Logical Methods in Computer Science
Volume 20, Issue 4, 2024, pp. 16:1–16:41
https://lmcs.episciences.org/

Submitted Oct. 28, 2021
Published Nov. 21, 2024

REFL-SPANNERS: A PURELY REGULAR APPROACH TO NON-REGULAR CORE SPANNERS

MARKUS L. SCHMID AND NICOLE SCHWEIKARDT

Humboldt-Universität zu Berlin, Unter den Linden 6, D-10099, Berlin, Germany
e-mail address: MLSchmid@MLSchmid.de, schweikn@informatik.hu-berlin.de

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 variant of core spanners that, while being incomparable to 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 for the typical problems in static analysis and query evaluation.

1. Introduction

The information extraction framework of *document spanners* has been introduced by Fagin, Kimelfeld, Reiss, and Vansummeren [FKRV15] 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 $0 \leqslant i \leqslant j \leqslant |w| + 1$.

Key words and phrases: Document spanners, regular expressions with backreferences.

This is the full version of the article [SS21a]. The first author has been funded by the German Research Foundation (Deutsche Forschungsgemeinschaft, DFG) – project number 416776735 (gefördert durch die Deutsche Forschungsgemeinschaft (DFG) – Projektnummer 416776735). The second author has been 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).

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).

DOI:10.46298/LMCS-20(4:16)2024

The document spanners (or simply *spanners*, for short) introduced in [FKRV15] follow a two-stage approach: *Primitive* spanners extract relations directly from the input document, which are then further manipulated by using particular algebraic operations. As primitive spanners, [FKRV15] introduces *vset-automata* and *regex-formulas*, which are variants of nondeterministic finite automata and regular expressions. They can use meta-symbols $\vdash_{\mathsf{x}}$ and $\dashv_{\mathsf{x}}$, 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 in $\mathcal{X}$. For example, $\alpha = (\,\vdash_{\mathsf{x}} (\mathtt{a} \vee \mathtt{b})^* \dashv_{\mathsf{x}}) \cdot (\,\vdash_{\mathsf{y}} (\mathtt{a}^* \vee \mathtt{b}^*) \dashv_{\mathsf{y}}) \mathtt{c}^*$ 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 $\vdash_{\mathsf{x}} (\mathtt{a} \vee \mathtt{b})^* \dashv_{\mathsf{x}}$ and $\vdash_{\mathsf{y}} (\mathtt{a}^* \vee \mathtt{b}^*) \dashv_{\mathsf{y}}$, respectively. This means that $[\![\alpha]\!](w)$ is a binary relation over w's spans. 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 $\vdash_{\mathsf{x}} \mathtt{ab} \dashv_{\mathsf{x}} \vdash_{\mathsf{y}} \mathtt{aa} \dashv_{\mathsf{y}} \mathtt{c}$, $\vdash_{\mathsf{x}} \mathtt{aba} \dashv_{\mathsf{x}} \vdash_{\mathsf{y}} \mathtt{a} \dashv_{\mathsf{y}} \mathtt{c}$ and $\vdash_{\mathsf{x}} \mathtt{abaa} \dashv_{\mathsf{x}} \vdash_{\mathsf{y}} \dashv_{\mathsf{y}} \mathtt{c}$. The rows of the extracted relation are also called *span-tuples*. Vset-automata follow the same principle, but take the form of nondeterministic finite automata. It is known that, with respect to defining spanners, vset-automata are more expressive than regex-formulas. The class of spanners expressible by vset-automata are called *regular spanners*; 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 [MRV18, FKRV15]); these aspects shall be discussed in more detail below).

The algebraic operations used for further manipulating the extracted span-relations comprise the union $\cup$, natural join $\bowtie$, projection π (with the obvious meaning) and string-equality selection $\zeta^{=}_{\mathcal{Z}}$. The latter is a unary operator that is 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 in $\mathcal{Z}$ 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 [FKRV15] are defined as $\mathsf{reg}\text{-}\mathfrak{S}^{\{\cup,\bowtie,\pi,\zeta^=\}}$, i. e., the closure of regular spanners under the operations $\cup$, $\bowtie$, π and $\zeta^=$ (these operations are interpreted as operations on spanners in the natural way). A central result of [FKRV15] 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* [MRV18] (i. e., variables in span-tuples can be undefined). 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 [FKRV15], we can nevertheless represent any core spanner $S \in \mathsf{reg}\text{-}\mathfrak{S}^{\{\cup,\bowtie,\pi,\zeta^=\}}$ in the form $\pi_{\mathcal{Y}} \zeta^{=}_{\mathcal{Z}_1} \zeta^{=}_{\mathcal{Z}_2} \dots \zeta^{=}_{\mathcal{Z}_k}(S')$ for a regular spanner S' (this is called the *core-simplification lemma* in [FKRV15]).

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 [ABMN21, FRU+20]), while spanner containment or equivalence is generally decidable, and can even be decided efficiently if we additionally require the spanner to be represented by

a certain deterministic vset-automaton (see [DKM$^+$19]).[1] 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 [FH18] 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 [FH18] 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 accordingly. We shall illustrate this with some examples.

Consider a regex formula $\alpha = {}^{\mathsf{x}_1\triangleright}\Sigma^*\triangleleft^{\mathsf{x}_1}\,{}^{\mathsf{x}_2\triangleright}\Sigma^*\triangleleft^{\mathsf{x}_2}\dots\,{}^{\mathsf{x}_n\triangleright}\Sigma^*\triangleleft^{\mathsf{x}_n}$. Then checking, for some $\mathcal{Z}_1, \mathcal{Z}_2, \dots, \mathcal{Z}_k \subseteq \{\mathsf{x}_1, \mathsf{x}_2, \dots, \mathsf{x}_n\}$, whether the empty tuple is in $(\pi_\emptyset \varsigma^{=}_{\mathcal{Z}_1}\varsigma^{=}_{\mathcal{Z}_2}\dots\varsigma^{=}_{\mathcal{Z}_k}([\![\alpha]\!]))(w)$, 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., [MS19]). However, checking for a (non-empty) span-tuple t whether it is in $(\varsigma^{=}_{\mathcal{Z}_1}\varsigma^{=}_{\mathcal{Z}_2}\dots\varsigma^{=}_{\mathcal{Z}_k}([\![\alpha]\!]))(w)$ 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 $r_1, r_2, \dots, r_n$ be some regular expressions, and let $\alpha = {}^{\mathsf{x}_1\triangleright}r_1\triangleleft^{\mathsf{x}_1}\,{}^{\mathsf{x}_2\triangleright}r_2\triangleleft^{\mathsf{x}_2}\dots\,{}^{\mathsf{x}_n\triangleright}r_n\triangleleft^{\mathsf{x}_n}$. Then there is a word w with $(\varsigma^{=}_{\{\mathsf{x}_1,\mathsf{x}_2,\dots,\mathsf{x}_n\}}([\![\alpha]\!]))(w) \neq \emptyset$ 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. For example, we can even express word equations as core spanners (see [FH18, 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

[1] As is common in the literature, the input vset-automata are assumed to be *sequential*, which means that every accepting run describes a valid span-tuple; it is well known that dropping this natural requirement yields intractability (see, e. g., [ABMN21]).

Problem	**Regular sp.**	**Refl-sp.**		**Core sp.** [FH18]
ModelChecking	$O(\|w\| \cdot \|M\| \log \|\mathcal{X}\|)$	$O(\|w\| \cdot \|M\| \log \|\mathcal{X}\|)$	[T. 5.4]	NP-c
NonEmptiness	$O(\|w\| \cdot \|M\|)$	NP-c	[T. 5.7]	NP-h
Satisfiability	$O(\|M\|)$	$O(\|M\|)$	[T. 5.8]	PSpace-c
Containment	PSpace-c [MRV18]	PSpace-c (for str. ref.)	[T. 5.13]	undec.
Equivalence	PSpace-c [MRV18]	PSpace-c (for str. ref.)	[T. 5.13]	undec.
Hierarchicality	$O(\|M\| \cdot \|\mathcal{X}\|^3)$	$O(\|M\| \cdot \|\mathcal{X}\|^3)$	[T. 5.8]	PSpace-c

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 (accepting a subword-marked language or a ref-language, respectively). The abbreviation "str. ref." means *strongly reference extracting*, a restriction for refl-spanners to be formally defined in Section 5. The bounds for ModelChecking are under the (rather weak) assumption $|\mathcal{X}| = O(|w|)$; without this assumption, $|w|$ has to be replaced with $|w| + |\mathcal{X}|$.

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 $L' = \{{}^{\mathtt{x}}\triangleright\,\mathtt{a}^n \triangleleft^{\mathtt{x}} \mathtt{b}\mathtt{x} \mid n \geqslant 0\}$ can be interpreted as a regular description of L by means of meta-symbols ${}^{\mathtt{x}}\triangleright$ and $\triangleleft^{\mathtt{x}}$ to *capture* a factor, and a meta-symbol $\mathtt{x}$ to *copy* or *reference* the captured factor (conceptionally, this is similar to so-called backreferences in practical regular expressions; see [FKRV15, FH18] for comparisons of regular expressions with backreferences and core spanners). In particular, all words of L can be easily obtained from the words of L' by simply replacing the occurrence of $\mathtt{x}$ 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 particular core spanners, e. g., the core spanner $\pi_{\{\mathtt{x},\mathtt{y}\}}\zeta^{=}_{\{\mathtt{x},\mathtt{x}'\}}\zeta^{=}_{\{\mathtt{y},\mathtt{y}'\}}([\![\alpha]\!])$ with $\alpha = {}^{\mathtt{x}}\triangleright\,\mathtt{a}^*\mathtt{b}\,{}^{\mathtt{y}}\triangleright\,\mathtt{c}\triangleleft^{\mathtt{x}}\mathtt{b}^*\,{}^{\mathtt{x}'}\triangleright\,\mathtt{a}^*\mathtt{b}\mathtt{c}\triangleleft^{\mathtt{x}'}\triangleleft^{\mathtt{y}}\,{}^{\mathtt{y}'}\triangleright\,\mathtt{c}\mathtt{b}^*\mathtt{a}^*\mathtt{b}\mathtt{c}\triangleleft^{\mathtt{y}'}$ could be represented as $[\![\,{}^{\mathtt{x}}\triangleright\,\mathtt{a}^*\mathtt{b}\,{}^{\mathtt{y}}\triangleright\,\mathtt{c}\triangleleft^{\mathtt{x}}\mathtt{b}^*\mathtt{x}\triangleleft^{\mathtt{y}}\mathtt{y}]\!]$.

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 $\Sigma \cup \mathcal{X} \cup \{{}^{\mathtt{x}}\triangleright, \triangleleft^{\mathtt{x}} \mid \mathtt{x} \in \mathcal{X}\}$ that has the additional property that the meta-symbols $\mathcal{X} \cup \{{}^{\mathtt{x}}\triangleright, \triangleleft^{\mathtt{x}} \mid \mathtt{x} \in \mathcal{X}\}$ are "well-behaved" in the sense that each word describes a valid span-tuple (one of this paper's 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 ${}^{\mathtt{x}}\triangleright \ldots \triangleleft^{\mathtt{x}}$, but all references $\mathtt{x}$ 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 for 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 [FKRV15] 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 automaton that represents the regular spanner, leaving us with an expression $\pi_{\mathcal{Y}}\varsigma^=_{\mathcal{Z}_1}\varsigma^=_{\mathcal{Z}_2}\dots\varsigma^=_{\mathcal{Z}_k}(M)$ for an automaton M that represents a regular spanner. We can show that if the string-equality selections $\varsigma^=_{\mathcal{Z}_1}\varsigma^=_{\mathcal{Z}_2}\dots\varsigma^=_{\mathcal{Z}_k}$ apply to a set of variables that never capture overlapping spans, then we can even "push" 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 [FKRV15, FKP18, PFKK19, ABMN21, MRV18, FRU+20, PtCFK19, Fre19, FH18, FT20, Pet19, SS22, SS21b, ABMN20, AJMR22, DKMP20, DBKM21, FT22, MR23]. 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 [ABMN21, FRU+20], the paper [MRV18] is concerned with different semantics of regular spanners and their expressive power, and [PFKK19] investigates the evaluation of relational algebra expressions over regular spanners.

Papers that are concerned with string-equality selection are the following. In [FH18] many negative results for core spanner evaluation are shown. By presenting a logic that exactly covers core spanners, the work [Fre19] answers questions on the expressive power of core spanners. That datalog over regular spanners covers the whole class of core spanners is shown in [PtCFK19]. In [FKP18], the authors consider conjunctive queries on top of regular spanners and, among mostly negative results, they also show 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 [FT20] 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 [FKRV15] 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 [Pet21]. However, the non-regular aspect of [Pet21] 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. The spanner class from [Pet21] is incomparable with core spanners, and the main focus of [Pet21] is on enumeration.

1.3. **Differences to the Preliminary Conference Version.** This paper is the full and substantially revised version of the extended abstract [SS21a], presented at the 24th International Conference on Database Theory (ICDT 2021). We will briefly describe the main changes.

The upper and lower complexity bounds for evaluation and static analysis problems of refl-spanners presented in Section 5 (see also Table 1) have been improved as follows. The upper bound for ModelChecking for refl-spanners reported in [SS21a] has been improved substantially, and, in fact, coincides with the (to our knowledge) best known upper bound for

ModelChecking for regular spanners (note that [SS21a] did not mention any upper bound for ModelChecking for regular spanners). The upper bounds for Containment and Equivalence for strongly reference extracting refl-spanners have been improved from membership in ExpSpace to PSpace-completeness.

With respect to the results on the expressive power of refl-spanners, this full version serves also as an erratum to [SS21a]. The main result with respect to the expressive power of refl-spanners holds as stated in [SS21a, Theorem 6.4], and is proven here in full detail (see Theorem 6.11). Unfortunately, the secondary result [SS21a, Theorem 6.5] does not hold as stated in [SS21a]. This error is corrected here.

1.4. **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 [FKRV15] 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.

2. Preliminaries

Let $\mathbb{N} = \{1, 2, 3, \ldots\}$ and $[n] = \{1, 2, \ldots, n\}$ for $n \in \mathbb{N}$. For a (partial) mapping $f\colon X \to Y$, we write $f(x) = \perp$ for some $x \in X$ to denote that $f(x)$ is not defined; we also set $\mathrm{dom}(f) = \{x \mid f(x) \neq \perp\}$. 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 $\mathsf{e}_B\colon A \to A \cup \{\varepsilon\}$ 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 write e_B to denote the natural extension of e_B to a morphism $A^* \to A^*$. Intuitively, for $w \in A^*$, the word $\mathsf{e}_B(w)$ is obtained from w by erasing all occurrences of letters in B. Technically, e_B depends on the alphabet A, but whenever we use $\mathsf{e}_B(w)$ we always assume that $\mathsf{e}_B\colon A \to A \cup \{\varepsilon\}$ 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: $\emptyset$ is in RE_Σ with $\mathcal{L}(\emptyset) = \emptyset$. Every $a \in \Sigma \cup \{\varepsilon\}$ is in RE_Σ with $\mathcal{L}(a) = \{a\}$. 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 (and use the usual precedences of operands), 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 finite set Q of states, a finite alphabet Σ, a start state q_0, a set $F \subseteq Q$ of accepting states, and a transition function $\delta\colon Q \times (\Sigma \cup \{\varepsilon\}) \to \mathcal{P}(Q)$. 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 as $|Q| + |\delta|$. 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 an *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 $\{[i, j\rangle \mid i, j \in \mathbb{N},\ i \leqslant j\}$ (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 operations for spans (note, however, that the union of two spans is not necessarily a span anymore). Two spans $s = [i, j\rangle$ and $s' = [i', j'\rangle$ are *equal* if $s = s'$ (i.e., $i = i'$ and $j = j'$), they are *disjoint* if $j \leqslant i'$ or $j' \leqslant i$ and they are *non-overlapping* 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. 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.

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 *non-overlapping* if, for every $\mathsf{x}, \mathsf{y} \in \mathrm{dom}(t)$, $t(\mathsf{x})$ and $t(\mathsf{y})$ are non-overlapping. An $(\mathcal{X}, w)$-relation is *functional*, *hierarchical* or *non-overlapping*, if all its elements are functional, hierarchical or non-overlapping, 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).

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 $w \in \Sigma^*$, we have $S_1(w) \subseteq S_2(w)$. A spanner S over Σ and $\mathcal{X}$ is *functional*, *hierarchical* or *non-overlapping* if, for every w, $S(w)$ is functional, hierarchical or non-overlapping, respectively. Note that, for span-tuples, span-relations and spanners, the non-overlapping property 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 $w \in \Sigma^*$.

To define the *natural join* $S_1 \bowtie S_2$ we need further notation: 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 $\mathsf{x} \in \mathrm{dom}(t_1) \cap \mathrm{dom}(t_2)$. 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 $i \in \{1,2\}$. For two $(\mathcal{X},w)$-relations R_1, R_2 we let $R_1 \bowtie R_2 = \{t_1 \bowtie t_2 \mid t_1 \in R_1, t_2 \in R_2, t_1 \sim t_2\}$. Finally, 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 $w \in \Sigma^*$.

The *projection* $\pi_{\mathcal{Y}}(S_1)$ for a set $\mathcal{Y} \subseteq \mathcal{X}$ is defined by letting $(\pi_{\mathcal{Y}}(S_1))(w) = \{t_{|\mathcal{Y}} \mid t \in S_1(w)\}$, 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 $\mathsf{x},\mathsf{y} \in \mathcal{Y} \cap \mathrm{dom}(t)$ with $t(\mathsf{x}) = [i,j\rangle$ and $t(\mathsf{y}) = [i',j'\rangle$, we have that $w[i,j\rangle = w[i',j'\rangle$. Note that here we require $w[i,j\rangle = w[i',j'\rangle$ only for $\mathsf{x},\mathsf{y} \in \mathcal{Y} \cap \mathrm{dom}(t)$ instead of all $\mathsf{x},\mathsf{y} \in \mathcal{Y}$.

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 a class $\mathfrak{S}$ of spanners and a set P of operations on spanners, $\mathfrak{S}^P$ (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 [FKRV15], 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 [FKRV15] for a formal definition of vset-automata and regex-formulas). It is known that, with respect to defining spanners, vset-automata are more expressive than regex-formulas; but for defining the class of core spanners, it does not matter whether one uses the class of regex-formulas or the class of vset-automata, i. e., $[\![\mathsf{RGX}]\!]^{\{\cup,\pi,\bowtie,\varsigma^{=}\}} = [\![\mathsf{VSet}]\!]^{\{\cup,\pi,\bowtie,\varsigma^{=}\}}$). A crucial result from [FKRV15] 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, $\mathcal{Y} \subseteq \mathcal{X}$ and $E \subseteq \mathcal{P}(\mathcal{X})$. The setting in [FKRV15] 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 [MRV18], and in the terminology of [PFKK19], we consider the *schemaless* semantics.

In [MRV18] it is shown that the classical framework for *regular spanners* with function semantics introduced in [FKRV15] can be extended to the schemaless case, i. e., vset-automata and regex-formulas 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 when considering the closure under union, projection and natural join, i.e., $[\![\mathsf{VSet}]\!]^{\{\cup,\pi,\bowtie\}} = [\![\mathsf{RGX}]\!]^{\{\cup,\pi,\bowtie\}}$). However, the string-equality selection operator — which turns regular spanners into the more powerful core spanners — is not treated in [MRV18]. Our definition of the string-equality selection operator given above extends the definition from [FKRV15] 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 [MRV18]. Due to [MRV18], 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 [FKRV15] 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 (see Theorem 6.11), and since this seems to be a worthwhile task in its own right, we show that the core-simplification lemma from [FKRV15] 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 [MRV18].

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

Proof. We proceed by induction on the construction of $S \in \mathsf{core}\text{-}\mathfrak{S} = [\![\mathsf{RGX}]\!]^{\{\cup,\pi,\bowtie,\varsigma^{=}\}}$.

For the induction base we consider $S \in [\![\mathsf{RGX}]\!]$ over $\mathcal{X}$. By Theorem 4.4 of [MRV18] there exists a (hierarchical) vset-automaton M such that $S = [\![M]\!]$. In particular, $S \in \mathsf{reg}\text{-}\mathfrak{S}$. Furthermore, it is obvious that $S = \pi_{\mathcal{X}}\varsigma^{=}_{\emptyset}(S)$, and hence we are done.

For the induction step we distinguish between two cases.

Case 1: S is of the form $\pi_{\mathcal{Y}}(S_1)$ or $\varsigma^{=}_{E}(S_1)$, where $S_1 \in \mathsf{core}\text{-}\mathfrak{S}$ over $\mathcal{X}$ and $\mathcal{Y} \subseteq \mathcal{X}$ or $E \subseteq \mathcal{P}(\mathcal{X})$. Applying the induction hypothesis to S_1, we obtain that there exist $S'_1 \in \mathsf{reg}\text{-}\mathfrak{S}$, $\mathcal{Y}_1 \subseteq \mathcal{X}$, and $E_1 \subseteq \mathcal{P}(\mathcal{X})$ such that $S_1 = \pi_{\mathcal{Y}_1}\varsigma^{=}_{E_1}(S'_1)$.

If S is of the form $\pi_{\mathcal{Y}}(S_1)$, we are done by noting that $S = \pi_{\mathcal{Y}\cap\mathcal{Y}_1}\varsigma^{=}_{E_1}(S'_1)$.

If S is of the form $\varsigma^{=}_{E}(S_1)$, we are done by noting that $S = \varsigma^{=}_{E}(\pi_{\mathcal{Y}_1}\varsigma^{=}_{E_1}(S'_1)) = \pi_{\mathcal{Y}_1}\varsigma^{=}_{E'\cup E_1}(S'_1)$ where $E' := \{C \cap \mathcal{Y}_1 \mid C \in E\}$.

Case 2: S is of the form $(S_1 * S_2)$ where $* \in \{\cup, \bowtie\}$ and $S_i \in \mathsf{core}\text{-}\mathfrak{S}$ over $\mathcal{X}$ for each $i \in \{1,2\}$. Applying the induction hypothesis to S_i for each $i \in \{1,2\}$, we obtain that there exist $S'_i \in \mathsf{reg}\text{-}\mathfrak{S}$, $\mathcal{Y}_i \subseteq \mathcal{X}$, and $E_i \subseteq \mathcal{P}(\mathcal{X})$ such that $S_i = \pi_{\mathcal{Y}_i}\varsigma^{=}_{E_i}(S'_i)$. In particular, for each $i \in \{1,2\}$ there is a vset-automaton M_i such that $S'_i = [\![M_i]\!]$.

Let $\tilde{\mathcal{Y}} := \mathcal{Y}_1 \cap \mathcal{Y}_2$. Let $\mathcal{Z}_i$ denote the set of all variables that occur in M_i and let $\mathcal{Z}'_i := \mathcal{Z}_i \setminus \tilde{\mathcal{Y}}$. By suitably renaming variables we can assume w.l.o.g. that $\mathcal{Z}'_1 \cap \mathcal{Z}'_2 = \emptyset$.

Furthermore, we assume w.l.o.g. (this can be achieved by suitably modifying M_i) that for every $y \in \tilde{\mathcal{Y}}$ and for each $i \in \{1,2\}$, there is a unique $z(y,i) \in \mathcal{Z}_i \setminus \mathcal{Y}_i$ such that whenever M_i reads ${}^{y}\triangleright$, it immediately afterwards has to read ${}^{z(y,i)}\triangleright$, and whenever it reads $\triangleleft^{y}$, this immediately follows after having read $\triangleleft^{z(y,i)}$. This ensures that for every $w \in \Sigma^*$ and every $t \in S'_i(w)$ we have

$$\big(\, y \in \mathrm{dom}(t) \iff z(y,i) \in \mathrm{dom}(t) \,\big) \quad \text{and} \quad \big(\, y \in \mathrm{dom}(t) \implies t(y) = t(z(y,i)) \,\big)\,.$$

Therefore, we can assume w.l.o.g. that $E_i \subseteq \mathcal{P}(\mathcal{Z}'_i)$ (this can be achieved by replacing y with $z(y,i)$ for every $y \in \tilde{\mathcal{Y}}$ that occurs in some string-equality constraint specified by E_i).

Recall that we assume that $S = (S_1 * S_2)$ with $* \in \{\cup, \bowtie\}$. Let

$$\tilde{S} \;=\; \pi_{\mathcal{Y}_1\cup\mathcal{Y}_2}\varsigma^{=}_{E_1\cup E_2}(S'_1 * S'_2)\,.$$

Since $S'_1, S'_2 \in \mathsf{reg}\text{-}\mathfrak{S}$ and $* \in \{\cup, \bowtie\}$ we obtain from Theorem 4.5 of [MRV18] that $(S'_1 * S'_2) \in \mathsf{reg}\text{-}\mathfrak{S}$. Therefore, in order to finish the proof of Lemma 2.2 it suffices to prove the following claim.

Claim 2.3. $S = \tilde{S}$, i.e., $S(w) = \tilde{S}(w)$ for every $w \in \Sigma^*$.

The proof is straightforward, but somewhat tedious: Consider an arbitrary $w \in \Sigma^*$ and show that $S(w) \subseteq \tilde{S}(w)$ and $S(w) \supseteq \tilde{S}(w)$.

We first consider the $\subseteq$-part and choose an arbitrary $t \in S(w)$. Our aim is to show that $t \in \tilde{S}(w)$.

Case 1: $* = \cup$. Since $S(w) = S_1(w) \cup S_2(w)$, there exists $i \in \{1,2\}$ such that $t \in S_i(w)$. Since $S_i = \pi_{\mathcal{Y}_i} \varsigma^{=}_{E_i}(S'_i)$, there exists $t' \in S'_i(w)$ such that $t = \pi_{\mathcal{Y}_i}(t')$ and t' satisfies the string-equality constraints specified by E_i — i.e., for all $C \in E$ and all variables $u, v \in C$ with $u, v \in \mathrm{dom}(t')$ we have $t'(u) = t'(v)$.

Furthermore, from $S'_i = [\![M_i]\!]$ we know that $\mathrm{dom}(t') \subseteq \mathcal{Z}'_i \cup \tilde{\mathcal{Y}}$. Since $E_{3-i} \subseteq \mathcal{P}(\mathcal{Z}'_{3-i})$ and $(\mathcal{Z}'_i \cup \tilde{\mathcal{Y}}) \cap \mathcal{Z}'_{3-i} = \emptyset$, t' trivially satisfies the string-equality constraints specified by E_{3-i}. Therefore, $t' \in (\varsigma^{=}_{E_1 \cup E_2}(S'_1 \cup S'_2))(w)$, and hence $t = \pi_{\mathcal{Y}_i}(t') = \pi_{\mathcal{Y}_1 \cup \mathcal{Y}_2}(t') \in (\pi_{\mathcal{Y}_1 \cup \mathcal{Y}_2} \varsigma^{=}_{E_1 \cup E_2}(S'_1 \cup S'_2))(w) = \tilde{S}(w)$.

Case 2: $* = \bowtie$. Since $S(w) = S_1(w) \bowtie S_2(w)$, for each $i \in \{1,2\}$ there is $t_i \in S_i(w)$ such that $t_1 \sim t_2$ and $t = t_1 \bowtie t_2$ (i.e., t_1 and t_2 are compatible, and t is their join result).

Since $S_i = \pi_{\mathcal{Y}_i} \varsigma^{=}_{E_i}(S'_i)$, there exists $t'_i \in S'_i(w)$ such that $t_i = \pi_{\mathcal{Y}_i}(t'_i)$ and t'_i satisfies the string-equality constraints specified by E_i. Moreover, we know that $\mathrm{dom}(t'_i) \subseteq \mathcal{Z}_i$. Hence, since $\mathcal{Z}_1 \cap \mathcal{Z}_2 \subseteq \tilde{\mathcal{Y}} = \mathcal{Y}_1 \cap \mathcal{Y}_2$, we obtain that t'_1 and t'_2 are compatible (because $t_1 \sim t_2$ and $t'_i = \pi_{\mathcal{Y}_i}(t'_i)$). Let $t' := t'_1 \bowtie t'_2$ be their join result. Clearly, $t' \in (S'_1 \bowtie S'_2)(w)$. Furthermore, t' satisfies all string-equality constraints specified by $E_1 \cup E_2$, and therefore, $t' \in (\varsigma^{=}_{E_1 \cup E_2}(S'_1 \bowtie S'_2))(w)$. Finally, observe that $\pi_{\mathcal{Y}_1 \cup \mathcal{Y}_2}(t') = t$, and hence $t \in (\pi_{\mathcal{Y}_1 \cup \mathcal{Y}_2} \varsigma^{=}_{E_1 \cup E_2}(S'_1 \bowtie S'_2))(w) = \tilde{S}(w)$. This finishes the proof of the $\subseteq$-part.

Now, we consider the $\supseteq$-part and choose an arbitrary $t \in \tilde{S}(w)$. Our aim is to show that $t \in S(w)$. By definition of $\tilde{S}$ there exists $t' \in (S'_1 * S'_2)(w)$ such that $t = \pi_{\mathcal{Y}_1 \cup \mathcal{Y}_2}(t')$ and t' satisfies the string-equality constraints specified by $E_1 \cup E_2$.

Case 1: $* = \cup$. Since $t' \in (S'_1 \cup S'_2)(w)$, there is an $i \in \{1,2\}$ such that $t' \in S'_i(w)$. In particular, $\mathrm{dom}(t') \subseteq \mathcal{Z}_i$, and t' satisfies the string-equality constraints specified by E_i. Hence, $t' \in (\varsigma^{=}_{E_i}(S'_i))(w)$. Furthermore, $t = \pi_{\mathcal{Y}_1 \cup \mathcal{Y}_2}(t') = \pi_{\mathcal{Y}_i}(t')$, and hence $t \in (\pi_{\mathcal{Y}_i} \varsigma^{=}_{E_i}(S'_i)(w) = S_i(w) \subseteq S_1(w) \cup S_2(w) = S(w)$.

Case 2: $* = \bowtie$. Since $t' \in (S'_1 \bowtie S'_2)(w)$, for each $i \in \{1,2\}$ there exists $t'_i \in S'_i(w)$ such that $t'_1 \sim t'_2$ and $t' = t'_1 \bowtie t'_2$ (i.e., t'_1 and t'_2 are compatible and t' is their join result). In particular, $\mathrm{dom}(t'_i) \subseteq \mathcal{Z}_i$, and t'_i satisfies the string-equality constraints specified by E_i. Thus, $t_i := \pi_{\mathcal{Y}_i}(t'_i) \in (\pi_{\mathcal{Y}_i} \varsigma^{=}_{E_i}(S'_i))(w) = S_i(w)$. Furthermore, $t_1 \sim t_2$ and $t = t_1 \bowtie t_2$. I.e., $t \in S_1(w) \bowtie S_2(w) = S(w)$.

This finishes the proof of the $\supseteq$-part, the proof of Claim 2.3, and the proof of Lemma 2.2. □

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. 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).[2]

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 [Pet21].

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 *non-overlapping*, if $\mathsf{st}(w)$ is hierarchical or non-overlapping, respectively.

Example 3.2. Let $\mathcal{X} = \{\mathsf{x}, \mathsf{y}, \mathsf{z}\}$ and $\Sigma = \{\mathtt{a}, \mathtt{b}, \mathtt{c}\}$. Then ${}^{\mathsf{x}}\triangleright\,\mathtt{aa}\,\triangleleft^{\mathsf{x}}\,\mathtt{ab}\,{}^{\mathsf{y}}\triangleright\;{}^{\mathsf{z}}\triangleright\,\mathtt{ca}\,\triangleleft^{\mathsf{z}}\,\mathtt{a}\triangleleft^{\mathsf{y}}$ is a functional and hierarchical subword-marked word. The subword-marked word $u = \mathtt{b}\,{}^{\mathsf{x}}\triangleright\,\mathtt{a}\,{}^{\mathsf{y}}\triangleright\,\mathtt{aba}^{\mathsf{z}}\triangleright\mathtt{a}\triangleleft^{\mathsf{z}}\mathtt{c}\triangleleft^{\mathsf{x}}\mathtt{ab}\triangleleft^{\mathsf{y}}\mathtt{c}$ is functional, but not hierarchical, while $v = {}^{\mathsf{x}}\triangleright\mathtt{a}^{\mathsf{y}}\triangleright\mathtt{ba}\triangleleft^{\mathsf{y}}\mathtt{cab}\triangleleft^{\mathsf{x}}\mathtt{caa}$ 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 ${}^{\mathsf{x}}\triangleright\,\mathtt{aa}\,\triangleleft^{\mathsf{x}}\,\mathtt{ab}\,{}^{\mathsf{y}}\triangleright\;{}^{\mathsf{x}}\triangleright\,\mathtt{ca}\,\triangleleft^{\mathsf{x}}\,\mathtt{a}\triangleleft^{\mathsf{y}}$ nor ${}^{\mathsf{x}}\triangleright\,\mathtt{a}\,{}^{\mathsf{y}}\triangleright\,\mathtt{ba}\,\triangleleft^{\mathsf{x}}\,\mathtt{c}$ are valid subword-marked words.

[2]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., [DKM$^+$19, Fre19, FKP18, FT20]). 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 [Sch16] (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 [Sch16], i. e., with actual references, but also the variants without references, we introduce the term subword-marked word for the latter.

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 *non-overlapping* if all $w \in L$ are functional, hierarchical or non-overlapping, 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 non-overlapping spanner if and only if L is functional, hierarchical or non-overlapping, 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 [FKRV15, MRV18]). 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. $\mathsf{reg\text{-}}\mathfrak{S} = [\![\mathsf{reg\text{-}swm\text{-}}\mathfrak{L}]\!]$.

Proof. Let M be a vset-automaton over Σ and with variables $\mathcal{X}$. Then we can intrepret M as an NFA M' over alphabet $\Sigma \cup \Gamma_{\mathcal{X}}$, by interpreting every variable operation $\mathsf{x}\vdash$ and $\dashv\mathsf{x}$ as an ${}^{\mathsf{x}}\!\triangleright$- and $\triangleleft^{\mathsf{x}}$-transition, respectively. We can then further modify M' such that it rejects all inputs that are not subword-marked words (this can be done by simply keeping track in the finite state control which markers have been read so far). With this further modification, we have that $[\![M]\!] = [\![\mathcal{L}(M')]\!]$.

On the other hand, let M be an NFA such that $\mathcal{L}(M)$ is a subword-marked language over Σ and $\mathcal{X}$. Then we can simply interpret all ${}^{\mathsf{x}}\!\triangleright$- and $\triangleleft^{\mathsf{x}}$-transitions as variable operations $\mathsf{x}\vdash$ and $\dashv\mathsf{x}$ in order to obtain a vset-automaton M' with $[\![\mathcal{L}(M)]\!] = [\![M']\!]$. □

Proposition 3.5 justifies that, throughout this paper, instead of using the vset-automata of [FKRV15], we will represent regular spanners by ordinary nondeterministic finite automata (NFA) over $\Sigma \cup \Gamma_{\mathcal{X}}$ that accept subword-marked languages. It is an easy exercise to see that any vset-automaton (as defined in [FKRV15]) can be transformed into an according NFA by increasing the automaton's state space by at most the factor $3^{|\mathcal{X}|}$. If the vset-automaton is *sequential* (see, e. g., [ABMN21]), then the NFA is even of linear size.

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* $\mathrm{O}(|M| \cdot |\mathcal{X}|^2)$ *if* $\mathcal{L}(M)$ *is a subword-marked language, and, if so, whether* $\mathcal{L}(M)$ *is functional in time* $\mathrm{O}(|M| \cdot |\mathcal{X}|^2)$*, and whether it is hierarchical or non-overlapping in time* $\mathrm{O}(|M| \cdot |\mathcal{X}|^3)$*.*

Proof. A word $w \in (\Sigma \cup \Gamma_{\mathcal{X}})^*$ is *not* a subword-marked word if and only if there is an $\mathsf{x} \in \mathcal{X}$ such that one of the following properties is satisfied:

- $w = w_1 \lhd^{\mathsf{x}} w_2$ with $|w_1|_{\mathsf{x}\rhd} = 0$,
- $w = w_1 \,{}^{\mathsf{x}}\!\rhd w_2$ with $|w_2|_{\lhd^{\mathsf{x}}} = 0$,
- $|w|_{\mathsf{x}\rhd} \geqslant 2$ or $|w|_{\lhd^{\mathsf{x}}} \geqslant 2$.

Moreover, a subword-marked word over Σ and $\mathcal{X}$ is *not* functional if and only if there is an $\mathsf{x} \in \mathcal{X}$ such that $|w|_{\mathsf{x}\rhd} = 0$; it is *not* hierarchical if there are $\mathsf{x}, \mathsf{y} \in \mathcal{X}$ such that $w = w_1 \,{}^{\mathsf{x}}\!\rhd w_2 \,{}^{\mathsf{y}}\!\rhd w_3 \lhd^{\mathsf{x}} w_4 \lhd^{\mathsf{y}} w_5$ with $\mathfrak{e}(w_2) \neq \varepsilon$, $\mathfrak{e}(w_3) \neq \varepsilon$ and $\mathfrak{e}(w_4) \neq \varepsilon$; and it is *not* non-overlapping if there are $\mathsf{x}, \mathsf{y} \in \mathcal{X}$ such that one of the following properties is satisfied:

- $w = w_1 \,{}^{\mathsf{x}}\!\rhd w_2 \,{}^{\mathsf{y}}\!\rhd w_3 \lhd^{\mathsf{x}} w_4 \lhd^{\mathsf{y}} w_5$ and $\mathfrak{e}(w_3) \neq \varepsilon$ and $\mathfrak{e}(w_2 w_4) \neq \varepsilon$
- $w = w_1 \,{}^{\mathsf{x}}\!\rhd w_2 \,{}^{\mathsf{y}}\!\rhd w_3 \lhd^{\mathsf{y}} w_4 \lhd^{\mathsf{x}} w_5$ and
  - $\mathfrak{e}(w_3) = \varepsilon$ and $\mathfrak{e}(w_2) \neq \varepsilon$ and $\mathfrak{e}(w_4) \neq \varepsilon$,
  - $\mathfrak{e}(w_3) \neq \varepsilon$ and $\mathfrak{e}(w_2 w_4) \neq \varepsilon$.

These considerations show that we can construct an NFA N that checks whether a given word over $\Sigma \cup \Gamma_{\mathcal{X}}$ is *not* a subword-marked word over Σ and $\mathcal{X}$, or whether a given subword-marked word over Σ and $\mathcal{X}$ is *not* functional. Note that N needs a constant number of states per $\mathsf{x} \in \mathcal{X}$, and the outdegree is bounded by $\mathrm{O}(|\Sigma \cup \Gamma_{\mathcal{X}}|) = \mathrm{O}(|\mathcal{X}|)$; thus, $|N| = \mathrm{O}(|\mathcal{X}|^2)$.

Moreover, we can construct an NFA N that checks whether a given subword-marked word over Σ and $\mathcal{X}$ is *not* hierarchical or *not* non-overlapping. Since we now need a constant number of states for each two variables $\mathsf{x}, \mathsf{y} \in \mathcal{X}$, we can assume that $|N| = \mathrm{O}(|\mathcal{X}|^3)$.

Finally, we can check whether $\mathcal{L}(M)$ is a subword-marked language by checking non-emptiness for the cross-product automaton of M and N (i. e., the automaton that accepts the intersection $\mathcal{L}(M) \cap \mathcal{L}(N)$). Since the cross-product automaton has size $\mathrm{O}(|M| \cdot |N|) = \mathrm{O}(|M| \cdot |\mathcal{X}|^2)$, this can be done in time $\mathrm{O}(|M| \cdot |\mathcal{X}|^2)$. Analogously, checking functionality for a subword-marked language given by an NFA M can be done in time $\mathrm{O}(|M| \cdot |\mathcal{X}|^2)$ by exploiting the respective automata N for checking non-functionality, and checking hierarchicality or the non-overlapping property can be done in time $\mathrm{O}(|M| \cdot |\mathcal{X}|^3)$ by exploiting the respective automata N for checking non-hierarchicality or the overlapping property, respectively. □

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 $w_1 = v_1 \,{}^{\mathsf{x}}\!\rhd v_2 \lhd^{\mathsf{x}} v_3$.

Since ref-words are subword-marked words, the properties *functional*, *hierarchical* and *non-overlapping* are well-defined.

Example 4.2. Let $\Sigma = \{\mathtt{a}, \mathtt{b}, \mathtt{c}\}$ and $\mathcal{X} = \{\mathtt{x}, \mathtt{y}\}$. The subword-marked words $u = \mathtt{ab}\ {}^{\mathtt{x}}\triangleright\ \mathtt{ab}\ \triangleleft^{\mathtt{x}}\ \mathtt{c}\ {}^{\mathtt{y}}\triangleright\ \mathtt{xaa}\ \triangleleft^{\mathtt{y}}\ \mathtt{y}$ and $v = \mathtt{a}\ {}^{\mathtt{x}}\triangleright\ \mathtt{ab}\ {}^{\mathtt{y}}\triangleright\ \mathtt{ab}\ \triangleleft^{\mathtt{x}}\ \mathtt{a}\ \triangleleft^{\mathtt{y}}\ \mathtt{xy}$ (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, $\mathtt{axb}\ {}^{\mathtt{x}}\triangleright\ \mathtt{ab}\ \triangleleft^{\mathtt{x}}\ \mathtt{c}\ {}^{\mathtt{y}}\triangleright\ \mathtt{xaa}\ \triangleleft^{\mathtt{y}}\ \mathtt{y}$ and $\mathtt{aa}\ {}^{\mathtt{x}}\triangleright\ \mathtt{ab}\ \triangleleft^{\mathtt{x}}\ \mathtt{c}\ {}^{\mathtt{y}}\triangleright\ \mathtt{ya}\triangleleft^{\mathtt{y}}$ 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 $\mathtt{x} \in \mathcal{X}$ are interpreted as *references* to the subword ${}^{\mathtt{x}}\triangleright\, v \triangleleft^{\mathtt{x}}$, which we will call the *definition* of $\mathtt{x}$. 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 $\mathtt{x}$ contains references of $\mathtt{y}$, and the definition of $\mathtt{y}$ contains references of $\mathtt{z}$ and so on. Next, we formally define this nested referencing process encoded by ref-words. Recall that for a subword-marked word w we denote by $\mathfrak{e}(w)$ 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 ${}^{\mathtt{x}}\triangleright\, v_{\mathtt{x}} \triangleleft^{\mathtt{x}}$ be a definition such that $\mathfrak{e}(v_{\mathtt{x}}) \in \Sigma^*$.
(2) Replace all occurrences of $\mathtt{x}$ in w by $\mathfrak{e}(v_{\mathtt{x}})$.

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 $\mathcal{X} = \{\mathtt{x}_1, \mathtt{x}_2, \mathtt{x}_3, \mathtt{x}_4\}$ and let

$$w = \mathtt{aa}\ {}^{\mathtt{x}_1}\triangleright\ \mathtt{ab}\ {}^{\mathtt{x}_2}\triangleright\ \mathtt{acc}\ \triangleleft^{\mathtt{x}_2}\ \mathtt{ax}_2\ \triangleleft^{\mathtt{x}_1}\ {}^{\mathtt{x}_4}\triangleright\ \mathtt{x}_1\mathtt{ax}_2\ \triangleleft^{\mathtt{x}_4}\ \mathtt{x}_4\mathtt{bx}_1\,.$$

Due to the definition ${}^{\mathtt{x}_2}\triangleright\ \mathtt{acc}\triangleleft^{\mathtt{x}_2}$, the procedure of Definition 4.3 will initially replace all references of $\mathtt{x}_2$ by $\mathtt{acc}$. Then, the definition for variable $\mathtt{x}_1$ is ${}^{\mathtt{x}_1}\triangleright\ \mathtt{ab}\ {}^{\mathtt{x}_2}\triangleright\ \mathtt{acc}\ \triangleleft^{\mathtt{x}_2}\ \mathtt{aacc}\triangleleft^{\mathtt{x}_1}$, so $\mathtt{abaccaacc}$ can be substituted for the references of $\mathtt{x}_1$. After replacing the last variable $\mathtt{x}_4$, we obtain

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

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

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

A set L of ref-words is called a *ref-language*. We extend the $\mathfrak{d}(\cdot)$-function to ref-languages L in the obvious way, i. e., $\mathfrak{d}(L) = \{\mathfrak{d}(w) \mid w \in L\}$. Note that $\mathfrak{d}(L)$ is necessarily a subword-marked language. 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* $\mathrm{O}(|M| \cdot |\mathcal{X}|^2)$ *if* $\mathcal{L}(M)$ *is a ref-language over* Σ *and* $\mathcal{X}$*.*

Proof. Since M accepts a subword-marked language over $\Sigma \cup \mathcal{X}$ and $\mathcal{X}$, it does *not* accept a ref-language over Σ and $\mathcal{X}$ if and only if it accepts a word $w \in (\Sigma \cup \mathcal{X} \cup \Gamma_{\mathcal{X}})^*$ such that, for some $\mathsf{x} \in \mathcal{X}$, $w = w_1 \mathsf{x} w_2$ with $|w_1|_{\dashv^{\mathsf{x}}} = 0$. It is straightforward to construct an NFA N that accepts all words over $\Sigma \cup \Gamma_{\mathcal{X}} \cup \mathcal{X}$ with this property, and also $|N| = \mathrm{O}(|\mathcal{X}|^2)$ (per each variable, we need a constant number of states with an outdegree bounded by $\mathrm{O}(|\Sigma \cup \Gamma_{\mathcal{X}} \cup \mathcal{X}|) = \mathrm{O}(|\mathcal{X}|)$). Therefore, we can check whether $\mathcal{L}(M)$ is a ref-language over Σ and $\mathcal{X}$ by constructing the cross-product automaton of M and N and check whether it accepts $\emptyset$. This can be done in time $\mathrm{O}(|M| \cdot |N|) = \mathrm{O}(|M| \cdot |\mathcal{X}|^2)$. □

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 $[\![\mathfrak{L}]\!]_{\mathfrak{d}} = \{[\![L]\!]_{\mathfrak{d}} \mid L \in \mathfrak{L}\}$. The class of *refl-spanners* is $\mathsf{refl}\text{-}\mathfrak{S} = [\![\mathsf{reg}\text{-}\mathsf{ref}\text{-}\mathfrak{L}]\!]_{\mathfrak{d}}$.

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 provides 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* (JACM) from 2000 to 2010). Let $\Sigma' = \Sigma \cup \{\#\}$. We define a refl-spanner

$$\alpha \quad = \quad \Sigma'^* \,\#\, \Sigma^* \;\mathsf{email}\colon\; {}^{\mathsf{x}}\!\triangleright\, \Sigma^+ \triangleleft^{\mathsf{x}} \;@\; r_{\mathsf{dom}}\, \Sigma^* \,\#\, \Sigma'^* \;\mathsf{email}\colon\; \mathsf{x} \;@\; r_{\mathsf{dom}}\, \Sigma^* \,\#\, \Sigma'^*\,,$$

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 a given spanner S over Σ and $\mathcal{X}$, $w \in \Sigma^*$ and $(\mathcal{X}, w)$-tuple t. The problem NonEmptiness is to decide whether $S(w) \neq \emptyset$ for given S and w. For the problem Satisfiability, we get a single spanner S as input and ask whether there is a $w \in \Sigma^*$ with $S(w) \neq \emptyset$, and for the problems Hierarchicality and Functionality, we get a single spanner S and ask 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 that accept a ref-language. Recall that a summary of our results is provided by Table 1 in the introduction.

We first define some concepts that shall be helpful for dealing with the issue that 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.

Let w be a subword-marked word over Σ and $\mathcal{X}$, let $t = \mathsf{st}(w)$, and let $\mathfrak{e}(w) = w_1 w_2 \dots w_n$ with $w_i \in \Sigma$ for every $i \in [n]$. The *marker-set representation* of w is a word $\mathsf{msrep}(w)$ over the extended alphabet $\Sigma \cup \mathcal{P}(\Gamma_{\mathcal{X}})$, where each subset of $\Gamma_{\mathcal{X}}$ serves as a letter of the alphabet, as follows. For each $i \in \{1, \dots, n+1\}$ let Γ_i be the set comprising of all symbols ${}^{\mathsf{x}}\!\triangleright$ where $\mathsf{x} \in \mathrm{dom}(t)$ and $t(\mathsf{x}) = [i, j\rangle$ for some j, and all symbols $\triangleleft^{\mathsf{x}}$ where $\mathsf{x} \in \mathrm{dom}(t)$ and $t(\mathsf{x}) = [j, i\rangle$ for some j. Then $\mathsf{msrep}(w) = (\Gamma_1)^{|\Gamma_1|} w_1 (\Gamma_2)^{|\Gamma_2|} w_2 \dots (\Gamma_n)^{|\Gamma_n|} w_n (\Gamma_{n+1})^{|\Gamma_{n+1}|}$. Let us illustrate these definitions with an example.[3]

Example 5.1. If $\mathcal{X} = \{\mathsf{x}_1, \mathsf{x}_2, \mathsf{x}_3\}$, $w = {}^{\mathsf{x}_1}\!\triangleright\ {}^{\mathsf{x}_3}\!\triangleright\ \mathtt{ab}\ \triangleleft^{\mathsf{x}_1}\ {}^{\mathsf{x}_2}\!\triangleright\ \mathtt{cb}\ \triangleleft^{\mathsf{x}_2}\ \mathtt{abca}\triangleleft^{\mathsf{x}_3}$ (and therefore $\mathsf{st}(w) = ([1,3\rangle, [3,5\rangle, [1,9\rangle)$), then $\Gamma_1 = \{{}^{\mathsf{x}_1}\!\triangleright, {}^{\mathsf{x}_3}\!\triangleright\}$, $\Gamma_3 = \{\triangleleft^{\mathsf{x}_1}, {}^{\mathsf{x}_2}\!\triangleright\}$, $\Gamma_5 = \{\triangleleft^{\mathsf{x}_2}\}$, $\Gamma_9 = \{\triangleleft^{\mathsf{x}_3}\}$, and $\Gamma_i = \emptyset$ for every $i \in \{2, 4, 6, 7, 8\}$. Consequently,

$$\mathsf{msrep}(w) = \{{}^{\mathsf{x}_1}\!\triangleright, {}^{\mathsf{x}_3}\!\triangleright\}\{{}^{\mathsf{x}_1}\!\triangleright, {}^{\mathsf{x}_3}\!\triangleright\}\mathtt{ab}\{\triangleleft^{\mathsf{x}_1}, {}^{\mathsf{x}_2}\!\triangleright\}\{\triangleleft^{\mathsf{x}_1}, {}^{\mathsf{x}_2}\!\triangleright\}\mathtt{cb}\{\triangleleft^{\mathsf{x}_2}\}\mathtt{abca}\{\triangleleft^{\mathsf{x}_3}\}\,.$$

We also extend the function $\mathsf{msrep}(\cdot)$ in the natural way to a subword-marked language L by setting $\mathsf{msrep}(L) = \{\mathsf{msrep}(w) \mid w \in L\}$.

As indicated above, the main idea of the marker-set representation is to represent a subword-marked word in a way that abstracts from the differences caused by reading the same markers from $\Gamma_{\mathcal{X}}$ just in a different order, which, for subword-marked words considered as strings makes a difference, but has no effect on the represented document or the represented span-tuple. Such a representation could also be defined without repeating each Γ_i for $|\Gamma_i|$ times, i. e., by defining $\mathsf{msrep}(w) = \Gamma_1 w_1 \Gamma_2 w_2 \dots \Gamma_n w_n \Gamma_{n+1}$ (or, illustrated with Example 5.1 from above, as $\mathsf{msrep}(w) = \{{}^{\mathsf{x}_1}\!\triangleright, {}^{\mathsf{x}_3}\!\triangleright\}\mathtt{ab}\{\triangleleft^{\mathsf{x}_1}, {}^{\mathsf{x}_2}\!\triangleright\}\mathtt{cb}\{\triangleleft^{\mathsf{x}_2}\}\mathtt{abca}\{\triangleleft^{\mathsf{x}_3}\}$). However, the definition with repeated occurrences of Γ_i will be more convenient for proving the results of this section. Later on, in Section 6.2, we shall also use this simplified version of the marker-set representation.

The next propositions point out the relevant properties of the marker-set representation. The first one is a direct consequence of the definition of the marker-set representation, the second one follows from the first one. We state these propositions mainly for demonstrating the meaning of the marker-set representation and for general illustrational purposes; for our results, the ref-word analogues of Proposition 5.2 and 5.3 (formally stated as Lemmas 5.11 and 5.12) are more important and shall be proven in detail.

[3] Also note that the marker-set representation is similar to the extended vset-automata used in [ABMN21].

Proposition 5.2. *Let w_1, w_2 be subword-marked words. Then $\mathfrak{e}(w_1) = \mathfrak{e}(w_2)$ and $\mathsf{st}(w_1) = \mathsf{st}(w_2)$ if and only if $\mathsf{msrep}(w_1) = \mathsf{msrep}(w_2)$.*

Proposition 5.3. *Let L_1, L_2 be subword-marked languages. Then $[\![L_1]\!] \subseteq [\![L_2]\!]$ if and only if $\mathsf{msrep}(L_1) \subseteq \mathsf{msrep}(L_2)$.*

Note that the above proposition states that $\mathsf{msrep}(L_1) \subseteq \mathsf{msrep}(L_2)$ is characteristic for the inclusion of the corresponding spanners (i. e., $[\![L_1]\!] \subseteq [\![L_2]\!]$), but not for the inclusion of the corresponding subword-marked languages L_1 and L_2. More precisely, $\mathsf{msrep}(L_1) \subseteq \mathsf{msrep}(L_2)$ is not a sufficient condition for $L_1 \subseteq L_2$ (although, as can be easily seen, it is a necessary condition).

5.1. **Model Checking and Non-Emptiness.** We now consider the problem ModelChecking (recall that this is the problem 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).

Theorem 5.4. ModelChecking *for* refl-$\mathfrak{S}$ *can be solved in time* $\mathrm{O}((|w| + |\mathcal{X}|)|M| \log(|\mathcal{X}|))$, *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.*

Proof. For convenience, we will assume that every symbol in $\Gamma_{\mathcal{X}}$ occurs in some transition of M (this is w.l.o.g., because if ${}^{\mathsf{x}}\triangleright$ or $\triangleleft^{\mathsf{x}}$ does not, we can safely remove x from $\mathcal{X}$ and reject t in case that $\mathsf{x} \in \mathrm{dom}(t)$). In particular, $|\mathcal{X}| \leqslant |M|$.

The proof is split in two parts. First, we consider the *regular case*, where $\mathcal{L}(M)$ is not a ref-language, but just a subword-marked language over Σ and $\mathcal{X}$ (i. e., $[\![\mathcal{L}(M)]\!]$ is a regular spanner). The algorithm for this case is then used for the general *refl case* where $\mathcal{L}(M)$ is a ref-language over Σ and $\mathcal{X}$ (i. e., $[\![\mathcal{L}(M)]\!]$ is a refl-spanner).

Regular case: We shall denote by $\mathsf{msrep}(M)$ the NFA M with the modification that we interpret each transition labelled with some $\sigma \in \Gamma_{\mathcal{X}}$ as a transition that can read any symbol $\Gamma \subseteq \Gamma_{\mathcal{X}}$ with $\sigma \in \Gamma$. This means that $\mathsf{msrep}(M)$ accepts a language over the alphabet $\Sigma \cup \mathcal{P}(\Gamma_{\mathcal{X}})$ and, moreover, for every $w \in \mathcal{L}(M)$ and arbitrary subword-marked word w' with $\mathsf{msrep}(w) = \mathsf{msrep}(w')$ (this includes w itself), we have that $w' \in \mathcal{L}(\mathsf{msrep}(M))$.

By definition, $t \in [\![\mathcal{L}(M)]\!](w)$ if and only if there exists a subword-marked word $v \in \mathcal{L}(M)$ such that $\mathsf{st}(v) = t$ and $\mathfrak{e}(v) = w$. This latter property can be checked as follows.

We first combine t and w into a subword-marked word w', i. e., $\mathsf{st}(w') = t$ and $\mathfrak{e}(w') = w$ (the order in which we place consecutive occurrences of symbols from $\Gamma_{\mathcal{X}}$ is not relevant). In the following, we shall consider $\mathsf{msrep}(w')$ (the construction of which we discuss later on), and the automaton $\mathsf{msrep}(M)$ as explained above.

Claim 5.5. The following statements are equivalent:
(1) There exists a subword-marked word $v \in \mathcal{L}(M)$ such that $\mathsf{st}(v) = t$ and $\mathfrak{e}(v) = w$.
(2) $\mathsf{msrep}(w') \in \mathcal{L}(\mathsf{msrep}(M))$.

Proof. We start with "(1) $\implies$ (2)" and assume that there exists a subword-marked word $v \in \mathcal{L}(M)$ such that $\mathsf{st}(v) = t$ and $\mathfrak{e}(v) = w$. The fact $\mathfrak{e}(v) = w$ means that v equals w with some (possibly empty) factors over $\Gamma_{\mathcal{X}}$ between the symbols of Σ. Moreover, $\mathsf{st}(v) = t$ means that if in v we replace each maximal factor $\sigma_1\sigma_2 \ldots \sigma_k$ over $\Gamma_{\mathcal{X}}$ with $\{\sigma_1, \sigma_2, \ldots, \sigma_k\}^k$, we get exactly $\mathsf{msrep}(w')$, and, by construction of $\mathsf{msrep}(M)$, the fact that $v \in \mathcal{L}(M)$ also implies that $\mathsf{msrep}(w') \in \mathcal{L}(\mathsf{msrep}(M))$ (i. e., $\mathsf{msrep}(M)$ accepts $\mathsf{msrep}(w')$ via the same sequence of states).

Next, we prove "(2) $\implies$ (1)" and assume that $\mathsf{msrep}(w') \in \mathcal{L}(\mathsf{msrep}(M))$, which means that there is an accepting run of $\mathsf{msrep}(M)$ on $\mathsf{msrep}(w')$. By construction, each transition of this run that reads a symbol $\Gamma \subseteq \Gamma_{\mathcal{X}}$ is a re-interpretation of an original σ-transition of M for some $\sigma \in \Gamma$. Consequently, for every symbol $\Gamma \subseteq \Gamma_{\mathcal{X}}$ of $\mathsf{msrep}(w')$, there is a well-defined *representative* $\sigma \in \Gamma$. If we replace every symbol $\Gamma \subseteq \Gamma_{\mathcal{X}}$ in $\mathsf{msrep}(w')$ by its representative, then we obtain a word v over $\Sigma \cup \Gamma_{\mathcal{X}}$, and by construction of $\mathsf{msrep}(M)$, we also have $v \in \mathcal{L}(M)$, which means that v is a subword-marked word. By construction of $\mathsf{msrep}(w')$, it is also obvious that $\mathfrak{e}(v) = \mathfrak{e}(\mathsf{msrep}(w')) = w$. Moreover, in the construction of v, every factor in $\mathsf{msrep}(w')$ of the form $\Gamma^{|\Gamma|}$ with $\Gamma \subseteq \Gamma_{\mathcal{X}}$, is replaced by a factor $\sigma_1\sigma_2\ldots\sigma_{|\Gamma|}$ with $\sigma_i \in \Gamma$ for every $i \in [|\Gamma|]$. Since v is a subword-marked word, this means that $\{\sigma_1, \sigma_2, \ldots, \sigma_{|\Gamma|}\} = \Gamma$, which means that $(\sigma_1, \sigma_2, \ldots, \sigma_{|\Gamma|})$ is some linear ordering of Γ. By construction of $\mathsf{msrep}(w')$, this directly implies that $\mathsf{st}(v) = t$. □

Claim 5.5 means that in order to check whether $t \in [\![\mathcal{L}(M)]\!](w)$, it is sufficient to check whether $\mathsf{msrep}(w') \in \mathcal{L}(\mathsf{msrep}(M))$. Therefore, we shall now discuss how we can efficiently check $\mathsf{msrep}(w') \in \mathcal{L}(\mathsf{msrep}(M))$. First, we have to construct (a suitable representation of) $\mathsf{msrep}(w')$, which we do in the following way.

Let us assume that $w' = \gamma_1 w_1 \gamma_2 w_2 \ldots \gamma_n w_n \gamma_{n+1}$, where $w = w_1 w_2 \ldots w_n$ and the $\gamma_1, \gamma_2, \ldots, \gamma_{n+1}$ are (possibly empty) factors over $\Gamma_{\mathcal{X}}$. Consequently,

$$\mathsf{msrep}(w') = (\Gamma_1)^{|\Gamma_1|} w_1 (\Gamma_2)^{|\Gamma_2|} w_2 \ldots (\Gamma_n)^{|\Gamma_n|} w_n (\Gamma_{n+1})^{|\Gamma_{n+1}|}$$

according to the definition of $\mathsf{msrep}(\cdot)$. Let us further assume that we have some fixed order on $\Gamma_{\mathcal{X}}$. We initialise an array A of size $n+1$ in time $O(n)$. Then we move over w' from left to right, and whenever we encounter some $\gamma_i = \sigma_1\sigma_2\ldots\sigma_{n_i}$, we construct a binary search tree (with respect to the order on $\Gamma_{\mathcal{X}}$) of the elements of $\Gamma_i = \{\sigma_1, \sigma_2, \ldots, \sigma_{n_i}\}$, and we store this search tree in entry i of A. For this, we have to consider each symbol from w' only once and we have to insert at most $|\Gamma_{\mathcal{X}}|$ symbols from $\Gamma_{\mathcal{X}}$ into a binary search tree of size at most $|\Gamma_{\mathcal{X}}|$. Consequently, this can be done in time $\mathrm{O}(|w'| + (|\mathcal{X}| \log(|\mathcal{X}|)) = \mathrm{O}(|w| + (|\mathcal{X}| \log(|\mathcal{X}|))$. We then represent $\mathsf{msrep}(w')$ as the string $1^{|\Gamma_1|} w_1 2^{|\Gamma_2|} w_2 \ldots n^{|\Gamma_n|} w_n (n+1)^{|\Gamma_{n+1}|}$ and the array A, where the symbols i point to the i^{th} entry of A.

With this representation of $\mathsf{msrep}(w')$, we can read $\mathsf{msrep}(w')$ with $\mathsf{msrep}(M)$, and every transition can be evaluated in constant time for symbols from Σ, and in time $\mathrm{O}(\log(|\mathcal{X}|))$ for symbols from $\Gamma_{\mathcal{X}}$. More precisely, transitions labelled with symbols from Σ can be evaluated in constant time by directly comparing the next input symbol with the transition label from Σ, while for evaluating transition labels from $\Gamma_{\mathcal{X}}$, we have to check whether the input symbol $\Gamma \subseteq \Gamma_{\mathcal{X}}$ (from $\mathsf{msrep}(w')$) contains the transition label, which, since we have Γ represented as binary search tree, can be done in time $\mathrm{O}(\log(|\mathcal{X}|))$. Consequently, we can check whether $\mathsf{msrep}(M)$ accepts $\mathsf{msrep}(w')$ in time $\mathrm{O}(|\mathsf{msrep}(w')| \cdot |\mathsf{msrep}(M)| \cdot \log(|\mathcal{X}|)) = \mathrm{O}((|w| + |\mathcal{X}|)|M| \log(|\mathcal{X}|))$. Since this also dominates the time needed for constructing the representation of $\mathsf{msrep}(w')$, the total running time is $\mathrm{O}((|w| + |\mathcal{X}|)|M| \log(|\mathcal{X}|))$.

Refl case: In this case, $t \in [\![\mathcal{L}(M)]\!](w)$ if and only if there is some $v \in \mathcal{L}(M)$ such that $\mathsf{st}(\mathfrak{d}(v)) = t$ and $\mathfrak{e}(\mathfrak{d}(v)) = w$. We next show how this latter property can be checked.

For every $\mathsf{x} \in \mathrm{dom}(t)$ let w_{x} be the subword of w that corresponds to $t(\mathsf{x})$, i.e., $w_{\mathsf{x}} = w[i, j\rangle$ if $[i, j\rangle = t(\mathsf{x})$. Next, we obtain an NFA M' from M by replacing every x-transition for every $\mathsf{x} \in \mathcal{X}$ by a path of transitions labelled with the word w_{x}. It is important to observe that M' accepts a subword-marked language over Σ and $\mathcal{X}$ (indeed, replacing the references of ref-words by some factors over Σ necessarily yields valid subword-marked words).

Claim 5.6. The following statements are equivalent:
(1) There exists a ref-word $v \in \mathcal{L}(M)$ such that $\mathsf{st}(\mathfrak{d}(v)) = t$ and $\mathfrak{e}(\mathfrak{d}(v)) = w$.
(2) There exists a subword-marked word $v' \in \mathcal{L}(M')$ such that $\mathsf{st}(v') = t$ and $\mathfrak{e}(v') = w$.

Proof. We start with "(1) $\implies$ (2)" and assume that there exists some $v \in \mathcal{L}(M)$ such that $\mathsf{st}(\mathfrak{d}(v)) = t$ and $\mathfrak{e}(\mathfrak{d}(v)) = w$. Since $\mathsf{st}(\mathfrak{d}(v)) = t$, we know that $\mathfrak{d}(v)$ equals the word obtained from v by replacing each occurrence of a reference x by the word w_x. Hence, defining $v' := \mathfrak{d}(v)$, we have $v' \in \mathcal{L}(M')$, and, by assumption, $\mathsf{st}(v') = t$ and $\mathfrak{e}(v') = w$.

Next, we prove "(2) $\implies$ (1)" and assume that there exists some $v' \in \mathcal{L}(M')$ such that $\mathsf{st}(v') = t$ and $\mathfrak{e}(v') = w$. We obtain a word v from the accepting run of M' on v' as follows. We simply trace all the symbols in the order read by the transitions, but whenever we read some w_x by some of the paths that M's x-transitions have been replaced with, we use the symbol x instead of the factor w_x. By construction of M', we have $v \in \mathcal{L}(M)$, which means that v is a ref-word and therefore $\mathfrak{d}(v)$ is well-defined. Furthermore, since $\mathsf{st}(v') = t$ and by construction of the w_x, we also have $\mathfrak{d}(v) = v'$. Finally, by assumption, we have $\mathsf{st}(v') = t$ and $\mathfrak{e}(v') = w$, which implies $\mathsf{st}(\mathfrak{d}(v)) = t$ and $\mathfrak{e}(\mathfrak{d}(v)) = w$. □

We observe that Claim 5.6 means that in order to check $t \in [\![\mathcal{L}(M)]\!](w)$, it is sufficient to check whether there exists a subword-marked word $v \in \mathcal{L}(M')$ (recall that M' accepts a subword-marked language) such that $\mathsf{st}(v) = t$ and $\mathfrak{e}(v) = w$. This latter task is exactly the *regular case* discussed above; thus, it can be done in time $\mathrm{O}((|w| + |\mathcal{X}|)|M'| \log(|\mathcal{X}|))$. However, since we obtained M' from M by replacing some transitions by paths of $\mathrm{O}(|w|)$ transitions, we can only bound $|M'| = \mathrm{O}(|w| \cdot |M|)$. This yields an overall upper bound of $\mathrm{O}((|w| + |\mathcal{X}|)|w| \cdot |M| \cdot \log(|\mathcal{X}|))$, which is quadratic in $|w|$. We shall now discuss how we can use a standard data-structure for strings in order to implement our algorithmic idea more efficiently.

Let us recall that the algorithm for the refl case first constructs M' by replacing x-transitions by w_x-paths, then we re-interpret the $\Gamma_\mathcal{X}$-transitions of M' as described in the *regular case*, and then we check whether this NFA accepts $\mathsf{msrep}(w')$ (for a suitable w' obtained from w and t). The whole improvement consists in just keeping the x-transitions instead of replacing them by w_x-paths, but then, when checking whether $\mathsf{msrep}(w')$ is accepted, we nevertheless want to treat x-transitions as w_x-transitions, but in constant time (i. e., we want to check whether the remaining input starts with w_x in constant time), for which we need the data structure. More precisely, we will compute a data structure that allows us to check, for every position i of $\mathsf{msrep}(w')$ and every $\mathsf{x} \in \mathcal{X}$, whether w_x is a prefix of $\mathsf{msrep}(w')[i..|\mathsf{msrep}(w')|]$ in constant time. With this addition to the algorithm, the overall running time is still $\mathrm{O}((|w| + |\mathcal{X}|)|M'| \log(|\mathcal{X}|))$ (note that for this it is vital that we can check *in constant time* whether the remaining input starts with w_x), but we have $|M'| = \mathrm{O}(|M|)$, which leads to the overall running time $\mathrm{O}((|w| + |\mathcal{X}|)|M| \log(|\mathcal{X}|))$. Let us now give the details for the improvement.

For a word u and for every $i \in [|w|+1]$ let $\mathsf{suff}_i(u)$ be the suffix of u starting at position i of u. A *longest common extension* data structure LCE_u for a word u is defined such that $\mathsf{LCE}_u(i, j)$, for i, j with $1 \leqslant i < j \leqslant |u|$, is the length of the longest common prefix of $\mathsf{suff}_i(u)$ and $\mathsf{suff}_j(u)$. It is known that (see, e. g., [FH06]) LCE_u can be constructed in time $\mathrm{O}(|u|)$ and afterwards $\mathsf{LCE}_u(i, j)$ can be retrieved in constant time for arbitrary given i and j.

We construct $\mathsf{LCE}_{w\hat{w}}$, where $\hat{w}$ is obtained from $\mathsf{msrep}(w')$ by replacing all symbols from $\mathcal{P}(\Gamma_\mathcal{X})$ by a new symbol $\#$, i. e., $\# \notin \Sigma \cup \Gamma_\mathcal{X}$ (note that $|\mathsf{msrep}(w')| = |\hat{w}|$ and these words only differ with respect to symbols $\mathcal{P}(\Gamma_\mathcal{X})$ and $\#$). Now, for a given $\mathsf{x} \in \mathcal{X}$ with $t(\mathsf{x}) = [i_\mathsf{x}, j_\mathsf{x}\rangle$,

and position i of $\mathsf{msrep}(w')$, we have that w_{x} is a prefix of $\mathsf{msrep}(w')[i..|\mathsf{msrep}(w')|]$ if and only if $\mathsf{LCE}_{w\hat{w}}(i_{\mathsf{x}}, |w|+i) \geqslant j_{\mathsf{x}} - i_{\mathsf{x}}$. Finally, observe that $\mathsf{LCE}_{w\hat{w}}$ can be constructed in time $\mathrm{O}(|w\hat{w}|) = \mathrm{O}(|w| + |\mathcal{X}|)$. This completes the proof of Theorem 5.4. □

We discuss a few particularities about Theorem 5.4. The result points out that ModelChecking for refl-spanners has the same complexity as for regular spanners,[4] which can be considered surprising, given the fact that refl-spanners cover a large class of core spanners and are generally much more expressive than regular spanners. Moreover, for core spanners the problem ModelChecking is NP-complete. Another interesting fact is that in data complexity, ModelChecking can be solved in linear time for both regular as well as refl-spanners (whereas the latter result is slightly more complicated, since it depends on using the longest common extension data structure).

Next, we consider the problem NonEmptiness (i. e., to decide whether $S(w) \neq \emptyset$ for given S and w), for which we can obtain the following theorem by standard methods.

Theorem 5.7. NonEmptiness *for* refl-$\mathfrak{S}$ *is* NP*-complete.*

Proof. The NP-hardness can be proven in a similar way as Theorem 3.3 in [FH18]. It follows easily by a reduction from the problem of matching patterns with variables, which is as follows: for a given pattern $\alpha \in (\Sigma \cup \mathcal{X})^*$ and a word $w \in \Sigma^*$, decide whether there is a mapping $h\colon \mathcal{X} \to \Sigma^*$ such that $\widehat{h}(\alpha) = w$, where $\widehat{h}$ is the natural extension of h to a morphism $(\mathcal{X} \cup \Sigma)^* \to (\mathcal{X} \cup \Sigma)^*$, by letting h be the identity on Σ. This problem is NP-complete (for more information see, e. g., [FSV16, FS15, FH18]).

Let $\alpha \in (\mathcal{X} \cup \Sigma)^*$ be a pattern with variables and let $w \in \Sigma^*$. Moreover, let β be the regular expression obtained from α by replacing each first occurrence of a variable x by ${}^{\mathsf{x}}\!\triangleright \Sigma^* \triangleleft^{\mathsf{x}}$. It can be easily seen that $\mathcal{L}(\beta)$ is a ref-language over Σ and $\mathcal{X}$, and that $w \in \mathcal{L}(\alpha)$ if and only if $[\![\mathcal{L}(\beta)]\!](w) \neq \emptyset$. This shows that NonEmptiness for refl-$\mathfrak{S}$ is NP-hard.

For membership in NP, let $S = [\![\mathcal{L}(M)]\!] \in \text{refl-}\mathfrak{S}_{\Sigma,\mathcal{X}}$ and let $w \in \Sigma^*$. In order to check whether $S(w) \neq \emptyset$, we guess some $(\mathcal{X}, w)$-tuple t, which can be done in polynomial time. Then we check whether $t \in S(w)$, which can also be done in polynomial time (see Theorem 5.4). □

5.2. **Static Analysis.** We start with the following straightforward result. Recall that for the problems Satisfiability, Hierarchicality and Functionality, we get a single spanner S, represented by an NFA, 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.

Theorem 5.8.

(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* $\mathrm{O}(|M| \cdot |\mathcal{X}|^3)$, *and*
(c) Functionality *for* refl-$\mathfrak{S}$ *can be solved in time* $\mathrm{O}(|M| \cdot |\mathcal{X}|^2)$,

where M is an NFA *that describes a refl-spanner over Σ and $\mathcal{X}$.*

Proof. (a): Let $S \in \text{refl-}\mathfrak{S}_{\Sigma,\mathcal{X}}$ be represented by an NFA M.

If $\mathcal{L}(M) \neq \emptyset$, then there is a $v \in \mathcal{L}(M)$ and therefore, by definition, $\mathsf{st}(\mathfrak{d}(v)) \in [\![\mathcal{L}(M)]\!](\mathfrak{e}(v))$, which means that $S(\mathfrak{e}(v)) \neq \emptyset$. On the other hand, if there is some $w \in \Sigma^*$

[4] To the best knowledge of the authors, the bound that is provided by Theorem 5.4 is also the currently best upper bound for model checking of regular spanners in the literature.

with $S(w) \neq \emptyset$, then there is a span-tuple $t \in [\![\mathcal{L}(M)]\!](w)$, which means that there is a ref-word $v \in \mathcal{L}(M)$ with $\mathfrak{e}(\mathfrak{d}(v)) = w$ and $\mathsf{st}(\mathfrak{d}(v)) = t$. Thus, $\mathcal{L}(M) \neq \emptyset$. Consequently, there is a word $w \in \Sigma^*$ with $S(w) \neq \emptyset$ if and only if $\mathcal{L}(M) \neq \emptyset$. Checking whether $\mathcal{L}(M) \neq \emptyset$ can be done in time $\mathrm{O}(|M|)$, by searching M for a path from the start state to an accepting state.

(b): Let $S \in \mathsf{refl}\text{-}\mathfrak{S}_{\Sigma,\mathcal{X}}$ be represented by an NFA M. It is straightforward to see that S is hierarchical if and only if $\mathcal{L}(M)$ is hierarchical. According to Proposition 3.6 the latter can be checked in time $\mathrm{O}(|M| \cdot |\mathcal{X}|^3)$.

(c): Let $S \in \mathsf{refl}\text{-}\mathfrak{S}_{\Sigma,\mathcal{X}}$ be represented by an NFA M. It is straightforward to see that S is functional if and only if $\mathcal{L}(M)$ is functional. According to Proposition 3.6 the latter can be checked in time $\mathrm{O}(|M| \cdot |\mathcal{X}|^2)$. □

With respect to Theorem 5.8, it is worth recalling that for core spanners, Satisfiability and Hierarchicality are PSpace-complete, even for restricted classes of core spanners (see [FH18]).

Finally, we investigate the problems Containment and Equivalence, which consist in deciding whether $S_1 \subseteq S_2$ or $S_1 = S_2$, respectively, for given spanners S_1 and S_2.

Recall that for core spanners Containment and Equivalence are not semi-decidable (see [FH18]). We now show that for refl-spanners we can achieve decidability of Containment and Equivalence by imposing suitable restrictions.

Let us first briefly discuss the case of regular spanners. Let S_1 and S_2 be regular spanners represented by NFAs M_1 and M_2. Due to Proposition 5.3, we can check whether $S_1 \subseteq S_2$ by checking whether $\mathsf{msrep}(\mathcal{L}(M_1)) \subseteq \mathsf{msrep}(\mathcal{L}(M_2))$. Checking whether $\mathsf{msrep}(\mathcal{L}(M_1)) \not\subseteq \mathsf{msrep}(\mathcal{L}(M_2))$ can be done in PSpace (i. e., we guess a witness w' and check whether $w' \in \mathsf{msrep}(\mathcal{L}(M_1)) \setminus \mathsf{msrep}(\mathcal{L}(M_2))$); we skip all the details here, since the result will follow from our more general result with respect to a subclass of refl-spanners; moreover, the PSpace-completeness of containment of regular spanners has already been mentioned in [MRV18].

Let us give some intuition of why the situation is more complicated for refl-spanners. We first note that also for ref-words (since they are subword-marked words), we can use the function $\mathsf{msrep}(\cdot)$. However, while $\mathsf{msrep}(w_1) = \mathsf{msrep}(w_2)$ for ref-words w_1, w_2 is still sufficient for $\mathfrak{e}(\mathfrak{d}(w_1)) = \mathfrak{e}(\mathfrak{d}(w_2))$ and $\mathsf{st}(\mathfrak{d}(w_1)) = \mathsf{st}(\mathfrak{d}(w_2))$, it is not a necessary condition. For example, the ref-words

$$w_1 = {}^{\mathsf{x}}\triangleright\, \mathtt{ab} \triangleleft^{\mathsf{x}} \mathtt{b}\, {}^{\mathsf{y}}\triangleright\, \mathtt{abb} \triangleleft^{\mathsf{y}} \mathtt{yab}\,,$$
$$w_2 = {}^{\mathsf{x}}\triangleright\, \mathtt{ab} \triangleleft^{\mathsf{x}} \mathtt{b}\, {}^{\mathsf{y}}\triangleright\, \mathtt{xb} \triangleleft^{\mathsf{y}} \mathtt{xbx}\,,$$
$$w_3 = {}^{\mathsf{x}}\triangleright\, \mathtt{ab} \triangleleft^{\mathsf{x}} \mathtt{b}\, {}^{\mathsf{y}}\triangleright\, \mathtt{abb} \triangleleft^{\mathsf{y}} \mathtt{yx}$$

have all the same image ${}^{\mathsf{x}}\triangleright\mathtt{ab}\triangleleft^{\mathsf{x}}\mathtt{b}{}^{\mathsf{y}}\triangleright\mathtt{abb}\triangleleft^{\mathsf{y}}\mathtt{abbab}$ under the function $\mathfrak{d}(\cdot)$ (and therefore they all describe the same document and the same span-tuple), but their marker-set representations are obviously pairwise different.

The main idea in the following is that we impose a restriction to ref-words (and therefore to ref-languages and refl-spanners) that allows us to obtain an analogue of Proposition 5.3 for refl-spanners. Let us first informally explain our approach. 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 ${}^{\mathsf{y}_\mathsf{x}}\triangleright\, \mathsf{x} \triangleleft^{\mathsf{y}_\mathsf{x}}$, where 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.

In the following, we assume that the set of variables is partitioned into the following sets. There is a set $\mathcal{X}_r$ of *reference-variables*, there is a set $\mathcal{X}_e$ of *extraction-variables*, and, for each reference-variable $\mathsf{x} \in \mathcal{X}_r$, there is a set $\mathcal{X}_{e,\mathsf{x}}$ of *extraction-variables for reference-variable* x. The intuitive idea is that for every $\mathsf{x} \in \mathcal{X}_r$, every reference of x is extracted by some $\mathsf{y} \in \mathcal{X}_{e,\mathsf{x}}$, i. e., it occurs directly between ${}^{\mathsf{y}}\triangleright$ and $\triangleleft^{\mathsf{y}}$; 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 from $\Sigma \cup \mathcal{X}$ which is x. The variables $\mathcal{X}_e$ are also only used for extraction (i. e., they are never referenced), but their respective markers can parenthesise any kind of factor. 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.

Definition 5.9. A ref-word w over Σ and $\mathcal{X}$ is a *strongly reference extracting* ref-word over Σ and $(\mathcal{X}_r, \mathcal{X}_e, \{\mathcal{X}_{e,\mathsf{x}} \mid \mathsf{x} \in \mathcal{X}_r\})$, if it satisfies the following:

- $\mathcal{X}_r, \mathcal{X}_e, \{\mathcal{X}_{e,\mathsf{x}} \mid \mathsf{x} \in \mathcal{X}_r\}$ is a partition of $\mathcal{X}$.
- For every $\mathsf{x} \in \mathcal{X}_r$, if $w[i, i+1\rangle = \mathsf{x}$,
  - then $w[i-1, i+2\rangle = {}^{\mathsf{y}}\triangleright\, \mathsf{x} \triangleleft^{\mathsf{y}}$ with $\mathsf{y} \in \mathcal{X}_{e,\mathsf{x}}$, and
  - $\mathsf{st}(\mathfrak{d}(w))(\mathsf{x}) = [i', j'\rangle$ with $j' - i' \geqslant 1$.
- For every $\mathsf{y} \in \mathcal{X}_e \cup \bigcup_{\mathsf{x} \in \mathcal{X}_r} \mathcal{X}_{e,\mathsf{x}}$ we have that $|w|_{\mathsf{y}} = 0$.
- For every $\mathsf{y} \in \mathcal{X}_{e,\mathsf{x}}$, if $w[i, i+1\rangle = {}^{\mathsf{y}}\triangleright$ then $w[i, i+3\rangle = {}^{\mathsf{y}}\triangleright\, \mathsf{x} \triangleleft^{\mathsf{y}}$.

A ref-language L over Σ and $\mathcal{X}$ is a *strongly reference extracting* ref-language over Σ and $(\mathcal{X}_r, \mathcal{X}_e, \{\mathcal{X}_{e,\mathsf{x}} \mid \mathsf{x} \in \mathcal{X}_r\})$ if every ref-word $w \in L$ is a strongly reference extracting ref-word over Σ and $(\mathcal{X}_r, \mathcal{X}_e, \{\mathcal{X}_{e,\mathsf{x}} \mid \mathsf{x} \in \mathcal{X}_r\})$.

Observation 5.10. Let w be a strongly reference extracting ref-word over Σ and $(\mathcal{X}_r, \mathcal{X}_e, \{\mathcal{X}_{e,\mathsf{x}} \mid \mathsf{x} \in \mathcal{X}_r\})$.

(1) For every $\mathsf{x} \in \mathcal{X}_r$, every occurrence of x in $\mathsf{msrep}(w)$ is directly preceded by (and followed by) a symbol $\Gamma \subseteq \Gamma_{\mathcal{X}}$ such that Γ contains some ${}^{\mathsf{y}}\triangleright$ (some $\triangleleft^{\mathsf{y}}$, respectively) with $\mathsf{y} \in \mathcal{X}_{e,\mathsf{x}}$ and no other ${}^{\mathsf{y}'}\triangleright$ ($\triangleleft^{\mathsf{y}'}$, respectively) with $\mathsf{y}' \in \bigcup_{\mathsf{x}' \in \mathcal{X}_r} \mathcal{X}_{e,\mathsf{x}'}$ and $\mathsf{y} \neq \mathsf{y}'$.
(2) Every occurrence of a symbol $\Gamma \subseteq \Gamma_{\mathcal{X}}$ with ${}^{\mathsf{y}}\triangleright \in \Gamma$ (or $\triangleleft^{\mathsf{y}} \in \Gamma$) for some $\mathsf{y} \in \mathcal{X}_{e,\mathsf{x}}$, is followed by (preceded by, respectively) a sequence of occurrences of Γ followed by (preceded by) a reference x.

Lemma 5.11. *Let w_1 and w_2 be strongly reference extracting ref-words over Σ and $(\mathcal{X}_r, \mathcal{X}_e, \{\mathcal{X}_{e,\mathsf{x}} \mid \mathsf{x} \in \mathcal{X}_r\})$. Then $\mathfrak{e}(\mathfrak{d}(w_1)) = \mathfrak{e}(\mathfrak{d}(w_2))$ and $\mathsf{st}(\mathfrak{d}(w_1)) = \mathsf{st}(\mathfrak{d}(w_2))$ if and only if $\mathsf{msrep}(w_1) = \mathsf{msrep}(w_2)$.*

Proof. If $\mathsf{msrep}(w_1) = \mathsf{msrep}(w_2)$, then we obviously also have $\mathsf{msrep}(\mathfrak{d}(w_1)) = \mathsf{msrep}(\mathfrak{d}(w_2))$. Since both $\mathfrak{d}(w_1)$ and $\mathfrak{d}(w_2)$ are subword-marked words, we can use Proposition 5.2 to conclude that $\mathfrak{e}(\mathfrak{d}(w_1)) = \mathfrak{e}(\mathfrak{d}(w_2))$ and $\mathsf{st}(\mathfrak{d}(w_1)) = \mathsf{st}(\mathfrak{d}(w_1))$.

Next, we assume that $\mathfrak{e}(\mathfrak{d}(w_1)) = \mathfrak{e}(\mathfrak{d}(w_2))$ and $\mathsf{st}(\mathfrak{d}(w_1)) = \mathsf{st}(\mathfrak{d}(w_2))$. Since both $\mathfrak{d}(w_1)$ and $\mathfrak{d}(w_2)$ are subword-marked words, we can conclude with Proposition 5.2 that $\mathsf{msrep}(\mathfrak{d}(w_1)) = \mathsf{msrep}(\mathfrak{d}(w_2))$. In order to conclude the proof, we have to show that $\mathsf{msrep}(w_1) = \mathsf{msrep}(w_2)$. To this end, we show that the assumption $\mathsf{msrep}(w_1) \neq \mathsf{msrep}(w_2)$ leads to a contradiction.

We prove by induction that, for every $i \in \{0, 1, \ldots, \max\{|\mathsf{msrep}(w_1)|, |\mathsf{msrep}(w_2)|\}\}$, $\mathsf{msrep}(w_1)[1, i+1\rangle = \mathsf{msrep}(w_2)[1, i+1\rangle$. As the base of the induction, we observe that

$\mathsf{msrep}(w_1)[1,0+1\rangle = \mathsf{msrep}(w_1)[1,1\rangle = \varepsilon$ and $\mathsf{msrep}(w_2)[1,0+1\rangle = \varepsilon$. Next, we assume that, for some $i \in \{0,1,\ldots,\max\{|\mathsf{msrep}(w_1)|,|\mathsf{msrep}(w_2)|\}-1\}$, we have that $\mathsf{msrep}(w_1)[1,i+1\rangle = \mathsf{msrep}(w_2)[1,i+1\rangle$. We let $\sigma_1 = \mathsf{msrep}(w_1)[i+1,i+2\rangle$ and $\sigma_2 = \mathsf{msrep}(w_2)[i+1,i+2\rangle$. We make a case distinction with respect to σ_1 (note that by symmetry this will cover all possible cases).

Case $\sigma_1 \subseteq \Gamma_{\mathcal{X}}$: In this case, $\sigma_1 \neq \sigma_2$ would directly imply that for some $\mathsf{x} \in \mathcal{X}$, we have that $\mathsf{st}(\mathfrak{d}(w_1))(\mathsf{x}) \neq \mathsf{st}(\mathfrak{d}(w_2))(\mathsf{x})$ (note that this holds for all possible choices of σ_2). Hence, we would get the contradiction that $\mathsf{st}(\mathfrak{d}(w_1)) \neq \mathsf{st}(\mathfrak{d}(w_2))$, which implies that we have $\sigma_1 = \sigma_2$.

Case $\sigma_1 \in \Sigma$ **and** $\sigma_2 \in \Sigma$: This directly implies that there is some position j such that $\mathsf{msrep}(\mathfrak{d}(w_1))[j,j+1\rangle = \sigma_1$ and $\mathsf{msrep}(\mathfrak{d}(w_2))[j,j+1\rangle = \sigma_2$. Thus, since $\mathsf{msrep}(\mathfrak{d}(w_1)) = \mathsf{msrep}(\mathfrak{d}(w_2))$, we can directly conclude that $\sigma_1 = \sigma_2$.

Case $\sigma_1 = \mathsf{x} \in \mathcal{X}_r$: If $\sigma_2 \subseteq \Gamma_{\mathcal{X}}$, then we can proceed like in the first case, but with σ_2 playing the role of σ_1. Hence, we may assume that $\sigma_2 \in \mathcal{X} \cup \Sigma$. By Point 1 of Observation 5.10, we can conclude that $\mathsf{msrep}(w_1)[i,i+1\rangle = \Gamma \subseteq \Gamma_{\mathcal{X}}$ such that ${}^{\mathsf{y}}\triangleright \in \Gamma$ for some $\mathsf{y} \in \mathcal{X}_{e,\mathsf{x}}$. By assumption, we therefore also have $\mathsf{msrep}(w_2)[i,i+1\rangle = \Gamma$. Hence, Point 2 of Observation 5.10 implies that symbol Γ at position i of $\mathsf{msrep}(w_2)$ is followed by a sequence of occurrences of Γ followed by x. Since we made the assumption that $\sigma_2 \in \mathcal{X} \cup \Sigma$, we know that $\sigma_2 \neq \Gamma$, which means that $\sigma_2 = \mathsf{x}$. Consequently, $\sigma_1 = \sigma_2$.

We have shown that $\mathsf{msrep}(w_1)[1,i+1\rangle = \mathsf{msrep}(w_2)[1,i+1\rangle$ holds for every $i \in \{0,1,\ldots,\max\{|\mathsf{msrep}(w_1)|,|\mathsf{msrep}(w_2)|\}\}$. This means that either $\mathsf{msrep}(w_1) = \mathsf{msrep}(w_2)$, or one of these words is a proper prefix of the other. Let us assume that the latter case applies and, without loss of generality, that $\mathsf{msrep}(w_1)$ is a proper prefix of $\mathsf{msrep}(w_2)$. More formally, for some $j \in \{|\mathsf{msrep}(w_1)|,|\mathsf{msrep}(w_1)|+1,\ldots,|\mathsf{msrep}(w_2)|\}$, we have that $\mathsf{msrep}(w_2)[1,j+1\rangle = \mathsf{msrep}(w_1)$, and $|\mathsf{msrep}(w_2)| > |\mathsf{msrep}(w_1)|$. If $\mathsf{msrep}(w_2)[j+1,j+2\rangle \in \Sigma$ or $\mathsf{msrep}(w_2)[j+1,j+2\rangle \in \mathcal{X}$, then $|\mathfrak{d}(w_2)| > |\mathfrak{d}(w_1)|$, which leads to the contradiction that $\mathsf{msrep}(\mathfrak{d}(w_1)) \neq \mathsf{msrep}(\mathfrak{d}(w_2))$ (note that in the latter case, i. e., $\mathsf{msrep}(w_2)[j+1,j+2\rangle \in \mathcal{X}$, it is important that for strongly reference extracting ref-words we require for every $\mathsf{x} \in \mathcal{X}_r$ with $|w|_{\mathsf{x}} \neq 0$ that $\mathsf{st}(\mathfrak{d}(w))(\mathsf{x}) = [i,j\rangle$ with $j - i \geqslant 2$). Moreover, if $\mathsf{msrep}(w_2)[j+1,j+2\rangle = \Gamma \subseteq \Gamma_{\mathcal{X}}$, then, for some $\mathsf{x} \in \mathcal{X}$, we have that $\mathsf{st}(\mathfrak{d}(w_1))(\mathsf{x}) \neq \mathsf{st}(\mathfrak{d}(w_2))(\mathsf{x})$. This leads to the contradiction that $\mathsf{st}(\mathfrak{d}(w_1)) \neq \mathsf{st}(\mathfrak{d}(w_2))$. Consequently, it is not possible that one of $\mathsf{msrep}(w_1)$ or $\mathsf{msrep}(w_2)$ is a proper prefix of the other; thus, $\mathsf{msrep}(w_1) = \mathsf{msrep}(w_2)$. This concludes the proof of Lemma 5.11. □

Lemma 5.12. *Let L_1, L_2 be strongly reference extracting ref-languages over Σ and $(\mathcal{X}_r, \mathcal{X}_e, \{\mathcal{X}_{e,\mathsf{x}} \mid \mathsf{x} \in \mathcal{X}_r\})$. Then $[\![L_1]\!] \subseteq [\![L_2]\!]$ if and only if $\mathsf{msrep}(L_1) \subseteq \mathsf{msrep}(L_2)$.*

Proof. Let us first assume that $[\![L_1]\!] \subseteq [\![L_2]\!]$. Let $v_1 \in \mathsf{msrep}(L_1)$ and let $v_1' \in L_1$ be such that $\mathsf{msrep}(v_1') = v_1$. Moreover, let $w = \mathfrak{e}(\mathfrak{d}(v_1'))$, which also means that $\mathsf{st}(\mathfrak{d}(v_1')) \in [\![L_1]\!](w)$. Since $[\![L_1]\!] \subseteq [\![L_2]\!]$, we have $[\![L_1]\!](w) \subseteq [\![L_2]\!](w)$ and therefore $\mathsf{st}(\mathfrak{d}(v_1')) \in [\![L_2]\!](w)$. By definition, this means that there is some $v_2' \in L_2$ with $\mathfrak{e}(\mathfrak{d}(v_2')) = w = \mathfrak{e}(\mathfrak{d}(v_1'))$ and $\mathsf{st}(\mathfrak{d}(v_2')) = \mathsf{st}(\mathfrak{d}(v_1'))$. By Proposition 5.11, this means that $\mathsf{msrep}(v_1') = \mathsf{msrep}(v_2')$, and, by definition, $\mathsf{msrep}(v_2') \in \mathsf{msrep}(L_2)$. Thus, $\mathsf{msrep}(v_1') = v_1 \in \mathsf{msrep}(L_2)$. Since $v_1 \in \mathsf{msrep}(L_1)$ has been chosen arbitrarily, we can conclude that $\mathsf{msrep}(L_1) \subseteq \mathsf{msrep}(L_2)$.

Next, we assume that $\mathsf{msrep}(L_1) \subseteq \mathsf{msrep}(L_2)$. Let $w \in \Sigma^*$ and $t \in [\![L_1]\!](w)$ be arbitrarily chosen. This means that there is some $v_1 \in L_1$ with $\mathfrak{e}(\mathfrak{d}(v_1)) = w$ and $\mathsf{st}(\mathfrak{d}(v_1)) = t$. By assumption, $\mathsf{msrep}(v_1) \in \mathsf{msrep}(L_2)$, which means that there is some $v_2 \in L_2$ with

$\mathsf{msrep}(v_1) = \mathsf{msrep}(v_2)$, which, by Proposition 5.11, means that $\mathfrak{e}(\mathfrak{d}(v_1)) = \mathfrak{e}(\mathfrak{d}(v_2)) = w$ and $\mathsf{st}(\mathfrak{d}(v_1)) = \mathsf{st}(\mathfrak{d}(v_2)) = t$. Thus, $t \in [\![L_2]\!](w)$. Since $w \in \Sigma^*$ and $t \in [\![L_1]\!](w)$ have been chosen arbitrarily, we can conclude that $[\![L_1]\!] \subseteq [\![L_2]\!]$. □

Theorem 5.13. Containment *and* Equivalence *for strongly reference extracting refl-spanners over* Σ *and* $(\mathcal{X}_r, \mathcal{X}_e, \{\mathcal{X}_{e,\mathsf{x}} \mid \mathsf{x} \in \mathcal{X}_r\})$ *are* PSpace*-complete.*

Proof. We first note that hardness follows from the fact that the inclusion and equivalence problem for NFAs is PSpace-hard (see [MS72, SM73]). Next, we prove the upper bound for the case of Containment (which also implies the upper bound for Equivalence). Note that the general proof idea is very similar to [MRV17, Thm. 6.4], i. e., the PSpace-completeness of regular spanners with schemaless semantics.

Let L_1, L_2 be strongly reference extracting ref-languages over Σ and $(\mathcal{X}_r, \mathcal{X}_e, \{\mathcal{X}_{e,\mathsf{x}} \mid \mathsf{x} \in \mathcal{X}_r\})$, represented by NFA M_1 and M_2. We wish to decide whether $[\![L_1]\!] \subseteq [\![L_2]\!]$. By Lemma 5.12, this can be done by checking whether $\mathsf{msrep}(L_1) \subseteq \mathsf{msrep}(L_2)$. We shall now devise a nondeterministic algorithm that checks whether $\mathsf{msrep}(L_1) \not\subseteq \mathsf{msrep}(L_2)$. Note that we cannot modify automata for L_1 and L_2 in polynomial space such that they accept the languages $\mathsf{msrep}(L_1)$ and $\mathsf{msrep}(L_2)$, respectively.

Without loss of generality, we assume that M_1 and M_2 are complete. The algorithm guesses words w_1, w_2 over $\Sigma \cup \Gamma_{\mathcal{X}} \cup \mathcal{X}$ letter by letter, which satisfy $\mathsf{msrep}(w_1) = \mathsf{msrep}(w_2)$, and checks whether w_1 is accepted by M_1 and rejected by M_2. To this end, we run the automata in parallel and, since the automata are nondeterministic, we have to explore all possible paths labelled with w_1 and w_2, which can be done by maintaining the sets S_1 and S_2 of current active states of M_1 and M_2, respectively. More precisely, we guess the next symbols σ_1 and σ_2 of w_1 and w_2, and then we update the sets of active states by applying the σ_1- and σ_2-transitions for the active states of M_1 and M_2, respectively. Obviously, in order to satisfy $\mathsf{msrep}(w_1) = \mathsf{msrep}(w_2)$, we must have $\sigma_1 = \sigma_2$ whenever $\sigma_1 \in \Sigma \cup \mathcal{X}$. If, however, in w_1 we guess that instead of a symbol from Σ or $\mathcal{X}$ a sequence over $\Gamma_{\mathcal{X}}$ follows, then we must process with M_1 and M_2 all possible permutations of this sequence. This can be done as follows. If we guess the next σ_1 to be neither from Σ or from $\mathcal{X}$, we guess a set $\Gamma \subseteq \Gamma_{\mathcal{X}}$ instead. Then we enumerate all linear orders $\vec{\Gamma}$ of the symbols from Γ, and for each such $\vec{\Gamma}$, we update the sets of active states by exploring all $\vec{\Gamma}$-labelled paths in M_1 and M_2 that start in states of S_1 and S_2, respectively. Moreover, after such a step, the algorithm must next again guess some σ_1 from Σ or $\mathcal{X}$.

We start this procedure with $S_1 = \{s_{0,1}\}$ and $S_2 = \{s_{0,2}\}$, where $s_{0,1}$ and $s_{0,2}$ are the initial states of M_1 and M_2, respectively. The algorithm terminates with output yes as soon as S_1 contains an accepting state and S_2 does not contain an accepting state. Note that if S_1 contains an accepting state, then this also means that the currently read input w_1 is a valid ref-word, which, since $\mathsf{msrep}(w_1) = \mathsf{msrep}(w_2)$, also means that w_2 is a valid ref-word.

We shall now consider the correctness of the algorithm. The basic observation is that, for some (nondeterministically chosen) word over $w \in \Sigma \cup \Gamma_{\mathcal{X}} \cup \mathcal{X}$, the algorithm simulates M_1 and M_2 in parallel on *all* inputs w' that satisfy $\mathsf{msrep}(w') = \mathsf{msrep}(w)$. If $\mathsf{msrep}(L_1) \not\subseteq \mathsf{msrep}(L_2)$, then there is some $w \in \mathcal{L}(M_1)$ such that for every w' over $\Sigma \cup \Gamma_{\mathcal{X}} \cup \mathcal{X}$ with $\mathsf{msrep}(w) = \mathsf{msrep}(w')$, we have that $w' \notin \mathcal{L}(M_2)$. Consequently, if the algorithm guesses w (this means it guesses w's symbols from $\Sigma \cup \mathcal{X}$ in the right order, and whenever some sequence γ over $\Gamma_{\mathcal{X}}$ follows, it guesses the set Γ that contains exactly the symbols from γ), then, after having completely read w, we will reach a set of active states S_1 with an accepting state, while, due to $w' \notin \mathcal{L}(M_2)$ for all w' with $\mathsf{msrep}(w) = \mathsf{msrep}(w')$, the set S_2 of active

states cannot contain an accepting state. This means that the algorithm produces output yes. On the other hand, if the algorithm produces output yes, then it has guessed some w such that, once it its completely consumed, S_1 contains an accepting state while S_2 does not contain any accepting state. Consequently, $\mathcal{L}(M_1)$ accepts some w' with $\mathsf{msrep}(w') = \mathsf{msrep}(w)$, while there is no $w'' \in \mathcal{L}(M_2)$ with $\mathsf{msrep}(w'') = \mathsf{msrep}(w)$. This directly implies that $\mathsf{msrep}(w) \in \mathsf{msrep}(L_1)$ and $\mathsf{msrep}(w) \notin \mathsf{msrep}(L_2)$; thus, $\mathsf{msrep}(L_1) \not\subseteq \mathsf{msrep}(L_2)$.

With respect to the complexity of the algorithm, we observe that we only store sets S_1 and S_2 that are subsets of the states of M_1 and M_2, respectively. Furthermore, updating these steps can be easily done in polynomial space by consulting the transition functions of M_1 and M_2. In particular, we observe that guessing a set $\Gamma \subseteq \Gamma_{\mathcal{X}}$ and enumerating all linear orders $\vec{\Gamma}$ of the symbols from Γ can also be done in polynomial space. □

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.

6.1. From Refl-Spanners to Core Spanners.

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 easy to see.

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

Proof. Let S be a refl-spanner over Σ and $\mathcal{X}$ represented by an NFA M, and let $k \in \mathbb{N}$ be such that $|w|_{\mathsf{x}} \leqslant k$ for every $\mathsf{x} \in \mathcal{X}$ and every $w \in \mathcal{L}(M)$. For every $\mathsf{x} \in \mathcal{X}$, let $\mathcal{Y}_{\mathsf{x}} = \{\mathsf{y}_{\mathsf{x},1}, \mathsf{y}_{\mathsf{x},2}, \ldots, \mathsf{y}_{\mathsf{x},k}\}$ be k fresh variables. We define an NFA M' as follows. The automaton M' simulates M, but for every $\mathsf{x} \in \mathcal{X}$, it maintains a counter c_{x} in its finite state control that is initially 0 and that can hold values from $\{0\} \cup [k]$. Whenever M takes an x-transition, M' increments c_{x} and then can read any word from $\mathcal{L}(\,{}^{\mathsf{y}_{\mathsf{x},c_{\mathsf{x}}}\triangleright}\,\Sigma^*\,\triangleleft^{\mathsf{y}_{\mathsf{x},c_{\mathsf{x}}}})$. It can be easily seen that $\mathcal{L}(M')$ is a subword-marked language over Σ and $\mathcal{X}$. Furthermore, it can be shown with moderate effort that $[\![\mathcal{L}(M)]\!] = \pi_{\mathcal{X}} \varsigma^{=}_{\{\mathcal{Y}_{\mathsf{x}} \mid \mathsf{x} \in \mathcal{X}\}}([\![\mathcal{L}(M')]\!])$. □

It is interesting to note that the refl-spanner $[\![\mathcal{L}(\mathsf{a}^+ \,{}^{\mathsf{x}}\!\triangleright\, \mathsf{b}^+ \triangleleft^{\mathsf{x}} (\mathsf{a}^+\mathsf{x})^*\mathsf{a}^+)]\!]$, which is *not* reference-bounded, is provably not a core spanner (see [FKRV15, Theorem 6.1])).

6.2. From Core Spanners to Refl-Spanners.

The question of 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 $L = \mathcal{L}({}^{\mathsf{x}}\!\triangleright(\mathsf{a}^* \vee \mathsf{b}^*)\triangleleft^{\mathsf{x}} \mathsf{c}\, {}^{\mathsf{y}}\!\triangleright(\mathsf{a}^* \vee \mathsf{b}^*)\triangleleft^{\mathsf{y}})$ can be represented as $[\![L']\!]$ where $L' = \mathcal{L}({}^{\mathsf{x}}\!\triangleright(\mathsf{a}^* \vee \mathsf{b}^*)\triangleleft^{\mathsf{x}}\mathsf{c}\mathsf{x})$. However, if we change L to $\mathcal{L}({}^{\mathsf{x}}\!\triangleright\Sigma^*\mathsf{a}\Sigma^*\triangleleft^{\mathsf{x}}\mathsf{c}\,{}^{\mathsf{y}}\!\triangleright\Sigma^*\mathsf{b}\Sigma^*\triangleleft^{\mathsf{y}})$, then neither of the ref-languages $\mathcal{L}({}^{\mathsf{x}}\!\triangleright\,\Sigma^*\mathsf{a}\Sigma^*\triangleleft^{\mathsf{x}}\,\mathsf{c}\mathsf{x})$ nor $\mathcal{L}({}^{\mathsf{x}}\!\triangleright\,\Sigma^*\mathsf{b}\Sigma^*\triangleleft^{\mathsf{x}}\,\mathsf{c}\mathsf{x})$ yield an equivalent refl-spanner and we have to use $\mathcal{L}({}^{\mathsf{x}}\!\triangleright\, r \triangleleft^{\mathsf{x}}\,\mathsf{c}\mathsf{x})$, where r is a regular expression for $\mathcal{L}(\Sigma^*\mathsf{a}\Sigma^*) \cap \mathcal{L}(\Sigma^*\mathsf{b}\Sigma^*)$.

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 $\varsigma^{=}_{\{\mathsf{x},\mathsf{y}\}}[\![\mathcal{L}({}^{\mathsf{x}}\!\triangleright\,\mathsf{a}^*\triangleleft^{\mathsf{x}}\,{}^{\mathsf{y}}\!\triangleright\,{}^{\mathsf{z}}\!\triangleright\,\mathsf{a}^*\triangleleft^{\mathsf{z}}\mathsf{a}^*\triangleleft^{\mathsf{y}})]\!]$ 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 $\varsigma^{=}_{\{\mathsf{x},\mathsf{y}\}}(\llbracket\mathcal{L}(\,{}^{\mathsf{x}}\triangleright \ldots\, {}^{\mathsf{y}}\triangleright \ldots \triangleleft^{\mathsf{x}} \ldots \triangleleft^{\mathsf{y}})\rrbracket)$. For an in-depth analysis of the capability of core spanners to describe word-combinatorial properties, we refer to [FH18, Fre19].

These considerations suggest that the refl-spanner formalism is 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 in the next subsection) that just combines several variables (or columns in the spanner result) into one variable (or column) in a natural way, and a projection; see Theorem 6.11 for our respective main result.

In this section, we shall also use the marker-set representation of subword-marked words (and therefore ref-words) as defined at the beginning of Section 5. However, in the following it will be more convenient to simplify this concept as follows. Let w be a subword-marked word over Σ and $\mathcal{X}$, let $t = \mathsf{st}(w)$, and let $\mathfrak{e}(w) = w_1 w_2 \ldots w_n$ with $w_i \in \Sigma$ for every $i \in [n]$. For each $i \in \{1, \ldots, n{+}1\}$ let Γ_i again be the set comprising of all symbols ${}^{\mathsf{x}}\triangleright$ where $\mathsf{x} \in \mathrm{dom}(t)$ and $t(\mathsf{x}) = [i, j\rangle$ for some j, and all symbols $\triangleleft^{\mathsf{x}}$ where $\mathsf{x} \in \mathrm{dom}(t)$ and $t(\mathsf{x}) = [j, i\rangle$ for some j. We now set $\mathsf{msrep}(w) = \Gamma'_1 w_1 \Gamma'_2 w_2 \ldots \Gamma'_n w_n \Gamma'_{n+1}$, where, for every $i \in \{1, 2, \ldots, n+1\}$, $\Gamma'_i = \Gamma_i$ if $\Gamma_i \neq \emptyset$, and $\Gamma'_i = \varepsilon$ otherwise. Note that this corresponds to our original version of the marker-set representation with the only difference that the factors $\Gamma_i^{|\Gamma_i|}$ are replaced by just one occurrence of the symbol Γ_i, or completely erased if $\Gamma_i = \emptyset$. Moreover, note that we have re-defined $\mathsf{msrep}(\cdot)$ and that from now on, we will only use this definition of the marker-set representation.

For example, if $w = {}^{\mathsf{x}_1}\triangleright\ {}^{\mathsf{x}_3}\triangleright\ \mathtt{ab}\ \triangleleft^{\mathsf{x}_1}\ {}^{\mathsf{x}_2}\triangleright\ \mathtt{cb}\ \triangleleft^{\mathsf{x}_2}\ \mathtt{abca}\triangleleft^{\mathsf{x}_3}$, then we have $\mathsf{msrep}(w) = \{{}^{\mathsf{x}_1}\triangleright, {}^{\mathsf{x}_3}\triangleright\}\mathtt{ab}\{\triangleleft^{\mathsf{x}_1}, {}^{\mathsf{x}_2}\triangleright\}\mathtt{cb}\{\triangleleft^{\mathsf{x}_2}\}\mathtt{abca}\{\triangleleft^{\mathsf{x}_3}\}$.

We again extend the function $\mathsf{msrep}(\cdot)$ in the natural way to a subword-marked language L by setting $\mathsf{msrep}(L) = \{\mathsf{msrep}(w) \mid w \in L\}$.

Remark 6.2. In this section, we assume that all our regular and refl-spanners are represented as (automata that accept) subword-marked languages and ref-languages, respectively, all the words of which are given in marker-set representation. It can be easily seen that any NFA accepting a subword-marked language or a ref-language can be transformed into an NFA that accepts the $\mathsf{msrep}(\cdot)$ image of that language. This transformation may cause an exponential blow-up, which is no problem, since we are here only concerned with questions of expressive power, and not with complexity issues.

We next give an intuitive explanation of the result to be proven in this subsection. As the central question of this subsection, we investigate which core spanners of the form $S = \varsigma^{=}_{E}(\llbracket L\rrbracket)$ (where L is a regular subword-marked language) can be described by $\llbracket L'\rrbracket$ for some regular ref-language L'. Our first respective observation is that S can in fact be described in the form $\llbracket L'\rrbracket$ for a regular ref-language L', provided that all the variables that are subject to the string-equality selection $\varsigma^{=}_{E}$ are *simple* (with respect to L). A variable x is called simple with respect to a subword-marked language L, if any definition for x (in any $w \in L$) is always of the form $\Gamma_1 u \Gamma_2$ with $\Gamma_1, \Gamma_2 \subseteq \Gamma_{\mathcal{X}}$, ${}^{\mathsf{x}}\triangleright \in \Gamma_1$, $\triangleleft^{\mathsf{x}} \in \Gamma_2$ and $u \in \Sigma^+$ (or it defines the empty string, i. e., there is an occurrence of $\Gamma_1 \subseteq \Gamma_{\mathcal{X}}$ with $\{{}^{\mathsf{x}}\triangleright, \triangleleft^{\mathsf{x}}\} \subseteq \Gamma_1$).

This result – i. e., that a core spanner $\varsigma^{=}_{E}(\llbracket L\rrbracket)$ is a refl-spanner, if all the variables that are subject to the string-equality selection are simple – is in fact a generalisation of the

following example already mentioned above (note that both x and y are simple):

$$\varsigma^{=}_{\{\mathsf{x},\mathsf{y}\}}[\![\mathcal{L}(\,{}^{\mathsf{x}}\triangleright\,(\mathtt{a}^*\vee\mathtt{b}^*)\,\triangleleft^{\mathsf{x}}\,\mathtt{c}\,{}^{\mathsf{y}}\triangleright\,(\mathtt{a}^*\vee\mathtt{b}^*)\triangleleft^{\mathsf{y}})]\!] \;=\; [\![\mathcal{L}(\,{}^{\mathsf{x}}\triangleright\,(\mathtt{a}^*\vee\mathtt{b}^*)\,\triangleleft^{\mathsf{x}}\,\mathtt{c}\,{}^{\mathsf{y}}\triangleright\,\mathsf{x}\triangleleft^{\mathsf{y}})]\!]\,.$$

Next, we show that we can use this result in order to prove that a much larger class of core spanners can also be described by ref-languages, if we also allow an additional operation on spanners, which we call the *span-fusion* $\biguplus$. Intuitively, for a set $\lambda=\{\mathsf{y}_1,\mathsf{y}_2,\ldots,\mathsf{y}_k\}\subseteq\mathcal{X}$ of variables, and a completely new variable x with $\mathsf{x}\notin\mathcal{X}$, the span-fusion $\biguplus_{\lambda\to\mathsf{x}}$ replaces in a span-relation the set of columns $\{\mathsf{y}_1,\mathsf{y}_2,\ldots,\mathsf{y}_k\}$ by a single new column x that, for each row t (i. e., span-tuple t), contains a single span that corresponds to the union of all the spans $t(\mathsf{y}_1),t(\mathsf{y}_2),\ldots,t(\mathsf{y}_k)$. For example, $\biguplus_{\{\mathsf{x}_1,\mathsf{x}_2\}\to\mathsf{z}}$ turns a $\{\mathsf{x}_1,\mathsf{x}_2,\mathsf{x}_3\}$-tuple $t=([3,17\rangle,[17,23\rangle,[4,20\rangle)$ into a $\{\mathsf{z},\mathsf{x}_3\}$-tuple t' with $t'(\mathsf{z})=[3,23\rangle$ and $t'(\mathsf{x}_3)=[4,20\rangle)$. The span-fusion operation will be formally defined below in Section 6.2.2.

We can show that provided that $\varsigma^{=}_{E}$ satisfies a certain *non-overlapping* property (basically, we require that any two variables that are subject to the string-equality selection are non-overlapping with respect to L), we can represent $\varsigma^{=}_{E}([\![L]\!])$ in the form $\biguplus_{\lambda_1\to\mathsf{x}_1}\biguplus_{\lambda_2\to\mathsf{x}_2}\cdots\biguplus_{\lambda_k\to\mathsf{x}_k}([\![L']\!])$ for a regular ref-language L' over Σ and a set $\mathcal{X}'$ of variables with $|\mathcal{X}'|=\mathrm{O}(|\mathcal{X}|^3)$.

The proof idea for this result is as follows (formal details are given later on). Every regular subword-marked language L over Σ and $\mathcal{X}$ can be transformed into a subword-marked language L' over Σ and $\mathcal{X}'$ such that, for every $w\in L$, there is a $w'\in L'$ with $\mathfrak{e}(w)=\mathfrak{e}(w')$ and $\mathsf{st}(w')$ is a *split* of $\mathsf{st}(w)$. An $\mathcal{X}'$-tuple t' is a split of an $\mathcal{X}$-tuple t, if every span $t(\mathsf{x})=[\ell_{\mathsf{x}},r_{\mathsf{x}}\rangle$ is factorised into $\wp$ factors with respect to t', i. e., there are variables $\mathsf{x}^1,\mathsf{x}^2,\ldots,\mathsf{x}^{\wp}\in\mathcal{X}'$ such that $t'(\mathsf{x}^1)=[\ell_{\mathsf{x}},k_1\rangle,t'(\mathsf{x}^2)=[k_1,k_2\rangle,\ldots,t'(\mathsf{x}^{\wp-1})=[k_{\wp-1},k_{\wp}\rangle,t'(\mathsf{x}^{\wp})=[k_{\wp},r_{\mathsf{x}}\rangle$ (the number $\wp$ is explained later and satisfies $\wp=\mathrm{O}(|\mathcal{X}|^2)$). Moreover, we require these factorisations to be such that the spans for any two variables are non-overlapping. The subword-marked language L' of all these splits of subword-marked words from L will be called the *split* of L, and it can be shown that if L is regular, then so is its split L'. In particular, by definition of the span-fusion, we immediately have that $\biguplus_{\Lambda}[\![L']\!]=[\![L]\!]$, where $\Lambda=\{\lambda_{\mathsf{x}}\to\mathsf{x}\mid\mathsf{x}\in\mathcal{X}\}$ with $\lambda_{\mathsf{x}}=\{\mathsf{x}^1,\mathsf{x}^2,\ldots,\mathsf{x}^{\wp}\}$ for every $\mathsf{x}\in\mathcal{X}$. The split-operation will be formally defined below in Section 6.2.3.

The next idea is that we can simulate the string-equality selection $\varsigma^{=}_{E}$ (for L) with $E=\{\mathcal{Z}_1,\mathcal{Z}_2,\ldots,\mathcal{Z}_k\}$ directly on the split L'. Intuitively speaking, for every $\mathsf{x},\mathsf{y}\in\mathcal{Z}_i$, we require the pairs $\mathsf{x}^j,\mathsf{y}^j$ to be subject to a string-equality selection for every $j\in[\wp]$, or, more formally, we define the string-equality selection $\varsigma^{=}_{E'}$, where $E'=\{\mathcal{Y}_i^j\mid i\in[k],j\in[\wp]\}$ and $\mathcal{Y}_i^j=\{\mathsf{x}^j\mid\mathsf{x}\in\mathcal{Z}_i\}$ for $i\in[k]$ and $j\in[\wp]$. It can be easily seen that if $t\in\biguplus_{\Lambda}\varsigma^{=}_{E'}([\![L']\!])(w)$ for some $w\in\Sigma^*$, then we also have $t\in\varsigma^{=}_{E}([\![L]\!])(w)$. The converse, however, is only true if t has a split t' that is in $\varsigma^{=}_{E'}([\![L']\!])(w)$. A split t' of t is only in $\varsigma^{=}_{E'}([\![L']\!])(w)$ if, for every $i\in[k]$, all $t(\mathsf{x})$ with $\mathsf{x}\in\mathcal{Z}_i$ are factorised into the parts $t'(\mathsf{x}^1),t'(\mathsf{x}^2),\ldots,t'(\mathsf{x}^{\wp})$ in exactly the same way, since otherwise t' would not satisfy $\varsigma^{=}_{E'}$. In the general case, the set $\mathcal{X}'$ of variables is not large enough to make such a split t' possible for every $t\in\varsigma^{=}_{E}([\![L]\!])(w)$. The problem is that due to the property of t' that all the spans for each two variables of $\mathcal{X}'$ are non-overlapping, the number of variables needed for this property may depend on $|w|$ (as will also be demonstrated below by an example). This, however, can only happen if E is *overlapping*, i. e., there are variables $\mathsf{x},\mathsf{y}\in\bigcup_{i\in[k]}\mathcal{Z}_i$ such that the spans of x and y may refer to overlapping spans. Consequently, we have $\biguplus_{\Lambda}\varsigma^{=}_{E'}([\![L']\!])=\varsigma^{=}_{E}([\![L]\!])$, if E is *non-overlapping* (see Lemmas 6.8 and 6.10).

Finally, since the split L' of L has necessarily only simple variables, we can express $\varsigma^{=}_{E'}(\llbracket L' \rrbracket)$ as $\llbracket L'' \rrbracket$ for a regular ref-language L'' (recall that this was our first result, sketched at the beginning of this subsection). Consequently, $\varsigma^{=}_{E}(\llbracket L \rrbracket)$ can be expressed as $\biguplus_{\Lambda}(\llbracket L'' \rrbracket)$, where L'' is a regular ref-language.

In the following subsections, we shall formally carry out this proof roadmap.

6.2.1. *Simple Variables.* As sketched above, a variable is called simple with respect to a subword-marked language if all its definitions do not contain parts of other variable definitions, i. e., symbols $\Gamma \subseteq \mathcal{X}$. We will show next that string-equality selections for simple variables can be expressed by the ref-language mechanism, i. e., by using variable references.

Let us now define simple variables formally.[5] Let L be a subword-marked language over Σ and $\mathcal{X}$, let $w \in L$ and $t = \mathsf{st}(w)$. We say that a variable $\mathsf{x} \in \mathcal{X}$ is *simple* (*with respect to* w) if $t(\mathsf{x})$ is undefined or the empty span, or $w = u_1 \Gamma u_2 \Gamma' u_3$ with ${}^{\mathsf{x}}\triangleright \in \Gamma$ and $\triangleleft^{\mathsf{x}} \in \Gamma'$ implies that $u_2 \in \Sigma^+$. For example, x and z are simple in $\mathtt{ab}\{{}^{\mathtt{x}}\triangleright, {}^{\mathtt{y}}\triangleright\}\mathtt{ab}\{\triangleleft^{\mathtt{x}}\}\mathtt{c}\{\triangleleft^{\mathtt{y}}, {}^{\mathtt{z}}\triangleright\}\mathtt{a}\{\triangleleft^{\mathtt{z}}\}$, whereas y is not simple. We extend this definition to subword-marked languages in the obvious way, i. e., x is simple *with respect to* L if x is simple with respect to every $w \in L$.

Our next goal is to show that for a subword-marked language L over Σ and $\mathcal{X}$, and an $E = \{\mathcal{Z}_1, \mathcal{Z}_2, \ldots, \mathcal{Z}_k\} \subseteq \mathcal{P}(\mathcal{X})$ such that all variables from $\bigcup_{i \in [k]} \mathcal{Z}_i$ are simple with respect to L, we can construct a regular ref-language L' such that $\llbracket L' \rrbracket = \varsigma^{=}_{\mathcal{Z}}(\llbracket L \rrbracket)$; see Theorem 6.3 below. We first sketch the intuitive idea of the construction and then give a formal proof.

For simplicitly, let us assume that $E = \{\mathcal{Z}\}$. Every $w \in L$ may contain some $\mathcal{Z}$-*definitions*, which are definitions that define at least one variable $\mathsf{z} \in \mathcal{Z}$. Since all variables from $\mathcal{Z}$ are simple, such $\mathcal{Z}$-definitions are either *non-empty* and of the form $\Gamma_1 u \Gamma_2$ with $\{\mathsf{z} \in \mathcal{Z} \mid {}^{\mathsf{z}}\triangleright \in \Gamma_1\} = \{\mathsf{z} \in \mathcal{Z} \mid \triangleleft^{\mathsf{z}} \in \Gamma_2\} \neq \emptyset$, and $u \in \Sigma^+$, or they are *empty*, which means that they are represented by a single symbol Γ with $\{{}^{\mathsf{z}}\triangleright, \triangleleft^{\mathsf{z}}\} \subseteq \Gamma$ for at least one $\mathsf{z} \in \mathcal{Z}$. The first step is to remove all subword-marked words that have both non-empty and empty $\mathcal{Z}$-definitions, since they describe span-tuples that are filtered out by the string-equality selection $\varsigma^{=}_{\mathcal{Z}}$. Moreover, this can be easily done by simply storing in the finite state control whether the first $\mathcal{Z}$-definition is empty or not, and then abort any computation that reads both types of $\mathcal{Z}$-definition.

The next task is to handle all other subword-marked words, i. e., those $w \in L$ in which some $\mathcal{Z}$-definitions assign to variables from $\mathcal{Z}$ spans referring to different non-empty strings (which we somehow have to get rid of, since they represent span-tuples filtered out by the string-equality selection), and those that assign the same non-empty string to all variables from $\mathcal{Z}$. These properties cannot be explicitly checked by a finite automaton. The idea of the construction is based on the observation that for all subword-marked words $w \in L$ that assign to every variable from $\mathcal{Z}$ spans referring to the same string, we can as well only keep the very first $\mathcal{Z}$-definition $\Gamma_1 u \Gamma_2$ of w, and replace every further $\mathcal{Z}$-definition $\Gamma'_1 u \Gamma'_2$ in w by $\Gamma'_1 \widehat{\mathsf{z}} \Gamma'_2$, where $\widehat{\mathsf{z}}$ is some arbitrarily chosen variable defined by the first $\mathcal{Z}$-definition. Such a modified w' satisfies $w = \mathfrak{d}(w')$ and therefore also $\mathsf{st}(w) = \mathsf{st}(\mathfrak{d}(w'))$. Consequently, this modification yields a ref-language L' with $\llbracket L' \rrbracket = \varsigma^{=}_{\mathcal{Z}}(\llbracket L \rrbracket)$. Obviously, we have to show that L' is regular. To this end, we can directly modify the NFA M for L as follows. We simulate M up to the point where we read the first $\mathcal{Z}$-definition $\Gamma_1 u \Gamma_2$. However, before reading u, we guess a state for every variable from $\mathcal{Z}$ that might be defined by a later $\mathcal{Z}$-definition.

[5] Recall that by Remark 6.2 we assume that all subword-marked and all ref-languages are given in marker-set representation.

Then, while reading u, we compute some states that are reachable from the guessed states by reading u (since M is nondeterministic, these target states are also implicitly guessed by the specific path that is chosen). The idea is of course that if the currently read input turns out to be of the kind that is not filtered out by $\zeta^{=}_{\mathcal{Z}}$, then it must also be possible to guess exactly the state pairs between which all the further occurrences of u are read in the following $\mathcal{Z}$-definitions of w (after all, these state pairs must be such that u can be read between them, so it is possible to guess them while processing the string u of the first $\mathcal{Z}$-definition). Then, in the rest of the computation, whenever we encounter a $\mathcal{Z}$-definition $\Gamma'_1 u' \Gamma'_2$, we simply verify if we have guessed valid state pairs (i. e., whether the computation of M also ends up in the states guessed for the variables of this $\mathcal{Z}$-definition when we encounter this $\mathcal{Z}$-definition), and then we simply read $\Gamma'_1 \widehat{\mathsf{z}} \Gamma'_2$ instead of $\Gamma'_1 u' \Gamma'_2$. Let us now give a formal proof.

Theorem 6.3. *Let L be a regular subword-marked language over Σ and $\mathcal{X}$, let $E = \{\mathcal{Z}_1, \mathcal{Z}_2, \ldots, \mathcal{Z}_k\} \subseteq \mathcal{P}(\mathcal{X})$ be such that all variables from $\bigcup_{i\in[k]} \mathcal{Z}_i$ are simple with respect to L. There is a regular ref-language L' over Σ and $\mathcal{X}$ with $[\![L']\!] = \zeta^{=}_{E}([\![L]\!])$. Furthermore, L' is reference-bounded.*

Proof. Let $\mathcal{Z} = \bigcup_{i\in[k]} \mathcal{Z}_i$. Since all variables from $\mathcal{Z}$ are simple with respect to L, we know that, for every $w \in \mathcal{L}(M)$ and $i \in [k]$, all definitions of variables from $\mathcal{Z}_i$ are either *non-empty $\mathcal{Z}_i$-definitions* of the form $\Gamma_1 u \Gamma_2$ with $\{\mathsf{z} \in \mathcal{Z} \mid {}^{\mathsf{z}}\triangleright \in \Gamma_1\} = \{\mathsf{z} \in \mathcal{Z} \mid \triangleleft^{\mathsf{z}} \in \Gamma_2\} \neq \emptyset$, and $u \in \Sigma^+$, or they are *empty $\mathcal{Z}_i$-definitions* represented by a single symbol Γ with $\{{}^{\mathsf{z}}\triangleright, \triangleleft^{\mathsf{z}}\} \subseteq \Gamma$ for at least one $\mathsf{z} \in \mathcal{Z}_i$.

We now make two assumptions. Firstly, we assume that $\mathcal{Z}_i \cap \mathcal{Z}_{i'} = \emptyset$ for all $i, i' \in [k]$ with $i \neq i'$. Secondly, we assume that, for every $w \in L$ and for every non-empty $\mathcal{Z}$-definition $\Gamma_1 u \Gamma_2$, there is some $i \in [k]$ such that $\{\mathsf{z} \in \mathcal{Z} \mid {}^{\mathsf{z}}\triangleright \in \Gamma_1\} \subseteq \mathcal{Z}_i$, and, likewise, for every empty $\mathcal{Z}$-definition Γ_1 in w, there is some $i \in [k]$ such that $\{\mathsf{z} \in \mathcal{Z} \mid {}^{\mathsf{z}}\triangleright \in \Gamma_1\} \subseteq \mathcal{Z}_i$. This means that every $\mathcal{Z}_i$-definition cannot be at the same time a $\mathcal{Z}_{i'}$-definition for some $i' \in [k]$ with $i \neq i'$. However, $|\{\mathsf{z} \in \mathcal{Z} \mid {}^{\mathsf{z}}\triangleright \in \Gamma_1\}| > 1$ is possible. These assumptions are obviously not without loss of generality, and we will explain later on how our construction can be adapted to the general case.

Let L_1 be the subset of L containing those $w \in L$ that, for every $i \in [k]$, have either only non-empty $\mathcal{Z}_i$-definitions or only empty $\mathcal{Z}_i$-definitions (i. e., not both types occur in the same subword-marked word). All $w \in L$ that, for some $i \in [k]$, have both empty and non-empty $\mathcal{Z}_i$-definitions, represent span-tuples that are filtered out by the string-equality selection. Thus, we have that $\zeta^{=}_{E}([\![L]\!]) = \zeta^{=}_{E}([\![L_1]\!])$. Moreover, it can be seen that L_1 is a regular subword-marked language. To this end, let M be an NFA that represents L. We describe how M can be changed into an NFA M_1 that accepts L_1. The NFA M_1 simulates M, but whenever we read the first $\mathcal{Z}_i$-definition for some $i \in [k]$, we store in the finite state control whether it is empty or non-empty. Then, we check whether all the following $\mathcal{Z}_i$-definitions are also empty or non-empty, respectively, and we stop the computation in a non-accepting state, if this is not the case.

For every $w \in L_1$, we now define a ref-word $\mu(w)$ as follows. If, for every $i \in [k]$, w contains at most one non-empty $\mathcal{Z}_i$-definition (this includes the case that it contains only empty $\mathcal{Z}_i$-definitions, or no $\mathcal{Z}_i$-definition at all), then we set $\mu(w) = w$. Note that in this case the span-tuple represented by w is not filtered out by $\zeta^{=}_{E}$. If, for some $i \in [k]$, w contains at least two non-empty $\mathcal{Z}_i$-definitions $\Gamma_1 u \Gamma_2$ and $\Gamma'_1 u' \Gamma'_2$ with $u \neq u'$, then $\mu(w) = \bot$ is undefined. Note that the string-equality selection $\zeta^{=}_{E}$ filters out the span-tuple represented

by such subword-marked words. If for a subword-marked word w neither of these two cases apply, then the span-tuple represented by w is not filtered out by $\zeta^{=}_{E}$, and in this case we define $\mu(w)$ as follows.

For every $i \in [k]$ such that w contains at least two non-empty $\mathcal{Z}_i$-definitions, we do the following replacement. Let $\Gamma_1 u \Gamma_2$ be the leftmost $\mathcal{Z}_i$-definition, and choose one fixed $\widehat{\mathsf{z}} \in \mathcal{Z}_i$ such that ${}^{\widehat{\mathsf{z}}}\triangleright \in \Gamma_1$. Then we replace every $\mathcal{Z}_i$-definition $\Gamma'_1 u' \Gamma'_2$ that follows the initial $\Gamma_1 u \Gamma_2$ by $\Gamma'_1 \widehat{\mathsf{z}} \Gamma'_2$ (note that, by assumption, we have $u = u'$). This is well-defined, since, by assumption, we have $\{\mathsf{z} \in \mathcal{Z} \mid {}^{\mathsf{z}}\triangleright \in \Gamma_1\} \subseteq \mathcal{Z}_i$ for every $\mathcal{Z}_i$-definition $\Gamma_1 u \Gamma_2$. The string obtained by these replacements will be denoted by $\mu(w)$. It can be easily seen that, for every $w \in L_1$, if $\mu(w) \neq \bot$, then $\mu(w)$ is a ref-word with $w = \mathfrak{d}(\mu(w))$. Finally, we define $\mu(L_1) = \{\mu(w) \mid w \in L_1, \mu(w) \neq \bot\}$.

We next observe that $[\![\mu(L_1)]\!] = \zeta^{=}_{E}([\![L_1]\!])$. To this end, let $t \in (\zeta^{=}_{E}([\![L_1]\!]))(v)$ for some $v \in \Sigma^*$. This means that there is some $w \in L_1$ with $v = \mathfrak{e}(w)$ and $t = \mathsf{st}(w)$. Moreover, due to the string-equality selection $\zeta^{=}_{E}$, we know that $\mu(w) \neq \bot$, and therefore $\mathfrak{d}(\mu(w)) = w$, which also implies that $\mathfrak{e}(\mathfrak{d}(\mu(w))) = \mathfrak{e}(w) = v$ and $\mathsf{st}(\mathfrak{d}(\mu(w))) = \mathsf{st}(w) = t$. Consequently, $t \in [\![\mu(L_1)]\!](v)$.

For the other direction, let $t \in [\![\mu(L_1)]\!](v)$ for some $v \in \Sigma^*$. This means that there is some $w \in \mu(L_1)$ with $v = \mathfrak{e}(\mathfrak{d}(w))$ and $t = \mathsf{st}(\mathfrak{d}(w))$. By definition of $\mu(L_1)$, we have $w = \mu(w')$ for some $w' \in L_1$ with $\mu(w') \neq \bot$, which also means that $w' = \mathfrak{d}(\mu(w')) = \mathfrak{d}(w)$. Hence, $\mathfrak{e}(\mathfrak{d}(w)) = \mathfrak{e}(w') = v$ and $\mathsf{st}(\mathfrak{d}(w)) = \mathsf{st}(w') = t$. Consequently, $t \in [\![L_1]\!](v)$. Moreover, since $\mu(w') \neq \bot$, we know that all (if any) $\mathcal{Z}$-definitions of w' define the same factor, which implies that $t \in (\zeta^{=}_{E}([\![L_1]\!]))(v)$.

Next, we have to show that $\mu(L_1)$ is regular. Let M be an NFA that represents L_1 with a set Q of states. We describe how M can be changed into an NFA $\mu(M)$ that accepts $\mu(L_1)$. The NFA $\mu(M)$ has the same states as M, but in each state it also stores, for every $\mathsf{z} \in \mathcal{Z}$, a pair $(s_{\mathsf{z},\triangleright}, s_{\mathsf{z},\triangleleft}) \in (Q \cup \{\bot\})^2$. Initially, we have $(s_{\mathsf{z},\triangleright}, s_{\mathsf{z},\triangleleft}) = (\bot, \bot)$ for every $\mathsf{z} \in \mathcal{Z}$.

The NFA $\mu(M)$ simulates M, but every encountered non-empty $\mathcal{Z}$-definition is processed in one of two possible ways, depending on whether or not it is the *first* $\mathcal{Z}_i$-definition for some $i \in [k]$.

Let us assume that for some $i \in [k]$ we reach the first non-empty $\mathcal{Z}_i$-definition $\Gamma_1 u \Gamma_2$. Moreover, let $\widehat{\mathcal{Z}}_i$ be the variables from $\mathcal{Z}$ defined by this definition, i. e., $\widehat{\mathcal{Z}}_i = \{\mathsf{z} \in \mathcal{Z}_i \mid {}^{\mathsf{z}}\triangleright \in \Gamma_1\}$ (it is helpful to note that $\mathcal{Z}_i \setminus \widehat{\mathcal{Z}}_i$ is therefore exactly the set of variables that might be defined by any other $\mathcal{Z}_i$-definition to appear later in the input). Let $\widehat{\mathsf{z}}_i \in \widehat{\mathcal{Z}}_i$ be arbitrarily chosen (this variable $\widehat{\mathsf{z}}_i$ will play a central role later on, when we describe how $\mathcal{Z}_i$-definitions are processed that are not *first* $\mathcal{Z}_i$-definitions). Let us further assume that after reading Γ_1, we are in some state s. Now, $\mu(M)$ sets every $s_{\mathsf{z},\triangleright}$ with $\mathsf{z} \in \mathcal{Z}_i \setminus \widehat{\mathcal{Z}}_i$ to some nondeterministically guessed state. It then reads u simultaneously from the states s and all the guessed states $s_{\mathsf{z},\triangleright}$ with $\mathsf{z} \in \mathcal{Z}_i \setminus \widehat{\mathcal{Z}}_i$. The states that are reached from the guessed states $s_{\mathsf{z},\triangleright}$ with $\mathsf{z} \in \mathcal{Z}_i \setminus \widehat{\mathcal{Z}}_i$ by reading u are stored in the components $s_{\mathsf{z},\triangleleft}$ for all $\mathsf{z} \in \mathcal{Z}_i \setminus \widehat{\mathcal{Z}}_i$ (or $s_{\mathsf{z},\triangleleft} = \bot$ if no state can be reached from $s_{\mathsf{z},\triangleright}$ by reading u). This means that after reading u, $\mu(M)$ is in some state t (the state reached from s by reading u) and the additionally stored pairs $(s_{\mathsf{z},\triangleright}, s_{\mathsf{z},\triangleleft})$ are such that, for every $\mathsf{z} \in \mathcal{Z}_i \setminus \widehat{\mathcal{Z}}_i$, reading u can change M from state $s_{\mathsf{z},\triangleright}$ to $s_{\mathsf{z},\triangleleft}$ (unless $s_{\mathsf{z},\triangleleft} = \bot$). In particular, note that the start states $s_{\mathsf{z},\triangleright}$ are non-deterministically guessed, while the states $s_{\mathsf{z},\triangleleft}$ (unless we have $s_{\mathsf{z},\triangleleft} = \bot$) are also non-deterministically determined, since there might be several states rechable from $s_{\mathsf{z},\triangleright}$ by reading u.

Whenever we encounter a non-empty $\mathcal{Z}_i$-definition $\Gamma'_1 u' \Gamma'_2$, for some $i \in [k]$, that is *not* a first $\mathcal{Z}_i$-definition, then we proceed as follows. We let again $\widehat{\mathcal{Z}}'_i$ be the variables from $\mathcal{Z}$ defined by this definition. By the way we have processed the first $\mathcal{Z}_i$-definition $\Gamma_1 u \Gamma_2$, we know that for every $\mathsf{z}' \in \widehat{\mathcal{Z}}'_i$, the pair $(s_{\mathsf{z}',\triangleright}, s_{\mathsf{z}',\triangleleft})$ is such that we can reach $s_{\mathsf{z}',\triangleleft}$ from $s_{\mathsf{z}',\triangleright}$ by reading u (or $s_{\mathsf{z}',\triangleleft} = \bot$). Moreover, recall that $\widehat{\mathsf{z}}_i$ is an arbitrary variable defined by this first $\mathcal{Z}_i$-definition $\Gamma_1 u \Gamma_2$. Let s' be the state reached by M after having read Γ'_1. We check whether $s_{\mathsf{z}',\triangleright} = s'$ for every $\mathsf{z}' \in \widehat{\mathcal{Z}}'_i$, and whether $s_{\mathsf{z}',\triangleleft} = t'$ for every $\mathsf{z}' \in \widehat{\mathcal{Z}}'_i$ and the same state t', and, if this is the case, we read the variable reference $\widehat{\mathsf{z}}_i$ and move to state t'.

This concludes the definition of $\mu(M)$. We will show next the correctness of the construction, i. e., $\mathcal{L}(\mu(M)) = \mu(L_1)$.

Let w be accepted by M and assume that $\mu(w) \neq \bot$, i. e., $\mu(w) \in \mu(L_1)$. If, for every $i \in [k]$, w has at most one non-empty $\mathcal{Z}_i$-definition, then $\mu(M)$ accepts $\mu(w) = w$ (by the same run as M accepts w). So let us assume that $i_1, i_2, \ldots, i_p \in [k]$ are such that w has at least two non-empty $\mathcal{Z}_{i_j}$-definitions for $j \in [p]$. Since $\mu(w) \neq \bot$, we know that, for every $j \in [p]$, all $\mathcal{Z}_{i_j}$-definitions of w define the same factor u_j, i. e., every non-empty $\mathcal{Z}_{i_j}$-definition has the form $\Gamma_1 u_j \Gamma_2$. We have to show that $\mu(M)$ accepts $\mu(w)$.

We assume that, for every $j \in [p]$, $\Gamma_1^j u_j \Gamma_2^j$ is the first $\mathcal{Z}_{i_j}$-definition of w, and let $\widehat{\mathcal{Z}}_{i_j}$ be the variables from $\mathcal{Z}_{i_j}$ defined by this $\mathcal{Z}_{i_j}$-definition $\Gamma_1^j u_j \Gamma_2^j$. For every $\mathsf{z} \in \mathcal{Z}_{i_j} \setminus \widehat{\mathcal{Z}}_{i_j}$ that is defined by some $\mathcal{Z}_{i_j}$-definition in w, let $\widetilde{s}_{\mathsf{z},\triangleright}$ and $\widetilde{s}_{\mathsf{z},\triangleleft}$ be the states between which M reads the factor u_j in the $\mathcal{Z}_{i_j}$-definition for z. The NFA $\mu(M)$ can now simulate M until it encounters the first $\mathcal{Z}_{i_j}$-definition $\Gamma_1^j u_j \Gamma_2^j$. Then, after reading Γ_1^j, $\mu(M)$ can guess for every $\mathsf{z} \in \mathcal{Z}_{i_j} \setminus \widehat{\mathcal{Z}}_{i_j}$ the states $\widetilde{s}_{\mathsf{z},\triangleright}$ and store them in the components $s_{\mathsf{z},\triangleright}$, if w contains some $\mathcal{Z}_{i_j}$-definition that defines z, and it can guess some arbitrary state otherwise. Then, it can read u_j in such a way that exactly the states $\widetilde{s}_{\mathsf{z},\triangleleft}$ are stored in the components $s_{\mathsf{z},\triangleleft}$, if w contains some $\mathcal{Z}_{i_j}$-definition that defines z, and $\widetilde{s}_{\mathsf{z},\triangleleft} = \bot$ otherwise. This must be possible by definition of these states $\widetilde{s}_{\mathsf{z},\triangleright}$ and $\widetilde{s}_{\mathsf{z},\triangleleft}$, and by $\mu(M)$'s definition. The NFA $\mu(M)$ can now proceed with simulating M, but whenever it encounters another $\mathcal{Z}_{i_j}$-definition $\Gamma_1'^j u_j \Gamma_2'^j$, due to the stored states, it will read $\Gamma_1'^j \widehat{\mathsf{z}}_j \Gamma_2'^j$ instead of $\Gamma_1'^j u_j \Gamma_2'^j$ (recall that $\widehat{\mathsf{z}}_i$ is the fixed variable from $\widehat{\mathcal{Z}}_{i_j}$ selected earlier). Hence, it accepts the word $\mu(w)$.

On the other hand, let some w' be accepted by $\mu(M)$. If, for every $i \in [k]$, w' has at most one non-empty $\mathcal{Z}_i$-definition, then M accepts w', and $\mu(w') = w'$, i. e., $w' \in \mu(L_1)$. So let us assume that $i_1, i_2, \ldots, i_p \in [k]$ are such that w' has at least two non-empty $\mathcal{Z}_{i_j}$-definitions for $j \in [p]$. For every $j \in [p]$, the first $\mathcal{Z}_{i_j}$-definition has the form $\Gamma_1^j u_j \Gamma_2^j$, and all other $\mathcal{Z}_{i_j}$-definitions have the form $\Gamma_1'^j \widehat{\mathsf{z}}_j \Gamma_2'^j$, where $\widehat{\mathsf{z}}_j$ is from the set $\widehat{\mathcal{Z}}_{i_j}$ of those variables from $\mathcal{Z}_{i_j}$ that are defined by the first $\mathcal{Z}_{i_j}$-definition $\Gamma_1^j u_j \Gamma_2^j$. Moreover, for every such $\mathcal{Z}_{i_j}$-definition $\Gamma_1'^j \widehat{\mathsf{z}}_j \Gamma_2'^j$ where $\widehat{\mathsf{z}}_j$ is read by going from some state s to some state t, we know that M can read u_j by going from state s to t (this follows from the definition $\mu(M)$). This directly implies that the word w obtained from w' by replacing each $\Gamma_1'^j \widehat{\mathsf{z}}_j \Gamma_2'^j$ by $\Gamma_1'^j u_j \Gamma_2'^j$ satisfies $\mu(w) = w'$ and $w \in \mathcal{L}(M) = L_1$. Thus, $w' \in \mu(L_1)$.

We conclude that we have shown that $\varsigma^{=}_E(\llbracket L \rrbracket) = \varsigma^{=}_E(\llbracket L_1 \rrbracket)$, that $\llbracket \mu(L_1) \rrbracket = \varsigma^{=}_E(\llbracket L_1 \rrbracket)$, and that $\mu(L_1)$ is a regular ref-language over Σ and $\mathcal{X}$. Thus, $L' := \mu(L_1)$ is a regular ref-language over Σ and $\mathcal{X}$ with $\llbracket L' \rrbracket = \varsigma^{=}_E(\llbracket L \rrbracket)$. Moreover, L' is obviously reference-bounded.

In order to conclude this proof, we have to show how to adapt the construction to the general case, i. e., the case where, for some $i, i' \in [k]$ with $i \neq i'$, it is possible that $\mathcal{Z}_i \cap \mathcal{Z}_j \neq \emptyset$, and $w \in L$ might contain $\mathcal{Z}$-definitions that are both $\mathcal{Z}_i$- and $\mathcal{Z}_{i'}$-definitions.

For a fixed $w \in L$, we can define an equivalence relation over $\mathcal{Z}$ as follows. We set $\mathsf{x} \sim'_w \mathsf{z}$ if $\mathsf{x}, \mathsf{z} \in \mathcal{Z}_i$ for some $i \in [k]$, or if w has a non-empty $\mathcal{Z}$-definition $\Gamma_1 u \Gamma_2$ or an empty $\mathcal{Z}$-definition Γ_1 such that $\{{}^{\mathsf{x}}\triangleright, {}^{\mathsf{z}}\triangleright\} \subseteq \Gamma_1$, and we let $\sim_w$ be the transitive closure of $\sim'_w$. Let $E_{\sim_w} = \{\mathcal{Z}_1^{\sim_w}, \mathcal{Z}_2^{\sim_w}, \ldots, \mathcal{Z}_{k_{\sim_w}}^{\sim_w}\}$ be the set of the equivalence classes of $\sim_w$. We note that, for every $w \in L$, we have that $E_{\sim_w}$ satisfies the two properties from above: (1) $\mathcal{Z}_i^{\sim_w} \cap \mathcal{Z}_{i'}^{\sim_w} = \emptyset$ for all $i, i' \in [k_{\sim_w}]$ with $i \neq i'$, and (2), for every $w \in L$ and for every non-empty $\mathcal{Z}$-definition $\Gamma_1 u \Gamma_2$ and every empty $\mathcal{Z}$-definition Γ_1 in w, there is some $i \in [k_{\sim_w}]$ such that $\{\mathsf{z} \in \mathcal{Z} \mid {}^{\mathsf{z}}\triangleright \in \Gamma_1\} \subseteq \mathcal{Z}_i^{\sim_w}$.

For the following, it is a crucial observation that, for every $w \in L$, the tuple described by w is filtered out by $\varsigma^{=}_{E}$ if and only if it is filtered out by $\varsigma^{=}_{E_{\sim_w}}$, i. e., $\varsigma^{=}_{E}(\{\mathsf{st}(w)\}) = \varsigma^{=}_{E_{\sim_w}}(\{\mathsf{st}(w)\})$. Indeed, if the tuple described by w is filtered out by $\varsigma^{=}_{E}$, then it must be filtered out by $\varsigma^{=}_{E_{\sim_w}}$ as well, since E is a refinement of $E_{\sim_w}$. On the other hand, if the tuple described by w satisfies $\varsigma^{=}_{E}$, then, by definition of $\sim_w$, it must also satisfy all the string-equality selections described by $E_{\sim_w}$.

For every $w \in L$, we have defined above the condition for w to be in L_1, and we have defined the ref-word $\mu(w)$. For every fixed $w \in L$, these definitions depend only on w and the sets $\mathcal{Z}_1, \mathcal{Z}_2, \ldots, \mathcal{Z}_k$. However, we had to require that these sets satisfied our two conditions: (1) $\mathcal{Z}_i \cap \mathcal{Z}_{i'} = \emptyset$ for every $i, i' \in [k]$ with $i \neq i'$, and (2), for every $w \in L$ and for every non-empty $\mathcal{Z}$-definition $\Gamma_1 u \Gamma_2$ and every empty $\mathcal{Z}$-definition Γ_1 in w, there is some $i \in [k]$ such that $\{\mathsf{z} \in \mathcal{Z} \mid {}^{\mathsf{z}}\triangleright \in \Gamma_1\} \subseteq \mathcal{Z}_i$. Since the sets from $E_{\sim_w}$ also satisfy these conditions, we can also define, for every $w \in L$, the condition for w to be in L_1, and the ref-word $\mu(w)$, but with respect to the sets $\mathcal{Z}_1^{\sim_w}, \mathcal{Z}_2^{\sim_w}, \ldots, \mathcal{Z}_{k_{\sim_w}}^{\sim_w}$.

It is not hard to see that for this adapted version of L_1, we also have that $\varsigma^{=}_{E}([\![L]\!]) = \varsigma^{=}_{E}([\![L_1]\!])$. Indeed, every word $w \in L_1$ that, for some $i \in [k_{\sim_w}]$, contains both a non-empty $\mathcal{Z}_i^{\sim_w}$-definition and an empty $\mathcal{Z}_i^{\sim_w}$-definition must be filtered out by $\varsigma^{=}_{E}$.

Moreover, the adapted version of μ also satisfies that, if $w \in L_1$ with $\mu(w) \neq \bot$, then $\mu(w)$ is a ref-word with $w = \mathfrak{d}(\mu(w))$. In particular, this also means that $[\![\mu(L_1)]\!] = \varsigma^{=}_{E}([\![L_1]\!])$.

Consequently, also in the general case, we have that $\varsigma^{=}_{E}([\![L]\!]) = [\![\mu(L_1)]\!]$. However, it remains to show that with the adapted definitions of L_1 and $\mu(w)$, L_1 and $\mu(L_1)$ are still regular. In the constructions of the NFAs for L_1 and $\mu(L_1)$, we have explained how the computation should proceed on a fixed input w, and in dependency of the sets $\mathcal{Z}_1, \mathcal{Z}_2, \ldots, \mathcal{Z}_k$. Consequently, for a fixed input w, we can carry out the constructions also for the sets $\mathcal{Z}_1^{\sim_w}, \mathcal{Z}_2^{\sim_w}, \ldots, \mathcal{Z}_{k_{\sim_w}}^{\sim_w}$, as long as the automaton has knowledge of these sets, which, after all, depend on the input. Hence, we let the NFAs simply initially guess the equivalence relation $\sim_w$, and then we will verify this guess while processing the input w, and we will abort the computation if the guess of $\sim_w$ has been incorrect. In this way, we can assume that for any input w the sets $\mathcal{Z}_1^{\sim_w}, \mathcal{Z}_2^{\sim_w}, \ldots, \mathcal{Z}_{k_{\sim_w}}^{\sim_w}$ are known to the NFA, which means that we can carry out the NFA constructions described above, just with respect to the sets $\mathcal{Z}_1^{\sim_w}, \mathcal{Z}_2^{\sim_w}, \ldots, \mathcal{Z}_{k_{\sim_w}}^{\sim_w}$ instead of the set $\mathcal{Z}_1, \mathcal{Z}_2, \ldots, \mathcal{Z}_k$.

This completes the proof of Theorem 6.3. □

6.2.2. *The Span Fusion Operation.* 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 $[i,j\rangle \uplus \bot = \bot \uplus [i,j\rangle = [i,j\rangle$. Intuitively speaking, the operation $\uplus$ constructs the set-union of two spans and fills in the gaps to turn it into a valid span. Note that $\uplus$ is obviously associative. 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 $s \in K$.

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 t (i. e., span-tuple t) in $S(w)$, contains the span $\biguplus(\{t(\mathsf{y}_i) \mid i \in [k]\})$.

Definition 6.4. 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 $((\mathcal{X}\setminus\lambda)\cup\{\mathsf{x}\})$ 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 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 $\biguplus_{\{\lambda_i\to\mathsf{x}_i \mid i\in[k]\}}$, respectively.

Example 6.5. Let $L = \mathcal{L}(\ {}^{\mathsf{x}_1}\!\triangleright\, \mathsf{a}^*\ {}^{\mathsf{x}_2}\!\triangleright\, \mathsf{b}^*\ \triangleleft^{\mathsf{x}_1}\ \mathsf{a}^*\triangleleft^{\mathsf{x}_2})$ be a subword-marked language over $\{\mathsf{a},\mathsf{b}\}$ and $\{\mathsf{x}_1,\mathsf{x}_2\}$. For $w = \mathtt{aabaaa}$, we have $[\![L]\!](w) = \{([1,4\rangle, [3,7\rangle)\}$. Moreover, let $L' = \mathcal{L}(\ {}^{\mathsf{y}_1}\!\triangleright\, \mathsf{a}^*\ \triangleleft^{\mathsf{y}_1}\ {}^{\mathsf{y}_2}\!\triangleright\ {}^{\mathsf{y}_3}\!\triangleright\, \mathsf{b}^*\ \triangleleft^{\mathsf{y}_3}\ \triangleleft^{\mathsf{y}_2}\ {}^{\mathsf{y}_4}\!\triangleright\, \mathsf{a}^*\triangleleft^{\mathsf{y}_4})$ be a subword-marked language over $\{\mathsf{a},\mathsf{b}\}$ and $\{\mathsf{y}_1,\mathsf{y}_2,\mathsf{y}_3,\mathsf{y}_4\}$, and let $\Lambda = \{\{\mathsf{y}_1,\mathsf{y}_2\} \to \mathsf{x}_1, \{\mathsf{y}_3,\mathsf{y}_4\} \to \mathsf{x}_2\}$. Then

$$\biguplus\nolimits_\Lambda [\![L']\!](w) = \{\biguplus\nolimits_\Lambda([1,3\rangle, [3,4\rangle, [3,4\rangle, [4,7\rangle)\} = \{([1,4\rangle, [3,7\rangle)\} = [\![L]\!](w)$$

In fact, it can be easily verified that $\biguplus_\Lambda [\![L']\!] = [\![L]\!]$.

6.2.3. *The Split Operation.* Intuitively speaking, a split t' of a span-tuple t is any span-tuple obtained from t by splitting each span $t(\mathsf{x})$ into several spans, e. g., splitting $[3,24\rangle$ into spans $[3,5\rangle, [5,17\rangle, [17,17\rangle, [17,24\rangle$. Any such split t' can be easily translated back into t by applications of the span fusion, i. e., $\biguplus_{\{\lambda_\mathsf{x}\to\mathsf{x} \mid \mathsf{x}\in\mathcal{X}\}}(t') = t$, where the sets λ_x are the variables used for the split version of $t(\mathsf{x})$. There are two important aspects of this operation. Firstly, if we allow large enough splits, i. e., if we choose the sets λ_x large enough, then we can always split in such a way that all variables are non-overlapping. Secondly, the split of a regular spanner (i. e., the spanner that extracts all splits of the original span-tuples) is also a regular spanner. Let us next define this formally.

Let us first recall the definition of non-overlapping variables, which has already been introduced in Section 2. Let L be a subword-marked language over Σ and $\mathcal{X}$, let $w \in L$ and $t = \mathsf{st}(w)$. We say that variables $\mathsf{x},\mathsf{y} \in \mathcal{X}$ are *non-overlapping (with respect to w (or t))* if $t(\mathsf{x}) = t(\mathsf{y})$ or $t(\mathsf{x})$ and $t(\mathsf{y})$ are disjoint (i. e., $t(\mathsf{x}) = [\ell_\mathsf{x}, r_\mathsf{x}\rangle$ and $t(\mathsf{y}) = [\ell_\mathsf{y}, r_\mathsf{y}\rangle$ with $r_\mathsf{x} \leqslant \ell_\mathsf{y}$ or $r_\mathsf{y} \leqslant \ell_\mathsf{x}$). The variables x and y are non-overlapping with respect to the subword-marked language L, if x and y are non-overlapping with respect to every $w \in L$.

For any set $\mathcal{X}$ of variables, we define the *extended variable set* $\mathsf{ext}(\mathcal{X})$ for $\mathcal{X}$ by $\mathsf{ext}(\mathcal{X}) := \bigcup_{\mathsf{x}\in\mathcal{X}}\{\mathsf{x}^1, \mathsf{x}^2, \ldots, \mathsf{x}^\wp\}$, where $\wp := 4|\mathcal{X}|^2 - 1$ (the choice of the number $\wp$ is crucial and will

become clear later). Intuitively, for every $\mathsf{x} \in \mathcal{X}$, the extended variable set for $\mathcal{X}$ contains $\wp$ new variables, e. g., $\mathsf{ext}(\{\mathsf{x},\mathsf{y},\mathsf{z}\}) = \{\mathsf{x}^1,\ldots,\mathsf{x}^{37},\mathsf{y}^1,\ldots,\mathsf{y}^{37},\mathsf{z}^1,\ldots,\mathsf{z}^{37}\}$. With respect to $\mathsf{ext}(\mathcal{X})$, we can also make the following observation.

Observation 6.6. For every set $\mathcal{X}$ of variables, we have that $|\mathsf{ext}(\mathcal{X})| = \mathrm{O}(|\mathcal{X}|^3)$.

Let t be an $\mathcal{X}$-tuple. Any $\mathsf{ext}(\mathcal{X})$-tuple t' is called a *split* of t if it satisfies the following properties:

- For every $\mathsf{x} \in \mathcal{X}$,
  - if $t(\mathsf{x}) = \bot$, then $t'(\mathsf{x}^1) = t'(\mathsf{x}^2) = \ldots = t'(\mathsf{x}^\wp) = \bot$,
  - if $t(\mathsf{x}) = [k,\ell\rangle$, then, for every $i \in [\wp]$, $t'(\mathsf{x}^i) = [k_i,\ell_i\rangle$ such that $k_1 = k$, $\ell_\wp = \ell$, and, for every $i \in [\wp-1]$, $\ell_i = k_{i+1}$.
- Any two variables of $\mathsf{ext}(\mathcal{X})$ are non-overlapping with respect to t.

For example, let t be an $\{\mathsf{x},\mathsf{y}\}$-tuple with $t(\mathsf{x}) = [3,24\rangle$ and $t(\mathsf{y}) = [7,15\rangle$. Then a possible split of t would be the $\mathsf{ext}(\mathcal{X})$-tuple t' with $t'(\mathsf{x}^1) = [3,7\rangle, t'(\mathsf{x}^2) = [7,12\rangle, t'(\mathsf{x}^3) = [12,15\rangle, t'(\mathsf{x}^4) = [15,24\rangle$ and $t'(\mathsf{x}^j) = [24,24\rangle$ for every $j \in \{5,6,\ldots,17\}$, and $t'(\mathsf{y}^1) = [7,12\rangle, t'(\mathsf{y}^2) = [12,12\rangle, t'(\mathsf{y}^3) = [12,15\rangle$ and $t'(\mathsf{y}^j) = [15,15\rangle$ for every $j \in \{4,5\ldots,17\}$.

For every $\mathsf{x} \in \mathcal{X}$, we define $\lambda_\mathsf{x}^\wp = \{\mathsf{x}^1,\mathsf{x}^2,\ldots,\mathsf{x}^\wp\}$. Hence, $\mathsf{ext}(\mathcal{X}) = \bigcup_{\mathsf{x}\in\mathcal{X}} \lambda_\mathsf{x}^\wp$. By definition, every split t' of t satisfies $\biguplus_{\{\lambda_\mathsf{x}^\wp \to \mathsf{x} \mid \mathsf{x}\in\mathcal{X}\}}(t') = t$.

We can define splits analogously for subword-marked words. Let L be a subword-marked language over Σ and $\mathcal{X}$. For any $w \in L$, we say that a subword-marked word w' over Σ and $\mathsf{ext}(\mathcal{X})$ is a *split* of w, if $\mathfrak{e}(w) = \mathfrak{e}(w')$ and $\mathsf{st}(w')$ is a split of $\mathsf{st}(w)$.

For an $\mathcal{X}$-tuple t (or subword-marked word w), let $\mathsf{split}_\mathcal{X}(t)$ ($\mathsf{split}_\mathcal{X}(w)$, respectively) be the set of all splits of t (splits of w, respectively). For any subword-marked language over Σ and $\mathcal{X}$, let $\mathsf{split}_\mathcal{X}(L) = \{\mathsf{split}_\mathcal{X}(w) \mid w \in L\}$, and let $\mathsf{split}_\mathcal{X}([\![L]\!]) = [\![\mathsf{split}_\mathcal{X}(L)]\!]$.

We will next see that the split of any regular spanner is also a regular spanner. We will show that any NFA M for a subword-marked language L can be modified such that it accepts $\mathsf{split}_\mathcal{X}(L)$. To this end, we simply non-deterministically use the markers ${}^{\mathsf{x}^1}\triangleright, \triangleleft^{\mathsf{x}^1}, {}^{\mathsf{x}^2}\triangleright, \triangleleft^{\mathsf{x}^2}, \ldots, {}^{\mathsf{x}^\wp}\triangleright, \triangleleft^{\mathsf{x}^\wp}$ for processing the part of the input that by M is read between markers ${}^{\mathsf{x}}\triangleright$ and $\triangleleft^{\mathsf{x}}$. Moreover, to ensure that each two variables are non-overlapping, we have to make sure that all these non-deterministic split points agree with each other in the sense that whenever some split point is created by reading $\triangleleft^{\mathsf{x}^j}$ and ${}^{\mathsf{x}^{j+1}}\triangleright$, then we also must create a split-point for all other variables by reading $\triangleleft^{\mathsf{y}^{j'}}$ and ${}^{\mathsf{y}^{j'+1}}\triangleright$ for the correct $j' \in [\wp]$. This construction is formally carried out in the proof of the next lemma.

Lemma 6.7. *Let L be a regular subword-marked language over Σ and $\mathcal{X}$. Then $\mathsf{split}_\mathcal{X}(L)$ is a regular subword-marked language over Σ and $\mathsf{ext}(\mathcal{X})$.*

Proof. Let M be an NFA for L. To transform M into an NFA for $\mathsf{split}_\mathcal{X}(L)$, we perform two transformation steps. In general, we use the following terminology. Reading a symbol $\Gamma \subseteq \Gamma_\mathcal{X}$ with ${}^{\mathsf{x}}\triangleright \in \Gamma$ is called *opening* variable x, and reading a symbol $\Gamma \subseteq \Gamma_\mathcal{X}$ with $\triangleleft^{\mathsf{x}} \in \Gamma$ is called *closing* variable x. Moreover, at any step of a computation, we say that variable x is *open*, if we have already read a symbol $\Gamma \subseteq \Gamma_\mathcal{X}$ with ${}^{\mathsf{x}}\triangleright \in \Gamma$, but we have not yet read a symbol $\Gamma' \subseteq \Gamma_\mathcal{X}$ with $\triangleleft^{\mathsf{x}} \in \Gamma$. We assume that any NFA always stores in its finite state control which variables are currently open (this information can be maintained during any computation).

Step 1 (Changing M into M'): The NFA M' simulates the computation of M on w, but with the following differences. Whenever M opens some variable $\mathsf{x} \in \mathcal{X}$, then M' opens variable

x^1. At any point in the computation, M' can nondeterministically choose some currently open variable x^j with $j < \wp$, and then close x^j and open x^{j+1} (observe that closing $\mathsf{x}^\wp$ is not included here). Whenever M closes some variable $\mathsf{x} \in \mathcal{X}$, then M' closes the currently open variable x^j (which may be $\mathsf{x}^\wp$) and then opens and closes all remaining variables $\mathsf{x}^{j+1}, \ldots, \mathsf{x}^\wp$.

Let us say that a subword-marked word w' over Σ and $\mathsf{ext}(\mathcal{X})$ is a *pseudo-split* of a subword marked word w over Σ and $\mathcal{X}$, if the first property of the definition of splits is satisfied:

- if $\mathsf{st}(w)(\mathsf{x}) = \bot$, then $\mathsf{st}(w')(\mathsf{x}^1) = \mathsf{st}(w')(\mathsf{x}^2) = \ldots = \mathsf{st}(w')(\mathsf{x}^\wp) = \bot$,
- if $\mathsf{st}(w)(\mathsf{x}) = [k, \ell\rangle$, then, for every $i \in [\wp]$, $\mathsf{st}(w')(\mathsf{x}^i) = [k_i, \ell_i\rangle$ such that $k_1 = k$, $\ell_\wp = \ell$, and, for every $i \in [\wp - 1]$, $\ell_i = k_{i+1}$.

It can be easily seen that M' accepts exactly all subword-marked words over Σ and $\mathsf{ext}(\mathcal{X})$ that are pseudo-splits of subword-marked words accepted by M.

We now have to filter out those pseudo-splits which are not valid splits, i. e., those that do not satisfy the second property of the definition of splits: any two variables of $\mathsf{ext}(\mathcal{X})$ are non-overlapping with respect to $\mathsf{st}(w')$.

Step 2 (Changing M' into M''): The NFA M'' simulates the computation of M' on w, but it can interrupt certain computations and reject the input as follows. Let us assume that M' reads a symbol $\Gamma \subseteq \Gamma_{\mathcal{X}}$, which, since Γ is non-empty, means that some variable is either opened or closed at this step. Then M' checks whether there is some variable y that is open right before M' reads Γ and $\lhd^\mathsf{y} \notin \Gamma$ and, if this is the case, M' interrupts the computation and rejects.

Obviously, if some $w' \in \mathcal{L}(M')$ is a valid split of some $w \in \mathcal{L}(M)$, then upon reading w' the situation that causes M'' to interrupt will never occur, which means that $w' \in \mathcal{L}(M'')$. On the other hand, if some $w' \in \mathcal{L}(M')$ is only a pseudo-split of some $w \in \mathcal{L}(M)$, but not a valid split, then there are two variables in $\mathsf{ext}(\mathcal{X})$ that are overlapping with respect to w', which means that any computation of M'' on w' must lead to the situation that causes M' to interrupt and reject. Consequently, M'' accepts the subset of $\mathcal{L}(M')$ of valid splits. Thus, $\mathcal{L}(M'') = \mathsf{split}_{\mathcal{X}}(L)$ □

Let L be a regular subword-marked language over Σ and $\mathcal{X}$, and let $\varsigma^{=}_{E}$ be a string-equality selection with $E = \{\mathcal{Z}_1, \mathcal{Z}_2, \ldots, \mathcal{Z}_k\} \subseteq \mathcal{P}(\mathcal{X})$. The next question is how we can represent the spanner $\varsigma^{=}_{E}(\llbracket L \rrbracket)$ by first applying the split version of $\llbracket L \rrbracket$, then applying some string-equality selections tailored to the split span relation, and then, finally, using the span-fusion operation to translate back the split span-tuples into the original span-tuples. More precisely, we want to represent $\varsigma^{=}_{E}(\llbracket L \rrbracket)$ in the form $(\biguplus_{\Lambda}(\varsigma^{=}_{E'}(\llbracket \mathsf{split}_{\mathcal{X}}(L) \rrbracket)))$, for some suitable string-equality selection $\varsigma^{=}_{E'}$.

Intuitively speaking, $\varsigma^{=}_{E'}$ simulates $\varsigma^{=}_{E}$ in the sense that for any $\mathsf{x}, \mathsf{y} \in \mathcal{Z}_i$ it checks the string-equality of $t(\mathsf{x})$ and $t(\mathsf{y})$ by checking the string-equalities for all the variable pairs $t(\mathsf{x}_j)$ and $t(\mathsf{y}_j)$ for every $j \in [\wp]$. More formally, for every $i \in [k]$ and every $j \in [\wp]$, we define $\mathcal{Y}^j_i = \{\mathsf{x}^j \mid \mathsf{x} \in \mathcal{Z}_i\}$, and we set $\mathsf{split}_{\mathcal{X}}(\varsigma^{=}_{E}) = \varsigma^{=}_{\{\mathcal{Y}^j_i \mid j \in [\wp], i \in [k]\}}$.

The following lemma follows more or less directly from the definitions. Recall that, for every $\mathsf{x} \in \mathcal{X}$, we have defined $\lambda^\wp_{\mathsf{x}} = \{\mathsf{x}^1, \mathsf{x}^2, \ldots, \mathsf{x}^\wp\}$.

Lemma 6.8. *Let L be a regular subword-marked language over Σ and $\mathcal{X}$, let $\varsigma^{=}_{E}$ be a string-equality selection with $E \subseteq \mathcal{P}(\mathcal{X})$, let $\Lambda = \{\lambda^\wp_{\mathsf{x}} \to \mathsf{x} \mid \mathsf{x} \in \mathcal{X}\}$ and let $w \in \Sigma^*$. Then $(\biguplus_{\Lambda}(\mathit{split}_{\mathcal{X}}(\varsigma^{=}_{E})(\llbracket \mathit{split}_{\mathcal{X}}(L) \rrbracket)))(w) \subseteq (\varsigma^{=}_{E}(\llbracket L \rrbracket))(w)$.*

Proof. Let $t \in (\biguplus_\Lambda(\mathsf{split}_\mathcal{X}(\varsigma^{=}_E)([\![\mathsf{split}_\mathcal{X}(L)]\!])))(w)$. This means that there is a split t' of t with $t' \in (\mathsf{split}_\mathcal{X}(\varsigma^{=}_E)(\mathsf{split}_\mathcal{X}([\![L]\!])))(w)$ and $\biguplus_\Lambda(t') = t$. Due to the string-equality selection $\mathsf{split}_\mathcal{X}(\varsigma^{=}_E)$, this means that, for every $i \in [k]$ and all $\mathsf{x}, \mathsf{y} \in \mathcal{Z}_i$, if $t'(\mathsf{x}^1) \neq \bot$ and $t'(\mathsf{y}^1) \neq \bot$, then the spans $t'(\mathsf{x}^j)$ and $t'(\mathsf{y}^j)$ refer to equal factors of w for every $j \in [\wp]$. This directly implies that also $(\biguplus_\Lambda(t'))(\mathsf{x})$ and $(\biguplus_\Lambda(t'))(\mathsf{y})$ refer to equal factors in w, which means that $\biguplus_\Lambda(t') \in (\varsigma^{=}_E([\![L]\!]))(w)$. Since $\biguplus_\Lambda(t') = t$, this means that $t \in (\varsigma^{=}_E([\![L]\!]))(w)$. □

The converse of Lemma 6.8 does not necessarily hold. This is due to the fact that for a $t \in (\varsigma^{=}_E([\![L]\!]))(w)$ and $\mathsf{x}, \mathsf{y} \in \mathcal{Z}_i$ such that $t(\mathsf{x})$ and $t(\mathsf{y})$ refer to equal factors, there might not be any split t' of t with the property that, for every $j \in [\wp]$, $t(\mathsf{x}^j)$ and $t(\mathsf{y}^j)$ refer to equal factors, which is necessary for t being in $(\biguplus_\Lambda(\mathsf{split}_\mathcal{X}(\varsigma^{=}_E)([\![\mathsf{split}_\mathcal{X}(L)]\!])))(w)$. Let us clarify this with an example.

Example 6.9. Let $L = \mathcal{L}(\,{}^{\mathsf{x}}\triangleright\, \mathsf{a}\, {}^{\mathsf{y}}\triangleright\, \mathsf{a}^*\, \triangleleft^{\mathsf{x}}\, \mathsf{a} \triangleleft^{\mathsf{y}})$ be a regular subword-marked language over $\Sigma = \{\mathsf{a}\}$ and $\mathcal{X} = \{\mathsf{x}, \mathsf{y}\}$. Then $t = ([1, 19\rangle, [2, 20\rangle)$ is in $(\varsigma^{=}_{\{\mathsf{x},\mathsf{y}\}}[\![L]\!])(\mathsf{a}^{19})$. Now consider an arbitrary split t' of t. By definition, t' is a span-tuple over $\mathsf{ext}(\mathcal{X}) = \{\mathsf{x}^1, \ldots, \mathsf{x}^{17}, \mathsf{y}^1, \ldots, \mathsf{y}^{17}\}$, such that

- $t'(\mathsf{x}^1) = [1, \ell_1\rangle, t'(\mathsf{x}^2) = [\ell_1, \ell_2\rangle, t'(\mathsf{x}^3) = [\ell_2, \ell_3\rangle, \ldots, t'(\mathsf{x}^{17}) = [\ell_{16}, 19\rangle$,
- $t'(\mathsf{y}^1) = [2, k_1\rangle, t'(\mathsf{y}^2) = [k_1, k_2\rangle, t'(\mathsf{y}^3) = [k_2, k_3\rangle, \ldots, t'(\mathsf{y}^{17}) = [k_{16}, 20\rangle$,
- any two variables from $\mathsf{ext}(\mathcal{X})$ are non-overlapping.

Now assume that t' also has the property that, for every $i \in [17]$, $t(\mathsf{x}^i)$ and $t(\mathsf{y}^i)$ refer to equal factors of a^{19}. For simplicity, let us assume that none of the spans are empty (we will briefly discuss the general case below). Since x^1 and y^1 are non-overlapping, we must have $\ell_1 = 2$, which means that $t'(\mathsf{x}^1)$ refers to a factor a. Since, by assumption, $t'(\mathsf{x}^1)$ and $t'(\mathsf{y}^1)$ refer to equal factors, we therefore must also have $k_1 = 3$. Analogously, we can conclude that $\ell_2 = 3$ and $k_2 = 4$, and inductively repeating this argument implies that every span must refer to a factor of size 1, which is not possible since $t(\mathsf{x})$ and $t(\mathsf{y})$ both refer to a factor of size 18, and we only have 17 variables $\mathsf{x}^1, \ldots, \mathsf{x}^{17}$ and 17 variables $\mathsf{y}^1, \ldots, \mathsf{y}^{17}$. Note that the same kind of pigeon-hole argument also applies if some of the spans of the variables from $\mathsf{ext}(\mathcal{X})$ are empty. Consequently, no split t' can have the property that, for every $i \in [17]$, $t(\mathsf{x}^i)$ and $t(\mathsf{y}^i)$ refer to equal factors of a^{19}, which means that t cannot be in $(\biguplus_\Lambda(\mathsf{split}_\mathcal{X}(\varsigma^{=}_E)([\![\mathsf{split}_\mathcal{X}(L)]\!])))(\mathsf{a}^{19})$.

Note that this problem cannot be resolved by simply enlarging the extended variable set $\mathsf{ext}(\mathcal{X})$ to $\{\mathsf{x}^1, \ldots, \mathsf{x}^p, \mathsf{y}^1, \ldots, \mathsf{y}^p\}$ for some sufficiently large p, since then the same contradiction can be obtained with the document a^{p+2}.

This example points out that using a string-equality selection with respect to variables that can overlap is the reason that the converse of Lemma 6.8 is not necessarily true. We will next restrict the string-equality selections accordingly.

Let L be a regular subword-marked language over Σ and $\mathcal{X}$, and let $E = \{\mathcal{Z}_1, \mathcal{Z}_2, \ldots, \mathcal{Z}_k\} \subseteq \mathcal{P}(\mathcal{X})$. We say that E is *non-overlapping* (*with respect to* L) if all $\mathsf{x}, \mathsf{y} \in \bigcup_{i\in[k]} \mathcal{Z}_i$ are non-overlapping with respect to L.

Lemma 6.10. *Let L be a regular subword-marked language over Σ and $\mathcal{X}$, let $\varsigma^{=}_E$ be a string-equality selection such that $E \subseteq \mathcal{P}(\mathcal{X})$ is non-overlapping with respect to L, let $\Lambda = \{\lambda^{\wp}_{\mathsf{x}} \to \mathsf{x} \mid \mathsf{x} \in \mathcal{X}\}$ and let $w \in \Sigma^*$. Then $(\varsigma^{=}_E([\![L]\!]))(w) \subseteq (\biguplus_\Lambda(\mathsf{split}_\mathcal{X}(\varsigma^{=}_E)([\![\mathsf{split}_\mathcal{X}(L)]\!])))(w)$.*

Proof. Let $E = \{\mathcal{Z}_1, \mathcal{Z}_2, \ldots, \mathcal{Z}_k\}$, and let $t \in (\varsigma^{=}_E([\![L]\!]))(w)$. We will show that there is a split t' of t, such that $t' \in (\mathsf{split}_\mathcal{X}(\varsigma^{=}_E)([\![\mathsf{split}_\mathcal{X}(L)]\!]))(w)$. If such a split exists, then, due to the fact that $\biguplus_\Lambda(t') = t$, we have $t \in (\biguplus_\Lambda(\mathsf{split}_\mathcal{X}(\varsigma^{=}_E)([\![\mathsf{split}_\mathcal{X}(L)]\!])))(w)$.

Since E is a non-overlapping string equality selection, we know that all the spans $t(\mathsf{x})$ with $\mathsf{x} \in \bigcup_{i=1}^{k} \mathcal{Z}_i$ are pairwise non-overlapping. However, it is not sufficient to simply create the span-tuple t' where, for every $\mathsf{x} \in \bigcup_{i=1}^{k} \mathcal{Z}_i$, $t'(\mathsf{x}^1) = t(\mathsf{x}) = [\ell, k\rangle$ and $t'(\mathsf{x}^i) = [k, k\rangle$ for every $i \in \{2, 3, \ldots, \wp\}$. In general, this definition would yield a span-tuple t' that is not a valid split of t, since t may have overlapping spans with respect to variables that are not in $\bigcup_{i=1}^{k} \mathcal{Z}_i$, i. e., for a variable $\mathsf{x} \in \mathcal{X} \setminus (\bigcup_{i=1}^{k} \mathcal{Z}_i)$ and some other variable $\mathsf{y} \in \mathcal{X}$, the spans $t(\mathsf{x})$ and $t(\mathsf{y})$ can be overlapping.

For every $\mathsf{x} \in \mathcal{X}$ with $t(\mathsf{x}) \neq \bot$, let $t(\mathsf{x}) = [\ell_\mathsf{x}, k_\mathsf{x}\rangle$. We observe that every $t(\mathsf{x})$ can be interpreted as the interval $\{\ell_\mathsf{x}, \ell_\mathsf{x} + 1, \ldots, k_\mathsf{x}\}$. A *split point of* $t(\mathsf{x})$ is any element from $\{\ell_\mathsf{x}, \ell_\mathsf{x}+1, \ldots, k_\mathsf{x}\}$. Now let us assume that, for every $\mathsf{x} \in \mathcal{X}$, we have a set $\{p_{\mathsf{x},1}, p_{\mathsf{x},2}, \ldots, p_{\mathsf{x},s_\mathsf{x}}\}$ of split points of $t(\mathsf{x})$, which satisfy the following properties.

(1) For every $\mathsf{x} \in \mathcal{X}$, $p_{\mathsf{x},1} = \ell_\mathsf{x}$, $p_{\mathsf{x},s_\mathsf{x}} = k_\mathsf{x}$ and $s_\mathsf{x} \leqslant \wp + 1$.
(2) For every $\mathsf{x}, \mathsf{y} \in \mathcal{X}$ and every $i_\mathsf{x} \in [s_\mathsf{x} - 1]$ and $i_\mathsf{y} \in [s_\mathsf{y} - 1]$, the spans $[p_{\mathsf{x},i_\mathsf{x}}, p_{\mathsf{x},i_\mathsf{x}+1}\rangle$ and $[p_{\mathsf{y},i_\mathsf{y}}, p_{\mathsf{y},i_\mathsf{y}+1}\rangle$ are non-overlapping.
(3) For every $i \in [k]$ and $\mathsf{x}, \mathsf{y} \in \mathcal{Z}_i$, $s_\mathsf{x} = s_\mathsf{y}$ and, for every $j \in [s_\mathsf{x}]$, $p_{\mathsf{x},j} - \ell_\mathsf{x} = p_{\mathsf{y},j} - \ell_\mathsf{y}$.

It can be easily seen that such split points induce a split t' of t such that $t' \in (\mathsf{split}_\mathcal{X}(\varsigma^{=}_{E})([\![\mathsf{split}_\mathcal{X}(L)]\!]))(w)$. Indeed, for every $\mathsf{x} \in \mathcal{X}$, we define $t'(\mathsf{x}^1) = [p_{\mathsf{x},1}, p_{\mathsf{x},2}\rangle$, $t'(\mathsf{x}^2) = [p_{\mathsf{x},2}, p_{\mathsf{x},3}\rangle$, …, $t'(\mathsf{x}^{s_\mathsf{x}-1}) = [p_{\mathsf{x},s_\mathsf{x}-1}, p_{\mathsf{x},s_\mathsf{x}}\rangle$, and we define $t'(\mathsf{x}^q) = [p_{\mathsf{x},s_\mathsf{x}}, p_{\mathsf{x},s_\mathsf{x}}\rangle$ for every q with $s_\mathsf{x} \leqslant q \leqslant \wp$. Since $s_\mathsf{x} \leqslant \wp + 1$ for every $\mathsf{x} \in \mathcal{X}$, this is well-defined. Due to the first property of the split points, the span-tuple t' satisfies the first condition of a split, and due to the second property of the split points, it also satisfies the second condition of a split. Thus, t' is a split of t. Since t satisfies the string-equality selection $\varsigma^{=}_{E}$, we must have $k_\mathsf{x} - \ell_\mathsf{x} = k_\mathsf{y} - \ell_\mathsf{y}$ for every $\mathsf{x}, \mathsf{y} \in \mathcal{Z}_i$ and $i \in [k]$. The fact that t satisfies the string equality selection $\varsigma^{=}_{E}$ together with the third property of the split points directly implies that t' satisfies the string-equality selection $\mathsf{split}_\mathcal{X}(\varsigma^{=}_{E})$. This means that $t' \in (\mathsf{split}_\mathcal{X}(\varsigma^{=}_{E})([\![\mathsf{split}_\mathcal{X}(L)]\!]))(w)$.

In order to complete the proof, we have to show that split points with the above properties can be created. Let us first observe that if we declare some elements of $\{1, 2, \ldots, |w| + 1\}$ to be *general split points*, then this defines a set $\{p_{\mathsf{x},1}, p_{\mathsf{x},2}, \ldots, p_{\mathsf{x},s_\mathsf{x}}\}$ of split points for every $\mathsf{x} \in \mathcal{X}$. More precisely, for every $\mathsf{x} \in \mathcal{X}$, $\{p_{\mathsf{x},1}, p_{\mathsf{x},2}, \ldots, p_{\mathsf{x},s_\mathsf{x}}\}$ is exactly the set of all general split points p that satisfy $p \in \{\ell_\mathsf{x}, \ell_\mathsf{x} + 1, \ldots, k_\mathsf{x}\}$. We declare the general split points in two steps.

- *First step*: For every $\mathsf{x} \in \mathcal{X}$ with $t(\mathsf{x}) \neq \bot$, both ℓ_x and r_x are declared general split points.
- *Second step*: Let us call the general split points added in the first step *old* general split points, and the ones to be added in this step *new* general split points. For every $i \in [k]$ we now proceed as follows.

  Let $\mathcal{Z}_i = \{\mathsf{z}_1, \mathsf{z}_2, \ldots, \mathsf{z}_{k_i}\}$. For every $j \in [k_i]$, let $\{q_{j,1}, q_{j,2}, \ldots, q_{j,s_j}\}$ be the old general split points in the interval $\{\ell_{\mathsf{z}_j}, \ell_{\mathsf{z}_j} + 1, \ldots, k_{\mathsf{z}_j}\}$, i. e., the old general split points defined in the first step that will be split points of $t(\mathsf{z}_j)$, and let $A_j = \{q_{j,1} - \ell_{\mathsf{z}_j}, q_{j,2} - \ell_{\mathsf{z}_j}, \ldots, q_{j,s_j} - \ell_{\mathsf{z}_j}\}$ be the set of these split points shifted to the left by ℓ_{z_j}. Then, we join all these sets into $B_i = \bigcup_{j=1}^{k_i} A_j$ and shift all these points back to their corresponding positions in each of the intervals $\{\ell_{\mathsf{z}_j}, \ell_{\mathsf{z}_j} + 1, \ldots, k_{\mathsf{z}_j}\}$, i. e., we declare all elements of $\{p + \ell_{\mathsf{z}_j} \mid j \in [k_i], p \in B_i\}$ to be new split points.

This concludes the definition of general split points and therefore defines, for every $\mathsf{x} \in \mathcal{X}$, a set $\{p_{\mathsf{x},1}, p_{\mathsf{x},2}, \ldots, p_{\mathsf{x},s_\mathsf{x}}\}$ of split points of $t(\mathsf{x})$. By definition, for every $\mathsf{x} \in \mathcal{X}$, $p_{\mathsf{x},1} = \ell_\mathsf{x}$ and $p_{\mathsf{x},s_\mathsf{x}} = k_\mathsf{x}$, since both ℓ_x and k_x are declared general split points in the first step.

We now estimate the total number of general split points introduced by the two steps from above. We first note that in the first step we have introduced at most $2|\mathcal{X}|$ general split points (the maximum $2|\mathcal{X}|$ is reached if $t(\mathsf{x}) \neq \bot$ for all variables $\mathsf{x} \in \mathcal{X}$). In the second step, for every $i \in [k]$, we create at most $k_i|B_i|$ new general split points, where $B_i = \bigcup_{j=1}^{k_i} A_j$. Since each A_j only contains old general split points, we know that $|A_j| \leqslant 2|\mathcal{X}|$ for every $j \in [k_i]$. Thus, $|B_i| \leqslant k_i 2|\mathcal{X}|$. This means that in the second step, we create at most $\sum_{i=1}^{k} k_i 2|\mathcal{X}| = 2|\mathcal{X}| \sum_{i=1}^{k} k_i \leqslant 2|\mathcal{X}|^2$. Finally, we conclude that the total number of general split points is at most $2|\mathcal{X}|+2|\mathcal{X}|^2 \leqslant 4|\mathcal{X}|^2$. In particular, this means that $s_{\mathsf{x}} \leqslant 4|\mathcal{X}|^2 = \wp+1$ for every $\mathsf{x} \in \mathcal{X}$. We therefore conclude that the defined split points satisfy the first property mentioned above.

That the split points satisfy the second and third property mentioned above is obvious by construction. □

6.2.4. *Core-Spanners with Non-Overlapping String-Equality Selections.* We can now plug together the previously proven lemmas in the way sketched at the beginning of Section 6 in order to obtain this section's main result. In particular, note that due to the normal form of Lemma 2.2, every core spanner (with non-overlapping string equality selections) has a representation as in the statement of the following theorem.

Theorem 6.11. *Let S be a core spanner 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 E is non-overlapping with respect to S'. There are a variable set $\mathcal{X}'$, a reference-bounded refl-spanner S'' over Σ and $\mathcal{X}'$, and a set $\Lambda \subseteq \mathcal{P}(\mathcal{X}')$ such that $S = \pi_{\mathcal{Y}} \biguplus_{\Lambda}(S'')$. Moreover, $|\mathcal{X}'| = \mathrm{O}(|\mathcal{X}|^3)$.*

Proof. Let L be a regular subword-marked language with $S' = [\![L]\!]$. Since E is non-overlapping with respect to L, Lemmas 6.8 and 6.10 imply that

$$\varsigma^{=}_{E}([\![L]\!]) \quad = \quad \Big\biguplus_{\Lambda} \mathsf{split}_{\mathcal{X}}(\varsigma^{=}_{E})([\![\mathsf{split}_{\mathcal{X}}(L)]\!]),$$

where $\Lambda = \{\lambda^{\wp}_{\mathsf{x}} \to \mathsf{x} \mid \mathsf{x} \in \mathcal{X}\}$.

Moreover, $\mathsf{split}_{\mathcal{X}}(L)$ is a subword-marked language over Σ and $\mathsf{ext}(\mathcal{X})$ such that any two variables from $\mathsf{ext}(\mathcal{X})$ are non-overlapping with respect to $\mathsf{split}_{\mathcal{X}}(L)$. This directly implies that every variable from $\mathsf{ext}(\mathcal{X})$ is simple with respect to $\mathsf{split}_{\mathcal{X}}(L)$. By applying Theorem 6.3, we can conclude that there is a reference-bounded regular ref-language L' over Σ and $\mathsf{ext}(\mathcal{X})$ with $[\![L']\!] = \mathsf{split}_{\mathcal{X}}(\varsigma^{=}_{E})([\![\mathsf{split}_{\mathcal{X}}(L)]\!])$. Hence, $S = \pi_{\mathcal{Y}} \varsigma^{=}_{E}([\![L]\!]) = \pi_{\mathcal{Y}} \biguplus_{\Lambda} \mathsf{split}_{\mathcal{X}}(\varsigma^{=}_{E})([\![\mathsf{split}_{\mathcal{X}}(L)]\!]) = \pi_{\mathcal{Y}} \biguplus_{\Lambda}([\![L']\!])$.

Finally, by Observation 6.6, we have that $|\mathsf{ext}(\mathcal{X})| = \mathrm{O}(|\mathcal{X}|^3)$. □

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 – followed by a projection, 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, followed by a projection, 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 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 that enables constant delay enumeration.

Acknowledgment

We thank the anonymous reviewers for their valuable comments. In particular, the differences between this paper and the preliminary conference version [SS21a], as described in Section 1.3, mainly go back to the reviewers' suggestions. They helped to considerably improve the paper both in terms of results and in terms of the presentation.

References

[ABMN20] Antoine Amarilli, Pierre Bourhis, Stefan Mengel, and Matthias Niewerth. Constant-delay enumeration for nondeterministic document spanners. *SIGMOD Rec.*, 49(1):25–32, 2020. doi:10.1145/3422648.3422655.

[ABMN21] Antoine Amarilli, Pierre Bourhis, Stefan Mengel, and Matthias Niewerth. Constant-delay enumeration for nondeterministic document spanners. *ACM Trans. Database Syst.*, 46(1):2:1–2:30, 2021. doi:10.1145/3436487.

[AJMR22] Antoine Amarilli, Louis Jachiet, Martin Muñoz, and Cristian Riveros. Efficient enumeration for annotated grammars. In *PODS '22: International Conference on Management of Data, Philadelphia, PA, USA, June 12 - 17, 2022*, pages 291–300, 2022. doi:10.1145/3517804.3526232.

[DBKM21] Johannes Doleschal, Noa Bratman, Benny Kimelfeld, and Wim Martens. The complexity of aggregates over extractions by regular expressions. In *24th International Conference on Database Theory, ICDT 2021, March 23-26, 2021, Nicosia, Cyprus*, pages 10:1–10:20, 2021. doi:10.4230/LIPICS.ICDT.2021.10.

[DKM+19] 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.

[DKMP20] Johannes Doleschal, Benny Kimelfeld, Wim Martens, and Liat Peterfreund. Weight annotation in information extraction. In *23rd International Conference on Database Theory, ICDT 2020, March 30-April 2, 2020, Copenhagen, Denmark*, pages 8:1–8:18, 2020. doi:10.4230/LIPIcs.ICDT.2020.8.

[FH06] Johannes Fischer and Volker Heun. Theoretical and practical improvements on the RMQ-problem, with applications to LCA and LCE. In *Combinatorial Pattern Matching, 17th Annual Symposium, CPM 2006, Barcelona, Spain, July 5-7, 2006, Proceedings*, pages 36–48, 2006.

[FH18] Dominik D. Freydenberger and Mario Holldack. Document spanners: From expressive power to decision problems. *Theory Comput. Syst.*, 62(4):854–898, 2018. URL: https://doi.org/10.1007/s00224-017-9770-0.

[FKP18] 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.

[FKRV15] 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.

[Fre19] Dominik D. Freydenberger. A logic for document spanners. *Theory Comput. Syst.*, 63(7):1679–1754, 2019. doi:10.1007/s00224-018-9874-1.

[FRU+20] Fernando Florenzano, Cristian Riveros, Martín Ugarte, Stijn Vansummeren, and Domagoj Vrgoc. Efficient enumeration algorithms for regular document spanners. *ACM Trans. Database Syst.*, 45(1):3:1–3:42, 2020. doi:10.1145/3351451.

[FS15] Henning Fernau and Markus L. Schmid. Pattern matching with variables: A multivariate complexity analysis. *Information and Computation (I&C)*, 242:287–305, 2015.

[FSV16] 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.

[FT20] 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. URL: https://doi.org/10.4230/LIPIcs.ICDT.2020.11.

[FT22] Dominik D. Freydenberger and Sam M. Thompson. Splitting spanner atoms: A tool for acyclic core spanners. In *25th International Conference on Database Theory, ICDT 2022, March 29 to April 1, 2022, Edinburgh, UK (Virtual Conference)*, pages 10:1–10:18, 2022. doi:10.4230/LIPIcs.ICDT.2022.10.

[MR23] Martin Muñoz and Cristian Riveros. Constant-delay enumeration for slp-compressed documents. In *26th International Conference on Database Theory, ICDT 2023, March 28-31, 2023, Ioannina, Greece*, pages 7:1–7:17, 2023. doi:10.4230/LIPICS.ICDT.2023.7.

[MRV17] Francisco Maturana, Cristian Riveros, and Domagoj Vrgoc. Document spanners for extracting incomplete information: Expressiveness and complexity. *CoRR*, abs/1707.00827, 2017. URL: http://arxiv.org/abs/1707.00827, arXiv:1707.00827.

[MRV18] 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.

[MS72] Albert R. Meyer and Larry J. Stockmeyer. The equivalence problem for regular expressions with squaring requires exponential space. In *13th Annual Symposium on Switching and Automata Theory, College Park, Maryland, USA, October 25-27, 1972*, pages 125–129, 1972. doi:10.1109/SWAT.1972.29.

[MS19] 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.

[Pet19] Liat Peterfreund. *The Complexity of Relational Queries over Extractions from Text*. PhD thesis, 2019. Computer science department, Technion.

[Pet21] Liat Peterfreund. Grammars for document spanners. In *24th International Conference on Database Theory, ICDT 2021, March 23-26, 2021, Nicosia, Cyprus*, pages 7:1–7:18, 2021. Extended version available at https://arxiv.org/abs/2003.06880. doi:10.4230/LIPIcs.ICDT.2021.7.

[PFKK19] 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.

[PtCFK19] 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.

[Sch16] Markus L. Schmid. Characterising REGEX languages by regular languages equipped with factor-referencing. *Information and Computation*, 249:1–17, 2016.

[SM73] Larry J. Stockmeyer and Albert R. Meyer. Word problems requiring exponential time: Preliminary report. In *Proceedings of the 5th Annual ACM Symposium on Theory of Computing, April 30 - May 2, 1973, Austin, Texas, USA*, pages 1–9, 1973. doi:10.1145/800125.804029.

[SS21a] Markus L. Schmid and Nicole Schweikardt. A purely regular approach to non-regular core spanners. In *24th International Conference on Database Theory, ICDT 2021, March 23-26, 2021, Nicosia, Cyprus*, pages 4:1–4:19, 2021. doi:10.4230/LIPIcs.ICDT.2021.4.

[SS21b] Markus L. Schmid and Nicole Schweikardt. Spanner evaluation over slp-compressed documents. In *PODS'21: Proceedings of the 40th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems, Virtual Event, China, June 20-25, 2021*, pages 153–165, 2021. doi:10.1145/3452021.3458325.

[SS22] Markus L. Schmid and Nicole Schweikardt. Query evaluation over slp-represented document databases with complex document editing. In *PODS '22: International Conference on Management of Data, Philadelphia, PA, USA, June 12 - 17, 2022*, pages 79–89, 2022. doi:10.1145/3517804.3524158.