsf{x}}\!\triangleright \, v \lhd^{\mathsf{x}}$, which we will call *definition* of x. Note that while a single variable can have several references, there is at most one definition per variable, and if there is no definition for a variable, then it also has no references. Variable definitions may contain other references or definitions of other variables, i. e., there may be chains of references, e. g., the definition of x contains references of y, and the definition of y contains references of z and so on. Next, we formally define this nested referencing process encoded by ref-words. Recall that for a subword-marked word w by $\mathfrak{e}(w)$ we denote the word obtained by removing all meta-symbols from $\Gamma_{\mathcal{X}}$ from w (however, if w is a ref-word, then $\mathfrak{e}(w)$ is a word over $\Sigma \cup \mathcal{X}$).

▶ **Definition 4.3** (Deref-Function). *For a ref-word w over Σ and $\mathcal{X}$, the subword-marked word $\mathfrak{d}(w)$ over Σ and $\mathcal{X}$ is obtained from w by repeating the following steps until we have a subword-marked word over Σ and $\mathcal{X}$:*

1. *Let ${}^{\mathsf{x}}\!\triangleright \, v_{\mathsf{x}} \lhd^{\mathsf{x}}$ be a definition such that $\mathfrak{e}(v_{\mathsf{x}}) \in \Sigma^*$.*
2. *Replace all occurrences of x in w by $\mathfrak{e}(v_{\mathsf{x}})$.*

It is straightforward to verify that the function $\mathfrak{d}(\cdot)$ is well-defined. By using this function and because ref-words encode subword-marked words, they can be interpreted as span-tuples. More precisely, for every ref-word w over Σ and $\mathcal{X}$, $\mathfrak{d}(w)$ is a subword-marked word over Σ and $\mathcal{X}$, $\mathfrak{e}(\mathfrak{d}(w)) \in \Sigma^*$ and $\mathsf{st}(\mathfrak{d}(w))$ is an $(\mathcal{X}, \mathfrak{e}(\mathfrak{d}(w)))$-tuple.

▶ **Example 4.4.** Let $\Sigma = \{\mathtt{a}, \mathtt{b}, \mathtt{c}\}$, let $\mathcal{X} = \{\mathsf{x}_1, \mathsf{x}_2, \mathsf{x}_3, \mathsf{x}_4\}$ and let

$$w = \mathtt{aa} \, {}^{\mathsf{x}_1}\!\triangleright \, \mathtt{ab} \, {}^{\mathsf{x}_2}\!\triangleright \, \mathtt{acc} \lhd^{\mathsf{x}_2} \mathtt{a}\mathsf{x}_2 \lhd^{\mathsf{x}_1} \; {}^{\mathsf{x}_4}\!\triangleright \, \mathsf{x}_1\mathtt{a}\mathsf{x}_2 \lhd^{\mathsf{x}_4} \mathsf{x}_4\mathtt{b}\mathsf{x}_1 \, .$$

Due to the definition ${}^{\mathsf{x}_2}\!\triangleright \, \mathtt{acc} \lhd^{\mathsf{x}_2}$, the procedure of Definition 4.3 will initially replace all references of x_2 by $\mathtt{acc}$. Then, the definition for variable x_1 is ${}^{\mathsf{x}_1}\!\triangleright \, \mathtt{ab} \, {}^{\mathsf{x}_2}\!\triangleright \, \mathtt{acc} \lhd^{\mathsf{x}_2} \mathtt{aacc} \lhd^{\mathsf{x}_1}$, so $\mathtt{abaccaacc}$ can be substituted for the references of x_1. After replacing the last variable x_4, we obtain

$$\mathfrak{d}(w) = \mathtt{aa} \, {}^{\mathsf{x}_1}\!\triangleright \, \mathtt{ab} \, {}^{\mathsf{x}_2}\!\triangleright \, \mathtt{acc} \lhd^{\mathsf{x}_2} \mathtt{aacc} \lhd^{\mathsf{x}_1} \; {}^{\mathsf{x}_4}\!\triangleright \, \mathtt{abaccaaccaacc} \lhd^{\mathsf{x}_4} \mathtt{abaccaaccaaccbabaccaacc} \, .$$

Moreover, we have $\mathsf{st}(\mathfrak{d}(w)) = ([3, 12\rangle, [5, 8\rangle, \bot, [12, 25\rangle)$.

As a non-hierarchical example, consider $u = {}^{\mathsf{x}_1}\!\triangleright\, \mathtt{a}\; {}^{\mathsf{x}_2}\!\triangleright\, \mathtt{aa} \lhd^{\mathsf{x}_1} \mathtt{c}\mathsf{x}_1\; {}^{\mathsf{x}_3}\!\triangleright\, \mathtt{ac} \lhd^{\mathsf{x}_2} \mathsf{x}_2\mathtt{a}\mathsf{x}_1 \lhd^{\mathsf{x}_3}$. It can be easily verified that $\mathfrak{d}(u) = {}^{\mathsf{x}_1}\!\triangleright\, \mathtt{a}\; {}^{\mathsf{x}_2}\!\triangleright\, \mathtt{aa} \lhd^{\mathsf{x}_1} \mathtt{caaa}\; {}^{\mathsf{x}_3}\!\triangleright\, \mathtt{ac} \lhd^{\mathsf{x}_2} \mathtt{aacaaaacaaaa} \lhd^{\mathsf{x}_3}$ and $\mathsf{st}(\mathfrak{d}(u)) = ([1,4\rangle, [2,10\rangle, [8,22\rangle, \bot)$.

A set L of ref-words is called *ref-language* and we extend the $\mathfrak{d}(\cdot)$-function to ref-languages L in the obvious way, i. e., $\mathfrak{d}(L) = \{\mathfrak{d}(w) \mid w \in L\}$. As for subword-marked languages, we are especially interested in ref-languages that are regular. By $\mathsf{reg\text{-}ref\text{-}}\mathfrak{L}_{\Sigma,\mathcal{X}}$ we denote the class of regular ref-languages over Σ and $\mathcal{X}$, and we set $\mathsf{reg\text{-}ref\text{-}}\mathfrak{L} = \bigcup_{\Sigma,\mathcal{X}} \mathsf{reg\text{-}ref\text{-}}\mathfrak{L}_{\Sigma,\mathcal{X}}$.

In analogy to Proposition 3.6, we can easily check for a given NFA over alphabet $\Sigma \cup \Gamma_{\mathcal{X}} \cup \mathcal{X}$ whether it accepts a ref-language over Σ and $\mathcal{X}$.

▶ **Proposition 4.5.** *Given an* NFA M *where* $\mathcal{L}(M)$ *is a subword-marked language over* $\Sigma \cup \mathcal{X}$ *and* $\mathcal{X}$*, we can decide in time* $\mathrm{O}(|M||\mathcal{X}|^2)$ *if* $\mathcal{L}(M)$ *is a ref-language over* Σ *and* $\mathcal{X}$*.*

4.2 Refl-Spanners

We shall now define spanners based on regular ref-languages.

▶ **Definition 4.6** (Refl-Spanners)**.** *Let* L *be a ref-language (over* Σ *and* $\mathcal{X}$*). Then the* refl-spanner $[\![L]\!]_{\mathfrak{d}}$ *(over* Σ *and* $\mathcal{X}$*) is defined by* $[\![L]\!]_{\mathfrak{d}} = [\![\mathfrak{d}(L)]\!]$*. For a class* $\mathfrak{L}$ *of ref-languages, we set* $[\![\mathfrak{L}]\!]_{\mathfrak{d}} = \{[\![L]\!]_{\mathfrak{d}} \mid L \in \mathfrak{L}\}$*, and the class of* refl-spanners *is* $\mathsf{refl\text{-}}\mathfrak{S} = [\![\mathsf{reg\text{-}ref\text{-}}\mathfrak{L}]\!]_{\mathfrak{d}}$.

Since any regular ref-language L over Σ and $\mathcal{X}$ is also a regular subword-marked language over $\Sigma \cup \mathcal{X}$ and $\mathcal{X}$, $[\![L]\!]$ is, according to Definition 3.3, also a well-defined spanner (but over $\Sigma \cup \mathcal{X}$ and $\mathcal{X}$). However, whenever we are concerned with a ref-language L over Σ and $\mathcal{X}$ that is not also a subword-marked language over Σ and $\mathcal{X}$ (i. e., L contains actual occurrences of symbols from $\mathcal{X}$), then we are never interested in $[\![L]\!]$, but always in $[\![L]\!]_{\mathfrak{d}}$. Consequently, by a slight abuse of notation, we denote in this case $[\![L]\!]_{\mathfrak{d}}$ simply by $[\![L]\!]$.

For a regular ref-language L the corresponding refl-spanner $[\![L]\!]$ produces for a given $w \in \Sigma^*$ all $(\mathcal{X}, w)$-tuples t that are represented by some $u \in \mathfrak{d}(L)$ with $\mathfrak{e}(u) = w$, or, equivalently, all $(\mathcal{X}, w)$-tuples t with $t = \mathsf{st}(\mathfrak{d}(v))$ and $\mathfrak{e}(\mathfrak{d}(v)) = w$ for some $v \in L$. It is intuitively clear that the use of variable references of refl-spanners provide a functionality that resembles string-equality selections for core spanners. However, there are also obvious differences between these two spanner formalisms (as already mentioned in the introduction and as investigated in full detail in Section 6).

Before moving on to the actual results about refl-spanners, we shall briefly discuss another example.

▶ **Example 4.7.** Assume that we have a document $w = p_1\#p_2\#\ldots\#p_n$, where each $p_i \in \Sigma^*$ is the title page of a scientific paper and $\# \notin \Sigma$ is some separator symbol (e. g., a list of all title pages of papers in the issues of *Journal of the ACM* from 2000 to 2010). Let $\Sigma' = \Sigma \cup \{\#\}$. We define a refl-spanner

$$\alpha = \Sigma'^* \,\#\, \Sigma^* \;\mathsf{email:}\; {}^{\mathsf{x}}\!\triangleright \Sigma^+ \lhd^{\mathsf{x}} \;\mathtt{@}\; r_{\mathsf{dom}}\, \Sigma^* \,\#\, \Sigma'^* \;\mathsf{email:}\; \mathsf{x} \;\mathtt{@}\; r_{\mathsf{dom}}\, \Sigma^* \,\#\, \Sigma'^* \,,$$

where $r_{\mathsf{dom}} = \mathsf{hu\text{-}berlin.de} \vee \mathsf{tu\text{-}berlin.de} \vee \mathsf{fu\text{-}berlin.de}$ is a regular expressions that matches the email-domains of the three universities in Berlin. It can be easily seen that $[\![\alpha]\!](w)$ contains the first parts of the email-addresses of authors that have at least two JACM-papers between year 2000 and 2010 while working at a university in Berlin.

5 Evaluation and Static Analysis of Refl-Spanners

The problem ModelChecking is to decide whether $t \in S(w)$ for given spanner S over Σ and $\mathcal{X}$, $w \in \Sigma^*$ and $(\mathcal{X}, w)$-tuple t, and NonEmptiness is to decide $S(w) \neq \emptyset$ for given S and w. For the problems Satisfiability, Hierarchicality and Functionality, we get a single spanner S as input and ask whether there is a $w \in \Sigma^*$ with $S(w) \neq \emptyset$, whether S is hierarchical, or whether S is functional, respectively. Finally, Containment and Equivalence is to decide whether $S_1 \subseteq S_2$ or $S_1 = S_2$, respectively, for given spanners S_1 and S_2. The input refl-spanners are always given as NFA. Recall that a summary of our results is provided by Table 1 in the introduction.

▶ **Theorem 5.1.** ModelChecking *for* refl-$\mathfrak{S}$ *can be solved in time poly*$(|M|)(|w| + (2|\mathcal{X}|)!)$*, where M is an* NFA *that represents a refl-spanner $S = [\![\mathcal{L}(M)]\!]$ over Σ and $\mathcal{X}$, $w \in \Sigma^*$, and t is an $(\mathcal{X}, w)$-tuple. In case that S is functional or M is normalised (in the sense of Definition 5.4 that follows further below), this can be improved to poly*$(|M|)(|w| + |\mathcal{X}|)$ *and* $\mathrm{O}(|M|(|w| + |\mathcal{X}|))$*, respectively.*

Proof Sketch. To check $t \in [\![\mathcal{L}(M)]\!](w)$ it is sufficient to check whether there is some $v \in \mathcal{L}(M)$ with $\mathsf{st}(\mathfrak{d}(v)) = t$ and $\mathfrak{e}(\mathfrak{d}(v)) = w$. To this end, we turn w into a subword-marked word $\tilde{w}$ by inserting the symbols $\Gamma_{\mathcal{X}}$ according to t, but, since we do not know in which order the factors over $\Gamma_{\mathcal{X}}$ are read by M, we represent the maximal factors over $\Gamma_{\mathcal{X}}$ as sets (represented as single symbols) rather than words over $\Gamma_{\mathcal{X}}$. Moreover, for every $\mathsf{x} \in \mathcal{X}$, let u_{x} be the factor of w that corresponds to $t(\mathsf{x})$, if defined. Then we check whether M can accept $\tilde{w}$, but we treat x-transitions with $\mathsf{x} \in \mathcal{X}$ of M as labelled with u_{x}, and whenever we encounter a symbol that represents a subset of $\Gamma_{\mathcal{X}}$, then we have to compute in a brute-force manner which states are reachable by a path that reads exactly the symbols from this set (which causes the factor $(2|\mathcal{X}|)!$ in the running-time). Moreover, in order to not introduce another $|w|$ factor, we use a longest common extension data-structure to be able to consume prefixes u_{x} from $\tilde{w}$, i. e., to handle the x-transitions, in constant time.

If S is functional, then each two states q, q' of M uniquely determine which $u \in (\Gamma_{\mathcal{X}})^*$, if any, can be read while moving from q to q' (this observation has been used in a similar way by Freydenberger, Kimelfeld, and Peterfreund [9] for vset-automata). If M is normalised, we know exactly in which order the symbols from $\Gamma_{\mathcal{X}}$ have to be inserted into w in order to construct $\tilde{w}$. In both cases, this means that the brute-force part is not necessary. ◀

We discuss a few particularities about Theorem 5.1. The mentioned general running-time is not polynomial (in combined complexity), but nevertheless fixed-parameter tractable with respect to parameter $|\mathcal{X}|$. This is worth mentioning since for core spanners ModelChecking is W[1]-hard with respect to parameter $|\mathcal{X}|$ (this can be concluded from [8] and [5]). If M is already normalised (in the sense of Definition 5.4), then, in the likely case that $|\mathcal{X}| \in \mathrm{O}(|w|)$, we obtain a running-time of $\mathrm{O}(|M||w|)$ which is also known to be a lower-bound (subject to the Strong Exponential Time Hypothesis SETH) for regular expression matching (see [2] for further details), which obviously reduces to ModelChecking for refl-$\mathfrak{S}$.

We further obtain the following two theorems, which are proved by standard methods.

▶ **Theorem 5.2.** NonEmptiness *for* refl-$\mathfrak{S}$ *is* NP*-complete.*

▶ **Theorem 5.3.**
(a) Satisfiability *for* refl-$\mathfrak{S}$ *can be solved in time* $\mathrm{O}(|M|)$*,*
(b) Hierarchicality *for* refl-$\mathfrak{S}$ *can be solved in time* $\mathrm{O}(|M||\mathcal{X}|^3)$*, and*
(c) Functionality *for* refl-$\mathfrak{S}$ *can be solved in time* $\mathrm{O}(|M||\mathcal{X}|^2)$*,*
where M is an NFA *that describes a refl-spanner over Σ and $\mathcal{X}$.*

For core spanners, Satisfiability and Hierarchicality are PSpace-complete, even for restricted classes of core spanners (see [8]), and Containment and Equivalence are not semi-decidable (see [8]). We now show that for refl-spanners we can achieve decidability of Containment and Equivalence by imposing suitable restrictions. The goal is to reduce the containment of refl-spanners to the containment of their corresponding ref-languages, but the problem is that the correspondence between ref-words and the span-tuples they describe is not unique in 2 ways. (1): Different subword-marked words w and w' with $\mathfrak{e}(w) = \mathfrak{e}(w')$ can nevertheless describe the same span-tuple, i. e., $\mathsf{st}(w) = \mathsf{st}(w')$, since the order of consecutive occurrences of symbols from $\Gamma_{\mathcal{X}}$ has no impact on the represented span-tuple. And (2): The same subword-marked word can be the $\mathfrak{d}(\cdot)$-image of two different ref-words v and v', i. e., $\mathfrak{d}(v) = \mathfrak{d}(v')$, e. g., the ref-words $w_1 = {}^{\mathsf{x}}\triangleright\mathtt{ab}\triangleleft^{\mathsf{x}}\mathtt{b}\,{}^{\mathsf{y}}\triangleright\mathtt{abb}\triangleleft^{\mathsf{y}}\mathtt{yab}$, $w_2 = {}^{\mathsf{x}}\triangleright\mathtt{ab}\triangleleft^{\mathsf{x}}\mathtt{b}\,{}^{\mathsf{y}}\triangleright\mathtt{xb}\triangleleft^{\mathsf{y}}\mathtt{xbx}$ and $w_3 = {}^{\mathsf{x}}\triangleright\mathtt{ab}\triangleleft^{\mathsf{x}}\mathtt{b}\,{}^{\mathsf{y}}\triangleright\mathtt{abb}\triangleleft^{\mathsf{y}}\mathtt{yx}$ are all $\preceq$-normalised (in the sense of Definition 5.4 that follows further below), where ${}^{\mathsf{x}}\triangleright \preceq \triangleleft^{\mathsf{x}} \preceq {}^{\mathsf{y}}\triangleright \preceq \triangleleft^{\mathsf{y}}$. However, $\mathfrak{d}(w_1) = \mathfrak{d}(w_2) = \mathfrak{d}(w_3) = {}^{\mathsf{x}}\triangleright\, \mathtt{ab} \triangleleft^{\mathsf{x}}\, \mathtt{b}\, {}^{\mathsf{y}}\triangleright\, \mathtt{abb} \triangleleft^{\mathsf{y}}\, \mathtt{abbab}$.

We can deal with issue (1) by requiring for any two subword-marked languages L and L' that all consecutive occurrences of symbols from $\Gamma_{\mathcal{X}}$ are ordered in the same way, since then $[\![L]\!] \subseteq [\![L']\!]$ is characterised by $L \subseteq L'$. This is actually a rephrasing of an analogous result about vset-automata from [3]. However, issue (2) is independent of the order of symbols from $\Gamma_{\mathcal{X}}$, i. e., even if we assume that the symbols from $\Gamma_{\mathcal{X}}$ are ordered in the same way in two ref-languages L and L', it is still not necessarily the case that $[\![L]\!] \subseteq [\![L']\!]$ is characterised by $L \subseteq L'$. In order to deal with issue (2) we need to impose some actual restrictions on ref-languages. Intuitively speaking, we require all variable references to be extracted by their own private extraction variable, i. e., in the ref-words we encounter all variable references x in the form ${}^{\mathsf{y_x}}\triangleright\, \mathsf{x} \triangleleft^{\mathsf{y_x}}$, where $\mathsf{y_x}$ has in all ref-words the sole purpose of extracting the content of some reference of variable x. With this requirement, the positions of the repeating factors described by variables and their references must be explicitly present as spans in the span-tuples. This seems like a strong restriction for refl-spanners, but we should note that for core spanners we necessarily have a rather similar situation: if we want to use string equality selections on some spans, we have to explicitly extract them by variables first.

Before we formally define the restriction of ref-languages mentioned above, we first develop some technical tools on the level of subword-marked languages.

▶ **Definition 5.4.** *A linear order $\preceq$ on $\Gamma_{\mathcal{X}}$ is* valid *if, for every $\mathsf{x} \in \mathcal{X}$, ${}^{\mathsf{x}}\triangleright \preceq \triangleleft^{\mathsf{x}}$. A subword-marked word w over Σ and $\mathcal{X}$ is* $\preceq$-normalised *for a valid linear order $\preceq$ on $\Gamma_{\mathcal{X}}$, if, for every $\sigma_1, \sigma_2 \in \Gamma_{\mathcal{X}}$ and every factor $\sigma_1\sigma_2$ in w, we have that $\sigma_1 \preceq \sigma_2$. A subword-marked word over Σ and $\mathcal{X}$ is* normalised, *if it is $\preceq$-normalised for some valid linear order $\preceq$ on $\Gamma_{\mathcal{X}}$.*

▶ **Example 5.5.** Let $\mathcal{X} = \{\mathsf{x}, \mathsf{y}, \mathsf{z}\}$ and let $\preceq$ be the valid order on $\Gamma_{\mathcal{X}}$ with ${}^{\mathsf{x}}\triangleright \preceq {}^{\mathsf{y}}\triangleright \preceq {}^{\mathsf{z}}\triangleright \preceq \triangleleft^{\mathsf{x}} \preceq \triangleleft^{\mathsf{y}} \preceq \triangleleft^{\mathsf{z}}$. Then $\mathtt{a}\, {}^{\mathsf{x}}\triangleright\, {}^{\mathsf{y}}\triangleright\, \mathtt{ab} \triangleleft^{\mathsf{y}}\, {}^{\mathsf{z}}\triangleright\, \triangleleft^{\mathsf{x}}\, \mathtt{b}\triangleleft^{\mathsf{z}}$ and ${}^{\mathsf{z}}\triangleright\, {}^{\mathsf{x}}\triangleright\, \mathtt{ab} \triangleleft^{\mathsf{z}}\, \mathtt{a}\triangleleft^{\mathsf{x}}$ are *not* $\preceq$-normalised. On the other hand, $\mathtt{a}\, {}^{\mathsf{x}}\triangleright\, {}^{\mathsf{y}}\triangleright\, \mathtt{ab}\, {}^{\mathsf{z}}\triangleright\, \triangleleft^{\mathsf{x}}\, \triangleleft^{\mathsf{y}}\, \mathtt{b}\triangleleft^{\mathsf{z}}$ and ${}^{\mathsf{z}}\triangleright\, \mathtt{a}\, {}^{\mathsf{x}}\triangleright\, \mathtt{ab} \triangleleft^{\mathsf{z}}\, \mathtt{a}\triangleleft^{\mathsf{x}}$ are $\preceq$-normalised.

▶ **Lemma 5.6.** *Let w_1 and w_2 be subword-marked words over Σ and $\mathcal{X}$ that are $\preceq$-normalised for a valid linear order $\preceq$ and that satisfy $\mathfrak{e}(w_1) = \mathfrak{e}(w_2)$. Then $\mathsf{st}(w_1) = \mathsf{st}(w_2)$ if and only if $w_1 = w_2$.*

This concept of normalised subword-marked words directly carries over to subword-marked languages: a subword-marked language L over Σ and $\mathcal{X}$ is $\preceq$-*normalised*, if every $w \in L$ is $\preceq$-normalised; and a subword-marked language over Σ and $\mathcal{X}$ is *normalised*, if it is $\preceq$-normalised for some valid linear order $\preceq$ on $\Gamma_{\mathcal{X}}$. Note that we do not only require all words of L to be normalised, but to be normalised with respect to the same valid order $\preceq$. And since ref-words are subword-marked words (and ref-languages are subword-marked languages), the concept of $\preceq$-normalisation also applies to ref-words and ref-languages.

Obviously, if L and K are $\preceq$-normalised subword-marked languages over Σ and $\mathcal{X}$, then $L \cup K$ is a $\preceq$-normalised subword-marked languages over Σ and $\mathcal{X}$.

For a given subword-marked language L over Σ and $\mathcal{X}$, and a valid order $\preceq$, we can obtain a $\preceq$-normalised subword-marked language L' over Σ and $\mathcal{X}$ with $[\![L]\!] = [\![L']\!]$ by simply reordering all maximal factors over $\Gamma_{\mathcal{X}}$ in the words of L according to $\preceq$. The next lemma shows how to effectively do this if L is regular and represented by an NFA M.

▶ **Lemma 5.7.** *Let M be an* NFA *that accepts a subword-marked language over Σ and $\mathcal{X}$, and let $\preceq$ be a valid linear order on $\Gamma_{\mathcal{X}}$. Then we can compute in time $\mathrm{O}(2^{|\Gamma_{\mathcal{X}}|}|M|)$ an* NFA *M' of size $\mathrm{O}(2^{|\Gamma_{\mathcal{X}}|}|M|)$ such that $[\![\mathcal{L}(M)]\!] = [\![\mathcal{L}(M')]\!]$ and $\mathcal{L}(M')$ is a $\preceq$-normalised subword-marked language over Σ and $\mathcal{X}$.*

The general idea for the lemma's proof is that whenever M reads a sequence of symbols from $\Gamma_{\mathcal{X}}$, then M' instead stores the letters of this sequence in the finite state control and makes ε-transitions instead. Whenever M stops reading the $\Gamma_{\mathcal{X}}$-sequence and makes the next Σ-transition, then M' first reads the recorded symbols from $\Gamma_{\mathcal{X}}$ from the input, but according to the order $\preceq$, and then reads the next Σ-transition.

From Lemmas 5.6 and 5.7, we can directly derive a procedure for deciding Containment for regular spanners: given $S_1, S_2 \in \text{reg-}\mathfrak{S}_{\Sigma,\mathcal{X}}$ represented by NFA M_1 and M_2, choose an arbitrary valid order $\preceq$ on $\mathcal{X}$ and compute NFA M'_1 and M'_2 according to Lemma 5.7. Now Lemma 5.6 directly implies that $[\![\mathcal{L}(M_1)]\!] \subseteq [\![\mathcal{L}(M_2)]\!]$ if and only if $\mathcal{L}(M_1) \subseteq \mathcal{L}(M_2)$. An analogue of Lemmas 5.6 and 5.7 on the level of vset-automata is given in [3], although without stating an explicit upper complexity bound for the construction (moreover, a similar construction (without complexity bounds) is also used in [4] to show that regular spanners are closed under the join operator).

We next turn to the problem that different ref-words w_1 and w_2, even though both $\preceq$-normalised, can describe the same span-tuple, simply because $\mathfrak{d}(w_1) = \mathfrak{d}(w_2)$. For example, the ref-words $w_1 = {}^{\mathsf{x}}\rhd\, \mathsf{ab} \lhd^{\mathsf{x}} \mathsf{b}\, {}^{\mathsf{y}}\rhd\, \mathsf{abb} \lhd^{\mathsf{y}} \mathsf{yab}$, $w_2 = {}^{\mathsf{x}}\rhd\, \mathsf{ab} \lhd^{\mathsf{x}} \mathsf{b}\, {}^{\mathsf{y}}\rhd\, \mathsf{xb} \lhd^{\mathsf{y}} \mathsf{xbx}$ and $w_3 = {}^{\mathsf{x}}\rhd\, \mathsf{ab} \lhd^{\mathsf{x}} \mathsf{b}\, {}^{\mathsf{y}}\rhd\, \mathsf{abb} \lhd^{\mathsf{y}} \mathsf{yx}$ are all $\preceq$-normalised, where ${}^{\mathsf{x}}\rhd \preceq \lhd^{\mathsf{x}} \preceq {}^{\mathsf{y}}\rhd \preceq \lhd^{\mathsf{y}}$. However, $\mathfrak{d}(w_1) = \mathfrak{d}(w_2) = \mathfrak{d}(w_3) = {}^{\mathsf{x}}\rhd\, \mathsf{ab} \lhd^{\mathsf{x}} \mathsf{b}\, {}^{\mathsf{y}}\rhd\, \mathsf{abb} \lhd^{\mathsf{y}} \mathsf{abbab}$.

It is our goal to restrict ref-words in such a way that $w_1 \neq w_2$ implies $\mathfrak{d}(w_1) \neq \mathfrak{d}(w_2)$. We first explain this restriction on an intuitive level. The set of variables $\mathcal{X}$ is partitioned into a set $\mathcal{X}_r$ of *reference-variables* and, for each such reference-variable $\mathsf{x} \in \mathcal{X}_r$, into a set $\mathcal{X}_{e,\mathsf{x}}$ of *extraction-variables.* For every $\mathsf{x} \in \mathcal{X}$, every reference of x is extracted by some $\mathsf{y} \in \mathcal{X}_{e,\mathsf{x}}$, i. e., it occurs between ${}^{\mathsf{y}}\rhd$ and $\lhd^{\mathsf{y}}$; moreover, every $\mathsf{y} \in \mathcal{X}_{e,\mathsf{x}}$ either does not occur at all, or it is used as extractor for x, i. e., y's definition contains exactly one occurrence of a symbol $\Sigma \cup \mathcal{X}$ which is x. In addition, we also require that for every $\mathsf{x} \in \mathcal{X}_r$ with at least one reference, the image of x under $\mathfrak{d}(\cdot)$ is non-empty (otherwise the deref-function may turn normalised ref-words into non-normalised ones by joining two previously separate factors over $\Gamma_{\mathcal{X}}$).

▶ **Definition 5.8.** *A ref-word w over Σ and $\mathcal{X}$ is a* strongly reference extracting *ref-word over Σ and $(\mathcal{X}_r, \{\mathcal{X}_{e,\mathsf{x}} \mid \mathsf{x} \in \mathcal{X}_r\})$, if it satisfies the following:*

- *$\mathcal{X} = \mathcal{X}_r \cup \bigcup_{\mathsf{x} \in \mathcal{X}_r} \mathcal{X}_{e,\mathsf{x}}$, where all sets $\mathcal{X}_r$ and $\mathcal{X}_{e,\mathsf{x}}$ with $\mathsf{x} \in \mathcal{X}_r$ are pairwise disjoint.*
- *Each reference of an $\mathsf{x} \in \mathcal{X}_r$ in w occurs in a factor ${}^{\mathsf{y}}\rhd \gamma \mathsf{x} \delta \lhd^{\mathsf{y}}$ with $\mathsf{y} \in \mathcal{X}_{e,\mathsf{x}}$, $\gamma, \delta \in (\Gamma_{\mathcal{X}})^*$.*
- *If an $\mathsf{y} \in \mathcal{X}_{e,\mathsf{x}}$ has a definition in w, then it has the form ${}^{\mathsf{y}}\rhd \gamma \mathsf{x} \delta \lhd^{\mathsf{y}}$ with $\gamma, \delta \in (\Gamma_{\mathcal{X}})^*$, and $|w|_{\mathsf{y}} = 0$.*
- *For every $\mathsf{x} \in \mathcal{X}_r$ with $|w|_{\mathsf{x}} \neq 0$, $\mathsf{st}(\mathfrak{d}(w))(\mathsf{x}) = [i, j\rangle$ with $j - i \geqslant 2$.*

A ref-language L over Σ and $\mathcal{X}$ is a *strongly reference extracting* ref-language over Σ and $(\mathcal{X}_r, \{\mathcal{X}_{e,\mathsf{x}} \mid \mathsf{x} \in \mathcal{X}_r\})$ if every ref-word $w \in L$ is a strongly reference extracting ref-word over Σ and $(\mathcal{X}_r, \{\mathcal{X}_{e,\mathsf{x}} \mid \mathsf{x} \in \mathcal{X}_r\})$. By a series of intermediate results we obtain:

▶ **Lemma 5.9.** *Let L, K be two strongly reference extracting and $\preceq$-normalised ref-languages over Σ and $(\mathcal{X}_r, \{\mathcal{X}_{e,\mathsf{x}} \mid \mathsf{x} \in \mathcal{X}_r\})$. Then $[\![L]\!] \subseteq [\![K]\!]$ if and only if $L \subseteq K$.*

Let $\mathsf{sre\text{-}refl\text{-}}\mathfrak{S}_{\Sigma,(\mathcal{X}_r,\{\mathcal{X}_{e,\mathsf{x}} \mid \mathsf{x} \in \mathcal{X}_r\})}$ be the class of strongly reference extracting refl-spanners over Σ and $(\mathcal{X}_r, \{\mathcal{X}_{e,\mathsf{x}} \mid \mathsf{x} \in \mathcal{X}_r\})$.

▶ **Theorem 5.10.** Containment *and* Equivalence *for* $\mathsf{sre\text{-}refl\text{-}}\mathfrak{S}_{\Sigma,(\mathcal{X}_r,\{\mathcal{X}_{e,\mathsf{x}} \mid \mathsf{x} \in \mathcal{X}_r\})}$ *is in* ExpSpace *if the refl-spanners are given as* NFA*, in* PSpace *if the refl-spanners are given as $\preceq$-normalised* NFA*, and in* NLogSpace *if the refl-spanners S_1 and S_2 are given as $\preceq$-normalised* DFA*.*

Proof Sketch. Let M_1 and M_2 be the NFA that represent S_1 and S_2. For deciding Containment we first turn M_1 and M_2 into normalised variants M_1' and M_2' with Lemma 5.7 (which requires exponential space) and then check whether $\mathcal{L}(M_1) \subseteq \mathcal{L}(M_2)$ (which is known to be possible in polynomial space for NFA and nondeterministic logarithmic space for DFA). The correctness is a direct consequence of Lemma 5.9. This also shows the other bounds. ◀

6 Expressive Power of Refl-Spanners

It is a straightfoward observation that the expressive power of refl-spanners properly exceeds the one of regular spanners, but it is less clear which refl-spanners are also core spanners and which core spanners are refl-spanners. We first briefly discuss the former question.

A ref-language L over Σ and $\mathcal{X}$ is *reference-bounded* if there is a number k with $|w|_{\mathsf{x}} \leqslant k$ for every $\mathsf{x} \in \mathcal{X}$ and every $w \in L$. A refl-spanner is *reference-bounded* if it is represented by a reference-bounded ref-language. The following is an easy exercise.

▶ **Theorem 6.1.** *Every reference-bounded refl-spanner is a core spanner.*

It is interesting to note that the not reference-bounded refl-spanner $[\![\mathcal{L}(\mathtt{a}^+ \, {}^{\mathsf{x}}\!\triangleright \, \mathtt{b}^+ \triangleleft^{\mathsf{x}} (\mathtt{a}^+\mathsf{x})^*\mathtt{a}^+)]\!]$ is provably not a core spanner (see [4, Theorem 6.1])).

The question which core spanners can be represented as refl-spanners is a much more difficult one and we shall investigate it in more detail. There are simple core spanners which translate to refl-spanners in an obvious way, e. g., $\pi_{\{\mathsf{x}\}}\varsigma^{=}_{\{\mathsf{x},\mathsf{y}\}}[\![L]\!]$ with $L = \mathcal{L}({}^{\mathsf{x}}\!\triangleright (\mathtt{a}^* \vee \mathtt{b}^*) \triangleleft^{\mathsf{x}} \mathtt{c} \, {}^{\mathsf{y}}\!\triangleright (\mathtt{a}^* \vee \mathtt{b}^*) \triangleleft^{\mathsf{y}})$ can be represented as $[\![L']\!]$ where $L' = \mathcal{L}({}^{\mathsf{x}}\!\triangleright (\mathtt{a}^* \vee \mathtt{b}^*) \triangleleft^{\mathsf{x}} \mathtt{c}\mathsf{x})$. However, if we change L to $\mathcal{L}({}^{\mathsf{x}}\!\triangleright \Sigma^*\mathtt{a}\Sigma^* \triangleleft^{\mathsf{x}} \mathtt{c} \, {}^{\mathsf{y}}\!\triangleright \Sigma^*\mathtt{b}\Sigma^* \triangleleft^{\mathsf{y}})$, then neither ref-language $\mathcal{L}({}^{\mathsf{x}}\!\triangleright \Sigma^*\mathtt{a}\Sigma^* \triangleleft^{\mathsf{x}} \mathtt{c}\mathsf{x})$ nor $\mathcal{L}({}^{\mathsf{x}}\!\triangleright \Sigma^*\mathtt{b}\Sigma^* \triangleleft^{\mathsf{x}} \mathtt{c}\mathsf{x})$ yield an equivalent refl-spanner and we have to use $\mathcal{L}({}^{\mathsf{x}}\!\triangleright r \triangleleft^{\mathsf{x}} \mathtt{c}\mathsf{x})$, where r is a regular expression for $\mathcal{L}(\Sigma^*\mathtt{a}\Sigma^*) \cap \mathcal{L}(\Sigma^*\mathtt{b}\Sigma^*)$.

Another problem is that core spanners can also use string-equality selections on spans that contain start or end positions of other spans. For example, it seems difficult to transform $\varsigma^{=}_{\{\mathsf{x},\mathsf{y}\}}[\![\mathcal{L}({}^{\mathsf{x}}\!\triangleright \mathtt{a}^* \triangleleft^{\mathsf{x}} \, {}^{\mathsf{y}}\!\triangleright \, {}^{\mathsf{z}}\!\triangleright \mathtt{a}^* \triangleleft^{\mathsf{z}} \mathtt{a}^* \triangleleft^{\mathsf{y}})]\!]$ into a refl-spanner. The situation gets even more involved if we use the string-equality selections directly on overlapping spans, e. g., as in core spanners of the form $\varsigma^{=}_{\{\mathsf{x},\mathsf{y}\}}([\![\mathcal{L}({}^{\mathsf{x}}\!\triangleright \ldots \, {}^{\mathsf{y}}\!\triangleright \ldots \triangleleft^{\mathsf{x}} \ldots \triangleleft^{\mathsf{y}})]\!])$. For an in-depth analysis of the capability of core spanners to describe word-combinatorial properties, we refer to [8, 7].

These considerations suggest that the refl-spanner formalism is much less powerful than core spanners, which is to be expected, since we have to pay a price for the fact that we can solve many problems for refl-spanners much more efficiently than for core spanners (see our results presented in Section 5 and summarised in Table 1). However, we can show that a surprisingly large class of core spanners can nevertheless be represented by a single refl-spanner along with the application of a simple spanner operation (to be defined next) that just combines several variables (or columns in the spanner result) into one variable (or column) in a natural way, and a projection.

The *span-fusion* $\uplus$ is a binary operation $\mathsf{Spans} \times \mathsf{Spans} \to \mathsf{Spans}$ defined by $[i, j\rangle \uplus [i', j'\rangle = [\min\{i, i'\}, \max\{j, j'\}\rangle$ and $[i, j\rangle \uplus \bot = \bot \uplus [i, j\rangle = [i, j\rangle$. For a set $K \subseteq \mathsf{Spans}(w)$, we define $\biguplus(K) = \bot$ if $K = \emptyset$ and $\biguplus(K) = \biguplus(K \setminus \{s\}) \uplus s$ if $s \in K$. Intuitively speaking, the operation $\uplus$ constructs the set-union of two spans and fills in the gaps to turn it into a valid span.

We next lift this operation to an operation on spanners with the following intended meaning. In a table $S(w)$ for some spanner S over Σ and $\mathcal{X}$, and $w \in \Sigma^*$, we want to replace a specified set of columns $\{\mathsf{y}_1, \mathsf{y}_2, \ldots, \mathsf{y}_k\} \subseteq \mathcal{X}$ by a single new column x that, for each row s (i. e., span-tuple s) in $S(w)$, contains the span $\biguplus(\{s(\mathsf{y}_i) \mid i \in [k]\})$.

▶ **Definition 6.2.** *Let $\lambda \subseteq \mathcal{X}$ and let x be a new variable with $\mathsf{x} \notin \mathcal{X} \setminus \lambda$. For any $\mathcal{X}$-tuple t, $\biguplus_{\lambda \to \mathsf{x}}(t)$ is the $((\mathcal{X} \setminus \lambda) \cup \{\mathsf{x}\})$-tuple with $(\biguplus_{\lambda \to \mathsf{x}}(t))(\mathsf{x}) = \biguplus(\{t(\mathsf{y}) \mid \mathsf{y} \in \lambda\})$ and $(\biguplus_{\lambda \to \mathsf{x}}(t))(\mathsf{z}) = t(\mathsf{z})$ for every $\mathsf{z} \in \mathcal{X} \setminus \lambda$. For a set R of $\mathcal{X}$-tuples, $\biguplus_{\lambda \to \mathsf{x}}(R) = \{\biguplus_{\lambda \to \mathsf{x}}(t) \mid t \in R\}$. Moreover, for a spanner S over Σ and $\mathcal{X}$, the spanner $\biguplus_{\lambda \to \mathsf{x}}(S)$ over Σ and $(\mathcal{X} \cup \{\mathsf{x}\})$ is defined by $(\biguplus_{\lambda \to \mathsf{x}}(S))(w) = \biguplus_{\lambda \to \mathsf{x}}(S(w))$ for every word w.*

We use the following generalised application of the operation $\biguplus_{\lambda \to \mathsf{x}}$. For $\Lambda = \{\lambda_1, \lambda_2, \ldots, \lambda_k\} \subseteq \mathcal{P}(\mathcal{X})$ such that all λ_i with $i \in [k]$ are pairwise disjoint, a spanner S over $\mathcal{X}$ and fresh variables $\mathsf{x}_1, \mathsf{x}_2, \ldots, \mathsf{x}_k$, we define $\biguplus_{\{\lambda_i \to \mathsf{x}_i \mid i \in [k]\}}(S) = \biguplus_{\lambda_1 \to \mathsf{x}_1}(\biguplus_{\lambda_2 \to \mathsf{x}_2}(\ldots \biguplus_{\lambda_k \to \mathsf{x}_k}(S) \ldots))$. If the new variables x_i are negligible or clear from the context, we also write $\biguplus_{\lambda}$ or $\biguplus_{\{\lambda_i \mid i \in [k]\}}$ instead of $\biguplus_{\lambda \to \mathsf{x}}$ or $\biguplus_{\{\lambda_i \to \mathsf{x}_i \mid i \in [k]\}}$.

▶ **Example 6.3.** Let $L = \mathcal{L}(\,{}^{\mathsf{x}}\!\rhd\, \mathsf{a}^* \,{}^{\mathsf{y}}\!\rhd\, \mathsf{b}^* \lhd^{\mathsf{x}}\, \mathsf{a}^* \lhd^{\mathsf{y}})$ be a non-hierarchical subword-marked language over $\Sigma = \{\mathsf{a}, \mathsf{b}\}$ and $\mathcal{X} = \{\mathsf{x}, \mathsf{y}, \mathsf{z}\}$. For $w = \mathsf{aabaaa}$, we have $[\![L]\!](w) = \{([1, 4\rangle, [3, 7\rangle)\}$. Moreover, let $L' = \mathcal{L}(\,{}^{\mathsf{x}}\!\rhd\, \mathsf{a}^* \lhd^{\mathsf{x}} \,{}^{\mathsf{y}}\!\rhd\, \mathsf{b}^* \lhd^{\mathsf{y}} \,{}^{\mathsf{z}}\!\rhd\, \mathsf{a}^* \lhd^{\mathsf{z}})$ be a *hierarchical* subword-marked language, and let $\lambda_1 = \{\mathsf{x}, \mathsf{y}\}$, $\lambda_2 = \{\mathsf{y}, \mathsf{z}\}$ and $\Lambda = \{\lambda_1, \lambda_2\}$. Then $\biguplus_{\Lambda}[\![L']\!](w) = \{\biguplus_{\Lambda}(([1, 3\rangle, [3, 4\rangle, [4, 7\rangle))\} = [\![L]\!](w)$. In fact, it can be easily verified that $\biguplus_{\Lambda}[\![L']\!] = [\![L]\!]$.

Two variables $\mathsf{x}, \mathsf{y} \in \mathcal{X}$ are *overlapping in a spanner* S over Σ and $\mathcal{X}$ if there is a $w \in \Sigma^*$ and $t \in S(w)$ such that $t(\mathsf{x}) \cap t(\mathsf{y}) \neq \emptyset$. A string equality-selection $\varsigma^{=}_{E}$ over $\mathcal{X}$ is *overlapping with respect to* S if there are variables $\mathsf{x}, \mathsf{y} \in \bigcup_{\mathcal{Z} \in E} \mathcal{Z}$ with $\mathsf{x} \neq \mathsf{y}$ that are overlapping in S. A string-equality selection $\varsigma^{=}_{E}$ is *non-overlapping with respect to* S if it is not overlapping with respect to S. We are now able to state the main result of this section:

▶ **Theorem 6.4.** *For every core spanner S with $S = \pi_{\mathcal{Y}} \varsigma^{=}_{E}(S')$, where S' is a regular spanner over Σ and $\mathcal{X}$, $\mathcal{Y} \subseteq \mathcal{X}$ and $E \subseteq \mathcal{P}(\mathcal{X})$ such that $\varsigma^{=}_{E}$ is a string-equality selection over $\mathcal{X}$ that is non-overlapping with respect to S', there is a reference-bounded refl-spanner S'' over Σ and $\mathcal{X}'$ with $|\mathcal{X}'| = \mathrm{O}(|\mathcal{X}|^3)$ and a set $\Lambda \subseteq \mathcal{P}(\mathcal{X}')$ such that $S = \pi_{\mathcal{Y}} \biguplus_{\Lambda}(S'')$.*

We discuss this result before giving a proof sketch. The core simplification lemma states that in every core spanner $S \in \mathsf{reg}\text{-}\mathfrak{S}^{\{\cup, \pi, \bowtie, \varsigma^{=}\}}$, we can "push" all applications of $\cup$ and $\bowtie$ into the NFA that represents the regular spanner, leaving us with an expression $\pi_{\mathcal{Y}} \varsigma^{=}_{E}(\mathcal{L}(M))$ for an NFA M that accepts a subword-marked language. Theorem 6.4 now assures that if $\varsigma^{=}_{E}$ is non-overlapping with respect to S, then we can further "push" even all string-equality selections into the NFA M, which then accepts a (reference-bounded) ref-language instead of a subword-marked language, i. e., we can represent the string-equality selections as mere variable references in the regular spanner representation. However, the construction will add (a polynomial number of) new variables, which have to be translated back by an application of the span-fusion.

A main building block for the proof of Theorem 6.4, which also constitutes an interesting result about core spanners in its own right, is the following normal-form result.

▶ **Theorem 6.5.** *Every core spanner S over Σ and $\mathcal{X}$ can be represented as $\pi_{\mathcal{Y}}(\biguplus_{\Lambda}(\varsigma^{=}_{E}(S')))$, where S' is a quasi-disjoint regular spanner over Σ and $\mathcal{X}'$ with $|\mathcal{X}'| = \mathrm{O}(|\mathcal{X}|^3)$, $\mathcal{Y} \subseteq \mathcal{X}$, $E, \Lambda \subseteq \mathcal{P}(\mathcal{X}')$.*

Proof Sketch. Due to the core-simplification lemma (Lemma 2.2), we can assume that $S = \pi_{\mathcal{Y}}\varsigma^{=}_{E}(S')$ for some regular spanner S' over Σ and $\mathcal{X}$, and $E = \{\mathcal{Z}_1, \mathcal{Z}_2, \ldots, \mathcal{Z}_\ell\} \subseteq \mathcal{P}(\mathcal{X})$. Let M be an NFA with $[\![\mathcal{L}(M)]\!] = S'$. We first transform M into M' by replacing every $\mathsf{x} \in \mathcal{X}$ by $m = |\mathcal{X}|^2$ new variables $\mathcal{SV}_m(\mathsf{x}) = \{[\mathsf{x}, 1], [\mathsf{x}, 2], \ldots, [\mathsf{x}, m]\}$, i. e., M' has variable set $\mathcal{X}' = \bigcup^{\mathsf{x} \in \mathcal{X}} \mathcal{SV}_m(\mathsf{x})$. Whenever M reads a factor ${}^{\mathsf{x}}{\triangleright}\, w_{\mathsf{x}} {\triangleleft}^{\mathsf{x}}$, then M' nondeterministically reads a factor ${}^{[\mathsf{x},1]}{\triangleright}\, w_{\mathsf{x},1} {\triangleleft}^{[\mathsf{x},1]}\ {}^{[\mathsf{x},2]}{\triangleright}\, w_{\mathsf{x},2} {\triangleleft}^{[\mathsf{x},2]} \ldots {}^{[\mathsf{x},m]}{\triangleright}\, w_{\mathsf{x},m} {\triangleleft}^{[\mathsf{x},m]}$ instead, where $w_{\mathsf{x},1} w_{\mathsf{x},2} \ldots w_{\mathsf{x},m}$ is a factorisation of w_{x}, such that, for every $j \in [m]$, $w_{\mathsf{x},j} \in (\Gamma_{\mathcal{X}'})^* \Sigma^* (\Gamma_{\mathcal{X}'})^*$. This can be done by keeping track in the finite state control for which $j \in [m]$ we have already read ${}^{[\mathsf{x},j]}{\triangleright}$ and then allowing M' to nondeterministically read factors ${\triangleleft}^{[\mathsf{x},j]}\ {}^{[\mathsf{x},j+1]}{\triangleright}$ while reading w_{x}. In order to make sure that $w_{\mathsf{x},j} \in (\Gamma_{\mathcal{X}'})^* \Sigma^* (\Gamma_{\mathcal{X}'})^*$ for every $j \in [m]$, we have to require that at least one such ${\triangleleft}^{[\mathsf{x},j]}\ {}^{[\mathsf{x},j+1]}{\triangleright}$ factor is read in every maximal factor of symbols from $\Gamma_{\mathcal{X} \setminus \{\mathsf{x}\}}$.

It can be shown that $\mathcal{L}(M')$ is in fact a quasi-disjoint subword-marked language over Σ and $\mathcal{X}'$. Moreover, we can express $\varsigma^{=}_{\mathcal{Z}_1} \varsigma^{=}_{\mathcal{Z}_2} \ldots \varsigma^{=}_{\mathcal{Z}_\ell}([\![\mathcal{L}(M)]\!])$ by turning each $\varsigma^{=}_{\mathcal{Z}_i}$ into string-equality selections $\varsigma^{=}_{E_i}$ with $E_i = \{\{[\mathsf{x}, 1] \mid \mathsf{x} \in \mathcal{Z}_i\}, \{[\mathsf{x}, 2] \mid \mathsf{x} \in \mathcal{Z}_i\}, \ldots, \{[\mathsf{x}, m] \mid \mathsf{x} \in \mathcal{Z}_i\}\}$ and applying them to $[\![\mathcal{L}(M)]\!]$, followed by a span-fusion that will combine back all variables from $\mathcal{SV}_m(\mathsf{x})$ into the single original variable x. More precisely, $\varsigma^{=}_{\mathcal{Z}_1} \varsigma^{=}_{\mathcal{Z}_2} \ldots \varsigma^{=}_{\mathcal{Z}_\ell}([\![\mathcal{L}(M)]\!]) = \biguplus_{\Lambda} \varsigma^{=}_{E_1} \varsigma^{=}_{E_2} \ldots \varsigma^{=}_{E_\ell}([\![\mathcal{L}(M')]\!])$, where $\Lambda = \{\mathcal{SV}_m(\mathsf{x}) \to \mathsf{x} \mid \mathsf{x} \in \mathcal{X}\}$. ◀

We are now ready to give a proof sketch for Theorem 6.4:

Proof Sketch of Theorem 6.4. Let $S = \pi_{\mathcal{Y}}\varsigma^{=}_{E}(S')$, where S' is a regular spanner over Σ and $\mathcal{X}$, $\mathcal{Y} \subseteq \mathcal{X}$ and $E = \{\mathcal{Z}_1, \mathcal{Z}_2, \ldots, \mathcal{Z}_\ell\} \subseteq \mathcal{P}(\mathcal{X})$ such that $\varsigma^{=}_{E}$ is non-overlapping with respect to S'. Let M be an NFA with $[\![\mathcal{L}(M)]\!] = S'$. The proof heavily relies on Theorem 6.5 in the sense that we only have to show that $\varsigma^{=}_{E_1} \varsigma^{=}_{E_2} \ldots \varsigma^{=}_{E_\ell}([\![\mathcal{L}(M')]\!])$ is a refl-spanner, where M' is obtained from M via the construction of the proof of Theorem 6.5. In particular, $\mathcal{L}(M')$ is quasi-disjoint and the string-equality selections $\varsigma^{=}_{E_1}, \varsigma^{=}_{E_2}, \ldots, \varsigma^{=}_{E_\ell}$ have been obtained from a string-equality selection $\varsigma^{=}_{E}$ that is non-overlapping with respect to S'.

We recall that, for every $i \in [\ell]$, $E_i = \{\{[\mathsf{x}, 1] \mid \mathsf{x} \in \mathcal{Z}_i\}, \{[\mathsf{x}, 2] \mid \mathsf{x} \in \mathcal{Z}_i\}, \ldots, \{[\mathsf{x}, m] \mid \mathsf{x} \in \mathcal{Z}_i\}\}$. The idea is to inductively incorporate all these string-equality selections $\varsigma^{=}_{\{[\mathsf{x},j] \mid \mathsf{x} \in \mathcal{Z}_i\}}$ one by one directly into the NFA M' by introducing variable references for the variables $\{[\mathsf{x}, j] \mid \mathsf{x} \in \mathcal{Z}_i\}$ (note that this changes the accepted language from M into a ref-language). We sketch one individual step of this construction. Any ref-word accepted by (the current version of) M' contains (possibly zero) factors of the form ${}^{[\mathsf{x},j]}{\triangleright}\, v_{[\mathsf{x},j]} {\triangleleft}^{[\mathsf{x},j]}$ with $\mathsf{x} \in \mathcal{Z}_i$. It can be shown that $v_{[\mathsf{x},j]} = \gamma u_{[\mathsf{x},j]} \delta$, where $u_{[\mathsf{x},j]} \in \Sigma^*$ and γ and δ contain neither symbols from Σ nor variable references (the former is a consequence from the fact that $\mathcal{L}(M')$ is quasi-disjoint, the latter follows from the non-overlapping property of $\varsigma^{=}_{E}$). Moreover, these factors are pairwise non-overlapping. Now we want to check whether all these $u_{[\mathsf{x},j]}$ are the same, since this is required by $\varsigma^{=}_{\{[\mathsf{x},j] \mid \mathsf{x} \in \mathcal{Z}_i\}}$. This property is non-regular and can therefore not be checked directly. Instead, we only check whether the first ${}^{[\mathsf{x},j]}{\triangleright}\, \gamma u_{[\mathsf{x},j]} \delta {\triangleleft}^{[\mathsf{x},j]}$ that we encounter is such that we can potentially read the exact same word $u_{[\mathsf{x},j]}$ between all the remaining pairs of brackets ${}^{[\mathsf{y},j]}{\triangleright} \ldots {\triangleleft}^{[\mathsf{y},j]}$ for $\mathsf{y} \in \mathcal{Z}_i \setminus \{\mathsf{x}\}$. To this end, we guess the state pairs between which these factors will be read and then check, while reading $u_{[\mathsf{x},j]}$ from the input, whether we can also read $u_{[\mathsf{x},j]}$ between these states. For such words, we can then on-the-fly just read the variable reference $[\mathsf{x}, j]$ instead $u_{[\mathsf{y},j]}$ when we reach the right states between which we have already checked that potentially $u_{[\mathsf{x},j]}$ could be read here. ◀

Theorem 6.5 says that if we allow an application of the span-fusion between the projection and the string-equality selections in the representation of the core-simplification lemma, then we can assume the regular spanner to be quasi-disjoint. Hence, we get the following:

▶ **Corollary 6.6.** $\mathsf{reg}\text{-}\mathfrak{S} = [\![\mathsf{RGX}]\!]^{\{\cup,\bowtie,\pi\}} = [\![\mathsf{RGX}]\!]^{\{\uplus,\pi\}}$ *and* $\mathsf{core}\text{-}\mathfrak{S} = [\![\mathsf{RGX}]\!]^{\{\cup,\bowtie,\pi,\varsigma^{=}\}} = [\![\mathsf{RGX}]\!]^{\{\uplus,\pi,\varsigma^{=}\}}$.

7 Conclusion

In this work, we introduced reference-bounded refl-spanners, a new fragment of core spanners. In terms of expressive power, this fragment is slightly less powerful than the class of core spanners, but has lower evaluation complexity (see Table 1, and note further that these upper bounds even hold for refl-spanners that are *not* necessarily reference-bounded). If we add the *span-fusion* – a natural binary operation on spanners – to reference-bounded refl-spanners (see Section 6) then they have the same expressive power as core spanners with *non-overlapping* string equality selections. This demonstrates that our formalism covers all aspects of core spanners except for the possibility of applying string equality selections on variables with overlapping spans. Moreover, since we achieve better complexities for refl-spanners compared to core spanners, this also shows that overlapping string equality selections are a source of complexity for core spanners.

From a conceptional point of view, our new angle was to treat the classical two-stage approach of core spanners, i. e., first producing the output table of a regular spanner and then filtering it by applying the string equality selections, as a single NFA. This is achieved by using ref-words in order to represent a document along with a span-tuple *that satisfies the string equality selections*, instead of just using subword-marked words to represent a document along with a span-tuple, which might not satisfy the string equality selections and therefore will be filtered out later in the second evaluation stage.

A question that is left open for further research is about constant delay enumeration for some fragment of core spanners strictly more powerful than the regular spanners. In this regard, we note that for efficient enumeration for (some fragments of) core spanners, we must overcome the general intractability of NonEmptiness (which we have for core spanners as well as for (reference bounded) refl-spanners). We believe that refl-spanners are a promising candidate for further restrictions that may lead to a fragment of core spanners with constant delay enumeration.

References

1 Antoine Amarilli, Pierre Bourhis, Stefan Mengel, and Matthias Niewerth. Constant-delay enumeration for nondeterministic document spanners. In *22nd International Conference on Database Theory, ICDT 2019, March 26–28, 2019, Lisbon, Portugal*, pages 22:1–22:19, 2019.

2 Arturs Backurs and Piotr Indyk. Which regular expression patterns are hard to match? In *IEEE 57th Annual Symposium on Foundations of Computer Science, FOCS 2016, 9–11 October 2016, Hyatt Regency, New Brunswick, New Jersey, USA*, pages 457–466, 2016. doi:10.1109/FOCS.2016.56.

3 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.

4 Ronald Fagin, Benny Kimelfeld, Frederick Reiss, and Stijn Vansummeren. Document spanners: A formal approach to information extraction. *J. ACM*, 62(2):12:1–12:51, 2015.

5 Henning Fernau, Markus L. Schmid, and Yngve Villanger. On the parameterised complexity of string morphism problems. *Theory of Computing Systems (ToCS)*, 59(1):24–51, 2016.

6 Fernando Florenzano, Cristian Riveros, Martín Ugarte, Stijn Vansummeren, and Domagoj Vrgoc. Constant delay algorithms for regular document spanners. In *Proceedings of the 37th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems, Houston, TX, USA, June 10–15, 2018*, pages 165–177, 2018.

7 Dominik D. Freydenberger. A logic for document spanners. *Theory Comput. Syst.*, 63(7):1679–1754, 2019. `doi:10.1007/s00224-018-9874-1`.

8 Dominik D. Freydenberger and Mario Holldack. Document spanners: From expressive power to decision problems. *Theory Comput. Syst.*, 62(4):854–898, 2018. `doi:10.1007/s00224-017-9770-0`.

9 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.

10 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. `doi:10.4230/LIPIcs.ICDT.2020.11`.

11 Florin Manea and Markus L. Schmid. Matching patterns with variables. In *Combinatorics on Words – 12th International Conference, WORDS 2019, Loughborough, UK, September 9–13, 2019, Proceedings*, pages 1–27, 2019. `doi:10.1007/978-3-030-28796-2_1`.

12 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.

13 Liat Peterfreund. *The Complexity of Relational Queries over Extractions from Text*. PhD thesis, Computer science department, Technion, 2019.

14 Liat Peterfreund. Grammars for document spanners. In *24th International Conference on Database Theory, ICDT 2021, March 23–26, 2021, Nicosia, Cyprus*, 2021. To appear. Extended version available at `https://arxiv.org/abs/2003.06880`.

15 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.

16 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.

17 Markus L. Schmid. Characterising REGEX languages by regular languages equipped with factor-referencing. *Information and Computation*, 249:1–17, 2016.

18 Markus L. Schmid and Nicole Schweikardt. A purely regular approach to non-regular core spanners. *CoRR*, abs/2010.13442, 2020. `arXiv:2010.13442`.