Journal of Visual Languages and Computing 24 (2013) 273–288

Contents lists available at SciVerse ScienceDirect

Journal of Visual Languages and Computing

journal homepage: www.elsevier.com/locate/jvlc

On the expressiveness of spider diagrams and commutative star-free regular languages

Aidan Delaney [a,*], Gem Stapleton [b], John Taylor [a], Simon Thompson [b]

[a] Visual Modelling Group, University of Brighton, UK
[b] School of Computing, University of Kent, UK

ARTICLE INFO

Article history:
Received 22 September 2010
Received in revised form
3 December 2012
Accepted 14 February 2013
This paper has been recommended for acceptance by S.-K. Chang
Available online 27 February 2013

Keywords:
Spider diagrams
Diagrammatic logics
Regular languages
Expressiveness
Star-free languages

ABSTRACT

Spider diagrams provide a visual logic to express relations between sets and their elements, extending the expressiveness of Venn diagrams. Sound and complete inference systems for spider diagrams have been developed and it is known that they are equivalent in expressive power to monadic first-order logic with equality, MFOL[=]. In this paper, we further characterize their expressiveness by articulating a link between them and formal languages. First, we establish that spider diagrams define precisely the languages that are finite unions of languages of the form $K\sqcup\!\sqcup\Gamma^*$, where K is a finite commutative language and Γ is a finite set of letters. We note that it was previously established that spider diagrams define commutative star-free languages. As a corollary, all languages of the form $K\sqcup\!\sqcup\Gamma^*$ are commutative star-free languages. We further demonstrate that every commutative star-free language is also such a finite union. In summary, we establish that spider diagrams define precisely: (a) languages definable in MFOL[=], (b) the commutative star-free regular languages, and (c) finite unions of the form $K\sqcup\!\sqcup\Gamma^*$, as just described.

1. Introduction

Spider diagrams [7,11] provide a visual notation for finite sets, their members and interrelationships. An example spider diagram is shown in Fig. 1. They are given a model theoretic semantics, starting with a finite universal set, U, and interpreting each closed curve, called a *contour*, as a subset of U. Each minimal area (i.e. a region not further subdivided by any segment of a contour), called a *zone*, represents an intersection of sets and their complements. Between them, the zones represent the universal set: the union of the sets represented by the zones is U. For instance, in Fig. 1, there are two contours, representing the sets P and Q, along with four zones. The zone inside the contour P but outside the contour Q represents the set $P \cap \overline{Q}$. Each dot (or set of joined dots), called a *spider*, is interpreted as a distinct element of the universe belonging to the appropriate set. For example, the spider comprising a single dot in Fig. 1 tells us that there is an element in the set $P \cap \overline{Q}$ and the spider comprising two dots tells us that there is another element which is in the set $P \cap \overline{Q}$ or in the set $\overline{P} \cap \overline{Q}$; the line connecting the two dots represents disjunction. Finally, the set represented by a shaded zone can only include elements represented by spiders with a dot in that zone. In Fig. 1, therefore, the zone inside P but outside Q contains at most two elements, since there are two dots in this zone. This diagram is an example of a *unitary* spider diagram. More complex spider diagrams are formed by using logical operators, such as $\wedge$ and $\vee$. Spider diagrams also have an associated sound and complete reasoning system [20] and, in [11], Stapleton et al. showed that spider diagrams are equally as expressive as monadic first-order logic with

* Corresponding author. Tel.: +44 1273 642455.
E-mail addresses: a.j.delaney@brighton.ac.uk (A. Delaney), g.e.stapleton@brighton.ac.uk (G. Stapleton), john.taylor@brighton.ac.uk (J. Taylor), s.j.thompson@kent.ac.uk (S. Thompson).

1045-926X/$ - see front matter © 2013 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.jvlc.2013.02.001

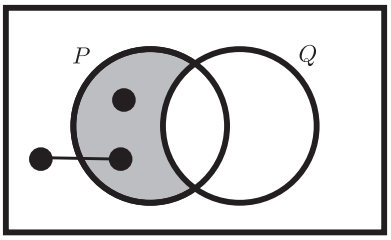

Fig. 1. A spider diagram.

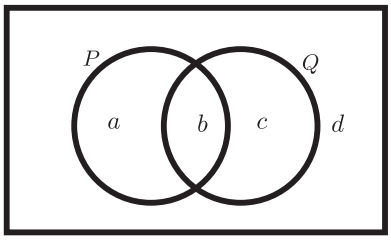

Fig. 2. An assignment of letters to zones.

equality, MFOL[=]. To illustrate this result, the spider diagram in Fig. 1 is equivalent to the MFOL[=] sentence

$$\exists x_1 \exists x_2 (P(x_1) \wedge \neg Q(x_1) \wedge \neg Q(x_2) \wedge x_1 \neq x_2 \wedge \forall y((P(y) \wedge \neg Q(y)) \Rightarrow (y = x_1 \vee y = x_2))).$$

In this paper, we present a novel approach to the study of spider diagrams, through examining their relationship with regular languages. Regular languages lie at the heart of theoretical computer science. Much is known about how they relate to finite automata, symbolic logic, and algebraic formalisms. Each of these relationships gives a different insight into regular languages as well as illuminating the other areas themselves. As with earlier work, our study of spider diagrams with regular languages provides insights into both regular languages and diagrammatic logic. For instance, we can now determine whether two spider diagrams are semantically equivalent by establishing whether they define the same language; two languages are equal if the minimal automata that accept them are the same.

We now explain how spider diagrams are associated with languages. The first step assigns sets of letters to contours, so that each zone corresponds to a single letter. If we have contours labelled P and Q, as in Fig. 1, and alphabet $\Sigma = \{a,b,c,d\}$ then we can assign $\{a,b\}$ to P and $\{b,c\}$ to Q. This induces an assignment of the letters to zones as follows:

1. a is assigned to the zone that is inside the contour P but outside the contour Q,
2. b is assigned to the zone that is inside both contours P and Q,
3. c is assigned to the zone that is inside the contour Q but outside the contour P, and
4. d is assigned to the zone that is outside both contours P and Q.

This assignment of letters to zones is illustrated in Fig. 2. Using this assignment, we can use spider diagrams to define languages by considering the information provided by the diagram. The presence of a spider in a diagram corresponds to the presence of a letter in a word. For instance, in Fig. 1, the spider comprising a single dot inside P but outside Q tells us that words must contain an a, and the other spider tells us that words must contain either an a (in addition to that present because of the first spider) or a d (because of the dot outside both contours). The disjunctive information arises from the fact that this spider comprises two dots connected by a line; the line represents disjunction. Thus, all words in the language defined by this spider diagram must contain one of the words aa, ad and da as a scattered subword (defined in Section 3) of a. The shading provides an upper bound on the number of occurrences of letters in words: all of the letters that are assigned to shaded zones must be represented by spiders. So, in Fig. 1, the shading tells us that the only a letters arise from spiders because the shaded zone is assigned the letter a. Apart from the restriction on the number of as, any other letters can be present. Thus, this spider diagram defines the language $\{aa,ad,da\} \sqcup\!\sqcup \{b,c,d\}^*$; this is the *shuffle product* of $\{aa,ad,da\}$ and $\{b,c,d\}^*$ which comprises of all words formed by interspersing the letters of words in $\{aa,ad,da\}$ with words in $\{b,c,d\}^*$. Of note is that spider diagrams cannot assert any ordering information between the letters of a word, so they define only commutative languages.

We connect our work to Thomas' definition of a language definable by a sentence in MFOL[$<$] [22]. Thomas proves that the star-free regular languages, including those which are not commutative, are precisely those definable in monadic first-order logic of order (MFOL[$<$]), in which the only binary predicate is $<$, interpreted as strict total order; the requirement for a strict total order arises from the fact that languages definable in MFOL[$<$] need not be commutative, so $<$ is necessary when placing constraints on the order of letters. The notion of when a MFOL[$<$] sentence defines a language requires a correspondence to be defined between monadic predicate symbols and sets of letters, just as we demonstrated when linking spider diagrams to languages in our example above by assigning sets of letters to contour labels. To illustrate, using the same example alphabet $\Sigma = \{a,b,c,d\}$, we assign the set $\{a,b\}$ to the predicate symbol P and the set $\{b,c\}$ to the predicate symbol Q, just as we assigned these sets to contours above. In this case, the MFOL[$<$] sentence

$$\exists x(P(x) \wedge \forall y(y \neq x \Rightarrow x < y))$$

defines the language consisting of all words that begin with a letter a or a letter b; intuitively, there is a letter ($\exists x$) that is in the set $\{a,b\}$ (since $P(x)$ holds and P is assigned $\{a,b\}$) that comes before every other letter (since $\forall y(y \neq x \Rightarrow x < y)$). Since this language is not commutative, it is not definable by a spider diagram or in MFOL[=].

In light of the observation that spider diagrams define commutative languages, spider diagrams of order were proposed [4], which are expressively equivalent to MFOL[$<$] [5], and therefore define all star-free regular

languages. In this paper, we prove that the following four statements are equivalent, where Σ is the alphabet:

1. L is the language of a spider diagram,
2. L is defined by a sentence in monadic first-order logic with equality, MFOL[=],
3. L is a commutative star-free regular language, and
4. L is a finite union of languages of the form $K \sqcup\!\sqcup \Gamma^*$ where K is a finite commutative language and $\Gamma \subseteq \Sigma$.

Thus, the work reported here complements Thomas' by showing that the commutative languages are precisely those definable in MFOL[=].

Section 2 illustrates our approach to prove that spider diagrams define precisely the commutative star-free regular languages by working through a set of representative examples. Section 3 presents our notation for discussing finite automata and regular languages. In Section 4 we define the syntax and semantics of spider diagrams. Section 5 formalises when an interpretation is a model for a word. Section 6 defines the language of a spider diagram and proves that the language is a finite union of languages of the form $K \sqcup\!\sqcup \Gamma^*$ as just described. Section 7 proceeds to derive our characterisation of commutative star-free languages as finite unions of shuffle products as previously described.

2. Illustrating our approach

To prove our main results, a key step requires us to prove that spider diagrams can define all star-free regular languages. To do this, we first characterise the set of languages defined by spider diagrams in terms of shuffle products; the shuffle products that correspond to spider diagrams have a particular form, $K \sqcup\!\sqcup \Gamma^*$, where K is a finite commutative language and Γ is a set of letters. Subsequently, by examining finite automata, we show that every commutative star-free regular language can be written as a finite union of shuffle products of the required form. We are then able to convert shuffle products in this form into spider diagrams.

Here, we work through a set of examples to illustrate the approach; the formal definitions of the concepts used will be given later in the paper. First, we provide examples of spider diagrams and derive the language they define in terms of shuffle products, following the strategy used in Section 6. We further illustrate how we convert such shuffle products into spider diagrams. Second, we provide an example of a commutative, star-free regular language and demonstrate how we identify a spider diagram defining the language via automata, following the strategy used in Section 7. Later in the paper, we make reference back to these examples where appropriate.

In the examples of this section, we take the alphabet to be $\Sigma = \{a,b,c,d\}$ and we assume the contour labels (analogous to predicate symbols in symbolic logic) are in the set $\mathcal{C} = \{P,Q\}$. Further, we assign the set of letters $\{a,b\}$ to P and $\{b,c\}$ to Q.

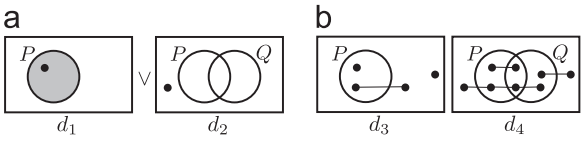

Fig. 3. Spider diagrams defining languages.

2.1. Converting spider diagrams to shuffle products

Consider the spider diagram $d_1 \vee d_2$ in Fig. 3a. This diagram is a disjunction of the so-called *unitary diagrams*; these are diagrams that do not involve any logical connectives. The diagram d_1 expresses that there is an element in the set P, by the use of the spider placed inside the contour labelled P. In addition, the shading tells us that there can be no other elements in P: shading asserts that all elements in the sets represented by shaded zones must be represented by spiders. The diagram d_2 expresses that there is an element that is not in P or Q.

In language terms, the diagram d_1 defines $L = \{a,b\}\sqcup\!\sqcup \{c,d\}^*$, since:

1. there is a spider inside the contour labelled P (assigned the set of letters $\{a,b\}$) so there must be an a or a b present in words of L; this gives rise to $\{a,b\}$ in the shuffle product and
2. there is shading inside the contour labelled P, so the only a or b letters present must be those arising from the spiders, and the lack of shading elsewhere means that we can have any number (including 0) occurrences of c and d; this gives rise to $\{c,d\}^*$ in the shuffle product.

We denote $\{a,b\}$ in the shuffle product by $K(d_1)$ and $\{c,d\}$ in the shuffle product by $\Gamma(d_1)$, so $L = K(d_1)\sqcup\!\sqcup \Gamma(d_1)^*$. The diagram d_2 similarly defines the language $K(d_2)\sqcup\!\sqcup \Gamma(d_2)^*$ where $K(d_2) = \{d\}$ and $\Gamma(d_2) = \{a,b,c,d\} = \Sigma$. Hence, the diagram $d_1 \vee d_2$ defines the language

$$(K(d_1)\sqcup\!\sqcup \Gamma(d_1)^*) \cup (K(d_2)\sqcup\!\sqcup \Gamma(d_2)^*).$$

Whilst in the example just given it was easy to identify the sets K and Γ for each unitary diagram, in more complex examples it is not so straightforward. Complexity arises because of the disjunctive information provided by spiders. In the previous example, the spider in d_1 was taken to represent either an a or a b since it was inside P; because this is the only spider in d_1, we do not have to worry about how the letters arising from this spider interact with letters arising from other spiders. When there are many spiders present, we must ensure that the set K fully represents the different possibilities for the letters arising from each spider.

For instance, the diagram d_3 in Fig. 3b contains three spiders, each of which provides disjunctive information. Moreover, each spider represents a different element, in set theory terms, and in language terms each spider gives rise to a letter in each word of the language defined by d_3. Of note here is that the spider comprising two dots represents an element which is either in P (the left most

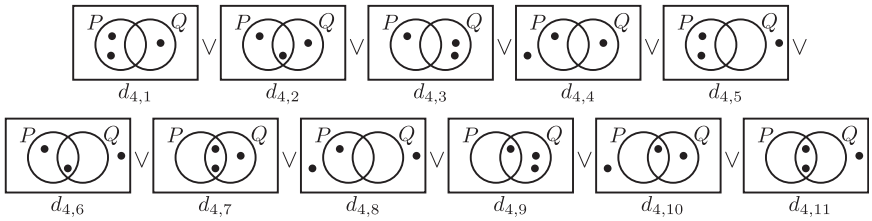

Fig. 4. Transforming to disjunctive normal form.

node) or not in P (the right most node). The fact that there are three spiders tells us that there are at least three letters in each word, namely:

1. an a or a b, arising from the spider inside P,
2. a c or a d, arising from the spider outside P, and
3. an a, a b, a c or a d, arising from the spider that comprises two dots.

Forming the set $K(d_3)$ is, therefore, not so straightforward: we need to pick one letter from each spider and arrange them in all possible orders. Then, we need to pick a different combination of letters from the spiders and arrange them in all possible orders, and repeat this process until we cannot form any more words. To simplify the details, we use the known result that tells us each spider diagram can be converted into a disjunction of unitary spider diagrams where all of the contour labels are present and each spider has just one node [11]. In the case of d_3, this diagram is obtained by first adding Q, shown in d_4, and then 'splitting' the spiders in d_4 so that they have only single dots, shown in Fig. 4.

Adding contours followed by splitting spiders means that all of the information provided by the spiders is now represented in *disjunctive normal form*[1] (DNF), from which we can easily derive the sets K and Γ, one of each from each unitary diagram. Here, we have $\Gamma(d_{4,i}) = \{a,b,c,d\} = \Sigma$ in each case (because there is no shading and there is no bound on the number of occurrences of any letter, so all letters are in $\Gamma(d_{4,i})$). To form each $K(d_{4,i})$, we start by mapping each spider to the letter corresponding to the zone in which it is placed. For instance, we define $\kappa : S(d_{4,1}) \to \Sigma$, where $S(d_{4,1}) = \{s_1,s_2,s_3\}$ is the set of spiders in $d_{4,1}$, by $\kappa(s_1) = \kappa(s_2) = a$, where s_1 and s_2 are the spiders inside P but outside Q and the letter a is assigned to P but not Q, and $\kappa(s_3) = c$, where s_3 is the spider inside Q but outside P and the letter c is assigned to Q but not P. We can now form $K(d_{4,1})$:

$$K(d_{4,1}) = \{\kappa(s_1)\} \sqcup\!\sqcup \{\kappa(s_2)\} \sqcup\!\sqcup \{\kappa(s_3)\} = \{a\} \sqcup\!\sqcup \{a\} \sqcup\!\sqcup \{c\}$$
$$= \{aac,aca,caa\}.$$

Each $K(d_{4,i})$ is similarly formed; equivalently, $K(d_{4,i})$ is the commutative closure of a word formed by ordering the letters arising from the spiders in any order. The language of d_3, denoted $\mathcal{L}(d_3)$, is given by the union of the languages

[1] Recall, a statement is in disjunctive normal form provided it is a disjunction of conjunctions of literals.

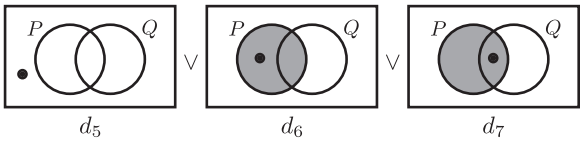

Fig. 5. Translating shuffle products to diagrams.

defined by each $d_{4,i}$, denoted $\mathcal{L}(d_{4,i})$, so

$$\mathcal{L}(d_3) = (K(d_{4,1}) \sqcup\!\sqcup \Gamma(d_{4,1})^*) \cup \cdots \cup (K(d_{4,11}) \sqcup\!\sqcup \Gamma(d_{4,11})^*).$$

We prove that spider diagrams define languages of the form above (Theorem 4 in Section 6).

In fact, we also prove that any language which can be written as a finite union of languages of the form $K \sqcup\!\sqcup \Gamma^*$, where K is a finite commutative language and Γ is a finite set of letters, can be defined by a spider diagram (Theorem 6 in Section 6). To illustrate the approach, suppose we have $K_1 = \{d\}$, $\Gamma_1 = \{a,b,c,d\}$, $K_2 = \{a,b\}$, and $\Gamma_2 = \{c,d\}$ and we want to define the language

$$(K_1 \sqcup\!\sqcup \Gamma_1^*) \cup (K_2 \sqcup\!\sqcup \Gamma_2^*) = (\{d\} \sqcup\!\sqcup \{a,b,c,d\}^*) \cup (\{a,b\} \sqcup\!\sqcup \{c,d\}^*)$$

using a spider diagram; as with our translation of spider diagrams into shuffle products, we convert shuffle products into spider diagram in disjunctive normal form. It is easy to convert $\{d\} \sqcup\!\sqcup \{a,b,c,d\}^*$ into a spider diagram in disjunctive normal form: the diagram must contain both contours P and Q and the letter d (in K_1) gives rise to a spider outside both P and Q (since neither P nor Q are assigned the letter d). There is no shading, since $\Gamma_1 = \Sigma$. The spider diagram d_5 in Fig. 5 defines $\{d\} \sqcup\!\sqcup \{a,b,c,d\}^*$. In the case of $\{a,b\} \sqcup\!\sqcup \{c,d\}^*$, the is no *unitary* diagram in disjunctive normal form that defines this language. This is because the set $K_2 = \{a,b\}$, whilst commutative and finite, cannot arise from such a diagram: a unitary diagram in DNF giving rise to K_2 can only contain one spider, since the words in K_2 have length 1, but that spider would have two nodes, one for the letter a and another for the letter b, contradicting the 'spiders have single nodes' condition of being in DNF. We partition K_2 into two sets, namely $\{a\}$ and $\{b\}$; in the general case, we partition K into sets that are the commutative closure of a single word, w, in K with each letter in w giving rise to a spider in an appropriate zone. We see that

$$\{a,b\} \sqcup\!\sqcup \{c,d\}^* = (\{a\} \sqcup\!\sqcup \{c,d\}^*) \cup (\{b\} \sqcup\!\sqcup \{c,d\}^*)$$

and we can now convert each of the shuffle products in this union into a disjunction of two unitary spider diagrams, each of which is in disjunctive normal form; these two diagrams

are d_6 and d_7 in Fig. 5. In both cases, we have shaded zones because the set Γ_2 is not equal to Σ: the zones corresponding to the letters not in Γ_2 are shaded. Thus, the language $(K_1 \sqcup\!\sqcup \Gamma_1^*) \cup (K_2 \sqcup\!\sqcup \Gamma_2^*)$ is defined by $d_5 \vee d_6 \vee d_7$. Following this approach, we are able to prove Theorem 6 in Section 6.

To summarise, in Section 6 we prove that spider diagrams can define precisely the languages that are finite unions of shuffle products of the form $K \sqcup\!\sqcup \Gamma^*$ where K is a finite commutative language and Γ is a finite set of letters.

2.2. Converting commutative star-free regular languages to spider diagrams

We now demonstrate how to covert commutative star-free regular languages to spider diagrams. Our strategy is to show that every such language can be written as a finite union of shuffle products of the form $K \sqcup\!\sqcup \Gamma^*$ where K is a finite commutative language and Γ is a finite set of letters. To do so, we analyse properties of minimal finite automata accepting such languages.

We illustrate our approach by considering the commutative star-free language, L, defined by the (star-free) regular expression

$$\overline{\emptyset} d \overline{\emptyset} \cup (\overline{\overline{\emptyset}\{a,b\}\overline{\emptyset}\{a,b\}\overline{\emptyset}\{a,b\}\overline{\emptyset}}).$$

A minimal finite automaton, $\mathcal{A}$, accepting $L = L(\mathcal{A})$ can be seen in Fig. 6a. We note that the language L can be described as a union of two languages: the set of words accepted at one of the final states together with the set of words accepted at the other final state. Thus, we can 'decompose' $\mathcal{A}$ into two automata, $\mathcal{A}_1$ and $\mathcal{A}_2$, each accepting one of these two sets of languages, say $L(\mathcal{A}_1)$ and $L(\mathcal{A}_2)$ respectively; these automata are also shown in Fig. 6b and c respectively.

We prove, in Theorem 8 of Section 7, that each of the automata arising from this decomposition accepts a commutative star-free regular language. In essence, this reduces our problem of establishing how to define commutative star-free regular languages using spider diagrams to only those languages accepted by automata with single final states. Moreover, as a corollary of the Hopcroft minimisation algorithm [9], we may minimise these deterministic complete finite automata without introducing additional final states (as the algorithm merges indistinguishable states and does not create new final states).

Consider the minimal automata $min(\mathcal{A}_1)$ and $min(\mathcal{A}_2)$ in Fig. 7a and b obtained from $\mathcal{A}_1$ in Fig. 6b and $\mathcal{A}_2$ in Fig. 6c respectively. We see that any letter which occurs on a cycle in the minimal automaton which is on a path to the final state also occurs on a loop at the final state. For instance, in $min(\mathcal{A}_2)$ there is a loop labelled c at the start state and c also occurs on a loop at the final state. Intuitively, the loop on the start state tells us that the number of cs at the beginning of any accepted word is not bounded. By commutativity, we can move all such cs to the end of the accepted word and remain in the accepted language. Thus, there must be a loop labelled c on the final state. Here, the intuitive reasoning about the existence of an appropriate loop at the final state is relatively straightforward, since c itself labelled a loop at a non-final state. We prove that this property of letters on cycles giving rise to loops on the final state is true of any minimal automaton with a single final state that accepts a commutative star-free regular language in Lemma 6. To extract a set, Γ, from such an automaton we simply read off the letters that occur on loops at the final state; intuitively, their occurrence on such a loop means that there are no restrictions on their use in words of the (commutative) language. In our example, we have

$$\Gamma(min(\mathcal{A}_1)) = \{a,b,c,d\}$$

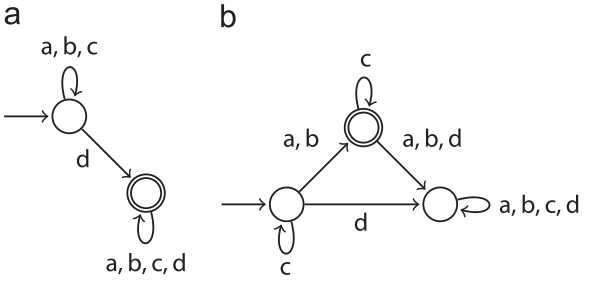

Fig. 7. Minimal automata obtained from $\mathcal{A}_1$ and $\mathcal{A}_2$.

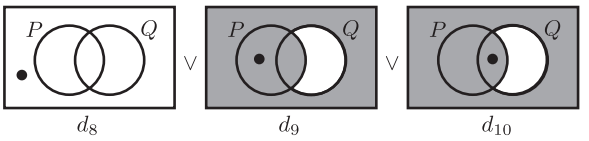

Fig. 8. Another translation of shuffle products to diagrams.

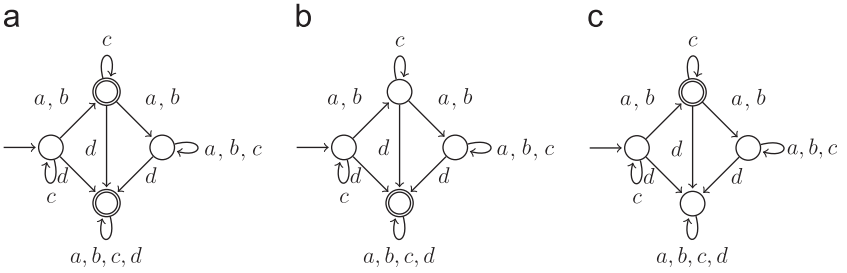

Fig. 6. Finite automata accepting commutative star-free regular languages.

and

$\Gamma(min(\mathcal{A}_2)) = \{c\}$.

All that remains is for us to determine how to extract the set K from such an automaton. To identify K, we simply read off the words that are accepted without following any cycles and take the commutative closure of this set; trivially, all words in K are accepted by the automaton, since the language is commutative. In our example, we have

$K(min(\mathcal{A}_1)) = \{d\}$

and

$K(min(\mathcal{A}_2)) = \{a,b\}$.

It can be shown that $min(\mathcal{A}_1)$ accepts $\{d\}\sqcup\sqcup\{a,b,c,d\}^*$ and $min(\mathcal{A}_2)$ accepts $\{a,b\}\sqcup\sqcup\{c\}^*$. Therefore, the original automaton $\mathcal{A}$ accepts

$(\{d\}\sqcup\sqcup\{a,b,c,d\}^*) \cup (\{a,b\}\sqcup\sqcup\{c\}^*)$.

A diagram defining this language is in Fig. 8, which is semantically equivalent to $d_5 \vee d_6 \vee d_7$ in Fig. 5. However, the diagram in Fig. 5 does not arise from the union of shuffle products given above: the shuffle product $\{d\}\sqcup\sqcup\{a,b,c,d\}^*$ gives rise to d_5 but $\{a,b\}\sqcup\sqcup\{c\}^*$ does not give rise to $d_6 \vee d_7$ since the lack of a d in the righthand component of the shuffle product, namely $\{c\}^*$, requires the zone outside both P and Q to be shaded.

We prove, in Theorem 9 in Section 7, that a minimal automaton, $\mathcal{A}$, with a single final state that accepts a commutative star-free regular language, accepts $K(\mathcal{A})\sqcup\sqcup\Gamma(\mathcal{A})^*$, where $K(\mathcal{A})$ and $\Gamma(\mathcal{A})$ are as just illustrated. Theorem 10 brings our results on automata together to establish that every commutative star-free language is a finite union of languages of the form $K\sqcup\sqcup\Gamma^*$. From this, we are able to prove our main result, in Theorem 11 of Section 7, that the following statements are equivalent:

1. L is the language of a spider diagram,
2. L is a commutative star-free regular language, and
3. L is a finite union of languages of the form $K\sqcup\sqcup\Gamma^*$ where K is a finite commutative language and $\Gamma \subseteq \Sigma$.

In addition, since it is known that spider diagrams are equivalent in expressiveness to monadic first-order logic with equality [20], the following statement is also equivalent: L is defined by a sentence in monadic first-order logic with equality.

3. Background: formal language theory

This section presents the notation we will use in respect of formal languages, alongside known results and concepts. The reader who is familiar with formal languages and properties of star-free regular languages may choose to skip this section.

Our notation for formal languages is primarily drawn from [13]. A regular language over a finite alphabet Σ is one that is defined using a *regular expression*:

1. the empty word, λ, and the empty language, $\emptyset$, are regular expressions,
2. each letter, a, in Σ is a regular expression, and
3. if r_1 and r_2 are regular expressions then so are:
   (a) $\overline{r_1}$,
   (b) $(r_1 \cup r_2)$,
   (c) $(r_1 \cap r_2)$,
   (d) $(r_1 \cdot r_2)$, and
   (e) r^*.

Regular expressions define regular languages in the obvious inductive way, where the base cases are given by: λ defines $\{\lambda\}$, $\emptyset$ defines $\emptyset$, and a defines $\{a\}$. Since we consider only regular languages, we shall simply say *language* to mean *regular language*. A language is *star-free* if it can be defined by a regular expression without using the Kleene star, *.

The following are regular expressions, given the alphabet $\Sigma = \{a,b\}$: a, b, $a \cdot a$, $b \cdot b$ and $(a \cdot a)^* \cup (b \cdot b)^*$. Thus, the regular expression $r = (a \cdot a)^* \cup (b \cdot b)^*$ (which can also be written as $(aa)^* \cup (bb)^*$ as an abuse of notation) defines the language containing all words with an even number of as and no bs or an even number of bs and no as. Examples of words in this language include λ, aa, $aaaa$, bb, and $bbbb$ whereas a, ab, and $aabb$ are examples of words not in the language. Where it is appropriate to do so, we blur the distinction between a regular expression and the language it defines.

The length of a word w over alphabet Σ is denoted $|w|$ whereas the number of occurrences of a letter, a, in w is denoted $|w|_a$. For example, aa and bb have length 2, $aabab$ has length 5 and so forth.

The *commutative closure* of a word w, denoted $comm(w)$, is the set

$comm(w) = \{w' \in \Sigma^* : \quad \text{for all } a \in \Sigma, |w'|_a = |w|_a\}$.

The *commutative closure* of a language L is $comm(L) = \bigcup_{w \in L} comm(w)$. If $L = comm(L)$ then L is *commutative*. The regular expression $r = (aa)^* \cup (bb)^*$ defines a commutative language.

The *shuffle product* of two words $u = u_1 \dots u_n$ and $v = v_1 \dots v_m$, denoted $u\sqcup\sqcup v$, is defined recursively as

$(u_1 \dots u_n \sqcup\sqcup v_1 \dots v_m) = \{u_1\}(u_2 \dots u_n \sqcup\sqcup v_1 \dots v_m)$

$\cup \{v_1\}(u_1 \dots u_n \sqcup\sqcup v_2 \dots v_m)$,

where $(\lambda \sqcup\sqcup u) = (u \sqcup\sqcup \lambda) = \{u\}$. The shuffle product of aa and bb is

$aa\sqcup\sqcup bb = \{aabb, abab, abba, baab, baba, bbaa\}$.

If $w \in u\sqcup\sqcup v$ then u is a *scattered subword* of w. For example, ab is a scattered subword of $abaab$. We define $w \rightarrow_s u$ to be the set of *scattered residuals* of u within w, obtained by deleting letters of u from w. Formally

$w \rightarrow_s u = \{v \in \Sigma^* : w \in u \sqcup\sqcup v\}$.

For example, the set of scattered residuals of ab in $aababaa$ is $\{aabaa, abaaa\}$.

The *shuffle product* of two languages L_1 and L_2, denoted $L_1\sqcup\sqcup L_2$, is

$$\bigcup_{u \in L_1 \wedge v \in L_2} u \sqcup\sqcup v.$$

Given a finite alphabet, Σ, a finite automaton, $\mathcal{A} = \langle Q, \Sigma, \delta, q_0, F \rangle$, is a 5-tuple where:

1. Q is a set of states,
2. δ is the transition function, so $\delta : Q \times \Sigma \to Q$, so if $\delta(q_1, a) = q_2$ then the machine makes a transition from q_1 to q_2 on reading the letter a,
3. q_0 is the initial state, and
4. F is the set of final states.

The transition function δ can be extended to $\delta^* : Q \times \Sigma^* \to Q$ where

$$\delta^*(q, \lambda) = q$$

and

$$\delta^*(q, a_1 a_2 \dots a_n) = \delta^*(\delta(q, a_1), a_2 \dots a_n).$$

The language *accepted* by A is the set of words, L, such that $w \in L$ if and only if $\delta^*(q_0, w) \in F$.

As is typical, we can represent a finite automaton, $\mathcal{A} = \langle Q, \Sigma, \delta, q_0, F \rangle$, using a directed edge-labelled graph whose nodes are Q and the edges are induced by δ: there is a directed edge labelled a from state q_i to q_j if and only if $\delta(q_i, a) = q_j$. This graph is the *transition diagram* of $\mathcal{A}$. For example, in Fig. 9a, the automaton with the depicted transition diagram accepts the language defined by the regular expression $(aa)^* \cup (bb)^*$. We blur the distinction between a finite automaton and its transition diagram where it is convenient to do so. For instance, we will refer to a path in the automaton when, strictly, we mean a path in the transition diagram.

A word $u \in L$ is said to *exercise* a path, p, in (the transition diagram of) $\mathcal{A} = \langle Q, \Sigma, \delta, q_0, F \rangle$ if performing the actions on the letters of the word u takes the path p through the automaton. A *cycle*, C, in finite automaton $\mathcal{A}$ is a sequence of edges in the transition diagram beginning at a state q and ending on q (i.e. $\delta^*(q, u) = q$ where u exercises C). A *trivial cycle* is a cycle of length 1. We define $\mathcal{A}|_a$ to be the graph obtained by deleting all edges from the transition diagram of the automaton except those labelled by the letter $a \in \Sigma$.

For example, in Fig. 9a, the word *aab* exercises the path starting at q_0, and then transitioning to q_1 then to q_3 and ending at q_5. It also exercises the path starting at q_3, then transitioning to q_1, then q_3 and ending at q_5. There is a trivial cycle at the non-final state q_5. The graph $\mathcal{A}|_a$ is obtained from the graph in Fig. 9a by deleting all of the b labels and then deleting all of the edges which are left with no label. The resulting graph can be seen in Fig. 9b.

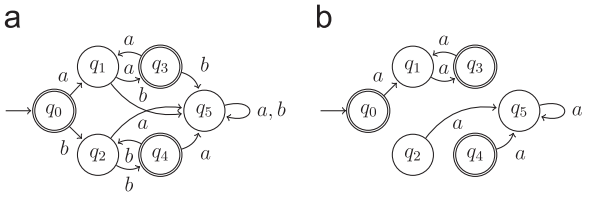

Fig. 9. The minimal finite automaton which recognises $(aa)^* \cup (bb)^*$ (left) and $\mathcal{A}|_a$ (right).

One key observation on which our arguments rely is captured by the following property:

Proposition 1 (*Modified from [12]*). *Let $\mathcal{A}$ be a minimal automaton accepting a star-free language. Then for each $a \in \Sigma$ the graph $\mathcal{A}|_a$ contains no non-trivial cycles.*

Given a language, L, a congruence relation, $\sim_L$, on Σ^* is defined such that for all $u, v \in \Sigma^*$ we have $u \sim_L v$ if and only if for all $x, y \in \Sigma^*$

$$xuy \in L \Leftrightarrow xvy \in L.$$

It can trivially be shown that $\sim_L$ is an equivalence relation. The equivalence class of $\sim_L$ of which u is a representative is denoted $[u]_L$ or, simply, $[u]$ when the language is clear from the context.

Continuing with the example $(aa)^* \cup (bb)^*$, we have:

1. $[\lambda] = \{\lambda\}$,
2. $[a] = a \cdot (aa)^*$,
3. $[b] = b \cdot (bb)^*$,
4. $[aa] = aa \cdot (aa)^*$,
5. $[bb] = bb \cdot (bb)^*$, and
6. $[ab] = (\Sigma^* ab \Sigma^*) \cup (\Sigma^* ba \Sigma^*)$.

Consider the prefix, x. When a minimal finite automaton, $\mathcal{A}$, accepting L, reads x it takes us to some state, q_i. From q_i, we can read u taking us to state q_j and, also from q_i, we can read v taking us to state q_k. Now, if xuy and xvy are both in L or both not in L, it must be that q_j and q_k are the same state (if different, q_j and q_k would be indistinguishable and this would contradict the minimality of $\mathcal{A}$). We have the following property:

Proposition 2 (*Modified from [21]*). *Let $\mathcal{A} = \langle Q, \Sigma, \delta, q_0, F \rangle$ be the minimal automaton accepting a regular language L and let $u, v \in \Sigma^*$. Then $u \sim_L v$ if and only if for each $q \in Q$ it is the case that $\delta^*(q, u) = \delta^*(q, v)$.*

Recall that a monoid is a set together with an associative, binary operation under which there is an identity element. So, Σ^* under concatenation is a monoid which has identity λ. Given a monoid, M, and a congruence relation, R, on M, one can form the quotient monoid M/R in the standard manner. The *syntactic monoid* of a language L is the quotient monoid $(\Sigma^*/\sim_L, \cdot, [\lambda]_L)$ where the binary operation $\cdot$ is defined by $[x]_L \cdot [y]_L = [xy]_L$ and $[\lambda]_L$ is the identity. A syntactic monoid, $(\Sigma^*/\sim_L, \cdot, [\lambda]_L)$, of L is *aperiodic* if, for each $[x]_L \in \Sigma^*/\sim_L$, there exists $n \in \mathbb{N}$ such that $[x]_L^n = [x]_L^{n+1}$.

In our running example, $(aa)^* \cup (bb)^*$, the syntactic monoid is $\mathcal{M} = (\Sigma^*/\sim_L, \cdot, [\lambda]_L)$ where

$$\Sigma^*/\sim_L = \{[\lambda], [a], [b], [aa], [bb], [ab]\}.$$

Under the binary operation, $\cdot$, we see that $[a] \cdot [a] = [aa]$, $[a] \cdot [ab] = [ab]$, and $[b] \cdot [\lambda] = [b]$ for example. The monoid M is not aperiodic, since $[a]^{2n} \neq [a]^{2n+1}$ for any integer n (intuitively, having an even number of as – $[a]^{2n}$ – is not equivalent to having an odd number of as – $[a]^{2n+1}$ – since the former describes words in L whereas the latter does not).

Since we are considering star-free languages, the following classic result is helpful to us.

Proposition 3 (*Schützenberger [16]*). *Let $L \subseteq \Sigma^*$ with syntactic monoid M. Then L is star-free if and only if M is finite and aperiodic.*

Hence the language $(aa)^* \cup (bb)^*$ is not star-free: whilst its syntactic monoid $\mathcal{M}$ is finite, it is not aperiodic. By contrast, the language $L = (ab)^*$ has a finite aperiodic monoid, namely $(\Sigma^*/\sim_L, \cdot, [\lambda]_L)$ where

$\Sigma^*/\sim_L = \{[\lambda],[a],[b],[aa],[ab],[ba]\}$,

since

1. $[\lambda] = [\lambda]^2$,
2. $[a]^2 = [a]^3$,
3. $[b]^2 = [b]^3$,
4. $[aa] = [aa]^2$,
5. $[ab] = [ab]^2$, and
6. $[ba] = [ba]^2$.

Hence, $(ab)^*$ defines a star-free regular language, even though this regular expression is not star-free itself. The regular expression $(ab)^*$, over alphabet $\Sigma = \{a,b\}$, can be written as a star-free regular expression

$$\overline{(b\overline{\emptyset})} \cap \overline{(\overline{\emptyset}aa\overline{\emptyset})} \cap \overline{(\overline{\emptyset}bb\overline{\emptyset})} \cap \overline{(\overline{\emptyset}a)}.$$

4. Spider diagrams

This section provides a brief overview of spider diagrams, modified from [11]. The contour labels in spider diagrams are selected from a finite set $\mathcal{C}$. A *zone*, denoted (in,out), is a pair of disjoint subsets of $\mathcal{C}$. 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 $\mathcal{Z}$. A *region* is a set of zones. To describe the spiders in a diagram, it is sufficient to say how many spiders are placed in each region. In the following definition, $\mathbb{P}Z$ denotes the power set of Z.

Definition 1 (*Howse et al. [11]*). A *unitary spider diagram*, d_1, is a quadruple $\langle C,Z,ShZ,SI \rangle$ where:

1. $C = C(d_1) \subseteq \mathcal{C}$ is a set of contour labels,
2. $Z = Z(d_1) \subseteq \{(in, C-in) : in \subseteq C\}$ is a set of zones,
3. $ShZ = ShZ(d_1) \subseteq Z(d_1)$ is a set of shaded zones, and
4. $SI = SI(d_1) \subsetneq \mathbb{N}^+ \times (\mathbb{P}Z - \{\emptyset\})$ is a finite set of spider identifiers such that for all $(n,r),(m,s) \in SI(d_1)$ if $r = s$ then $n = m$.

The set of *spiders* in d_1 is defined to be

$S(d_1) = \{(i,r) : (n,r) \in SI(d_1) \wedge 1 \leq i \leq n\}$.

For each $(i,r) \in S(d_1)$, the *habitat* of (i,r), denoted $\eta(i,r)$, is r; that is, $\eta(i,r) = r$. The symbol $\perp$ is also a unitary spider diagram. 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*.

The abstract syntax of the diagram d_1 in Fig. 3a is $C(d_1) = \{P\}$, with zones

$Z(d_1) = \{(\emptyset,\{P\}),(\{P\},\emptyset)\}$,

shaded zones $ShZ(d_1) = \{(\{P\},\emptyset)\}$, and spider identifiers

$SI(d_1) = \{(1,\{(\{P\},\emptyset)\})\}$.

By convention, we employ a lower-case d_i to denote a unitary spider diagram. An upper case D_i will denote an arbitrary diagram. It is useful to identify which zones could be present in a unitary diagram, given the label set, but are not present; semantically, *missing* zones provide information.

Definition 2 (*Howse et al. [11]*). A zone (in,out) is *missing* from unitary diagram d_1 if it is in $MZ(d_1) = \{(in, C(d_1) - in) : in \subseteq C(d_1)\} - Z(d_1)$.

Formally, the semantics of spider diagrams are model-based: a model is an assignment of sets to contour labels that agrees with the intended meaning of the diagram. Our definition of an interpretation includes $<$ as a strict total order on the universal set U. Whilst spider diagrams place no constraints on $<$, regular expressions are able to do so. Thus, the presence of $<$ will be meaningful when we define models for words later in the paper.

Definition 3 (*Delaney et al. [5]*). An *interpretation* is a triple $I = (U, <, \Psi)$ where U is a finite set, $<$ is a strict total order over U and $\Psi : \mathcal{C} \to \mathbb{P}U$ assigns a subset of U to each contour label. We extend Ψ to zones and regions:

1. each zone, $(in,out) \in \mathcal{Z}$, represents the set $\bigcap_{P_i \in in} \Psi(P_i) \cap \bigcap_{P_i \in out} (U - \Psi(P_i))$ and
2. each region, $r \in \mathbb{P}\mathcal{Z}$, represents the set which is the union of the sets represented by r's constituent zones.

The formal definition of a model for a unitary spider diagram differs, in presentation, from that in Howse et al. [11] because our interpretations contain a strict total order $<$. However, the relation $<$ is irrelevant when determining whether an interpretation is a model for a diagram. Further, our definition closely mirrors the one given for constraint diagrams in [19] (constraint diagrams extend spider diagrams by adding more syntax). In any case, an interpretation is a model under our definition precisely when it is a model under the definition given in [11].

Definition 4. Let $I = (U, <, \Psi)$ be an interpretation and let d_1 ($\neq \perp$) be a unitary spider diagram. Then I is a *model* for d_1, denoted $m \models d_1$, if and only if the following conditions hold:

1. *The missing zones condition*. All of the missing zones represent the empty set, that is, $\bigcup_{z \in MZ(d_1)} \Psi(z) = \emptyset$.
2. *The spider mapping condition*. There exists an injective function, $f : S(d_1) \to U$, called a *valid* function, such that the following conditions hold:
   (a) *The spiders' locations condition*. All spiders represent elements in the sets represented by the

regions in which they are placed: $\forall(s) \in S(d_1)$ $f(s) \in \Psi(\eta(s))$.

(b) *The shading condition*. Shaded regions represent a subset of elements denoted by spiders: $\forall z \in ShZ(d_1)$ $\Psi(z) \subseteq im(f)$, where $im(f)$ denotes the image of f (i.e. the set of elements in the codomain, U, to which the function f maps elements of the domain, $S(d_1)$).

If $d_1 = \perp$ then no interpretation is a model for d_1. For compound diagrams, the definition of a model extends in the obvious inductive way.

The interpretation $m = (U, <, \Psi)$ where $U = \{1,2,3\}$, $<$ is the natural order over U, $\Psi(P) = \{2\}$ and $\Psi(Q) = \{1,2,3\}$ is a model for the diagram d_1 in Fig. 3a but not for d_2. Therefore m is a model for $d_1 \vee d_2$.

5. Word models

In order to discuss the language of a spider diagram, we associate sets of letters with contour labels, as first done in [5], following Thomas' approach in [22]. We illustrated this in Section 2 and we now formalise the approach using a function called a letter map.

Definition 5. A function, $lm : \mathcal{C} \to \mathbb{P}\Sigma$, that maps contour labels to sets of letters is called a *potential letter map*. A potential letter map is a *letter map* if its extension to zones, $lm : \mathcal{C} \cup \mathcal{Z} \to \mathbb{P}\Sigma$, defined by

$$lm(in,out) = \bigcap_{P_i \in in} lm(P_i) \cap \bigcap_{P_i \in out} (\Sigma - lm(P_i))$$

ensures that any zone, (in,out), where $in \cup out = \mathcal{C}$ is assigned at most one letter, that is $|lm(in,out)| \leq 1$.

The letter map condition ensures that the spider diagram logic is capable of distinguishing each letter: given any letter, $a \in \Sigma$, there is a zone, $z \in (in, \mathcal{C} - in)$, where $lm(z) = \{a\}$. Thus, a letter map establishes a one-to-one correspondence between zones that partition $\mathcal{C}$ and letters. We are, therefore, assuming $|\Sigma| \leq 2^{|\mathcal{C}|}$. Further, when we consider letter map functions, we assume that they are extended to zones as defined above.

We define a function $zone : \Sigma \to \{(in, \mathcal{C} - in) : in \subseteq \mathcal{C}\}$ by $zone(a) = (in, \mathcal{C} - in)$ where $lm(in, \mathcal{C} - in) = \{a\}$. Given such a correspondence between letters and zones, we define when an interpretation models a word.

Definition 6. An interpretation $I = (U, <, \Psi)$ is a *model* for a word $w = a_1 a_2 \ldots a_n$ if there exists a bijection, Υ, from the multi-set $\{a_1, a_2, \ldots, a_n\}$ to U such that the following conditions hold:

1. *Letter location condition*. Each letter a_i interprets an element in the set represented by the zone to which the letter a_i is assigned: for each a_i, $\Upsilon(a_i) \in \Psi(zone(a_i))$.
2. *Letter order condition*. The order relation $<$ respects the order of letters in w: for each a_i where $i > 1$, $\Upsilon(a_{i-1}) < \Upsilon(a_i)$.

Such an Υ is said to be *valid*.

To illustrate the above concepts, suppose we have $\Sigma = \{a,b,c\}$ and $\mathcal{C} = \{P,Q\}$. Define a letter map by $lm(P) = \{a,b\}$ and $lm(Q) = \{b\}$. Consider an interpretation $I = (\{1,2,3\}, <, \Psi)$ where $<$ is the natural order over U, $\Psi(P) = \{1,2\}$ and $\Psi(Q) = \{2\}$. Then I models the word abc and only the word abc. If, instead, we had $\Psi(P) = \{1\}$ then I would not model any word since I has an element in the set $\Psi(Q) - \Psi(P)$, in other words, there is an element in the set represented by the zone $(\{Q\},\{P\})$, but $lm(\{Q\},\{P\}) = \emptyset$. Consequently, given any word, w, no valid Υ can exist in this interpretation.

Thus, in this section, we have demonstrated that a letter map function provides us with a link between the diagrams world and the formal language world. From this point forward we assume, in our theoretical exposition, that a particular letter map has been identified. Using a letter map, we are able to identify when interpretations model words. In the next section, we use models to define the language of a spider diagram, given a letter map.

6. The language of a spider diagram

A spider diagram, D_1, defines a language, L_1, if the model set of D_1 equals the model set of L_1; the models of L_1 are precisely those that model each of its words. Since spider diagrams place no constraint on $<$, it naturally follows that they place no constraints on the order of letters in words; hence, their languages are commutative.

For illustrative purposes, let $\Sigma = \{a,b,c,d\}$, $\mathcal{C} = \{P,Q\}$ and a letter map defined by $lm(P) = \{a,b\}$ and $lm(Q) = \{b,c\}$. The interpretation $m = (U, <, \Psi)$ where $U = \{1,2,3\}$, $<$ is the natural order over U, $\Psi(P) = \{1\}$ and $\Psi(Q) = \{2,3\}$ is a model for d_1 in Fig. 3a and is also a model for the word acc. Therefore, the word acc is in the language of d_1.

Definition 7. Let D_1 be a spider diagram and M be the set of models for D_1. Let L_1 be a subset of Σ^* and M' be the set models for words in L_1. L_1 is the *language of* D_1, denoted $\mathcal{L}(D_1)$, if and only if $M = M'$. Given D_1, if D_1 has a language, L_1, then we say that D_1 *defines* L_1.

We now use some key results from the literature [18] to characterise precisely the languages of spider diagrams: every spider diagram is semantically equivalent to a disjunction of unitary spider diagrams, each of which is $\perp$ or

1. contains all of the contour labels in $\mathcal{C}$,
2. has no missing zones, and
3. has only spiders placed in single zones.

Thus, we can identify the class of languages defined by spider diagrams by analysing diagrams in this *disjunctive normal form* (DNF). We note that a single unitary diagram meeting the above 'DNF' conditions is itself a diagram in DNF. Since unitary diagrams in DNF have only spiders with single zone habitats, to aid our exposition, we will abuse notation by abbreviating $\eta(s) = \{z\}$ to $\eta(s) = z$.

6.1. Diagrams with no language

Whether any given diagram defines a language depends on the letter map chosen. This is because a letter map need not assign letters to all zones. If this is the case then there are interpretations that do not model any word but these interpretations model diagrams. Trivially:

Theorem 1. *Let D_1 be a spider diagram in DNF. Then D_1 defines a language if and only if each unitary part of D_1 defines a language.*

We use Fig. 10 to illustrate when diagrams define languages. Suppose that $\Sigma=\{a,b,c\}$, $\mathcal{C}=\{P,Q\}$ and a letter map is defined as illustrated in d_1. Thus, the zone $(\emptyset,\{P,Q\})$ does not contain any letter. The diagram d_2 defines the language Σ^*, since all of its models also model words, and every word model also models d_2. However, the diagram d_3 does not define any language. The spider placed in the zone $(\emptyset,\{P,Q\})$ forces this zone to represent a non-empty set in any model. Such a model cannot satisfy any word due to the lack of a letter assigned to this zone. Although the diagrams d_4 and d_5 are not in DNF, they provide some insights into the subtlety of this issue. The diagram d_4 forces all elements to be in P and defines the language $\{a,b\}^*$. However, the diagram d_5 does not define a language. Whilst the element represented by the spider could correspond to a letter c in a word, since $lm(P)$ does not include c, the diagram has models where there are elements that are not in P or Q; such models for d_5 do not model words. Our results in this section are presented for diagrams in DNF for simplicity.

In the unitary case, a spider diagram, d_1, in DNF defines a language if and only if the zones, z, for which $lm(z)=\emptyset$, are shaded and have no spider in them. To prove that this characterisation does indeed identify those unitary diagrams in DNF that define languages, we make use of the following lemma, obtained from results in [11].

Lemma 1. *Let d_1 be a unitary diagram in DNF*:

1. *if $d_1\neq\perp$ then d_1 has a model and*
2. *in any interpretation, any two distinct zones in d_1 represent disjoint sets.*

Lemma 2. *Let d_1 be a unitary spider diagram in DNF. Then d_1 defines a language, L_1, if and only if $d_1=\perp$ or for all z in d_1, if $lm(z)=\emptyset$ then*

1. *z is shaded in d_1 and*
2. *for all spiders, s, in d_1, $\eta(s)\neq\{z\}$.*

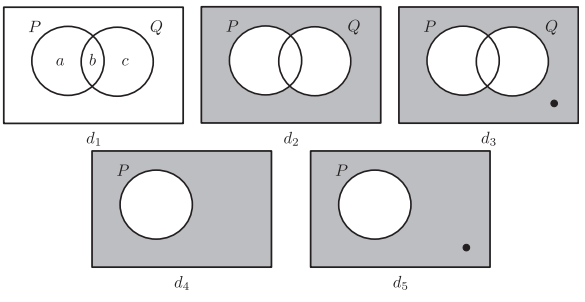

Fig. 10. The definability of languages.

Proof. If $d_1=\perp$ then d_1 defines the empty language. Suppose that d_1 is not $\perp$ and defines a language, L. For a contradiction, suppose that there exists a zone, $z=(in,\mathcal{C}-in)$, in d_1 where $lm(z)=\emptyset$ and one of the above conditions does not hold.

First, suppose that z in not shaded. Choose an interpretation $I=(S(d_1)\cup\{z\},<,\Psi)$ where $<$ is any ordering and $\Psi:\mathcal{C}\to S(d_1)\cup\{z\}$ is defined by

$$\Psi(L)=\begin{cases}\{s\in S(d_1): L\in in' \quad \text{where } \eta(s) \quad \text{if } L\notin in\\ \quad =(in',\mathcal{C}-in')\},\\ \{s\in S(d_1): L\in in' \quad \text{where } \eta(s) \quad \text{otherwise.}\\ =(in',\mathcal{C}-in')\}\cup\{z\}.\end{cases}$$

It is straightforward to show that I models d_1 where a valid mapping, f, of spiders to universal set elements simply maps them to themselves (that is, f is the identity map). Since L_1 is the language of d_1, there exists a word, $w=a_1\ldots a_n$, in L_1 modelled by I. Choose a valid Υ for w. Since Υ is bijective, a letter, a_i in w maps to z and, by the validity of Υ, we have

$$z=\Upsilon(a_i)\in\Psi(zone(a_i)).$$

It can be shown that $z\in\Psi(z)$. Therefore, $\Psi(zone(a_i))\cap\Psi(z)\neq\emptyset$. By Lemma 1, distinct zones in d_1 represent disjoint sets, so $zone(a_i)=z$. By the definition of *zone*, $lm(z)=a_i$ reaching a contradiction.

Alternatively, suppose that z contains a spider, s, that is $\eta(s)=\{z\}$. By Lemma 1, d_1 has a model, I. Clearly, in I, $f(s)\in\Psi(z)$ for some valid mapping, f, of spiders to universal set elements. Since d_1 defines a language, I models some w in L. But then some letter, a_i, in w must map to $f(s)$ under a valid Υ. Then we have

$$\Upsilon(a_i)=f(s)\in\Psi(zone(a_i)).$$

But $f(s)\in\Psi(z)$, so we deduce that $zone(a_i)=z$ (again using Lemma 1), giving a contradiction. Hence, in either case we have derived a contradiction so the two conditions must hold.

For the converse, suppose that the two conditions hold for each z in d_1. We must show that d_1 defines a language. Let $I=(U,<,\Psi)$ be a model for d_1. It is sufficient to show that I models a word. To construct such a word $w=a_1\ldots a_n$ where $|U|=n$, simply define a_i to be the letter such that $\Psi(zone(a_i))$ contains the ith element of U under $<$. It is straightforward to show that I models w. □

6.2. Diagrams that define languages

For diagrams that do define languages, we will now proceed to classify those languages. First, we will establish that the defined languages are all commutative and star-free. We will then proceed to show that they are finite unions of shuffle products of languages K and Γ^* where K is a finite commutative language and Γ is a finite set of letters.

6.2.1. Commutative and star-free languages

Spider diagrams are unable to specify any order information. From this it follows, intuitively, that the language of a spider diagram is commutative. Moreover, Thomas [22] establishes that MFOL[$<$] can define precisely the star-free languages. Since spider diagrams are equivalent to MFOL[$=$], they too can define only star-free languages. Hence:

Theorem 2. *The language of spider diagram D_1 is commutative and star-free.*

Proof. Let $w \in \mathcal{L}(D_1)$ then $m = (U, <, \Psi)$ is a model for w and D_1. Then any $m' = (U, <', \Psi)$ where $<'$ is any strict total order on U is a model for D_1. Furthermore, each m' models a permutation of w. Thus, the language of D_1 is commutative.

Let M be the set of interpretations that model D_1. Then M is the set of models for a sentence in *MFOL*[$=$] [20]. Furthermore M is the set of models for a star-free regular language as each star-free regular language definable in *MFOL*[$<$] [22]. Thus, as *MFOL*[$=$]$\subsetneq$*MFOL*[$<$], the language of D_1 is star-free. Therefore, the language of D_1 is both commutative and star-free. □

6.2.2. Shuffle products

For a unitary diagram, d_1, we will now define $K(d_1)$ to be the set of words derived from the spiders in d_1, and $\Gamma(d_1)$ to be the set of letters arising from the non-shaded zones, as illustrated in Section 2.

Definition 8. Let d_1 ($\neq \perp$) be a unitary diagram in DNF with $S(d_1) = \{s_1, s_2, \ldots, s_n\}$. Define $\kappa : S(d_1) \to \Sigma$ by $\{\kappa(s)\} = lm(\eta(s))$. If $S(d_1) \neq \emptyset$, we define

$$K(d_1) = \{\kappa(s_1)\} \sqcup\!\sqcup \{\kappa(s_2)\} \sqcup\!\sqcup \cdots \sqcup\!\sqcup \{\kappa(s_n)\}$$

and $K(d_1) = \{\lambda\}$ otherwise. For $d_1 = \perp$ we define $K(d_1) = \emptyset$.

Definition 9. Let d_1($\neq \perp$) be a unitary diagram in DNF. We define

$$\Gamma(d_1) = \{a \in \Sigma : zone(a) \in Z(d_1) - ShZ(d_1)\}.$$

For $d_1 = \perp$ we define $\Gamma(d_1) = \emptyset$.

Given $lm(P) = \{a,b\}$ and $lm(Q) = \{b,c\}$ over the alphabet $\Sigma = \{a,b,c,d\}$, we note that for a diagram containing only shading, such as that in Fig. 11, $K = \{\lambda\}$ and $\Gamma = \emptyset$ and we have $\{\lambda\} \sqcup\!\sqcup \emptyset^* = \{\lambda\} \sqcup\!\sqcup \{\lambda\} = \{\lambda\}$.

The following two lemmas are necessary to prove Theorem 3 which states that the language of a unitary spider diagram, d_1, is $K(d_1) \sqcup\!\sqcup \Gamma(d_1)^*$.

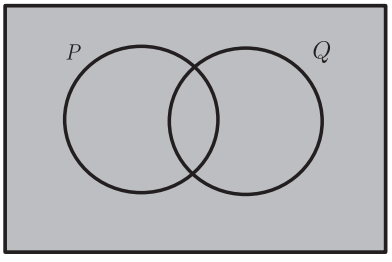

Fig. 11. A shaded diagram.

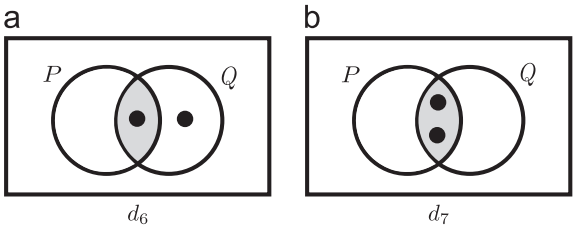

Fig. 12. Illustrating the proofs of Lemmas 3 and 4.

Consider the diagram d_6 in Fig. 12a. We show, in the following lemma, that d_6 defines the language $K(d_6) \sqcup\!\sqcup \Gamma(d_6)^*$ where $K(d_6) = \{bc,cb\}$ and $\Gamma(d_6) = \{a,c,d\}$. We take a model for d_6, say $m = (\{1,2,3,4\}, <, \Psi)$, where $\Psi(P) = \{1\}$ and $\Psi(Q) = \{1,3\}$ and define $lm(P) = \{a,b\}$ and $lm(Q) = \{b,c\}$. Then m models a unique word *bdcd*. Using the notation $x_{[i]}$ to assert that x is the letter in the ith position in w, there exists a Υ where $\Upsilon b_{[1]} = 1, \Upsilon d_{[2]} = 2, \Upsilon c_{[3]} = 3$ and $\Upsilon d_{[4]} = 4$. From Υ and f we define κ from spiders to letters, where s_1 is the spider in both P and Q and s_2 is the other spider: $\kappa(s_1) = b$ as $f(s_1) = \Upsilon b_{[1]} = 1$, and $\kappa(s_2) = c$ as $f(s_2) = \Upsilon c_{[3]} = 3$. We show that $\{\kappa(s_1)\} \sqcup\!\sqcup \{\kappa(s_2)\} = K(d_6)$: $K(d_6) = \{b\} \sqcup\!\sqcup \{c\} = \{bc,cb\}$ as expected. Furthermore, as $\Gamma(d_6) = \{a,c,d\}$ the scattered residual of $bdcd \to_s bc$, namely dd, is in $\Gamma(d_6)^*$.

Lemma 3. *Let d_1 be a unitary diagram in disjunctive normal form. Then either d_1 defines no language or d_1 defines a language and $\mathcal{L}(d_1) \subseteq K(d_1) \sqcup\!\sqcup \Gamma(d_1)^*$.*

Proof. Suppose d_1 defines a language. If $d_1 = \perp$ then $\mathcal{L}(d_1) = \emptyset$ and $K(d_1) \sqcup\!\sqcup \Gamma(d_1) = \emptyset$ and establishing the result. Otherwise, let $w = a_1 a_2 \ldots a_n \in \mathcal{L}(d_1)$ and let $m = (U, <, \Psi)$ be a model for d_1 and w. We show that $w \in K(d_1) \sqcup\!\sqcup \Gamma(d_1)^*$. Since m models both d_1 and w, choose a valid f from spiders in d_1 to U and a valid Υ mapping of letters in w to elements of U. Define a function $\kappa : S(d_1) \to \{a_1, a_2, \ldots, a_n\}$ by $\kappa(s) = a_i$ where $f(s) = \Upsilon(a_i)$; κ exists since Υ is bijective. Furthermore, since Υ is bijective and f is injective, we deduce κ is injective. Therefore, assuming $S(d_1)$ of the form $\{s_1, s_2, \ldots, s_x\}$, we can define

$$k \in \{\kappa(s_1)\} \sqcup\!\sqcup \{\kappa(s_2)\} \sqcup\!\sqcup \cdots \sqcup\!\sqcup \{\kappa(s_x)\},$$

where k is a scattered subword of w. To show $k \in K(d_1)$ we prove $zone(\kappa(s)) = \eta(s)$. By spiders' locations condition

$$f(s) \subseteq \Psi(\eta(s)).$$

Given $f(s) = \Upsilon(\kappa(s))$ we deduce

$$\Upsilon(\kappa(s)) \in \Psi(\eta(s)). \tag{1}$$

Since Υ is valid, we also have

$$\Upsilon(\kappa(s)) \in \Psi(zone(\kappa(s))). \tag{2}$$

Thus, from (1) and (2), the zones $\eta(s)$ and $zone(\kappa(s_i))$ do not represent disjoint sets. Since d_1 is in DNF, by definition we know that the contour label set is $\mathcal{C}$ and d_1 has no missing zones, so $zone(\kappa(s_i)) \in Z(d_1)$. We know that $\eta(s)$ is a zone in $Z(d_1)$. By Lemma 1, distinct zones in d_1 represent

disjoint sets, so we deduce

$zone(\kappa(s)) = \eta(s)$

as required. Hence $k \in K(d_1)$.

Let $\gamma = \gamma_1\gamma_2 \ldots \gamma_y$ be a scattered residual of $w \rightarrow_s k$. We show that each γ_i is in $\Gamma(d_1)$. As each spider s in $S(d_1)$ is mapped, via the bijective function Υ, to a letter of k then $\Upsilon(\gamma_i) \notin \Psi(s)$. Then, by the shading condition

$$\Psi(\gamma_i) \notin \bigcup_{z \in ShZ(d_1)} \Psi(z)$$

and since $\Upsilon(\gamma_i)$ ensures $\Upsilon(\gamma_i) \in \Psi(zone(\gamma_i))$ we deduce $zone(\gamma_i)$ is not shaded in d_1. Thus

$\gamma_i \in \{a : zone(a) \in Z(d_1) - ShZ(d_1)\} = \Gamma(d_1)$.

Therefore $\gamma \in \Gamma(d_1)^*$.

Thus, given a unitary diagram d_1 and an interpretation m such that m models d_1, m is a model for $w \in k \sqcup\!\sqcup \gamma$ where $k \in K(d_1), \gamma \in \Gamma(d_1)^*$. Hence either d_1 defines no language or $\mathcal{L}(d_1) \subseteq K(d_1) \sqcup\!\sqcup \Gamma(d_1)^*$. □

Whereas Lemma 3 shows that the language of a unitary spider diagram is of the form $K(d_1) \sqcup\!\sqcup \Gamma(d_1)^*$, the following lemma shows that any language of the form $K(d_1) \sqcup\!\sqcup \Gamma(d_1)^*$ is a subset of the language of that unitary spider diagram. Consider d_7 in Fig. 12b. Here $K(d_7) = \{bb\}$ and $\Gamma(d_7) = \{a,c,d\}$. Given $w \in k \sqcup\!\sqcup \gamma, k \in K(d_7), \gamma \in \Gamma(d_7)^*$, we show that an interpretation m that models w is a model for d_7. To do this we define f and show the required properties hold. Taking $w = bbcd$ as an example word and $m = (\{1,2,3,4\}, <, \Psi)$, where $\Psi(P) = \{1,2\}, \Psi(Q) = \{1,2,3\}$, as a model for w there exists Υ and κ from which we define f. As κ is injective from spiders to letters in w we use the bijection Υ to extend the mapping from spiders to elements of the universe. Denoting the spiders by s_1 and s_2, we have $f(s_1) = \Upsilon(\kappa(s_1)) = \Upsilon(b_{[1]})$ and $f(s_2) = \Upsilon(\kappa(s_2)) = \Upsilon(b_{[2]})$. Then both the spiders' locations condition and the shading condition hold, largely, by definition of κ, Υ and f.

Lemma 4. *Let d_1 be a unitary diagram in DNF. Then either d_1 defines no language or $K(d_1) \sqcup\!\sqcup \Gamma(d_1)^* \subseteq \mathcal{L}(d_1)$.*

Proof. Let $d_1 = \perp$ then, by definition, $K(d_1) = \Gamma(d_1) = \emptyset$ and $\emptyset \sqcup\!\sqcup \emptyset = \mathcal{L}(d_1) = \emptyset$. Otherwise, let $w \in K(d_1) \sqcup\!\sqcup \Gamma(d_1)^*$. Then there exists $k = k_1k_2 \ldots k_x \in K(d_1)$ and $\gamma = \gamma_1\gamma_2 \ldots \gamma_y \in \Gamma(d_1)^*$ such that $w \in k \sqcup\!\sqcup \gamma$. Since $k \in K(d_1)$ there exists $\kappa : S(d_1) \rightarrow \Sigma$ that maps spiders to letters in a manner that respects lm. In particular, without loss of generality, κ ensure that $\kappa(s_i) = k_i$ and $\{\kappa(s_i)\} = lm(\eta(s))$, where $S(d_1) = \{s_1, s_2, \ldots, s_x\}$; from this it follows that $zone(k_i) = \eta(s_i)$. We show that any model, $m = (U, <, \Psi)$, for w also models d_1. Suppose $w = a_1a_2 \ldots a_n$ then choose a valid Υ from $\{a_1, a_2, \ldots, a_n\}$ to U.

1. The missing zones condition holds as d_1 is in DNF and, therefore, has no missing zones.
2. Define $f : S(d_1) \rightarrow U$ such that, for each $s \in S(d_1)$ by $f(s) = \{\Upsilon(\kappa(s))\}$. We now show that f is valid.
   (a) We first show the spiders' locations condition holds. Let $s \in S(d_1)$ and we have $f(s) = \{\Upsilon(\kappa(s))\}$ by definition of f. Then

   $\Upsilon(\kappa(s)) \in \Psi(zone(\kappa(s)))$ by definition of Υ,
   $\in \Psi(\eta(s))$ by definition of κ.

   Thus $f(s) \in \Psi(\eta(s))$ and the spiders' locations condition holds.
   (b) For the shading condition let $z \in ShZ(d_1)$ and assume $\Psi(z) \not\subseteq im(f)$. Then there exists $e \in \Psi(z)$ such that $e \notin im(f)$. Furthermore if $e \notin im(f)$ then $\Upsilon(a_i) = e \in \Psi(z)$ where $a \in \Gamma(d_1)$. However, by definition of $\Gamma(d_1)$, $zone(a) \notin ShZ(d_1)$ and a contradiction arises. Therefore there are no elements in $\Psi(z)$ where $z \in ShZ(d_1)$ that are not in the image of f and

   $\forall z \in ShZ(d_1)(\Psi(z) \subseteq im(f))$

   as required.

Hence Ψ is valid. Therefore m models d_1, and $K(d_1) \sqcup\!\sqcup \Gamma(d_1)^* \subseteq \mathcal{L}(d_1)$. □

Theorem 3. *For any unitary spider diagram d_1 in DNF, either d_1 defines no language or the language $K(d_1) \sqcup\!\sqcup \Gamma(d_1)^*$.*

Proof. Suppose d_1 defines a language. By Lemma 3, $\mathcal{L}(d_1) \subseteq K(d_1) \sqcup\!\sqcup \Gamma(d_1)$, and by Lemma 4, $K(d_1) \sqcup\!\sqcup \Gamma(d_1) \subseteq \mathcal{L}(d_1)$. Therefore the language of d_1 is $K(d_1) \sqcup\!\sqcup \Gamma(d_1)^*$. □

We now extend our result for unitary spider diagrams to the compound case.

Theorem 4. *For any spider diagram $D_1 = d_1 \vee \ldots \vee d_n$ in DNF, either D_1 defines no language or the language of D_1 is $\mathcal{L}(D_1) = \bigcup_{1 \le i \le n} K(d_i) \sqcup\!\sqcup \Gamma(d_i)^*$.*

Theorem 4, proved by induction over the base case provided by Theorem 3, leads us to conjecture that spider diagrams can define all languages of this form, i.e. finite unions of languages of the form $K \sqcup\!\sqcup \Gamma^*$ where K is a finite commutative language and Γ is a finite set of letters. We now show that is indeed the case.

Lemma 5. *Let $w = a_1a_2 \ldots a_n$ be a (possibly empty) word, let $K = \{a_1\} \sqcup\!\sqcup \{a_2\} \sqcup\!\sqcup \cdots \sqcup\!\sqcup \{a_n\}$ and let $\Gamma \subseteq \Sigma$. Then $K \sqcup\!\sqcup \Gamma^*$ is the language of a unitary spider diagram.*

Proof. We construct a unitary diagram $d_1 = \langle C, Z, ShZ, SI \rangle$ in DNF where $C(d_1) = \mathcal{C}$ and

$ShZ(d_1) = Z(d_1) - \{zone(a) : a \in \Gamma\}$,

$SI(d_1) = \{(|w|_a, zone(a)) : a \in \Sigma \wedge |w|_a \neq 0\}$.

Obviously $K = K(d_1)$ and $\Gamma = \Gamma(d_1)$. Therefore, by Theorem 4, $\mathcal{L}(d_1) = K \sqcup\!\sqcup \Gamma^*$. □

Consider $K = \{ab, ba, cd, dc, abb\}$ and $\Gamma = \Sigma$. We partition K into $K_1 = \{ab, ba\}, K_2 = \{cd, dc\}$ and $K_3 = \{abb, bab, bba\}$ and, as in Lemma 5, create d_8, d_9 and d_{10}, depicted in Fig. 13, such that $\mathcal{L}(d_8) = K_1 \sqcup\!\sqcup \Gamma^*, \mathcal{L}(d_9) = K_2 \sqcup\!\sqcup \Gamma^*$, and

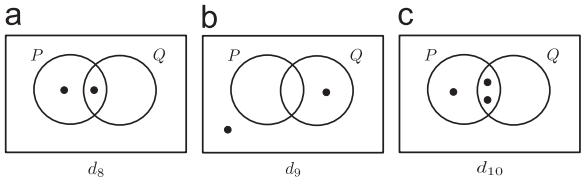

Fig. 13. Spider diagrams constructed from an arbitrary K and Γ.

$\mathcal{L}(d_{10}) = K_3 \sqcup\!\sqcup \Gamma^*$. Given $D_1 = d_8 \vee d_9 \vee d_{10}$ we have

$$\mathcal{L}(D_1) = (K_1 \sqcup\!\sqcup \Gamma^*) \cup (K_2 \sqcup\!\sqcup \Gamma^*) \cup (K_3 \sqcup\!\sqcup \Gamma^*)$$
$$= (K_1 \cup K_2 \cup K_3) \sqcup\!\sqcup \Gamma^* = K \sqcup\!\sqcup \Gamma^*.$$

The next theorem takes a finite commutative language K and a set of letters, Γ, and shows that their shuffle product is the language of a compound diagram.

Theorem 5. *Let $L_1 = K \sqcup\!\sqcup \Gamma^*$ where K is a finite commutative language and $\Gamma \subseteq \Sigma$. Then there exists a diagram D_1 such that $\mathcal{L}(D_1) = L_1$.*

Proof. If $K \neq \emptyset$ we partition K into $K_1, K_2, \ldots, K_n$ where each $K_i = \{a_1\} \sqcup\!\sqcup \{a_2\} \sqcup\!\sqcup \cdots \sqcup\!\sqcup \{a_{|w|}\}$ for some $w = a_1 a_2 \ldots a_{|w|} \in K$. By Lemma 5, for each K_i, we can construct a spider diagram, d_i, such that $\mathcal{L}(d_i) = K_i \sqcup\!\sqcup \Gamma^*$. Take D_1 to be the disjunction of all d_i's constructed from $K_i \sqcup\!\sqcup \Gamma^*$. Since

$$K \sqcup\!\sqcup \Gamma^* = \bigcup_{0 \leq i \leq n} (K_i \sqcup\!\sqcup \Gamma^*)$$

we deduce $\mathcal{L}(D_1) = K \sqcup\!\sqcup \Gamma^*$. Where $K = \emptyset$ the unitary diagram $\perp$ defines $K \sqcup\!\sqcup \Gamma^*$. □

Theorem 5 is extended to consider finite unions of languages of the form $K \sqcup\!\sqcup \Gamma^*$.

Theorem 6. *Let $L_1 = \bigcup_{i=1}^{n} (K_i \sqcup\!\sqcup \Gamma_i^*)$ where each K_i is a finite commutative language and $\Gamma_i \subseteq \Sigma$. Then there exists a diagram D_1 such that $\mathcal{L}(D_1) = L_1$.*

Proof. By Theorem 5, for each $K_i \sqcup\!\sqcup \Gamma_i^*$ there exists D_i such that $\mathcal{L}(D_i) = K_i \sqcup\!\sqcup \Gamma_i$. Then $L_1 = \mathcal{L}(\vee_{i=1}^{n} D_i)$. □

To conclude this section, from Theorems 2 and 6 we have the following corollary.

Corollary 1. *Let $L_1 = \bigcup_{i=1}^{n} (K_i \sqcup\!\sqcup \Gamma_i)$ where each K_i is a finite commutative language and $\Gamma_i \subseteq \Sigma$. Then L_1 is commutative and star-free.*

In the next section we show all commutative star-free language are of this form.

7. Characterisations of commutative star-free languages

In this section, we show that a language is commutative and star-free if and only if it is a finite union of languages of the form $K \sqcup\!\sqcup \Gamma^*$, where K is a finite commutative language and Γ is a finite set of letters. This is similar to Higman's characterisation of the shuffle ideal languages [8]. We consider minimal finite automata that accept commutative star-free languages, such as $\mathcal{A}$ in Fig. 14. We show that such an automaton can be decomposed into n automata $\mathcal{A}_1, \ldots, \mathcal{A}_n$ where n is the number of final states in $\mathcal{A}$ and the union of the languages of the component automata is the language of $\mathcal{A}$. Moreover, each $\mathcal{A}_i$ has a single final state. Given each $\mathcal{A}_i$, such as $\mathcal{A}_1$ in Fig. 14, we show that a finite commutative language, K, and a set of letters, Γ, may be extracted from $min(\mathcal{A}_i)$ where $\mathcal{L}(\mathcal{A}_i) = K \sqcup\!\sqcup \Gamma$, as demonstrated in Section 2. In order to determine the set Γ we will use Proposition 1 given in Section 3, which states that for a minimal automaton accepting a star-free language the graph $min(\mathcal{A})|_a, a \in \Sigma$ contains no non-trivial cycles. We show any letter occurring on a cycle in $min(\mathcal{A}_i)$ from which we can reach the final state, also occurs on a trivial cycle at the final state of $min(\mathcal{A}_i)$. The set Γ contains precisely the letters occurring on trivial cycles at the final state.

We begin with a restatement of the fact that the language of an automaton is the union of the sets of words accepted at each final state.

Theorem 7. *Let $\mathcal{A} = \langle Q, \Sigma, \delta, q_0, \{f_1, f_2, \ldots, f_n\} \rangle$ be a minimal finite automaton accepting a commutative star-free language L and $\mathcal{A}_1, \mathcal{A}_2, \ldots, \mathcal{A}_n$ be decomposed automata where $\mathcal{A}_i = \langle Q, \Sigma, \delta, q_0, \{f_i\} \rangle$. Then $\mathcal{L}(\mathcal{A}) = \bigcup_{i=1}^{n} \mathcal{L}(\mathcal{A}_i)$.*

Proposition 2 is used in the proof of the following theorem in order to show each $\mathcal{A}_i$ accepts a commutative language. The property states that $u \sim_L v$ if and only if for each $q \in Q$ it is the case that $\delta^*(q,u) = \delta^*(q,v)$. Moreover, it is possible to construct an epimorphism from the syntactic monoid, $\mathcal{M}_L$, of $L = \mathcal{L}(\mathcal{A})$ to that of $L_i = \mathcal{L}(\mathcal{A}_i)$, namely $\mathcal{M}_{L_i}$, to show $\mathcal{M}_{L_i}$ is both finite and aperiodic. Thus it follows that L_i is commutative and star-free.

Theorem 8. *Let $\mathcal{A} = \langle Q, \Sigma, \delta, q_0, \{f_1, f_2, \ldots, f_n\} \rangle$ be a minimal finite automaton accepting a commutative star-free language L. The automaton $\mathcal{A}$ may be decomposed into n automata, $\mathcal{A}_i = \langle Q, \Sigma, \delta, q_0, \{f_i\} \rangle$, where each $\mathcal{A}_i$ accepts a commutative star-free language L_i.*

Proof. Let $M = (\Sigma^*/\sim_L, \cdot, \lambda)$ be the syntactic monoid of L and $M_i = (\Sigma^*/\sim_{L_i}, \cdot, \lambda)$ be the syntactic monoid of L_i. We show that each $\mathcal{A}_i$ accepts a commutative star-free language by establishing that the function $\phi : M \to M_i$ defined by $\phi([u]_L) = [u]_{L_i}$ is an epimorphism.

We first prove that ϕ is well-defined by showing that if $[u]_L = [v]_L$ then $\phi([u]_L) = \phi([v]_L)$. Let $min(\mathcal{A}_i) = \langle Q_i, \Sigma, \delta_i, q_{0_i}, \{f_i'\} \rangle$ and define $g : Q \to Q_i$ where g is the natural mapping induced by the minimisation of $\mathcal{A}_i$ to $min(\mathcal{A}_i)$. Then g ensures that for all q and q' in Q, if q and q' are distinguishable in $\mathcal{A}$ but indistinguishable in $\mathcal{A}_i$ then $g(q) = g(q')$ and if q and q' are distinguishable in both $\mathcal{A}$

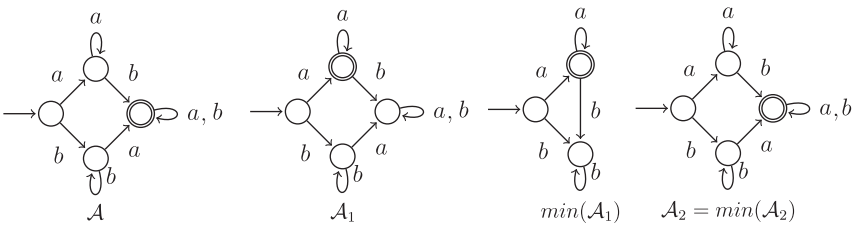

Fig. 14. Finite automata.

and $\mathcal{A}_i$ then we have $g(q) \neq g(q')$. Obviously, the function g is surjective.

As stated in Proposition 2, given any u and v in Σ^* we have $u \sim_L v$ if and only if for each $q \in Q, \delta^*(q,u) = \delta^*(q,v)$. Observing $g(\delta(q,a)) = \delta_i(g(q),a)$ it follows that for each q in Q, $\delta_i^*(g(q),u) = \delta_i^*(g(q'),v)$. Since g is surjective

$$\delta_i^*(q,u) = \delta_i^*(q,v).$$

Thus if $u \sim_L v$ then $u \sim_{L_i} v$ and $[u]_L \subseteq [v]_{L_i}$ and it follows that $[u]_L \subseteq [u]_{L_i}$. Hence $\phi([u]_L) = \phi([v]_L)$, so ϕ is well-defined.

We now show that ϕ is an epimorphism. Let $[u]_L, [v]_L \in M$ then

$$\phi([u]_L[v]_L) = \phi([uv]_L) = [uv]_{L_i} = [u]_{L_i}[v]_{L_i} = \phi([u]_L)\phi([v]_L)$$

Trivially ϕ is surjective. Hence ϕ is an epimorphism.

Since L is commutative and star-free we know that M is commutative and aperiodic. Furthermore, as $\phi : M \to M_i$ is an epimorphism, M_i is commutative and aperiodic. Hence, by Property 3, L_i is commutative and star-free.

We now proceed to derive results on automata with single final states. Since minimizing automata does not introduce any new final states, without loss of generality we can proceed by considering only minimal automata. In the previous section, for each unitary diagram d_1, in DNF we defined a finite commutative set $K(d_1)$ and a set of letters $\Gamma(d_1)$ such that $\mathcal{L}(d) = K(d_1) \sqcup\!\sqcup \Gamma(d_1)$. We now define $K(\mathcal{A})$ and $\Gamma(\mathcal{A})$ as analogous sets derived from the automaton $\mathcal{A}$.

Definition 10. Let $\mathcal{A} = \langle Q, \Sigma, \delta, q_0, \{f\} \rangle$ be a finite automaton accepting a commutative star-free language. We define $K(\mathcal{A})$ to be the set of words accepted by $\mathcal{A}$ where no cycle is followed.

In Fig. 14, $K(min(\mathcal{A}_1)) = \{a\}$ and $K(min(\mathcal{A}_2)) = \{ab, ba\}$, both of which are commutative.

Definition 11. Let $\mathcal{A} = \langle Q, \Sigma, \delta, q_0, \{f\} \rangle$ be a finite automaton accepting a star-free commutative language. We define $\Gamma(\mathcal{A})$ to be $\{a \in \Sigma : (f, a, f) \in \delta\}$.

In Fig. 14, $\Gamma(min(\mathcal{A}_1)) = \{a\}$ and $\Gamma(min(\mathcal{A}_2)) = \{a, b\}$. The language of automata $\mathcal{A}$ is $comm(K(\mathcal{A})) \sqcup\!\sqcup \Gamma(\mathcal{A})^*$, where $comm(X)$ denotes the commutative closure of X. This is established in Theorem 9 using the following key lemma:

Lemma 6. *Let $\mathcal{A} = \langle Q, \Sigma, \delta, q_0, \{f\} \rangle$ be a minimal automaton with a single final state accepting a commutative star-free language L. Let c be a cycle from which we can reach f. Then any letter that occurs on a transition that is part of c also occurs on a trivial cycle at the final state.*

Proof. Suppose $\mathcal{A}$ contains such a cycle, c, starting on state q and exercised by the word $w = a_1 a_2 \dots a_k$. Then there exists $u, v \in \Sigma^*$ such that $u \cdot w \cdot v \in L$, since $\mathcal{A}$ is minimal. Since w exercises c, we have $\delta(q_0, u) = \delta(q_0, uw)$. Then, for any $n \geq 0$, $u \cdot w^n \cdot v \in L$ by traversing, c, n times. We choose $n > |Q|$. The word $w_p^n = a_1^n a_2^n \dots a_k^n$ is a permutation of w^n and, as L is a commutative language

$$u \cdot v \cdot a_1^n a_2^n \dots a_k^n \in L. \tag{3}$$

Then $\mathcal{A}$ must contain a cycle c_{a_k} exercised by a word containing only the letter a_k, as the postfix a_k^n of w_p^n is longer than the number of states in $\mathcal{A}$. Furthermore, by Property 1, the cycle c_{a_k} is trivial. Then we can rewrite (3) as

$$u \cdot v \cdot a_1^n a_2^n \dots a_k^{x+y+z} \in L \quad \text{where } x+y+z = n, y \geq 1$$

and a_k^y exercises the trivial cycle c_{a_k}. Then we can add any number of a_ks to the word $u \cdot v \cdot a_1^n a_2^n \dots a_k^{x+y+z}$ and remain within L by traversing c more times. By commutativity, $u \cdot v \cdot w^n \cdot a_k^m \in L$ for any $m \geq 0$. Since $v \cdot v \cdot w_n \in L$, there is a trivial cycle labelled a_k at the final state, shown by choosing $m = 1$. Similarly, each letter, a_i, in w_n therefore gives rise to a trivial cycle at the final state, as required. □

Theorem 9. *Let $\mathcal{A} = \langle Q, \Sigma, \delta, q_0, \{f\} \rangle$ be a minimal automaton with a single final state accepting a commutative star-free language L. Then*

$$L = comm(K(\mathcal{A})) \sqcup\!\sqcup \Gamma(\mathcal{A})^*.$$

Proof. Let w be in the language of $\mathcal{A}$. We show $w \in comm(K(\mathcal{A})) \sqcup\!\sqcup \Gamma(\mathcal{A})^*$. If w exercises a path that gave rise to a word in $K(\mathcal{A})$ or some permutation of w exercises such a path then $w \in comm(K(\mathcal{A}))$, establishing the result. Otherwise, w exercises a path that includes at least one cycle. Let $w = u_0 v_1 u_1 \dots u_{n-1} v_n u_n$ where each v_i is a word that exercises a cycle c_i in $\mathcal{A}$ and $k = u_0 u_1 \dots u_n \in K$. Reorder the letters in w so that we obtain a word of the form kx so $kx = k v_1 v_2 \dots v_n$. Then each letter in x arises from a letter on a transition in each c_i. Therefore, each such letter is in $\Gamma(\mathcal{A})$. Hence kx is in $comm(K(\mathcal{A})) \sqcup\!\sqcup \Gamma(\mathcal{A})^*$. That is, a permutation of w is in $K \sqcup\!\sqcup \Gamma^*$. As $comm(K(\mathcal{A}))$ is commutative it follows that $comm(K(\mathcal{A})) \sqcup\!\sqcup \Gamma(\mathcal{A})^*$ is commutative (so $w \in comm(K(\mathcal{A})) \sqcup\!\sqcup \Gamma(\mathcal{A})^*$).

The converse, $K(A) \sqcup\!\sqcup \Gamma(A) \subseteq L(A)$, is shown by a similar argument. By definition $K(\mathcal{A}) \subseteq \mathcal{L}(\mathcal{A})$. By Theorem 8, $\mathcal{A}$ accepts a commutative language, therefore $comm(K(\mathcal{A})) \subseteq \mathcal{L}(\mathcal{A})$. Let $k \in comm(K(\mathcal{A}))$, $\gamma \in \Gamma(\mathcal{A})^*$ then $k\gamma \in \mathcal{L}(\mathcal{A})$ as, by definition of $\Gamma(\mathcal{A})$, for each letter γ_i in γ there is a transition $\delta(f, \gamma_i) = f$. Again, as $\mathcal{A}$ accepts a commutative language, $k \sqcup\!\sqcup \gamma \subseteq \mathcal{L}(\mathcal{A})$. Therefore $comm(K(\mathcal{A})) \sqcup\!\sqcup \Gamma(\mathcal{A})^* \subseteq \mathcal{L}(\mathcal{A})$, as required. Hence, $\mathcal{L}(\mathcal{A}) = comm(K(\mathcal{A})) \sqcup\!\sqcup \Gamma(\mathcal{A})^*$. □

We now derive a characterisation of commutative star-free languages.

Theorem 10. *Let L be a commutative star-free language over Σ. Then L is a finite union of languages of the form $K \sqcup\!\sqcup \Gamma^*$ where K is a finite commutative language and $\Gamma \subseteq \Sigma$.*

Proof. Where $L = \emptyset$ then $K = \Gamma = \emptyset$. Otherwise, Theorems 7 and 8 establish that, as L is commutative and star-free, an automaton

$$\mathcal{A} = \langle Q, \Sigma, \delta, q_0, \{f_1, f_2, \dots, f_n\} \rangle$$

accepting L may be decomposed into $\mathcal{A}_1, \dots, \mathcal{A}_n$ with $\mathcal{A}_i = \langle Q, \Sigma, \delta, q_0, \{f_i\} \rangle$ such that $\bigcup_{i=1}^n \mathcal{L}(\mathcal{A}_i) = L$. By Theorem 8, $\mathcal{L}(\mathcal{A}_i)$ is commutative and star-free. Therefore, by Theorem 9, $\mathcal{L}(\mathcal{A}_i) = comm(K(\mathcal{A}_i)) \sqcup\!\sqcup \Gamma(\mathcal{A}_i)^*$. Hence $L = \mathcal{L}(\mathcal{A}) = \bigcup_{i=1}^n (comm(K(\mathcal{A}_i)) \sqcup\!\sqcup \Gamma(\mathcal{A}_i)^*)$. □

Thus, we have considered the commutative star-free languages which we have shown to be finite unions of shuffle products $K \sqcup\!\sqcup \Gamma$, where K is a finite commutative language and $\Gamma \subseteq \Sigma$. Using the results we have derived and other results from the literature [11], we present our main theorem.

Theorem 11. *The following statements are equivalent*:

1. *L is the language of a spider diagram*,
2. *L is defined by a sentence in monadic first-order logic with equality*,
3. *L is a commutative star-free regular language, and*
4. *L is a finite union of languages of the form* $K \sqcup\!\sqcup \Gamma^*$ *where K is a finite commutative language and* $\Gamma \subseteq \Sigma$.

Proof. That $3 \Rightarrow 4$ is shown in Theorem 10 and $4 \Rightarrow 3$ is shown in Corollary 1. The proof of $4 \Leftrightarrow 1$ is given by Theorems 4 and 5. Stapleton proved that for all sentences in monadic first-order logic with equality there exists a spider diagram with the same set of models and vice-versa [20]. Therefore, by our definition of the language of a diagram $1 \Leftrightarrow 2$.

8. Related work and discussion

The literature concerning star-free regular languages provides a syntactic characterisation, a logical characterisation and an algebraic characterisation of the language class. Using the syntactic characterisation, as we have seen, star-free regular languages can be defined using a star-free regular expression. This syntactic approach has led to the definition of infinite hierarchies that characterise the class of star-free regular languages. One such infinite hierarchy, the so-called dot-depth hierarchy [1], can be syntactically defined using the empty language and the complement of the empty language as the base cases. The base cases are referred to as level 0 of the hierarchy. Level $n+\frac{1}{2}$ of the hierarchy $(n \geq 0)$ is the Boolean closure of level n and level $n+1$ of the hierarchy is defined using a closure involving the concatenation of languages at level $n+\frac{1}{2}$ of the hierarchy.

Thomas [22] proved that languages in the class of star-free regular languages are exactly those which are defined by sentences in monadic first-order logic of order. In that paper, Thomas shows that the level at which a star-free language, L, first appears in the dot-depth hierarchy is the same as the minimum number of quantifier alternations in an MFOL$[<]$ sentence, S, in prenex normal form that defines L. For instance, a language L definable by such an S drawn from $\exists^*\forall^*$ will first appear at level two in the hierarchy.

Using an algebraic approach to defining regular languages [15], Schützenberger [16] proved that the syntactic monoid of a star-free regular language is finite and aperiodic. Eilenberg [6] extended these results to consider varieties of finite monoids, and established a correspondence between varieties and well known subclasses of regular languages. More recently, Eilenberg's variety theorem has been extended to consider ordered monoids [14]. In our work, we have used these results in establishing that spider diagrams define commutative star-free regular languages as the syntactic monoid of the language defined by a spider diagram is commutative.

Shin [17] and Stapleton et al. [11], amongst others, have examined the expressiveness of diagrammatic logics. In Shin's case, she showed that Venn-II (a logic based on Venn diagrams) is exactly as expressive as monadic first-order logic, MFOL. Stapleton et al. showed that spider diagrams are exactly as expressive as monadic first order logic with equality, MFOL$[=]$. In each case, the proof strategies begin by establishing that for every diagram there is a semantically equivalent sentence in the corresponding logic by providing a syntactic translation. Thereafter, every sentence in the corresponding logic is established to be semantically equivalent to a diagram. In the Venn-II case, this strategy involves defining syntactic translations from MFOL to Venn-II. However, in the spider diagram case, a model theoretic analysis is conducted of MFOL$[=]$ sentences in order to prove that MFOL$[=]$ is no more expressive than spider diagrams. These proof techniques give us ways in which to view diagrams as sentences and vice versa. This means that we can use results from either paradigm and translate them to the other paradigm. For instance, we can now adopt theorem proving support developed for symbolic logics and utilise it for these diagrams to establish properties like semantic equivalence, as demonstrated in [23].

The connections between formal language theory and diagrammatic logic, developed in this paper, are also of practical use when performing reasoning with diagrams. Given two diagrams d_1 and d_2, Howse, Stapleton and Taylor's sound and complete reasoning system for spider diagrams can be used to determine semantic equivalence, but the algorithm to do so is computationally complex [11]; the exact complexity has not been computed, but it is far from being polynomial. The results in this paper give us another, more efficient, route to decide whether two diagrams are semantically equivalent. Suppose an automaton $\mathcal{A}(d_1)$ was constructed such that $\mathcal{A}(d_1)$ accepts the star-free regular language defined by d_1. An automaton $\mathcal{A}(d_2)$ may be similarly constructed. The classic Hopcroft algorithm [9] can be used to minimize each of $\mathcal{A}(d_1)$ and $\mathcal{A}(d_2)$ in $O(n \log(n))$, where n is the number of states in each automaton. The equivalence of minimal automaton can be checked in $O(n^{2/5})$ to decide whether d_1 is semantically equivalent to d_2 [10].

9. Conclusion

The main contributions of this paper are the development of a formal framework within which we can study spider diagrams by investigating commutative star-free languages, and various characterisations of the expressiveness of spider diagrams derived from results concerning formal languages. In particular, we have presented various characterisations of the expressiveness of spider diagrams with respect to formal languages, specifically that they define precisely the languages definable by MFOL$[=]$, the star-free regular languages, and languages that are finite unions of languages of the form $K \sqcup\!\sqcup \Gamma^*$ where K is a finite commutative language and $\Gamma \subseteq \Sigma$.

This research was originally inspired by Thomas' paper on the definability of star-free languages in MFOL[$<$] [22]. From the results in [20], it immediately follows that sentences in MFOL[$=$] are all semantically equivalent to MFOL[$<$] sentences drawn from the set of sentences in prenex normal-form with alternating quantifier blocks $\exists^*\forall^* \cup \exists^* \cup \forall^*$; one can obtain sentences in this form by converting spider diagrams in DNF to MFOL[$<$] sentences. Thus, as a consequence of the results in this paper, we can deduce that all commutative star-free languages are at level 2 of the dot-depth hierarchy, and may have appeared at level 1 or level 0. Indeed, we fully expect to be able to generalise results concerning spider diagrams in DNF to provide an effective procedure for determining the level in this hierarchy at which such a language first appears: spiders correspond to the presence of $\exists$ and shading corresponds to the presence of $\forall$, so to derive a procedure one needs to produce a 'minimal' disjunctive normal form for spider diagrams.

Future plans also include the development of a diagrammatic logic with the expressive power of monadic second-order logic, begun in [3] which has previously been shown by Büchi to define the class of regular languages [2]. In [3] the syntax of spider diagrams has be extended with an $\lhd$ operator, unlabelled curves and arrows. We believe that it may well be possible to derive new insights into the properties of regular languages via such a diagrammatic logic, akin to the results that we have presented in this paper. The different characteristics of the syntax of the various approaches to defining regular languages (using diagrams, finite automata, symbolic logics, and regular expressions) imply that the study of each can provide unique insight into properties of the others, as we have demonstrated in this paper.

Acknowledgements

This research forms part of the Defining Regular Languages with Diagrams (EPSRC Grant EP/H012311/1) and Sketching Euler Diagrams (EPSRC Grant EP/H048480/1) projects.

References

[1] Janusz A. Brzozowski, Hierarchies of aperiodic languages, RAIRO—Theoretical Informatics and Applications, 1976, pp. 33–49.

[2] J. Richard Büchi, Weak second order arithmetic and finite automata, Zeitschrift für Mathematische Logik und Grundlagen der Mathematik 6 (1–6) (1960) 66–92.

[3] Peter Chapman, Gem Stapleton, Introducing second order spider diagrams for defining regular languages, in: Proceedings of the Symposium on Visual Languages and Human-Centric Computing, IEEE, 2010, pp. 159–167.

[4] Aidan Delaney, Gem Stapleton, Spider diagrams of order, in: Proceedings of the International Workshop on Visual Languages and Logic, vol. 274, CUER, September 2007, pp. 27–39.

[5] Aidan Delaney, John Taylor, Simon Thompson, Spider diagrams of order and a hierarchy of star-free regular languages, in: Proceedings of the International Conference on the Theory and Application of Diagrams, Springer, 2008, pp. 172–187.

[6] Samuel Eilenberg, Automata, Languages, and Machines, vol. Part B, Academic Press Inc., 1974.

[7] J. Gil, John Howse, S. Kent, Formalising spider diagrams, in: Symposium on Visual Languages, IEEE Computer Society Press, September 1999, pp. 130–137.

[8] Graham Higman, Ordering by divisibility in abstract algebras, Proceedings of the London Mathematical Society s3–2 (January (1)) (1952) 326–336.

[9] John E. Hopcroft. An $n \log n$ algorithm for minimizing states in a finite automaton, Technical Report, Stanford, CA, USA, 1971.

[10] John E. Hopcroft, Richard M. Karp, An $n^{5/2}$ algorithm for maximum matchings in bipartite graphs, SIAM Journal on Computing 2 (4) (1973) 225–231.

[11] John Howse, Gem Stapleton, John Taylor, Spider diagrams, LMS Journal of Computation and Mathematics 8 (2005) 145–194.

[12] Mark V. Lawson, Finite Automata, CRC Press, 2003.

[13] Alexandru Mateescu, Arto Salomaa, Formal Languages: An Introduction and a Synopsis, Handbook of Formal Languages, Word, Language, Grammar, vol. 1, Springer-Verlag, New York, USA, 1997, pp. 1–39.

[14] Jean-Eric Pin, Syntactic Semigroups, Handbook of formal languages, Word, Language, Grammar, vol. 1, Springer-Verlag, New York, USA, 1997, pp. 679–746.

[15] Michael O Rabin, Dana Scott, Finite automata and their decision problems, IBM Journal of Research Development (1959) 114–125.

[16] Marcel-Paul Schützenberger, On finite monoids having only trivial subgroups, Information and Control (1965) 190–194.

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

[18] Gem Stapleton, Reasoning with Constraint Diagrams, PhD Thesis, University of Brighton, August 2004.

[19] Gem Stapleton, John Howse, John Taylor, A decidable constraint diagram reasoning system, Journal of Logic and Computation 15 (December (6)) (2005) 975–1008.

[20] Gem Stapleton, Simon Thompson, John Howse, John Taylor, The expressiveness of spider diagrams, Journal of Logic and Computation 14 (December (6)) (2004) 857–880.

[21] Howard Straubing, Finite Automata, Formal Logic, and Circuit Complexity, Birkhäuser, 1994.

[22] Wolfgang Thomas, Classifying regular events in symbolic logic, Journal of Computer and System Sciences 25 (1982) 360–376.

[23] Matej Urbas, Mateja Jamnik, Heterogeneous proofs: spider diagrams meet higher-order provers, in: Marko van Eekelen, Herman Geuvers, Julien Schmaltz, Freek Wiedijk (Eds.), Interactive Theorem Proving, Lecture Notes in Computer Science, vol. 6898, Springer-Verlag, 2011, pp. 376–382.