On the Descriptional Complexity of a Diagrammatic Notation

Aidan Delaney and Gem Stapleton
Visual Modelling Group,
University of Brighton, UK
{a.j.delaney,g.e.stapleton}@brighton.ac.uk

Abstract

Spider diagrams are a widely studied, visual logic that are able to make statements about relationships between sets and their cardinalities. Various meta-level results for spider diagrams have been established, including their soundness, completeness and expressiveness. In order to further enhance our understanding of spider diagrams, we can compare them with other languages; in the case of this paper we consider star-free regular languages. We establish relationships between various fragments of the spider diagram language and certain well-known subclasses of the star-free regular class. Utilising these relationships, given any spider diagram, we provide an upper-bound on the state complexity of minimal deterministic finite automata corresponding to that spider diagram. We further demonstrate cases where this bound is tight.

1 Introduction

It is widely recognised that diagrams play an important role in various areas, including visualizing information and reasoning about that information. They are often useful for conveying (sometimes complex) information in accessible and intuitive ways. This is one reason behind the widening perception of the importance of diagrams. Traditionally in mathematics and logic, diagrams have been excluded from formal proof techniques and were considered only as a heuristic aid. Whilst some people have held the view that diagrams *cannot be formalised*, so as to be permitted when reasoning formally, it has been shown that this view is incorrect: Shin devised a sound and complete diagrammatic logic [19]. Her work is widely regarded as a seminal piece, overturning the view that diagrams could not yield a formal reasoning system. Thus, diagrams are now being recognised as a valuable tool that can be exploited in a logical setting (see [9] for an extensive discussion on the importance of diagrams in numerous reasoning contexts).

Since the work of Shin, many other diagrammatic logics have emerged. One such logic is the widely studied language of spider diagrams (see, for example [3, 7, 8, 12, 20]). With regard to applications of spider diagrams, they have been used to assist with the task of identifying component failures in safety critical hardware designs [1]. They have also been (implicitly) used for displaying the results of database queries [21], representing non-hierarchical computer file systems [2], in a visual semantic web editing environment [10, 24] and for viewing clusters which contain concepts from multiple ontologies [5]. In all of these application areas, there are other languages that could be used instead. It is, therefore, useful if we can compare spider diagrams to other languages. If we can show that, for example, spider diagrams offer significant descriptional savings over another descriptional system then practitioners may choose to represent their specifications in this more succinct form.

This paper shows that spider diagrams can be used to define languages from a particular subset of the star-free regular languages. Sections 2, 3 and 4 consider preliminaries of spider diagrams, star-free regular languages and descriptional complexity respectively. Section 5 identifies various relationships between spider diagrams and star-free regular languages and section 6 investigates descriptional complexity by providing upper bounds on the size of a deterministic finite automata accepting the language generated by a spider diagram.

2 Spider Diagrams

This section will provide a brief overview of the spider diagram syntax presented in [8]. In figure 1 the spider diagram d_1 contains two labelled *contours*, A and B. Contours are simple closed curves. The diagram also contains three minimal regions, called *zones*. There is one zone inside A, another inside B and the

other zone is outside both A and B. Each zone can be described by a two-way partition of the contour label set. The zone inside A can be described as inside A but outside B and contains two *spiders*; spiders are trees whose vertices, called *feet*, are placed in zones (in d_1, the spiders each consist of a single vertex). Spider diagrams can also contain *shading*, as in d_2 (which contains three spiders and four zones).

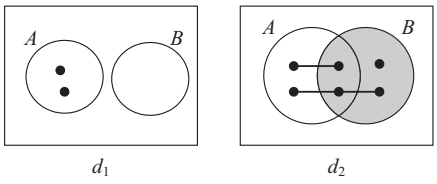


Figure 1. Two spider diagrams.

The syntax is defined at an abstract level. The contour labels in spider diagrams are selected from a finite set $\mathcal{L}$. A **zone** is defined to be a pair, (in, out), of finite disjoint subsets of $\mathcal{L}$. The set in contains the labels of the contours that the zone is inside whereas out contains the labels of the contours that the zone is outside. The set of all zones is denoted $\mathcal{Z}$. To describe the spiders in a diagram, it is sufficient to say how many spiders are placed in each region. Thus, the abstract definition of a spider diagram will specify the labels used, the zones, the shaded zones and use a set of spider identifiers to describe the spiders.

Definition 2.1. *A **unitary spider diagram**, d, is a quadruple $\langle L, Z, ShZ, SI\rangle$ where*

$L = L(d) \subseteq \mathcal{L}$ *is a set of contour labels,*

$Z = Z(d) \subseteq \{(a, L-a) : a \subseteq L\}$ *is a set of zones,*

$ShZ \subseteq Z(d)$ *is a set of shaded zones,*

$SI = SI(d) \subset \mathbb{Z}^+ \times (\mathbb{P}Z - \{\emptyset\})$ *is a finite set of spider identifiers such that for all* $(n_1, r_1), (n_2, r_2) \in SI(d)$ $(r_1 = r_2 \implies n_1 = n_2)$.

*The symbol $\perp$ is also a unitary spider diagram. We define $L(\perp) = Z(\perp) = ShZ(\perp) = SI(\perp) = \emptyset$. If d_1 and d_2 are spider diagrams then $(d_1 \vee d_2)$, $(d_1 \wedge d_2)$ and $\neg d_1$ are **compound spider diagrams**. Given a unitary diagram, d, a zone (a, b) is said to be **missing** if it is in the set $\{(a, L-a) : a \subseteq L\} - Z(d)$. If d has no missing zones then d is in **Venn form**. The set of **spiders** in d is defined to be*

$$S(d) = \{(i, r) : (n, r) \in SI(d) \wedge 1 \leq i \leq n\}.$$

*For spider (i, r), each zone in r is a **foot** of (i, r).*

So S(d) can be thought of as a bag of spiders generated by the identifiers.

By convention, we employ a lower-case d with or without subscripts to denote a unitary spider diagram. An upper case D with or without subscripts will denote an arbitrary spider diagram. The usual convention of omitting brackets where no ambiguity arises is adopted.

Our attention now turns to the semantics. Spider diagrams make statements about sets (represented by contours) and their cardinalities (by using spiders and shading). In figure 1, d_1 expresses that A and B are disjoint, because there are no points interior to both of the contours. Spiders assert the existence of elements, so d_1 specifies that there are (at least) two elements in A. The spiders in d_2 assert that there are at least three elements, one of which is in A, another is in $A \cup B$ and the third is in $B - A$. Shading is used to place upper bounds on set cardinality. For example, d_2 expresses that the set $B - A$ contains at most two elements, $A \cap B$ contains at most two elements and B contains at most three elements.

The semantics of spider diagrams are model-based. An *interpretation* is a universal set, U, together with an assignment of a subset of U to each contour (strictly, to contour labels) which is extended to interpret zones and regions. A zone, (a, b), represents the set $\bigcap_{l \in a} set(l) \cap \bigcap_{l \in b} (U - set(l))$ where $set(l)$ is the set assigned to constant label l. A set of zones, Z, represents the set which is the union of the sets represented by Z's constituent zones. An interpretation is a **model** for unitary diagram $d(\neq \perp)$ whenever

1. all of the zones which are missing represent the empty set,
2. all of the regions represent sets whose cardinality is at least the number of spiders placed entirely within that region and
3. all of the entirely shaded regions represent sets whose cardinality is at most the number of spiders with a foot in that region.

If $d = \perp$ then the interpretation is not a model for d (i.e. $\perp$ is a contradiction). The definition of a model extends to compound diagrams in the obvious (inductive) manner. The semantics are formalised in [20].

The operators $\neg$ and $\wedge$ are syntactic sugar in the spider diagram language, captured by the following theorem.

Theorem 2.1. *The language of spider diagrams is equivalent in expressive power to the fragment in which the only operator is $\vee$.*

3 Star-Free Regular Languges

The class of star-free regular languages is the set of languages which are a subset of the free monoid over an alphabet Σ including the finite languages which are closed under finite boolean operations and catenation (catenation is the product operation of the monoid) [13]. The Straubing-Thérin hierarchy describes some well studied subclasses of star-free regular languages. The more well known subclasses being the class of shuffle-ideal languages and piecewise testable languages.

A deep result by Schützenberger characterises star-free languages as those which have an aperiodic syntactic monoid [18]. Alternative characterisations of subsets of star-free languages are given later based on the shuffle product of languages. The shuffle product of two languages L_1, L_2 denoted $L_1 \sqcup L_2$ informally takes all words from L_1 and intersperses letters from all words in L_2. More formally, the words in $L_1 \sqcup L_2$ are precisely those of the form $w_0 w_1 w_2 \dots w_n$ where, for some subset $I = \{p_1, p_2, \dots, p_m\}$ of $\{1, \dots, n\}$

1. $w_{p_1} w_{p_2} \dots w_{p_m} \in L_1$ where $p_i < p_{i+1}$
2. $w_{q_1} w_{q_2} \dots w_{q_{n-m}} \in L_2$ where $\{q_1, \dots, q_{n-m}\} = \{1, \dots, n\} - I$ and $q_i < q_{i+1}$.

Considering the shuffle product and boolean operations of $\cup, \cap$ and $\subset$ gives us an immediate insight into the Straubing-Thérin catenation hierarchy. Level 0 is the set of languages $\{\Sigma^*, \emptyset\}$. Level 1/2 is the well known shuffle ideal set, which is the polynomial closure of Level 0. An alternative characterisation is that languages of catenation level 1/2 are of the form $k \sqcup \Sigma^*$ where k is a finite set of words. Level 1 is defined as the boolean closure of 1/2. This hierarchy has been extended by Pin to consider varieties of languages [14].

Spider diagrams are a monadic first order logic with equality (MOFLe). Therefore we are interested in the relationship between logics and star-free regular languages. The main body of literature discussing this relationship [6, 14, 15, 17, 22] assumes the existence of an order relation $<$ adjunct to the standard monadic first order operators of $\neg, \vee, \wedge, \iff$, the quantifiers $\exists$ and $\forall$ and predicates of the form $P_a(x)$ which states that the letter a is at positive position x in word w. Intuitively, if we do not have an order relation, $<$, as in MOFLe then any language corresponding to a formula will be closed under permutation. In other words, languages of the form, for example, $\Sigma^* A \Sigma^* B$ (A comes before B in every word) do not correspond to languages arising from formulae in MOFLe. Consequently spider diagrams do not contain facilities for ordering elements.

4 Descriptional Complexity

Descriptional complexity is concerned with the economy of representation offered by descriptional systems [11]. Descriptional systems consist of a set of finite descriptors which describe each instance of a language in a class of a languages [4]. For example, finite automata are a descriptional system which describe languages in the class of regular languages (REG). Other well-known descriptional systems include pushdown automata, linear bounded automata, Turing machines and spider diagrams.

Economy of representation is computed through the use of some metric. In the case of automata as descriptional systems, common metrics include the length of the automaton description or the number of states in the automaton. Other typical metrics include the number of productions in a grammar system or a measure of non-determinism in a system. Rabin and Scott's [16] well known result on the state complexity of deterministic and non-deterministic finite automata neatly illustrates the utility of such metrics: there exists a non-deterministic finite automaton with n states where the minimal deterministic automaton accepting the same language contains 2^n states. By convention, Rabin and Scott's result can be stated as (taken from [4])

$$\mathrm{nFA}_n \xrightarrow{states} \mathrm{dFA}_{\leq 2^n}$$

To draw conclusions on the descriptional complexity of spider diagrams we require a metric, which establishes their complexity, and a descriptional system for comparison. We have chosen the number of spiders in a diagram as the metric. This measure is independent of the actual layout of the diagram and is directly linked with diagram semantics. We establish relationships between spider diagrams and a descriptional system for regular languages. More precisely, we establish a relationship between the number of spiders in a spider diagram and the number of states in the minimal deterministic finite automaton which accepts the same language, thus comparing their descriptional complexity.

5 Relationships Between Spider Diagrams and Star-Free REG

The finite models for a spider diagram can be considered as giving rise to a star-free REG language. For example, in figure 2, the diagram d_1 has a model with universal set $U = \{1, 2\}$, with A representing the set $\{1\}$ and B representing $\{2\}$. So, in this model, there is exactly one element in $(\{A\}, \{B\})$ and exactly one

element in $(\{B\},\{A\})$ but the model does not specify any order on these elements. We can think of the zones as being the letters in a language; with this alphabet, the word $(\{A\},\{B\})(\{B\},\{A\})$ indicates that there is an element in $(\{A\},\{B\})$ which comes before an element in $(\{B\},\{A\})$; the word $(\{B\},\{A\})(\{A\},\{B\})$ indicates the existence of the same two elements, but in the opposite order. Consequently, the information provided by the two words matches the information provided by the model. The set of words in a language corresponding to d_1 contains precisely the words that arise from the finite models for d_1 in this way. We observe that isomorphic models give rise to the same words. In what follows, we formalise the *language of a spider diagram* and establish various relationships between spider diagrams and some subsets of star-free REG.

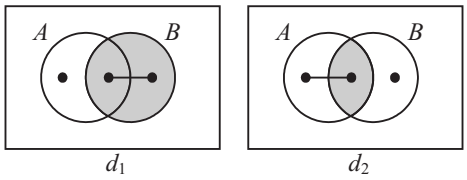


Figure 2. Words for a diagram.

To begin, we note that it can be shown that every spider diagram can be transformed into a syntactically equivalent diagram (see [8] for a set of sound and complete reasoning rules) involving only disjunctions of unitary diagrams, $D = d_1 \vee ... \vee d_n$ where each d_i has no missing zones and the same zone set. We assume, without loss of generality, that each d_i contains all of the labels in $\mathcal{L}$. Such a disjunction, D, is said to be in **disjunctive Venn form**; we call this fragment of the spider diagram language **disjunctive-SD** denoted $\mathcal{DSD}$. There is an algorithm that translates arbitrary diagrams into disjunctive Venn form (see [8] for the conversion of diagrams into *associated zone diagrams* and adapt the details in the obvious way).

To identify a star-free REG language for an arbitrary diagram, D, we can first convert D into disjunctive Venn form and then identify such a language for this disjunction. Thus, we need only identify a star-free REG language for each diagram in disjunctive Venn form. From this point forward all diagrams will be assumed to be part of the disjunctive fragment $\mathcal{DSD}$. The approach we adopt is to first identify a language for each unitary diagram and then extend to the whole of $\mathcal{DSD}$.

For example, d_2 in figure 2 can be translated into the star-free REG language that has alphabet $Z(d_2)$ and words that contain the letters (breaking into cases determined by the two-footed spider):

1. $(\{A\},\{B\})$ (for the two-footed spider) and $(\{B\},\{A\})$ (for the one-footed spider) but not $(\{A,B\},\emptyset)$ (because of the shading), or

2. $(\{A,B\},\emptyset)$ (for the two-footed spider) and $(\{B\},\{A\})$ (for the one-footed spider) but not more than one occurrence of $(\{A,B\},\emptyset)$ (because of the shading).

In general, the alphabet over which our star-free REG language is constructed is

$$\Sigma = \mathcal{Z}_{\mathcal{L}} = \{(a,b) \in \mathcal{Z} : a \cup b = \mathcal{L}\}$$

and the language generated by Σ is $\Sigma^* = \{\text{finite words over } \mathcal{Z}_{\mathcal{L}}\}$, including the empty word λ. A letter of the alphabet is simply a zone, $(\{in_1,...,in_n\},\{out_1,...,out_m\})$; informally, we may instead write $in_1...in_n\overline{out_1}...\overline{out_m}$. As a notational convenience we may also write letters of a word within square brackets, thus "$[AB][\overline{AB}]$" is a two letter word containing AB and $\overline{AB}$. We define, for unitary diagram d, $\Gamma(d) = Z(d) - ShZ(d)$ (recall that $Z(d) = \mathcal{Z}_{\mathcal{L}}$), so $\Gamma(d) \subseteq \Sigma$. Given an arbitrary diagram, D, some words in Σ^* correspond to the meaning of the diagram and the rest do not[1].

Definition 5.1. *Let w be a word in Σ^* and d $(\neq\perp)$ be a unitary diagram. The bag (or multi-set) of letters of which w consists is denoted $bag(w)$. The word w* ***conforms****, to d if and only if there exists an injection, $f\colon S(d) \to bag(w)$ satisfying*

1. *$f(s)$ is a foot of s,*

2. *f is bijective when the image is restricted to the maximal sub-bag of w whose elements are shaded zones in d.*

For $d = \perp$, no words in Σ^* conform to d.

So, w conforms to unitary diagram $d \neq \perp$ provided, for each spider, s, in d,

1. each spider in d gives rise to a letter in w by way of selecting a foot,

2. for each shaded zone, z, the number of occurrences of z in w is precisely the number of spiders whose selected foot is z.

[1] There are two special cases: when D is universally valid, all of the words in Σ^* correspond to the meaning of D and when D is a contradiction none of the words in Σ^* correspond to D.

To illustrate, in figure 2, the following are examples of words that conform to d_2:

$$[A\overline{B}][B\overline{A}],\ [B\overline{A}][A\overline{B}],\ [AB][B\overline{A}],$$
$$[B\overline{A}][A\overline{B}][AB],\ [B\overline{A}][A\overline{B}][\overline{AB}],\ [B\overline{A}][AB][A\overline{B}].$$

The word $[AB][AB][B\overline{A}]$ does not conform to d_2 because the letter $[AB]$ occurs twice (d_2 asserts that the zone $(\{A, B\}, \emptyset)$ contains at most one element).

Definition 5.2. *Let d be a unitary diagram. The* ***language of d*** *is the set of words in Σ^* that conform to d denoted $\mathscr{L}(d)$. Let $D = d_1 \vee ... \vee d_n$ be a spider diagram. The* ***language of D*** *is $\bigcup_{1 \leq i \leq n} \mathscr{L}(d_i)$.*

Definition 5.3. *Let d be a unitary diagram. We define $k(d)$ to be the set of* ***words generated by S(d)****:*

$k(d) = \{w \in \Sigma^* : w$ conforms to d where f is bijective$\}$

(taking f as in definition 5.1). For $d = \perp$, $k(d) = \emptyset$.

Lemma 5.1. *For unitary diagram d, $\mathscr{L}(d) = k(d) \sqcup \Gamma^*$ where $\Gamma = \Gamma(d)$.*

Proof. $\mathscr{L}(d)$ is of the form $\Gamma^* a_1 \Gamma^* a_2 \Gamma^* \dots \Gamma^* a_{|S(d)|} \Gamma^*$ where $a_1 a_2 \dots a_{|S(d)|} \in k(d)$. □

Corollary 5.1. *Let d be a unitary diagram such that $ShZ(d) = \emptyset$. Then $k(d) \sqcup \Gamma(d)^*$ is a shuffle-ideal.*

Proof. In this case, $\Gamma(d)^* = \Sigma^*$. □

Let $\mathcal{DSD}_{\mathcal{NS}}$ be the class of spider diagrams in $\mathcal{DSD}$ that do not contain shading.

Theorem 5.1. *Let $D = d_1 \vee ... \vee d_n \in \mathcal{DSD}_{\mathcal{NS}}$. Then $\mathscr{L}(D)$ is a shuffle-ideal.*

Proof. $\mathscr{L}(D) = \bigcup_{1 \leq i \leq n} \mathscr{L}(d_i)$ is a shuffle-ideal because the class of shuffle-ideals is closed under union. □

Theorem 5.2. *The class $\bigcup_{D \in \mathcal{DSD}_{NS}} \{\mathscr{L}(D)\}$ is a strict subclass of the class of shuffle-ideal languages.*

Proof. By theorem 5.2, $\bigcup_{D \in \mathcal{DSD}_{NS}} \{\mathscr{L}(D)\}$ is a class of shuffle-ideals. For strictness, we start by observing that, for each unitary diagram $d \in \mathcal{DSD}_{\mathcal{NS}}$, the language of d is closed under permutation of words. Therefore, the language of any diagram in $\mathcal{DSD}_{NS}$ is closed under permutation of words. However, the shuffle-ideal $l = \{z_1 z_2\} \sqcup \Sigma^*$, where $z_1, z_2 \in Z(d)$ are two distinct zones, is not closed under permutation of words since, for example, the word $z_1 z_2$ is in l but $z_2 z_1$ is not in l. Consequently, l is not the language of any diagram in $\mathcal{DSD}_{NS}$. □

There are compound diagrams whose languages are not equal to the language of any unitary diagram, captured by the following lemma.

Lemma 5.2.

$$\bigcup_{\{d \in \mathcal{DSD}: d \text{ is unitary}\}} \{\mathscr{L}(d)\} \subset \bigcup_{D \in \mathcal{DSD}} \{\mathscr{L}(D)\}.$$

Proof. (Sketch) Taking $D = d_1 \vee d_2$, where d_1 and d_2 are shown in figure 3, provides an example of a compound diagram whose language, $\mathscr{L}(D)$, is not the language for any unitary diagram. □

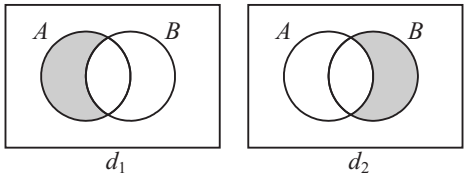


Figure 3. Classes of languages.

Theorem 5.3. *The class $\bigcup_{D \in \mathcal{DSD}} \{\mathscr{L}(D)\}$ is a strict subset of languages of catenation order 3/2.*

Proof. (Sketch) From [14] we know that sets of level 3/2 are finite unions of sets of the form $\Gamma_0^* a_1 \Gamma_1^* a_2 \Gamma_2^* \dots a_j \Gamma_j^*$ where $a_1, a_2, \dots, a_j \in \Sigma$ and $\Gamma_0, \Gamma_1, \dots, \Gamma_j \subseteq \Sigma$. From lemma 5.1 and lemma 5.2 we can see that $w \in \mathscr{L}(D)$ is of a similar, but restricted, form where $\Gamma_0 = \Gamma_1 = \dots = \Gamma_j \subseteq \Sigma$. □

Corollary 5.2. *The set of shuffle-ideal languages is not a subset of $\bigcup_{D \in \mathcal{DSD}} \{\mathscr{L}(D)\}$.*

Proof. From theorem 5.2 we know that there exists l such that $l \notin \bigcup_{D \in \mathcal{DSD}_{NS}} \{\mathscr{L}(D)\}$ but is a shuffle-ideal. The argument extends to show that $l \notin \bigcup_{D \in \mathcal{DSD}} \{\mathscr{L}(D)\}$. □

Corollary 5.3. *The class $\bigcup_{\{d \in \mathcal{DSD}: d \text{ is unitary}\}} \{\mathscr{L}(d)\}$ is not a subset of the class of shuffle-ideal languages.*

Proof. Let d be a unitary diagram in $\mathcal{DSD}$ such that $ShZ(d) \neq \emptyset$. Then $\Gamma(d) \subset \Sigma$. A simple application of the `TestShuffle` [6] to the automaton accepting $\mathscr{L}(d) = k(d) \sqcup \Gamma(d)^*$ verifies that $\bigcup_{D \in \mathcal{DSD}} \{\mathscr{L}(D)\}$ contains languages outside the class of shuffle ideal languages. □

The spider diagram in figure 4 illustrates the relationships between various classes of languages, established in this section, where DSD denotes the class $\bigcup_{D \in \mathcal{DSD}} \{\mathscr{L}(D)\}$ and DSD_{NS} denotes $\bigcup_{D \in \mathcal{DSD}_{NS}} \{\mathscr{L}(D)\}$.

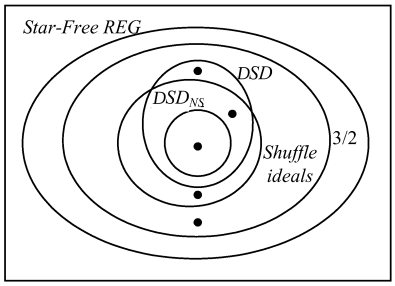


Figure 4. Relationships between languages.

6 The Descriptional Complexity of Spider Diagrams

For any given spider diagram, a regular language arises from the finite models, as demonstrated in the previous section. We now provide upper bounds on the number of states required in the minimal deterministic finite automaton accepting the language that arises from these models of a spider diagram, and show that our upper bound is exact in some cases. Given a unitary diagram d we will establish an upper bound on $|k(d)|$ in terms of the number of spiders in d. When $|k(d)|$ meets this upper bound, we establish the exact minimal number of states required in a finite state machine.

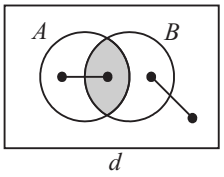


Figure 5. Large $k(d)$.

As an illustration, d in figure 5 has

$$k(d) = \{[A\overline{B}][B\overline{A}], [B\overline{A}][A\overline{B}], [A\overline{B}][\overline{AB}], [\overline{AB}][A\overline{B}], [AB][B\overline{A}], [B\overline{A}][AB], [AB][\overline{AB}], [\overline{AB}][AB]\}.$$

Given that there are two spiders, s_1 and s_2 say, in d, each with two feet, the largest number of words that can be in $k(d)$ is 8: suppose a word, w, has first letter which arises from s_1 and second letter from s_2, then there are $2 \times 2 = 4$ possible choices for w; since $k(d)$ is closed under permutation of words, every permutation of w must also be in $k(d)$, giving $4 \times 2 = 8$ words in total. Since the two spiders in d are placed in disjoint regions, every such permutation gives rise to a distinct word. Hence there are 8 words in $k(d)$.

Lemma 6.1. *Given a unitary diagram d*

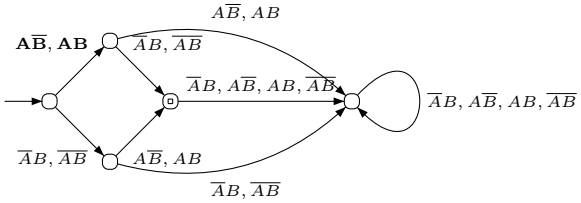
$$|k(d)| \leq |S(d)|! \times \prod_{(i,r) \in S(d)} |r|.$$

Given a unitary diagram d with set of words, $k(d)$, satisfying $|k(d)| = |S(d)|! \times \prod_{(i,r) \in S(d)} |r|$, we can construct a minimal finite automaton accepting precisely the words in $k(d)$. If d is entirely shaded, then this automaton accepts the language of d (since, in such a case, $\mathscr{L}(d) = k(d)$). Alternatively, the automaton can be trivially modified (adding loops, removing edges and possibly removing the sink state) to accept $\mathscr{L}(d)$. The number of states remains the same or reduces by 1 (the sink state is necessary only when at least one zone is shaded or $d = \perp$). For example, considering d in figure 5, a minimal finite automaton accepting precisely the words in $k(d)$ is $\mathcal{A}(k(d))$ in figure 6(a). To convert $\mathcal{A}(k(d))$ to a finite state automaton accepting $\mathscr{L}(d)$, we add loops and remove edges, enlarging the set of words accepted, in the obvious way (see figure 6(b)). Given any unitary diagram d we define $\mathcal{A}(k(d))$ to be the minimal automaton accepting precicely the words in $\mathscr{L}(k(d))$.

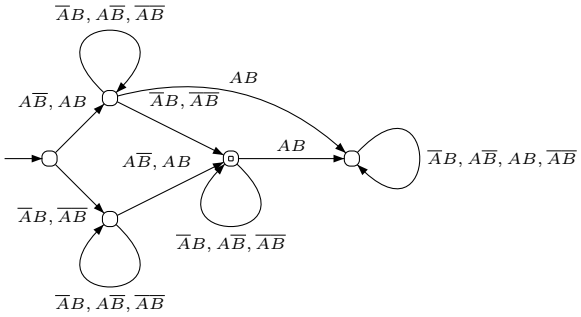
The bold typeface transition label in figure 6(a) represents two transitions: a transition from the start state to a state on the letter $A\overline{B}$ and another transition between the same two states on the letter AB. Each tranistion occurs on a letter that arises from a unique spider foot and is labeled with the appropriate letter. We allow the commonly accepted transition label shorthand in figure 6 by defining the function $l(s_i), s_i = (i, r) \in S(d)$ which returns a label consisting of the zones in r.

Theorem 6.1. *Let d be a unitary diagram with $|k(d)| = |S(d)|! \times \prod_{(i,r) \in S(d)} |r|$. The minimal complete automaton accepting precisely the words in $k(d)$ has $2^{|S(d)|} + 1$ states.*

Proof. (Sketch)We prove this by induction over the size of $S(d)$. In the base case, seen in figure 7(a), we accept all one-letter words corresponding to the single spider s in $S(d), |S(d)| = 1$. The cases where $|S(d)| = 2$ and $|S(d)| = 3$ are presented in figure 7. We assume that

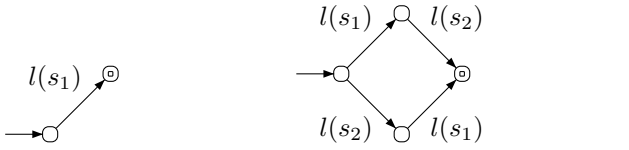


(a) Automaton accepting $k(d)$

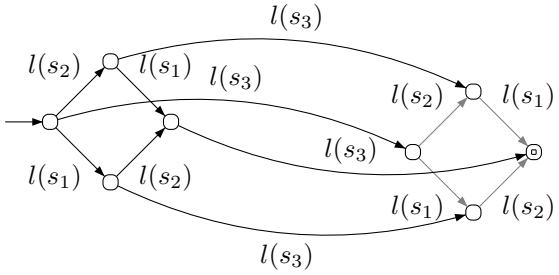


(b) Automaton accepting $\mathscr{L}(d)$

Figure 6. Constructing a finite state machine.



(a) $k(d)$ where $S(d) = \{s_1\}$ (b) $k(d)$ where $S(d) = \{s_1, s_2\}$



(c) $k(d)$ where $S(d) = \{s_1, s_2, s_3\}$

Figure 7. Constructing an automaton accepting *k(d)*.

where $|S(d)| = n$ the size of the automaton is 2^n. We may construct the automaton for the $n+1$ case by duplicating the automaton for the n case and "shifting" the duplicated structure by one letter. The inductive step is visible in the examples 7(a) through 7(c). To complete the automaton we add a sink state q_{sink} as the target of all missing transitions. The proof of minimality is omitted. □

Theorem 6.2. *Let d be a unitary diagram such that $|k(d)| = |S(d)|! \times \prod_{(i,r)\in S(d)} |r|$. The minimal automaton accepting $\mathscr{L}(d) = k(d) \sqcup \Gamma(d)^*$ has exatly $2^{|S(d)|} + 1$ states provided $\Gamma(d) \subset \Sigma$. Alternatively the minimal automaton has $2^{|S(d)|}$ (when $\Gamma(d) = \Sigma$).*

Proof. By theorem 6.1 $\mathcal{A}(k(d))$ has $2^{|S(d)|} + 1$ states. Intuitively we remove all transitions from $\mathcal{A}(k(d))$ either sourced on or targetted at q_{sink}. We add transitions from each state s of $\mathcal{A}(k(d))$ as follows. Let σ be a letter not labelling any transition sourced on s. If σ is not a shaded zone, label a transition from s to itself with σ. Otherwise, label a transition from s to q_{sink} with σ. □

This construction process is demonstrated in figure 6.

Corollary 6.1. *Let d be a unitary diagram. The minimal automaton accepting $\mathscr{L}(d)$ has at most $2^{|S(d)|} + 1$ states:*

$$d_n \xrightarrow{spiders,states} \mathrm{dFA}_{\leq 2^n+1}$$

Theorem 6.3. *Let $D = d_1 \vee \ldots \vee d_m$ be a spider diagram. The minimal automaton accepting $\mathscr{L}(D)$ has at most $2^{\sum_{i=1}^{m} |S(d_i)|} + 1$ states.*

$$D_{\sum_{i=1}^{m} |S(d_i)|} \xrightarrow{spiders,states} \mathrm{dFA}_{\leq 2^{\sum_{i=1}^{m} |S(d_i)|}+1}$$

Proof. By the well known upper-bound [23] for the union of m deterministic finite state machines $\mathcal{A}_1, \mathcal{A}_2, \ldots, \mathcal{A}_m$ is of order $O(\prod_{i=1}^{m}(|\mathcal{A}_i|))$. This gives us an upper bound for the size of $\mathcal{A}(\mathscr{L}(D))$ where, by corollary 6.1, $\mathcal{A}_1$ accepts $\mathscr{L}(d_1)$, $\mathcal{A}_2$ accepts $\mathscr{L}(d_2)\ldots$ and $\mathcal{A}_m$ accepts $\mathscr{L}(d_m)$. □

7 Conclusion

We have established relationships between spider diagrams and various subsets of star-free REG (see figure 4 for a summary). These results show that it is possible to use spider diagrams to visualise languages drawn from these various subclasses.

We have also demonstrated economy of representation offered by spider diagrams over deterministic finite state machines: in some cases a unitary spider diagram containing n spiders is equivalent to a finite state machine containing $2^n + 1$ states, for example. The in-

tuition behind the economy of representation that spider diagrams bring over deterministic finite state machines is that, to not specify an order on letters using deterministic FAs many states are required (ensuring one path exists from the start state to the final state for each ordering of the letters) whereas spider diagrams achieve this naturally. Deterministic FAs are more expressive than spider diagrams (partly due to being able to specify an ordering) but their restrictive syntax forces us to use many states when we do not wish to specify any ordering.

Acknowledgement Gem Stapleton is supported by a Leverhulme Trust Early Career Fellowship and by UK EPSRC grant EP/E011160/1 for the Visualization with Euler Diagrams project.

References

[1] R. Clark. Failure mode modular de-composition using spider diagrams. In *Proc. of Euler Diagrams 2004*, volume 134 of *Electronic Notes in Theoretical Computer Science*, pages 19–31, 2005.

[2] R. DeChiara, U. Erra, and V. Scarano. VennFS: A Venn diagram file manager. In *Proc. of Information Visualisation*, pages 120–126. IEEE, 2003.

[3] J. Flower, J. Masthoff, and G. Stapleton. Generating readable proofs: A heuristic approach to theorem proving with spider diagrams. In *Proc. of 3rd Intl. Conf. on the Theory and Application of Diagrams*, volume 2980 of *LNAI*, pages 166–181. Springer, 2004.

[4] J. Goldstine, M. Kappes, C. M. Kintala, H. Leung, A. Malcher, and D. Wotschke. Descriptional complexity of machines with limited resources. *J. of Universal Computer Science*, 2002.

[5] P. Hayes, T. Eskridge, R. Saavedra, T. Reichherzer, M. Mehrotra, and D. Bobrovnikoff. Collaborative knowledge capture in ontologies. In *Proc. 3rd Intl. Conf. on Knowledge Capture*, pages 99–106, 2005.

[6] P.-C. Héam. On shuffle ideals. *Theoretical Informatics Applications*, 36:359–384, 2002.

[7] J. Howse, F. Molina, J. Taylor, S. Kent, and J. Gil. Spider diagrams: A diagrammatic reasoning system. *J. of Visual Languages and Computing*, 12(3):299–324, 2001.

[8] J. Howse, G. Stapleton, and J. Taylor. Spider diagrams. *LMS J. of Computation and Mathematics*, 8:145–194, 2005.

[9] J. Larkin and H. Simon. Why a diagram is (sometimes) worth ten thousand words. *J. of Cognitive Science*, 11:65–99, 1987.

[10] J. Lovdahl. *Towards a Visual Editing Environment for the Languages of the Semantic Web*. PhD thesis, Linkoping University, 2002.

[11] A. Meyer and M. Fischer. Economy of description by automata, grammars, and formal systems. In *Proc. 12th Annual Symposium on Switching and Automata Theory*, pages 188–191, 1971.

[12] O. Patrascoiu, S. Thompson, and P. Rodgers. Tableaux for diagrammatic reasoning. In *Proc. 2005 Intl. Workshop on Visual Languages and Computing*, pages 279–286. Knowledge Systems Institute, 2005.

[13] J.-E. Pin. Finite semigroups and recognizable languages: an introduction. In *NATO Advanced Study Institute: Semigroups, Formal Languages and Groups*, pages 1–32. Kluwer Academic Publishers, 1995.

[14] J.-E. Pin. *Syntactic semigroups*, pages 679–746. Springer-Verlag New York, Inc., 1997.

[15] J.-E. Pin and P. Weil. Polynomial closure and unambiguous product. *Theoretical Computer Science*, 30:1–39, 1997.

[16] M. O. Rabin and D. Scott. Finite automata and their decision problems. *IBM J. of Research Development*, pages 114–125, 1959.

[17] A. Rabinovich. Star free expressions over the reals. *Theor. Comput. Sci.*, 233(1–2):233–245, 2000.

[18] M. Schtzenberger. On finite monoids having only trivial subgroups. *Information and Control*, pages 190–194, 1965.

[19] S.-J. Shin. *The Logical Status of Diagrams*. Cambridge University Press, 1994.

[20] G. Stapleton, S. Thompson, J. Howse, and J. Taylor. The expressiveness of spider diagrams. *J. of Logic and Computation*, 14(6):857–880, 2004.

[21] J. Thièvre, M. Viaud, and A. Verroust-Blondet. Using Euler diagrams in traditional library environments. In *Euler Diagrams 2004*, volume 134 of *ENTCS*, pages 189–202. ENTCS, 2005.

[22] W. Thomas. *Languages, automata, and logic*, pages 389–455. Springer-Verlag New York, Inc., 1997.

[23] S. Yu. State complexity of regular languages. In *IWD-CAGRS: Proc. Intl. Workshop on Descriptional Complexity of Automata, Grammars and Related Structures*, 1999.

[24] Y. Zhao and J. Lövdahl. A reuse based method of developing the ontology for e-procurement. In *Proc. Nordic Confernce on Web Services*, pages 101–112, 2003.