J Log Lang Inf
DOI 10.1007/s10849-016-9239-6

Some Properties of Iterated Languages

Shane Steinert-Threlkeld[1]

Abstract A special kind of substitution on languages called *iteration* is presented and studied. These languages arise in the application of semantic automata to iterations of generalized quantifiers. We show that each of the star-free, regular, and deterministic context-free languages are closed under iteration and that it is decidable whether a given regular or determinstic context-free language is an iteration of two such languages. This result can be read as saying that the van Benthem/Keenan 'Frege Boundary' is decidable for large subclasses of natural language quantifiers. We also determine the state complexity of iteration of regular languages.

Keywords Semantic automata · Generalized quantifiers · Iteration · Frege boundary · Decidability

1 Introduction

Starting with Barwise and Cooper (1981), generalized quantifiers—introduced into mathematical logic by Mostowski (1957) and Lindström (1966)—have been very

I thank Johan van Benthem, Thomas Icard, Makoto Kanazawa, Christopher Potts, and Jakub Szymanik for helpful discussions. Two anyonymous referees provided very helpful comments. An earlier version of this paper was presented at *Philosophy, Mathematics, Linguistics: Aspects of Interaction (PhML2014)* in St. Petersburg, Russia (see Steinert-Threlkeld 2014). I am grateful to the organizers and participants there. In particular, I must express deep gratitude to the late Grisha Mints, whose love of logic was matched by his warmth of heart and sense of humor. You were an inspiration.

✉ Shane Steinert-Threlkeld
shanest@stanford.edu

[1] Department of Philosophy, Stanford University, Stanford, CA, USA

Published online: 02 April 2016

fruitfully applied in the formal semantics of natural language. In particular, the meanings of determiners in sentences like

(1) Every student attends classes.
(2) At least three people will attend.
(3) Most people enjoyed the show.

have been modeled as type $\langle 1, 1\rangle$ generalized quantifiers. In other words, their denotations have been given as binary relations between subsets of a domain of discourse. For example, (3) is true iff

$$|[\![\text{people}]\!] \cap [\![\text{enjoyed}]\!]| > |[\![\text{people}]\!] \setminus [\![\text{enjoyed}]\!]|$$

In natural language, it is also common that quantified expressions occur not just as subjects of sentences but also as the object of transitive verbs:

(4) Every student takes at least three classes.
(5) Most professors teach two classes.

To model these sentences, one must *iterate* the relevant type $\langle 1, 1\rangle$ quantifiers to express a property of the respective transitive verbs. These will be type $\langle 1, 1, 2\rangle$ quantifiers: relations between two subsets (the denotations of the restrictors of the determiners) and a binary relation (the denotation of the transitive verb) on the domain of discourse. For instance, (5) is true iff $[\![\text{prof}]\!]$, $[\![\text{classes}]\!]$, and $[\![\text{teach}]\!]$ stand in the relation $\text{It}([\![\text{most}]\!], [\![\text{two}]\!])$ which will hold iff

$$|[\![\text{prof}]\!] \cap [\![\text{teach\_two\_classes}]\!]| > |[\![\text{prof}]\!] \setminus [\![\text{teach\_two\_classes}]\!]|$$

A question then arises: are all type $\langle 1, 1, 2\rangle$ quantifiers expressed in natural language iterations of unary quantifiers? Frege can be read as advocating a 'yes' answer to this question and so the line dividing iterated quantifiers from the rest of the type $\langle 1, 1, 2\rangle$ quantifiers has been dubbed *the Frege Boundary* by van Benthem (1989). Keenan (1992, 1996) does find examples of natural language sentences whose truth-conditions lie on the other side of the Frege boundary, such as:

(6) Different students answered different questions.
(7) A majority of the students read those two books.
(8) The two professors graded a total of fifty exams.

The sentence (6) is to be understood as having the following truth-conditions: any two distinct students answered distinct sets of questions.

Because there do appear to be sentences lying on the other side of the Frege boundary, one wants to know exactly which sentences these are. To this end, Keenan (1992, 1996) also provides an exact characterization of which type $\langle 1, 1, 2\rangle$ quantifiers are iterations of unary quantifiers.[1] This characterization, however, has a highly complex formulation, making it hard to apply in practice. One would like to be able to turn the characterization into an algorithm for deciding whether a given type $\langle 1, 1, 2\rangle$ quantifier

[1] Dekker (2003) generalizes these results to handle more than one iteration.

is an iteration or not. This would allow one to determine whether a given sentence with truth-conditions of the appropriate kind is equivalent to one with iterated quantifiers. But it has remained unknown whether his characterization is *effective*: given a type $\langle 1, 1, 2\rangle$ quantifier, can we decide whether it is an iteration? In other words: is the Frege boundary decidable?

In this paper, we take steps towards answering that question by using the semantic automata framework pioneered by van Benthem (1986) and recently extended to handle iterations by Steinert-Threlkeld and Icard III (2013). After a section containing notation and preliminaries in formal language and automata theory, Sect. 3 introduces semantic automata as well as an operation on languages called *iteration*, denoted by $\bullet$. The operation is so-named because if L_1 and L_2 are languages associated with type $\langle 1, 1\rangle$ quantifiers Q_1 and Q_2, then $L_1 \bullet L_2$ will be the language of the iteration of Q_1 and Q_2. Section 4 shows that the regular and star-free languages are closed under iteration. Section 5 determines the state complexity of iteration of regular languages. Then, in Sect. 6 we return to the decidability question and show that it is decidable whether a given regular language is an iteration of two languages. We conclude in Sect. 7 by discussing the prospects for extending the decidability result beyond the regular languages. In particular, we show that the deterministic context-free languages are closed under iteration and that the decidability proof carries over to these languages.

2 Preliminaries

Though some familiarity with the theory of regular languages and deterministic finite-state automata will be assumed, we here introduce basic concepts and notation. A *deterministic finite-state automaton* (DFA) is a tuple $\langle \Sigma, Q, \delta, F, q_0\rangle$ where Σ is an alphabet (a finite set of letters), Q is a set of states, $q_0 \in Q$ is the starting state, $F \subseteq Q$ is the set of final (or accepting) states, and $\delta : Q \times \Sigma \to Q$ is the transition function. We will use $\mathsf{M}, \mathsf{M}_1, \mathsf{M}_2$ to denote automata. The components of an automaton are denoted by $Q(\mathsf{M})$, $\delta(\mathsf{M})$, et cetera.

A language L is a subset of Σ^*, where Σ^* is the set of all finite sequences in Σ. Elements $w \in \Sigma^*$ are called *words*. ε denotes the empty sequence. For $w \in \Sigma^*$ for finite alphabet Σ, $w_i \in \Sigma$ denotes the character at the ith position of w. We will use L, L_1, L_2 to denote languages. A DFA accepts a language in the following sense. Define $\delta^* : Q \times \Sigma^* \to Q$ by induction on Σ^*:

$$\begin{aligned} \delta^*(q, \varepsilon) &= q \\ \delta^*(q, aw) &= \delta\left(\delta^*(q, w), a\right) \end{aligned}$$

In other words, $\delta^*(q, w)$ is the state arrived at by starting at q, transitioning according to w_1, then according to w_2, and so on through w_n, where n is the length of w. We can now define the language of an automaton M by

$$L(\mathsf{M}) = \left\{w \in \Sigma^* : \delta^*(q_0, w) \in F\right\}$$

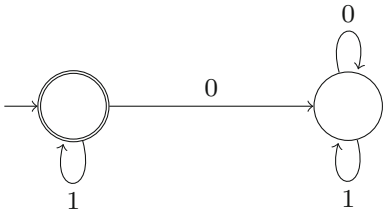

Fig. 1 The automaton min $(L_\forall)$

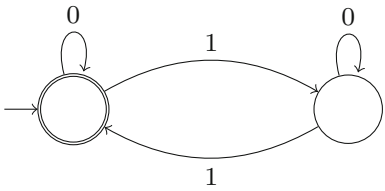

Fig. 2 The automaton min (L_{even})

The *regular languages* are the languages L such that $L = L(\mathsf{M})$ for some DFA M.[2]

We use $|\cdot|$ as both a set cardinality function and a word-length function. We often write $|\mathsf{M}|$ instead of $|Q(\mathsf{M})|$. The functions $\#_a : \Sigma^* \to \mathbb{N}$ for each $a \in \Sigma$ are recursively defined in such a way to return the number of as in a word $w \in \Sigma^*$. We write

$$\text{sgn}(q, \mathsf{M}) = \chi_{F(\mathsf{M})}(q)$$

and sgn (M) as an abbreviation for sgn $(q_0(\mathsf{M}), \mathsf{M})$ where χ_S is the characteristic function of set S. In other words, sgn (q, M) is 1 if q is an accepting state of M, 0 otherwise. We omit the second argument when context permits. If L is a regular language, we denote the minimal automaton accepting L by $\min(L)$.

For an example, consider the language $L_\forall = \left\{w \in \{0,1\}^* : \#_0(w) = 0\right\}$ of words in $\{0,1\}^*$ containing all and only 1s. This language is regular; its minimal automaton is depicted in Fig. 1. The choice of name for this language will become clear in the next section. In this diagram and others like it later in the paper, states are denoted by circles. q_0 is the state with an arrow leading in to it. States in F have two circles. An arrow from state q to q' labeled by a means that $\delta(q, a) = q'$.

The *star-free languages* in Σ is the smallest set of languages which contains Σ^*, $\{a\}$ for each $a \in \Sigma$ and which is closed under finite union, concatenation, and complementation. These languages are accepted by the *acyclic* or counter-free DFAs; see McNaughton and Papert (1971). Thus, every star-free language is regular. The converse, however, is not true: a paradigmatic regular language which is not star-free is $L_{\text{even}} = \left\{w \in \{0,1\}^* : \#_1(w) \text{ is even}\right\}$. The minimal automaton accepting this language is depicted in Fig. 2. A similar construction shows that $L_{/m} = \left\{w \in \{0,1\}^* : \#_1(w) \text{ is divisible by } m\right\}$ is also regular and that its minimal automaton has m states.

[2] Many alternative characterizations exist: acceptance by a non-deterministic finite-state automaton and generation by a regular expression, for example. See Hopcroft and Ullman (1979) for details.

We will use another characterization of the star-free languages in terms of first-order definability.[3] Given an alphabet Σ, consider a standard first-order language with unary predicate symbols P_a for each $a \in \Sigma$, a binary relation symbol $<$, and a countably infinite set of variables denoted VAR. We write Form (Σ) and Sent (Σ) for, respectively, the set of first-order formulas and sentences in this signature. We interpret formulas in this language in words $w \in \Sigma^*$ with respect to variable assignments $g : \text{VAR} \to \{1, 2, \ldots, |w|\}$. The relevant semantic clauses are given by

$$w, g \models x_i < x_j \text{ iff } g(x_i) < g(x_j)$$
$$w, g \models P_a(x_i) \text{ iff } w_{g(x_i)} = a$$

A first-order sentence φ defines the language

$$L_\varphi = \{w \in \Sigma^* \mid w \models \varphi\}$$

where the variable assignment g is omitted since a sentence is satisfied with respect to some g iff it is with respect to all g. The following theorem was mentioned above.

Theorem 1 (McNaughton and Papert 1971) *A language $L \subset \Sigma^*$ is star-free iff it is first-order definable in the following sense: there is a sentence $\varphi \in$ Sent (Σ) such that $L = L_\varphi$.*

3 Semantic Auomata and Iterated Languages

We will now see how these tools can be used to study quantification in natural language. In formal semantics of natural language, the denotations of the determiners 'every', 'at least three', and 'most' in examples (1)–(3) above have been given as type $\langle 1, 1\rangle$ generalized quantifiers.[4] A type $\langle 1, 1\rangle$ generalized quantifier is a class of finite models of the form $\langle M, A, B\rangle$ with $A, B \subseteq M$. For example, the determiners above have the following denotations:[5]

$$\forall = \{\langle M, A, B\rangle : A \subseteq B\}$$
$$\geq_3 = \{\langle M, A, B\rangle : |A \cap B| \geq 3\}$$
$$\text{most} = \{\langle M, A, B\rangle : |A \cap B| > |A \setminus B|\}$$

Semantic automata theory shows how such denotations can be given corresponding formal languages and automata accepting these languages. This works as follows.

[3] See Diekert and Gastin (2007) for a self-contained presentation of this equivalence and others.

[4] Barwise and Cooper (1981) pioneered this approach. See Peters and Westerståhl (2006) for a comprehensive overview. Szabolcsi (2010) is an overview of quantifiers from a linguistic perspective, including some problems for the generalized quantifier approach.

[5] Note that I am using the symbols $\forall$ and $\exists$ to denote type $\langle 1, 1\rangle$ generalized quantifiers whereas they are standardly used to denote type $\langle 1\rangle$ quantifiers. They are, however, intimately related: the $\langle 1, 1\rangle$ ones here are the *relativizations* of the normal type $\langle 1\rangle$ quantifiers. See §4.4 of Peters and Westerståhl (2006).

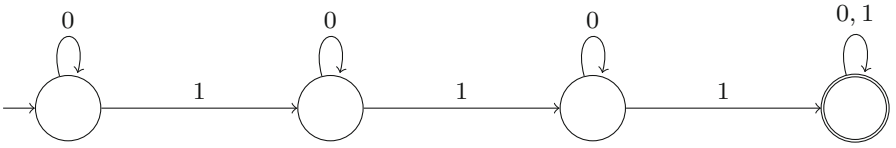

Fig. 3 The automaton min $(L_{\geq 3})$

Under standard assumptions about generalized quantifiers—conservativity and extension[6]—it follows that $\langle M, A, B\rangle \in Q$ iff $\langle A, A, A \cap B\rangle \in Q$. Now, given an enumeration $\mathbf{a}$ of A, a word in alphabet $\{0, 1\}$ corresponding to such a model can be defined by

$$(\tau\,(\mathbf{a}, B))_i = \begin{cases} 0 & a_i \in A \setminus B \\ 1 & a_i \in A \cap B \end{cases}$$

The *language of* Q—denoted L_Q—is then defined as the image of Q under τ; that is, as the set of all strings that can be generated from models in Q (together with an enumeration of A) by τ. One can verify that $L_\forall$ from the previous section is the language of $\forall$ in precisely this sense because $A \subseteq B$ iff $|A \setminus B| = 0$ iff $\#_0\,(\tau\,(\mathbf{a}, B)) = 0$ for any enumeration $\mathbf{a}$. Similarly, we have

$$L_{\geq 3} = \left\{w \in \{0, 1\}^* : \#_1\,(w) \geq 3\right\}$$
$$L_{\text{most}} = \left\{w \in \{0, 1\}^* : \#_1\,(w) > \#_0\,(w)\right\}$$

Note that the quantifier $\geq_3$ and its corresponding language are instances of a schema $\geq_n$ for any n. In particular, $\geq_1$ is the standard denotation for 'some' and so will be denoted $\exists$; the corresponding language is $L_\exists = \left\{w \in \{0, 1\}^* : \#_1\,(w) \geq 1\right\}$.

Example 1 Consider a model given as follows: $M = B = \{a, b, c, d\}$, $A = \{a, b, c\}$ with $[\![\text{student}]\!] = A$ and $[\![\text{attended}]\!] = B$. Clearly, (1) will be true in this model with this interpretation, i.e. $M \in \forall$. For any ordering $\mathbf{a}$, we have that $\tau\,(\mathbf{a}, B) = 111$, which is in $L_\forall$. The automaton in Fig. 1 can thus be seen as a verifier of the truth of sentences of the form 'Every A is a B'.

Example 2 Figure 3 shows an automaton accepting $L_{\geq 3}$.

Definability of a generalized quantifier in certain logics corresponds to the type of automaton recognizing its language as the following results show.

Theorem 2 (van Benthem 1986) *Q is definable in first-order logic iff L_Q is star-free.*

Theorem 3 (Mostowski 1998) *L_Q is regular iff Q is definable in first-order logic augmented with all divisibility quantifiers $D_n = \{\langle M, A\rangle : |A|$ is divisible by $n\}$.*

[6] Conservativity states that $\langle M, A, B\rangle \in Q$ iff $\langle M, A, A \cap B\rangle \in Q$. Extension states that if $\langle M, A, B\rangle \in Q$, then $\langle M', A, B\rangle \in Q$ for any $M' \supseteq M$.

Corollary 1 L_{most} *is not regular.*

Moreover, both neuroimaging and behavioral experiments have shown that competent speakers verify the truth of sentences with first-order definable quantifiers differently than sentences with determiners like 'most' in a way that seems to track the difference in automata needed.[7]

While the framework as described has proven fruitful both logically and empirically, the above methods can only handle sentences of the form '$Q\ A\ B$'. A natural larger fragment of natural language comes from considering examples like (4) and (5) in which a quantified noun phrase appears as both subject and object of a transitive verb. These sentences can be given truth-conditions using *iteration* of quantifiers. This operation takes two type $\langle 1, 1\rangle$ quantifiers and generates a new type $\langle 1, 1, 2\rangle$ quantifier as follows:

$$\text{It}\,(Q_1, Q_2) = \{\langle M, A, B, R\rangle : \langle M, A, \{x \in M : \langle M, B, R_x\rangle \in Q_2\}\rangle \in Q_1\}$$

where $R_x = \{y : Rxy\}$. To see how this works, consider again

(4) Every student takes at least three classes.

It $(\forall, \geq_3)$ will be the class of models $\langle M, A, B, R\rangle$ such that $A \subseteq \{x : |B \cap R_x| \geq 3\}$. Interpreting (4) in such a model, we find that the sentence is true iff

$$[\![\text{student}]\!] \subseteq \left\{x : |[\![\text{class}]\!] \cap [\![\text{take}_x]\!]| \geq 3\right\}$$

which says that each student is such that the number of classes taken by that student is at least three.

To extend the semantic automata framework to handle iterated quantifiers, one must show how to define formal languages corresponding to iterations. Steinert-Threlkeld and Icard III (2013) take an approach modeled on the previous case: they define an encoding $\tau\ (\mathbf{a}, \mathbf{b}, R)$ for finite models of the form $\langle M, A, B, R\rangle$ (with orderings for A and B) as strings in an alaphabet $\{0, 1, \boxdot\}$. A slightly different approach is pursued here: because $\text{It}(Q_1, Q_2)$ is 'built from' Q_1 and Q_2, we provide a definition of a language for iterations in terms of two given languages. In particular, given a language L in some alphabet Σ, define the function $\sigma_L : \{0, 1\} \to \mathcal{P}\left((\Sigma \cup \{\boxdot\})^*\right)$ by

$$1 \mapsto L \cdot \{\boxdot\} \qquad 0 \mapsto L^c \cdot \{\boxdot\}$$

where $L^c = \Sigma^* \setminus L$ and $\boxdot$ is a fresh symbol not in Σ. Here, $\cdot$ is concatenation of languages. For example, $L \cdot \{\boxdot\} := \{w\boxdot : w \in L\}$. Given $L_1, L_2 \subseteq \{0, 1\}^*$, we are interested in languages of the form

$$L = \sigma_{L_2}[L_1]$$

[7] See McMillan et al. (2005) for a neuroimaging experiment and Szymanik and Zajenkowski (2010) for a behavioral one. Clark (2011) is a useful overview of this literature.

where σ_{L_2} is extended to a substitution on the whole language L_1 in the usual way.[8] We call this language *the iteration of* L_1 *and* L_2 and denote it by $L_1 \bullet L_2$. This language is aptly named: if L_1 and L_2 are the languages of type $\langle 1, 1\rangle$ quantifiers Q_1 and Q_2, then $L_1 \bullet L_2$ will be the language of the iteration of Q_1 and Q_2.[9,10]

Example 3 We can apply this definition to sentence (4), by observing:

$$L_\forall \bullet L_{\geq 3} = \left\{w \in (w_i \boxdot)^* : |\{w_i : \#_1 (w_i) < 3\}| = 0\right\}$$

where w_i ranges over maximal subwords of w containing only 0s and 1s. Consider the following model and interpretation (where the s_i are the students, the c_i are the classes, and a ✓ means that $\langle s_i, c_j\rangle \in [\![\text{take}]\!]$):

	c_1	c_2	c_3	c_4
s_1	✓	✓	✓	✓
s_2	✓		✓	✓
s_3	✓	✓		✓

The encoding from Steinert-Threlkeld and Icard III (2013) will generate the string $1111 \boxdot 1011 \boxdot 1101 \boxdot$ which is in $L_\forall \bullet L_{\geq 3}$. This is at it should be: every student does take at least three classes.

Now that we have shown how to define formal languages for iterations of quantifiers, we can turn towards addressing our motivating question: is it decidable whether a given type $\langle 1, 1, 2\rangle$ quantifier is an iteration? En route to answering this question, we will prove the closure of the regular and star-free and languages under iteration and define automata for iterated languages in order to establish the state-complexity of iteration. After answering the decidability question for regular languages, we look at semantic automata for iterations of context-free and deterministic context-free languages. These languages are needed for sentences like (3) and (5) with proportional quantifiers.

4 Closure

Fact 1 The regular languages are closed under iteration. In other words, if L_1 and L_2 are regular, then $L_1 \bullet L_2$ is regular.

Proof This follows immediately from the closure of the regular languages under complement and substitution. □

[8] That is: $\sigma_{L_2}(\varepsilon) = \varepsilon$, $\sigma_{L_2}(aw) = \sigma_{L_2}(a)\,\sigma_{L_2}(w)$ and $\sigma_{L_2}[L_1] = \bigcup_{w \in L_1} \sigma_{L_2}(w)$.

[9] See Steinert-Threlkeld and Icard III (2013) and references therein. The present definition of iteration can be seen as a more concise representation of their Definition 8.

[10] Szymanik et al. (2013) contains a preliminary experiment looking at processing consequences of semantic automata for iterated quantifiers.

The proof for star-free langauges will have to be more complex because this class of languages is not closed under substitution in general. As an example, let $\Sigma = \{1\}$ and consider $L_1 = \{1\}^*$, $L_2 = \{11\}$ and $\sigma\ (1) = L_2$. Then $\sigma\ [L_1] = \{w \mid \#_1\ (w)\ \text{is even}\}$, which we have seen is quintessentially not star-free.

Proposition 1 *The star-free languages are closed under iteration.*

Proof Let $\varphi_1(P_0, P_1)$ and $\varphi_2(P_0, P_1)$ be first-order sentences defining L_1 and L_2 (see Theorem 1). We will use the $\boxdot$ symbol, with its corresponding predicate $P_{\boxdot}$ to convert predications in φ_1 into quantifications over words in $\{0, 1\}^*$. We need the following string of defined symbols:

$$\begin{aligned}
\text{prev}\left(x_j\right) = x_k &:= \left(x_k < x_j \wedge \forall x_i \left(x_i < x_j \rightarrow x_i \leq x_k\right)\right) \\
&\quad \vee \left(x_k = x_j \wedge \neg\exists x_i \left(x_i < x_j\right)\right) \\
\text{next}\left(x_j\right) = x_k &:= \left(x_j < x_k \wedge \forall x_i \left(x_j < x_i \rightarrow x_k \leq x_i\right)\right) \\
&\quad \vee \left(x_k = x_j \wedge \neg\exists x_i \left(x_j < x_i\right)\right) \\
\text{Start}\left(x_j\right) &:= \text{prev}\left(x_j\right) = x_j \vee P_{\boxdot}\left(\text{prev}\left(x_j\right)\right) \\
\text{End}\left(x_j\right) &:= \text{next}\left(x_j\right) = x_j \vee P_{\boxdot}\left(\text{next}\left(x_j\right)\right) \\
\text{Word}\left(x_j, x_k\right) &:= x_j \leq x_k \wedge \text{Start}\left(x_j\right) \wedge \text{End}\left(x_k\right) \\
&\quad \wedge \forall x_i \left(x_j < x_i < x_k \rightarrow \neg P_{\boxdot}\left(x_i\right)\right)
\end{aligned}$$

Inspection of these definitions shows that $w, g \models \text{Word}\left(x_j, x_k\right)$ iff the subword $w_{g(x_j)} \cdots w_{g(x_k)}$ is a maximal sub-word of w containing only 0s and 1s.

Given a formula φ, we denote by $\varphi^{[x_i, x_j]}$ the formula obtained by gaurding all quantifiers in φ over x with $x_i \leq x \leq x_j$. We will now define a translation

$$\tau : \text{Form}\left(\{0, 1\}\right) \times \text{Form}\left(\{0, 1\}\right) \rightarrow \text{Form}\left(\{0, 1, \boxdot\}\right)$$

In the definition below, we stipulate that if $i \neq j$, then $i_k \neq j_k$ for $k \in \{1, 2\}$.

$$\begin{aligned}
\tau\left(x_i = x_j, \varphi_2\right) &= x_{i_1} = x_{j_1} \wedge x_{i_2} = x_{j_2} \\
\tau\left(x_i < x_j, \varphi_2\right) &= x_{i_2} < x_{j_1} \\
\tau\left(P_1\left(x_i\right), \varphi_2\right) &= \varphi_2^{\left[x_{i_1}, x_{i_2}\right]} \\
\tau\left(P_0\left(x_i\right), \varphi_2\right) &= \neg\varphi_2^{\left[x_{i_1}, x_{i_2}\right]} \\
\tau\left(\neg\varphi_1, \varphi_2\right) &= \neg\tau\left(\varphi_1, \varphi_2\right) \\
\tau\left(\varphi_1 \wedge \psi_1, \varphi_2\right) &= \tau\left(\varphi_1, \varphi_2\right) \wedge \tau\left(\psi_1, \varphi_2\right) \\
\tau\left(\exists x_i \varphi_1, \varphi_2\right) &= \exists x_{i_1} \exists x_{i_2} \left(\text{Word}\left(x_{i_1}, x_{i_2}\right) \wedge \tau\left(\varphi_1, \varphi_2\right)\right)
\end{aligned}$$

It's easy to see that if φ_1 and φ_2 are sentences, then so too is $\tau\left(\varphi_1, \varphi_2\right)$.

Claim *Let* $\varphi_1, \varphi_2 \in Sent\,(\{0,1\})$. *Then*

$$L_{\tau(\varphi_1,\varphi_2)} = L_{\varphi_1} \bullet L_{\varphi_2}$$

Proof Put φ_1 in prenex normal form and generate $\tau\,(\varphi_1, \varphi_2)$ 'from the outside in'. Quantifiers over positions in words in L_1 are converted into quantifiers over maximal subwords in $\{0,1\}^*$. The translation of $<$ ensures that the order of the subwords reflects the order of the characters. The translation of P_0 and P_1 ensure that 0s are replaced by words in $L^c_{\varphi_2}$ and 1s by words in L_{φ_2}. But that is just the definition of $L_{\varphi_1} \bullet L_{\varphi_2}$. □

By Theorem 1, this shows that $L_1 \bullet L_2$ is star-free. □

Example 4 $\varphi_\forall := \forall x P_1\,(x)$ defines $L_\forall$ and $\varphi_\exists := \exists x P_1\,(x)$ defines $L_\exists$. We have that

$$\begin{aligned}
\tau\,(\varphi_\forall, \varphi_\exists) &= \forall x_1 \forall x_2\,(\text{Word}\,(x_1, x_2) \to \tau\,(P_1(x), \varphi_\exists)) \\
&= \forall x_1 \forall x_2 \left(\text{Word}\,(x_1, x_2) \to \varphi_\exists^{[x_1,x_2]}\right) \\
&= \forall x_1 \forall x_2\,(\text{Word}\,(x_1, x_2) \to \exists x\,(x_1 \leq x \leq x_2 \wedge P_1\,(x)))
\end{aligned}$$

which clearly defines $L_\forall \bullet L_\exists$.

5 State Complexity

The state complexity of a (binary) operation O on regular languages is the number of states sufficient and necessary in the worst case for a DFA M to accept $O(L_1, L_2)$ given DFAs M_1 and M_2 for the languages L_1 and L_2; see Yu et al. (1994), Cui et al. (2012). In our case, this requires knowing how to build an automaton for an iterated language out of automata for the two given languages. To rule out certain edge cases (see Example 5 below), we need one helper definition.

Definition 1 We will define the *one-step unraveling of* M, denoted M^+. If $\delta\,(q_0, c) \neq q_0$ for all $c \in \Sigma$, then $\mathsf{M}^+ = \mathsf{M}$. Otherwise:

- $Q^+ = \{*\} \cup Q\,(\mathsf{M})$
- $q_0^+ = *$
- $F^+ = F\,(\mathsf{M}) \cup \begin{cases} \{*\} & \text{if sgn}\,(\mathsf{M}) = 1 \\ \emptyset & \text{otherwise} \end{cases}$
- $\delta^+ = \delta\,(\mathsf{M}) \cup \{\langle *, c, q\rangle \mid \langle q_0, c, q\rangle \in \delta\,(\mathsf{M})\}$

Essentially, if the start state has any loops, we unwind those loops by one step. It's easy to verify that the above definition does not change the language accepted.

Lemma 1 $L\left(\mathsf{M}^+\right) = L\,(\mathsf{M})$

We are now in a position to define an automaton for the iteration of two languages. First, we intuitively describe how it works. Let M_1 and M_2 be DFAs for the two languages. By the definition of iteration, we want to replace 1 transitions in M_1 by

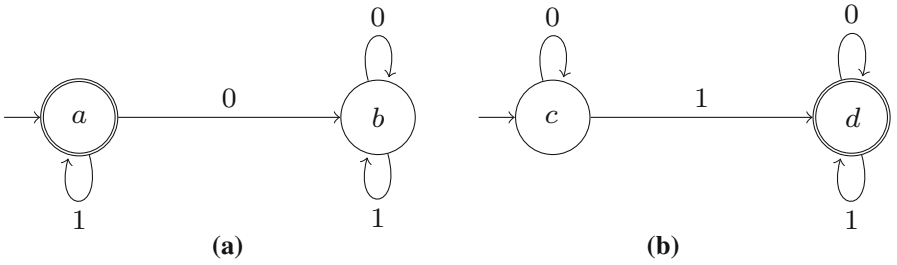

Fig. 4 The minimal automaton for **a** $L_\forall$, **b** $L_\exists$

accepting runs of M_2. We do this as follows: at each state of M_1, we attach a copy of M_2^+ and remove all 0 and 1 transitions from M_1. Now, any time M_1 would have made a 1 transition from q to q', we add a ⊡ transition from all of the accepting states of the q copy of M_2^+ to the start state of the q' copy of M_2^+. This works because reading a ⊡ while in an accepting state of M_2^+ means the previous subword is done and has been accepted, corresponding to a 1 in M_1; the machine then goes to the start state of the q' copy of M_2^+ to begin processing the next subword. Similarly, 0 transitions in M_1 are replaced by ⊡ transitions from rejecting states of the relevant copy of M_2^+. These intuitions are made precise in the following definition, after which some examples are given.

Definition 2 (*Iteration automaton*) Let M_1 and M_2 be DFAs in alphabet $\{0, 1\}$. We define *the iteration of* M_1 *and* M_2, denoted $\mathsf{It}(\mathsf{M}_1, \mathsf{M}_2)$ as follows:

- $\Sigma = \{0, 1, ⊡\}$
- $Q = Q(\mathsf{M}_1) \times Q\left(\mathsf{M}_2^+\right)$
- $q_0 = \langle q_0(\mathsf{M}_1), q_0\left(\mathsf{M}_2^+\right)\rangle$
- $F = F(\mathsf{M}_1) \times \left\{q_0\left(\mathsf{M}_2^+\right)\right\}$
- Transition function:[11]

$$\delta = \left\{\left\langle\langle q, q_1\rangle, c, \langle q, q_2\rangle\right\rangle \mid q \in Q(\mathsf{M}_1) \text{ and } \langle q_1, c, q_2\rangle \in \delta\left(\mathsf{M}_2^+\right)\right\} \quad (1)$$

$$\cup \left\{\left\langle\langle q_1, q\rangle, ⊡, \langle q_2, q_0\left(\mathsf{M}_2^+\right)\rangle\right\rangle \mid \langle q_1, \mathrm{sgn}(q, \mathsf{M}_2^+), q_2\rangle \in \delta(\mathsf{M}_1)\right\} \quad (2)$$

Example 5 Figure 4 shows the minimal automata for $L_\forall$, $L_\exists$, Fig. 5 shows $\mathsf{It}(\mathsf{M}_\exists, \mathsf{M}_\forall)$, and Fig. 6 shows $\mathsf{It}(\mathsf{M}_\forall, \mathsf{M}_\exists)$. The states are labelled to enhance readability; the pair (a, b) is abbreviated ab.

Both iteration machines illustrate the need to use one-step unraveling. In the machine $\mathsf{It}(\mathsf{M}_\exists, \mathsf{M}_\forall)$ (Fig. 5), if $\mathsf{M}_\forall$ were not unraveled, the word 111⊡11, which does

[11] This definition can be re-written in more traditional function notation:

$$\delta(\langle q_1, q_2\rangle, c) = \begin{cases} \left\langle q_1, \delta\left(\mathsf{M}_2^+\right)(q_2, c)\right\rangle & c \in \{0, 1\} \\ \left\langle \delta(\mathsf{M}_1)\left(q_1, \mathrm{sgn}\left(q, \mathsf{M}_2^+\right)\right), q_0\left(\mathsf{M}_2^+\right)\right\rangle & c = ⊡ \end{cases}$$

Some readers may find this definition easier to comprehend.

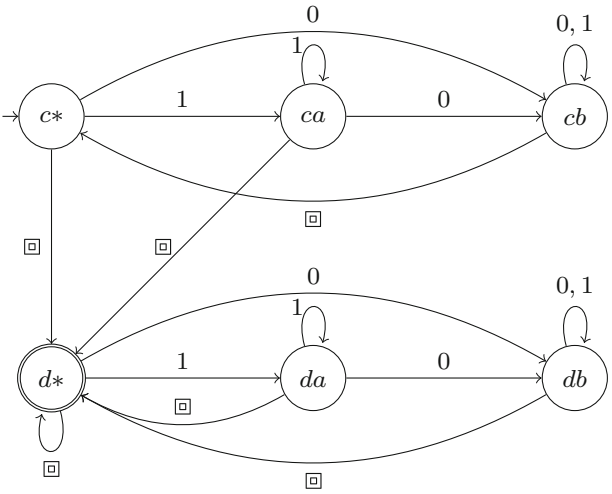

Fig. 5 The automaton $\mathsf{It}\left(\min\left(L_{\exists}\right), \min\left(L_{\forall}\right)\right)$

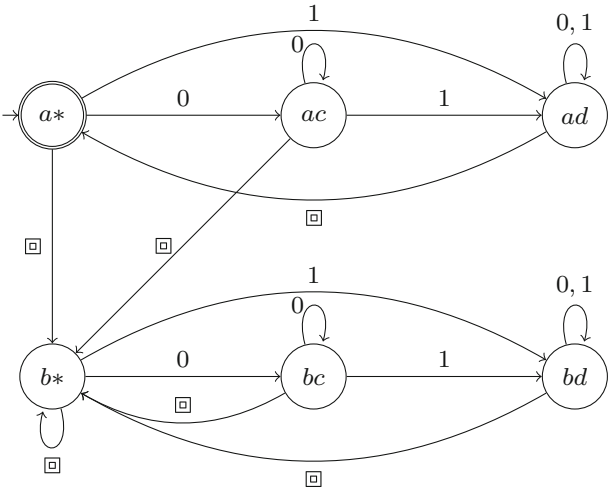

Fig. 6 The automaton $\mathsf{It}\left(\min\left(L_{\forall}\right), \min\left(L_{\exists}\right)\right)$

not end in ⧈, would be accepted. Similarly, if $\mathsf{M}_{\exists}$ were not unraveled in $\mathsf{It}\left(\mathsf{M}_{\forall}, \mathsf{M}_{\exists}\right)$, then $0^* \subset L\left(\mathsf{It}\left(\mathsf{M}_{\forall}, \mathsf{M}_{\exists}\right)\right)$, which we want to prevent.

Fact 2 $\left|\mathsf{It}\left(\mathsf{M}_1, \mathsf{M}_2\right)\right| = \left|\mathsf{M}_1\right| \cdot \left|\mathsf{M}_2^+\right| \leq \left|\mathsf{M}_1\right| \cdot \left|\mathsf{M}_2 + 1\right|$

Proposition 2 $L\left(\mathsf{It}\left(\mathsf{M}_1, \mathsf{M}_2\right)\right) = L\left(\mathsf{M}_1\right) \bullet L\left(\mathsf{M}_2\right)$

Proof For simplicity, in this proof we write L_1 and L_2 for $L\left(\mathsf{M}_1\right)$ and $L\left(\mathsf{M}_2\right)$ and It for $\mathsf{It}\left(\mathsf{M}_1, \mathsf{M}_2\right)$.

$\supseteq$: Write $L_1^n = \left\{w \in \{0,1\}^n \mid w \in L_1\right\}$. We show by induction on n that $L_1^n \bullet L_2 \subseteq L\left(\mathsf{It}\right)$ for all n.

The base case is $n = 0$. We have that L_1^0 is either $\emptyset$ or $\{\varepsilon\}$. The former case is trivial. For the latter case, we have that $L_1^0 = \{\varepsilon\}$ iff $\mathrm{sgn}\,(\mathsf{M}_1) = 1$ iff $q_0\,(\mathsf{M}_1) \in F\,(\mathsf{M}_1)$. By the definition of the iteration automaton, this holds iff $q_0\,(\mathsf{It}) \in F\,(\mathsf{It})$ iff $\varepsilon \in L\,(\mathsf{It})$. Since $L_1^0 \bullet L_2 = \{\varepsilon\}$ iff $L_1^0 = \{\varepsilon\}$, this completes the base case.

Now, suppose that $L_1^n \bullet L_2 \subseteq L\,(\mathsf{It})$ and let $w \in L_1^{n+1} \bullet L_2$. This means that there is a word $w' = a_1 \ldots a_{n+1} \in L_1$ and $w_i \in \{0, 1\}^*$ such that $w = w_1 \boxdot \ldots w_{n+1} \boxdot$ with $w_i \in L_2$ iff $a_i = 1$.

By clause (2) of the definition of $\delta\,(\mathsf{It})$, the run on $w_1 \boxdot \ldots w_n \boxdot$ ends in a state $\langle q, q_0\left(\mathsf{M}_2^+\right)\rangle$. By our inductive hypothesis, this state is in $F\,(\mathsf{It})$ iff $a_1 \ldots a_n \in L_1$. Now, we have that $w_{n+1} \in L_2$ iff $a_{n+1} = 1$. Moreover, by clause (1) of the definition of $\delta\,(\mathsf{It})$ and Lemma 1, reading w_{n+1} will lead to a state $\langle q, q_2\rangle$ such that $\mathrm{sgn}\left(q_2, \mathsf{M}_2^+\right) = 1$ iff $w_{n+1} \in L_2$. In other words, $\mathrm{sgn}\left(q_2, \mathsf{M}_2^+\right) = a_{n+1}$. Since, by assumption, $a_1 \ldots a_{n+1} \in L_1$, we know that $\langle q, a_{n+1}, q_1\rangle \in \delta\,(\mathsf{M}_1)$ for some $q_1 \in F\,(\mathsf{M}_1)$. Then clause (2) of the definition of $\delta\,(\mathsf{It})$ yields a transition $\langle\langle q, q_2\rangle, \boxdot, \langle q_1, q_0\left(\mathsf{M}_2^+\right)\rangle\rangle$. This latter state, by definition, is in $F\,(\mathsf{It})$, so $w \in L\,(\mathsf{It})$, as desired.

This completes the induction. Now, $w \in L_1 \bullet L_2$ iff $w \in L_1^n \bullet L_2$ for some n, whence it follows that $L_1 \bullet L_2 \subseteq L\,(\mathsf{It})$.

$\subseteq$: The definitions of iteration automaton and one-step unraveling ensure that It accepts only words of the form $(w_i \boxdot)^*$ where $w_i \in \{0, 1\}^*$. By reasoning similar to the previous direction, we have that

$$\langle\langle q_1, q_0\left(\mathsf{M}_2^+\right)\rangle, w_i \boxdot, \langle q_2, q_0\left(\mathsf{M}_2^+\right)\rangle\rangle \in \delta^*\,(\mathsf{It}) \Leftrightarrow \langle q_1, \chi_{L_2}\,(w_i), q_2\rangle \in \delta\,(\mathsf{M}_1)$$

where δ^* is as defined in Sect. 2. Combined with the definition of $F\,(\mathsf{It})$, this shows that $w \in L\,(\mathsf{It}) \Rightarrow w \in L_1 \bullet L_2$. □

Inspection of the definition above shows that if M_1 and M_2 are counter-free (and thus accept star-free languages), then so too is the iterated machine. This provides an alternative proof of Proposition 1.

The above proposition and fact show that given DFAs with m and n states, there is a DFA with $m \cdot (n + 1)$ states which accepts the iteration of the languages of the given DFAs. Moreover, this number of states is necessary in the worst case. The following fact can be verified using standard techniques from automata theory (e.g. by counting the number of equivalence classes of the Myhill-Nerode equivalence relation). It provides an exact characterization of the state complexity of iteration.

Proposition 3 *The minimal automaton accepting $L_{/m} \bullet L_{\geq n-1}$ has $m \cdot (n + 1)$ states, where the minimal automata accepting $L_{/m}$ and $L_{\geq n-1}$ have m and n states respectively.*

Theorem 4 *Iteration has state complexity $m \cdot (n + 1)$.*

6 Decidability

We now pursue the original question: given a language $L \subseteq \{0, 1, \boxdot\}^*$, is it decidable whether or not there are languages L_1 and L_2 such that $L = L_1 \bullet L_2$? We start with the regular case.

Theorem 5 (Decidability) *Let $L \subseteq \{0, 1, \boxdot\}^*$ be a regular language. Then it is decidable whether there are regular languages $L_1, L_2 \subseteq \{0, 1\}^*$ such that $L = L_1 \bullet L_2$.*

The proof works as follows: first, a case distinction is made between L having a certain 'uniformity' property or not. It's shown to be decidable which case obtains. Then, in each case, languages L_1 and L_2 are extracted such that L is an iteration iff it is the iteration of L_1 and L_2. All of the methods used for extraction are effective and language equality is decidable, so this suffices to prove the main result.

Proof [12] Let $L \subseteq \{0, 1, \boxdot\}^*$ be a regular language. Write $S_n := \left(\{0, 1\}^* \boxdot\right)^n$ and $S_* := \left(\{0, 1\}^* \boxdot\right)^*$. There are two cases:

$$\forall n \geq 1, w, w' \in S_n, \ w \in L \Leftrightarrow w' \in L \tag{1}$$

$$\exists n \geq 1, w, w' \in S_n, \ w \in L \text{ and } w' \notin L \tag{2}$$

First, we can decide which case holds. Let $h : \{0, 1, \boxdot\} \to \{0, 1, \boxdot\}^*$ be the string homomorphism given by

$$\begin{aligned} h(0) = h(1) &= \varepsilon \\ h(\boxdot) &= 0\boxdot \end{aligned}$$

It is clear that (1) holds iff $L \cap S_* = h^{-1}(L) \cap S_*$. Because automata for intersection and inverse homormosphism can be effectively constructed and language equality is decidable, we can check the right-hand side of this equivalence.

If (1) holds, we proceed as follows. Define the homomorphism $g : \{0, 1\} \to \{0, 1, \boxdot\}^*$ by

$$g(0) = g(1) = 0\boxdot$$

and let

$$\begin{aligned} L_1 &= g^{-1}(L) \\ L_2 &= \emptyset \end{aligned}$$

It is clear that L_1 and L_2 are regular and that $L \cap S_* = L_1 \bullet L_2$. Thus, L is an iteration of two regular languages iff $L = L \cap S_*$. This can easily be decided.

If (2) holds, there are words $w_1, \ldots, w_n, w'_1, \ldots, w'_n \in \{0, 1\}^*$ such that

$$\begin{aligned} w &= w_1 \boxdot \cdots w_n \boxdot \\ w' &= w'_1 \boxdot \cdots w'_n \boxdot \end{aligned}$$

with $w \in L$ and $w' \notin L$. Then there must be an $i \in \{1, \ldots, n\}$ s.t.

[12] Thanks to Makoto Kanazawa for suggesting this proof.

$$w_1 \boxdot \cdots w_{i-1} \boxdot w_i \boxdot w'_{i+1} \boxdot \cdots w'_n \boxdot \in L$$
$$w_1 \boxdot \cdots w_{i-1} \boxdot w'_i \boxdot w'_{i+1} \boxdot \cdots w'_n \boxdot \notin L \tag{3}$$

Let $f : \{0, 1\} \to \{0, 1, \boxdot\}^*$ be the string homomorphism

$$f(0) = w'_i \boxdot$$
$$f(1) = w_i \boxdot$$

and

$$L_1 = f^{-1}(L)$$
$$L_2 = \left\{ v \in \{0, 1\}^* \mid w_1 \boxdot \cdots w_{i-1} \boxdot v \boxdot w'_{i+1} \boxdot \cdots w'_n \boxdot \in L \right\}$$

It is clear that L_1 and L_2 are regular.

Now, L is an iteration $L'_1 \bullet L'_2$ iff $L = L_1 \bullet L_2$. To see this, suppose that $L = L'_1 \bullet L'_2$. The condition (3) implies that $\chi_{L'_2}(w_i) \neq \chi_{L'_2}(w'_i)$ and that

$$w_1 \boxdot \cdots w_{i-1} \boxdot v \boxdot w'_{i+1} \boxdot \cdots w'_n \boxdot \in L \text{ iff } \chi_{L'_2}(v) = \chi_{L'_2}(w_i)$$

Here again there are two cases. (i) $\chi_{L'_2}(w_i) = 1$ and $\chi_{L'_2}(w'_i) = 0$. Then $L'_2 = L_2$ and $L'_1 = L_1$. (ii) $\chi_{L'_2}(w_i) = 0$ and $\chi_{L'_2}(w'_i) = 1$. Then $L'_2 = \{0, 1\}^* \setminus L_2$ and $L'_1 = k(L_1)$ where k is the homomorphism $k(0) = 1$ and $k(1) = 0$. It's easy to see that $L'_1 \bullet L'_2 = L_1 \bullet L_2$.

All that remains is to show that automata for L_1 and L_2 can be found effectively. There must be $w_1, \ldots, w_n, w'_1, \ldots, w'_n \in \{0, 1\}^*$ satisfying (3). We can find these by brute-force search (since membership in L is decidable). With these strings in hand, automata for L_1 and L_2 can be obtained by the usual constructions. □

7 Beyond

7.1 Context-Free Languages

A natural class of languages beyond the regular languages are the context-free languages. To understand these, we again look at the class of machines accepting them. Essentially, these are automata which extend DFAs with a form of memory called a stack. A stack stores symbols. The top symbol can be read and removed (referred to as popping). Moreover, a string of symbols can be pushed on to the top of the stack.

A *pushdown automaton* (PDA) is a tuple $\langle Q, \Sigma, \Gamma, \delta, q_0, Z_0, F \rangle$ where Q, Σ, q_0, and F are as in a DFA. Γ is another alphabet called the stack alphabet; Z_0 is a special symbol designating an empty stack; $\delta : Q \times (\Sigma \cup \{\varepsilon\}) \times \Gamma \to \mathcal{P}(Q \times \Gamma^*)$ is the transition function. Intuitively, $(q', \gamma) \in \delta(q, a, b)$ means that if the PDA reads letter a while in state q_1 with letter $b \in \Gamma$ on top of the stack, it can transition into q_2 and

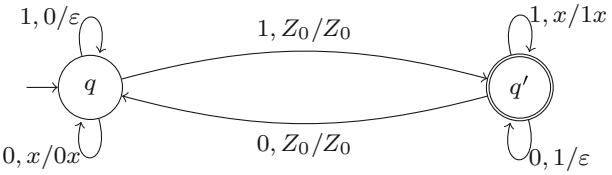

Fig. 7 PDA accepting L_{most}

replace b with γ on top of the stack.[13] The case when $\gamma = \varepsilon$ captures popping the top of the stack. In transition diagrams, we will draw an arrow from q to q' labelled by $a, b/\gamma$.

To define the language accepted by a PDA, we need a bit more notation. A triple $\langle q, w, \gamma \rangle$ of a state, word in Σ, and word in Γ will be called an *instantaneous description* of a PDA. We write $\langle q_1, aw, b\gamma \rangle \vdash_{\mathsf{M}} \langle q_2, w, \gamma'\gamma \rangle$ whenever $\langle q_2, \gamma' \rangle \in \delta(q_1, a, b)$. We omit the subscript when context allows and write $\vdash^*$ for the reflexive, transitive closure of $\vdash$. The language accepted by a PDA M is:

$$L(\mathsf{M}) = \left\{ w \in \Sigma^* : \langle q_0, w, Z_0 \rangle \vdash^* \langle q, \varepsilon, \gamma \rangle \text{ for some } q \in F \text{ and } \gamma \in \Gamma^* \right\}$$

A language L is *context-free* iff $L = L(\mathsf{M})$ for some PDA M.[14] See van Benthem (1986) for a logical characterization of the quantifiers with context-free languages.

Example 6 The language

$$L_{\text{most}} = \left\{ w \in \{0, 1\}^* : \#_1(w) > \#_0(w) \right\}$$

is context-free. Figure 7 depicts a pushdown automaton accepting that language. The states are labeled for future reference.

The closure under iteration result does not immediately extend to context-free languages because these languages are not closed under complement. But certain subclasses are. For instance, the permutation-closed languages on a two-letter alphabet are:

Theorem 6 (van Benthem 1986) *The permutation-closed context-free languages on a two-letter alphabet are closed under complement.*

We cannot conclude from this, however, that these languages are closed under iteration since, in general, an iterated language will not be closed under permutations. Nevertheless, we get a 'semi-closure' result:[15]

[13] The 'can' here reflects that the present definition defines nondeterministic PDAs which, unlike in the finite-state case, are strictly more powerful than their deterministic counterparts, to be discussed below.

[14] These languages can equally be characterized as those generated by a context-free grammar.

[15] Theorem 6 of Steinert-Threlkeld and Icard III (2013) states this result as a full closure result for context-free languages. This is because in that paper, the authors assumed—in addition to conservativity and extension—the isomorphism closure of quantifiers, which has the result that all of the languages discussed were permutation-closed. While this assumption is often made, natural language determiners like 'the first five' and 'every other' seem to be counterexamples.

Proposition 4 *If L_1 and L_2 are permutation-closed context-free languages in a two-letter alphabet, then $L_1 \bullet L_2$ is context-free.*

Proof Immediate from Theorem 6 and the closure of context-free languages under substitution. □

The main obstacle to using the strategy in Sect. 6 to try and prove the decidability of iteration for context-free languages is that language equality is undecidable. Several steps in the proof of Theorem 5 depended on the fact that language equality is decidable for regular languages. This leads us to conjecture:

Conjecture 1 It is undecidable whether a given context-free language $L \subseteq \{0, 1, \boxdot\}^*$ is an iteration of two context-free languages in alphabet $\{0, 1\}$.

7.2 Deterministic Context-Free Languages

Some hope, however, comes from finding a large subclass of the context-free languages for which equality is decidable. One such class is the *deterministic* context-free languages; these are the languages accepted by deterministic pushdown automata, which are those having at most one choice at every machine configuration in a sense to be made precise below. It is famously known that language equality is decidable for deterministic context-free langauges; see Sénizergues (1997, 2001, 2002). In this section, we will introduce these languages and then extend both the closure and decidability results to them.

A *deterministic pushdown automaton* (DPDA) is a PDA such that:

- for every $q \in Q$ and $b \in \Gamma$, if $\delta(q, \varepsilon, b) \neq \emptyset$, then $\delta(q, a, b) = \emptyset$ for every $a \in \Sigma$
- for every $q \in Q$, $a \in \Sigma \cup \{\varepsilon\}$, and $b \in \Gamma$, $|\delta(q, a, b)| \leq 1$.

These two constraints have the effect that at most one transition can occur at any instantaneous description of the DPDA.[16] A language L is deterministic context-free iff $L = L(\mathsf{M})$ for some DPDA M. Note that the PDA in Fig. 7 is in fact a DPDA, so L_{most} is in fact deterministic context-free. Another example is given below. For a complete characterization of the generalized quantifiers with languages accepted by DPDAs, see Kanazawa (2013).

Example 7 The language $L_{\geq 1/3} = \left\{w \in \{0, 1\}^* : \#_1(w) \geq \frac{1}{3}(\#_1(w) + \#_0(w))\right\}$, which can be used to verify the truth of sentences like (9), is deterministic context-free.

(9) At least one third of the students are happy.

A DPDA accepting the language is given in Fig. 8. The states are labeled for future reference.

[16] As presently defined, a DPDA may not read the entire input. However, for every DPDA M, there is another DPDA M' that does read the entire input such that $L(\mathsf{M}) = L(\mathsf{M}')$. See Lemma 10.3 (p. 236) of Hopcroft and Ullman (1979). Because of this, in what follows we will assume that all DPDAs read the entire input.

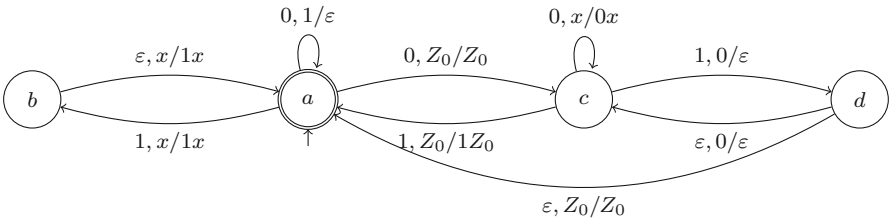

Fig. 8 A DPDA accepting $L_{\geq 1/3}$

Importantly, the deterministic context-free languages are in fact closed under complement; see, e.g., Theorem 10.1 on p. 238 of Hopcroft and Ullman (1979). Unfortunately, however, closure under iteration does not follow immediately since these languages are not closed under substitution. However, as in the star-free case, we can use the separator ⧈ to prove closure under iteration. We first offer a proof of closure under complement since some intermediate steps will be useful in defining a DPDA for iteration.

As a first step, we will define a DPDA that keeps track of whether or not a given DPDA has entered a final state since reading the last input symbol. This helps in distinguishing acceptance from rejection because a DPDA may make ε-transitions through both accepting and rejecting states after reading an input symbol. The states of the new DPDA will have a second component: 0, 1, or 2. A 0 will mean that the machine has not entered an accepting state since the last input, while a 1 means that it has entered an accepting state.

Definition 3 Given a DPDA M, define the *tracking extension of* M—denoted M^T—as follows:

- $Q = Q(\mathsf{M}) \times \{0, 1, 2\}$
- $\Sigma = \Sigma(\mathsf{M})$
- $\Gamma = \Gamma(\mathsf{M})$
- Transition function:

$$\delta = \left\{ \begin{array}{c} \left\langle \langle q, 1\rangle, \varepsilon, b, \left\langle q', 1\right\rangle, \gamma\right\rangle \\ \left\langle \langle q, 0\rangle, \varepsilon, b, \left\langle q', \text{sgn}\left(q', \mathsf{M}\right)\right\rangle, \gamma\right\rangle \end{array} : \langle q, \varepsilon, b, q', \gamma\rangle \in \delta(\mathsf{M}) \right\} \cup$$
$$\left\{ \begin{array}{c} \langle \langle q, 0\rangle, \varepsilon, b, \langle q, 2\rangle, b\rangle \\ \left\langle \langle q, 1\rangle, a, b, \left\langle q', \text{sgn}\left(q', \mathsf{M}\right)\right\rangle, \gamma\right\rangle \\ \left\langle \langle q, 2\rangle, a, b, \left\langle q', \text{sgn}\left(q', \mathsf{M}\right)\right\rangle, \gamma\right\rangle \end{array} : \langle q, a, b, q', \gamma\rangle \in \delta(\mathsf{M}) \text{ for } a \in \Sigma \right\}$$

- $q_0 = \langle q_0, \text{sgn}(\mathsf{M})\rangle$
- $F = \{\langle q, 1\rangle : q \in Q(\mathsf{M})\}$

Lemma 2 $L\left(\mathsf{M}^T\right) = L(\mathsf{M})$

Theorem 7 *The deterministic context-free languages are closed under complement.*

Proof Given L, let M be a DPDA such that $L = L(\mathsf{M})$. Consider the automaton M^c constructed as follows: M^c is just like M^T except that

$$F\left(\mathsf{M}^c\right) = \{\langle q, 2\rangle : q \in Q(\mathsf{M})\}$$

Clearly, M^c is a DPDA. The accepting states of M^c are those ending in a 2. Note that these are only entered from states ending in a 0 by an ε-transition. But a state ending in a 0 encodes that the machine has not entered an accepting state since reading the last input. The reader can use these ideas to show that $L\left(\mathsf{M}^c\right) = L^c$ or consult p. 239 of Hopcroft and Ullman (1979) for details. □

This proof shows that membership in L versus L^c is fully encoded by the second component of states of M^T. This allows us to prove the following proposition—a light generalization of Lemma 6.1.1 of McWhirter (2014)—which will be crucial in proving closure under iteration.

Proposition 5 *Given a DPDA* M*, there is another DPDA* $M^{\boxdot}$ *with* $\Sigma\left(\mathsf{M}^{\boxdot}\right) = \Sigma(\mathsf{M}) \cup \{\boxdot\}$ *and unique states* q_{acc} *and* q_{rej} *such that for any* γ *with* $\gamma_1 \notin \Gamma(\mathsf{M})$*:*

- $\langle q_0, w\boxdot, \gamma\rangle \vdash^* \langle q_{acc}, \varepsilon, \gamma\rangle$ *iff* $w \in L(\mathsf{M})$
- $\langle q_0, w\boxdot, \gamma\rangle \vdash^* \langle q_{rej}, \varepsilon, \gamma\rangle$ *iff* $w \notin L(\mathsf{M})$

Proof $\mathsf{M}^{\boxdot}$ will be just like M^T but with two new states q_{acc} and q_{rej} and the following modification to the transition function:

$$\begin{aligned}\delta\left(\mathsf{M}^{\boxdot}\right) = \delta\left(\mathsf{M}^T\right) &\cup \left\{\langle q, \boxdot, b, q_{\text{acc}}, b\rangle : q \in F\left(\mathsf{M}^T\right) \text{ and } b \in \Gamma(\mathsf{M})\right\} \\ &\cup \left\{\langle q, \boxdot, b, q_{\text{rej}}, b\rangle : q \in F\left(\mathsf{M}^c\right) \text{ and } b \in \Gamma(\mathsf{M})\right\} \\ &\cup \{\langle q_i, \varepsilon, b, q_i, \varepsilon\rangle : i \in \{\text{acc}, \text{rej}\} \text{ and } b \in \Gamma(\mathsf{M})\}\end{aligned}$$

In other words: upon reading $\boxdot$, the machine transitions to q_{acc} or q_{rej} according to whether the machine has entered an accepting state since the last input or not and then empties the stack of all contents from $\Gamma(\mathsf{M})$. This clearly implements the desired behavior. □

We are now in position to define a DPDA for the iteration of two others. As before, we want to replace 1 transitions from M_1 with accepting runs from M_2 (and *mutatis mutandis* for 0 transitions and rejecting runs). By Proposition 5, we know that an accepting run of M_2 corresponds to a run ending in q_{acc} without altering the stack in $\mathsf{M}_2^{\boxdot}$. So we do the following: we have a copy both of M_1 and of $\mathsf{M}_2^{\boxdot}$. All ε-transitions in M_1 are left intact. When M_1 is about to read a 1 in state q, it does the following: it takes an ε-transition to the start state of M_2 while pushing a q on to the stack. This has two effects: the q both 'insulates' M_1's use of the stack from $\mathsf{M}_2^{\boxdot}$'s and also tells the latter machine which state M_1 was previously in. Now, when $\mathsf{M}_2^{\boxdot}$ gets to q_{acc}, it sees the q on the stack and transitions to a new state named a_q while popping the q. This has the effect of 'lifting the lid' off of M_1's stack. The state a_q then makes an

ε-transition to the state that M_1 would have transitioned on reading a 1 in q while also making the appropriate stack manipulations. The 0 transitions are treated similarly, but with q_{rej} and other new states r_q. The following definition and lemma make this all precise. An example demonstrating the construction is given at the end of the section.

Definition 4 (*Iteration of DPDAs*) Let M_1 and M_2 be DPDAs in alphabet $\{0, 1\}$. We define *the iteration of* M_1 *and* M_2, denoted $\mathsf{It}(\mathsf{M}_1, \mathsf{M}_2)$ as follows:

- $Q = Q(\mathsf{M}_1) \cup Q\left(\mathsf{M}_2^{\boxdot}\right) \cup \{a_q, r_q : q \in Q(\mathsf{M}_1)\}$
- $\Sigma = \{0, 1, \boxdot\}$
- $\Gamma = \Gamma(\mathsf{M}_1) \cup \Gamma(\mathsf{M}_2) \cup Q(\mathsf{M}_1)$
- Transition function:

$$\begin{aligned}\delta = {} & \delta(\mathsf{M}_2) \cup \left\{\langle q, \varepsilon, b, q', \gamma\rangle : \langle q, \varepsilon, b, q', \gamma\rangle \in \delta(\mathsf{M}_1)\right\} \\ & \cup \left\{\langle q, \varepsilon, b, q_0(\mathsf{M}_2), qb\rangle : \langle q, a, b, q', \gamma\rangle \in \delta(\mathsf{M}_1) \text{ for some } a \in \{0, 1\}\right\} \\ & \cup \left\{ \begin{matrix} \langle q_{\text{acc}}, \varepsilon, q, a_q, \varepsilon\rangle \\ \langle a_q, \varepsilon, b, q', \gamma\rangle \end{matrix} : \langle q, 1, b, q', \gamma\rangle \in \delta(\mathsf{M}_1) \right\} \\ & \cup \left\{ \begin{matrix} \langle q_{\text{rej}}, \varepsilon, q, r_q, \varepsilon\rangle \\ \langle r_q, \varepsilon, b, q', \gamma\rangle \end{matrix} : \langle q, 0, b, q', \gamma\rangle \in \delta(\mathsf{M}_1) \right\}\end{aligned}$$

- $q_0 = q_0(\mathsf{M}_1)$
- $F = F(\mathsf{M}_1)$

When context allows, $\mathsf{It}(\mathsf{M}_1, \mathsf{M}_2)$ will be abbreviated as It.

Lemma 3 *Let* M_1 *and* M_2 *be as in Definition* 4.

(1) If $\langle q, 1w, x\beta\rangle \vdash_{\mathsf{M}_1} \langle q', w, \gamma\beta\rangle$, *then*

$$\langle q, w \boxdot w', x\beta\rangle \vdash_{\mathsf{It}}^{*} \langle q', w', \gamma\beta\rangle \text{ iff } w \in L(\mathsf{M}_2)$$

(2) If $\langle q, 0w, x\beta\rangle \vdash_{\mathsf{M}_1} \langle q', w, \gamma\beta\rangle$, *then*

$$\langle q, w \boxdot w', x\beta\rangle \vdash_{\mathsf{It}}^{*} \langle q', w', \gamma\beta\rangle \text{ iff } w \notin L(\mathsf{M}_2)$$

Proof We only show (1), since (2) is completely analogous. Suppose $\langle q, 1w, x\beta\rangle \vdash_{\mathsf{M}_1} \langle q', w, \gamma\beta\rangle$, i.e. $\langle q, a, x, q', \gamma\rangle \in \delta(\mathsf{M}_1)$. First, in a configuration $\langle q, w \boxdot w', x\beta\rangle$, It will take an ε-transition to $q_0\left(\mathsf{M}_2^{\boxdot}\right)$ while pushing a q on to the stack. By Proposition 5,

$$\langle q_0\left(\mathsf{M}_2^{\boxdot}\right), w \boxdot w', qx\beta\rangle \vdash_{\mathsf{It}}^{*} \langle q_{\text{acc}}, w', qx\beta\rangle$$

iff $w \in L(\mathsf{M}_2)$. Then, from q_{acc}, It will make an ε-transition to a_q while popping the q off the top of the stack. It will then make an ε-transition to q' while replacing x by γ on top of the stack. □

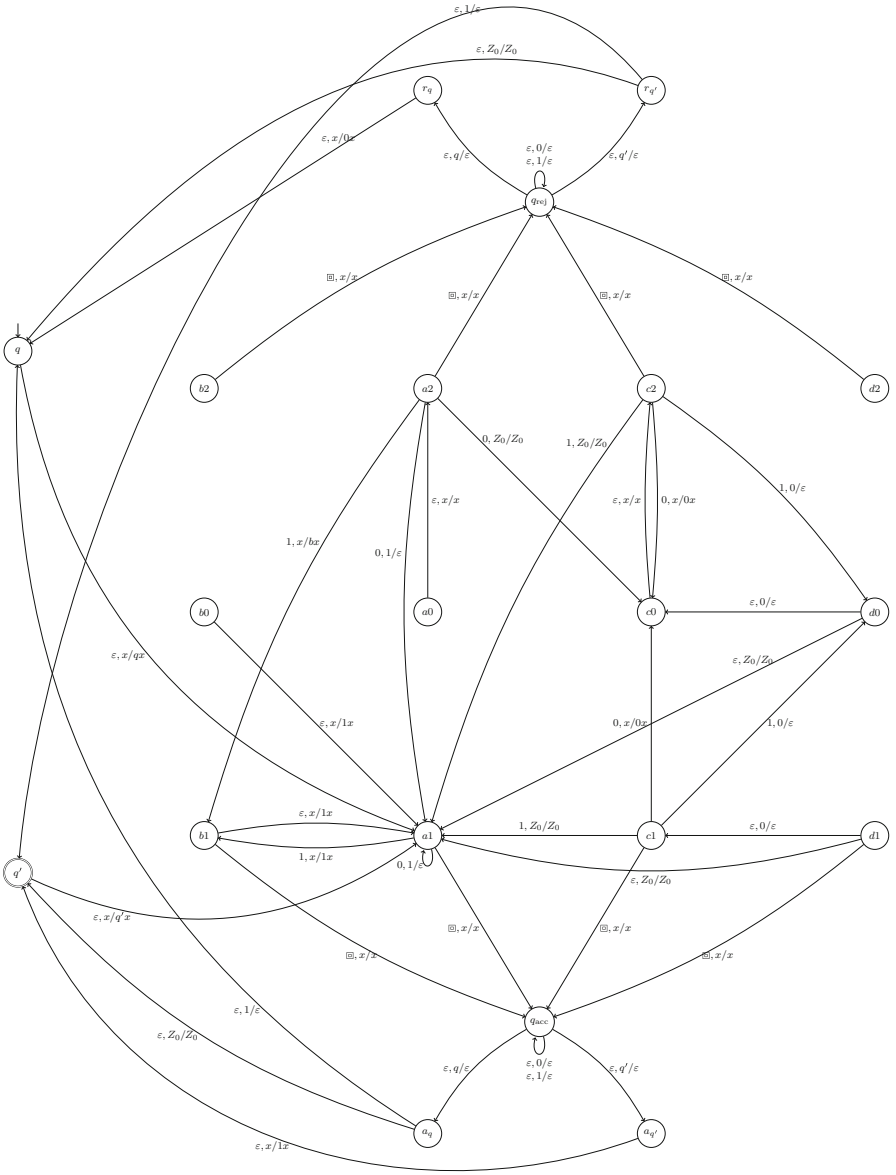

Fig. 9 The DPDA $\mathsf{It}\left(\mathsf{M}_{\text{most}}, \mathsf{M}_{\geq 1/3}\right)$

Theorem 8 $\mathsf{It}\left(\mathsf{M}_1, \mathsf{M}_2\right)$ *is a DPDA with* $L\left(\mathsf{It}\left(\mathsf{M}_1, \mathsf{M}_2\right)\right) = L\left(\mathsf{M}_1\right) \bullet L\left(\mathsf{M}_2\right)$. *Thus, the deterministic context-free languages are closed under iteration.*[17]

[17] McWhirter (2014) has obtained this result independently. My exposition of this result has improved greatly from reading her work.

Proof That It is a DPDA can be seen from inspection of Definition 4. ε-transitions in M_1 are left in-tact, while 0 and 1 transitions are replaced by ε-transitions to $q_0(\mathsf{M}_2)$, which preserves determinism. Transitions into and out of the new states a_q and r_q are also deterministic.

That $L(\mathsf{It}) = L(\mathsf{M}_1) \bullet L(\mathsf{M}_2)$ follows from repeated application of Lemma 3 and the observation that the only paths to accepting states which are not $q_0(\mathsf{It})$ must process an input of the form $w\boxdot$ for $w \in \{0, 1\}^*$. □

Theorem 9 *It is decidable whether a given deterministic context-free language $L \subseteq \{0, 1, \boxdot\}^*$ is an iteration.*

Proof The proof of Theorem 5 carries over unchanged into the DCFL case because language equality for DCFLs is decidable and the DCFLs are effectively closed under inverse homomorphism and intersection with a regular language (and S_* is regular). □

Example 8 Figure 9 depicts the automaton $\mathsf{It}\left(\mathsf{M}_{\text{most}}, \mathsf{M}_{\geq 1/3}\right)$. Note that the DPDA could be pruned: the states $d1$, $b0$, $a0$, $b2$, $a2$, and $d2$ are not reachable. They have been included to illustrate Definition 4 exactly.

8 Conclusion

This paper has taken up the question of whether the Frege boundary is decidable by using tools from formal language and automata theory. In particular, an operation on languages called iteration was introduced, after which it was shown that the star-free, regular, and deterministic context-free languages are closed under iteration. The state complexity of iteration of regular languages was also determined. Then, the decidability of the Frege boundary was addressed via the following question: given a language $L \subseteq \{0, 1, \boxdot\}^*$, is it decidable whether there are languages L_1 and L_2 such that $L = L_1 \bullet L_2$? We answered this question in the affirmative for both the regular and deterministic context-free languages.

There are two main lines of work to pursue. First, one would like to settle Conjecture 1 above. That is: is it undecidable whether a given (non-deterministic) context-free language is an iteration? Second, although this paper demonstrates that formal languages can be very useful in studying iterated quantifiers, one would like a characterization in language/automata terms of the Frege Boundary. A more natural setting for that pursuit may be the so-called 'picture languages', which are binary matrices (instead of words) of alphabet symbols.[18] We leave both of these pursuits to future work.

[18] See Giammarresi and Restivo (1997) for details.

References

Barwise, J., & Cooper, R. (1981). Generalized quantifiers and natural language. *Linguistics and Philosophy*, *4*(2), 159–219.

Clark, R. (2011). Generalized quantifiers and number sense. *Philosophy Compass*, *6*(9), 611–621.

Cui, B., Gao, Y., Kari, L., & Sheng, Y. (2012). State complexity of combined operations with two basic operations. *Theoretical Computer Science*, *437*, 82–102. doi:10.1016/j.tcs.2012.02.030.

Dekker, P. (2003). Meanwhile, within the Frege boundary. *Linguistics and Philosophy*, *26*(5), 547–556.

Diekert, V., & Gastin, P. (2007). First-order definable languages. In J. Flum, E. Grädel, & T. Wilke (Eds.), *Logic and automata: History and perspectives, Texts in Logic and Games* (pp. 261–306). Amsterdam: Amsterdam University Press.

Giammarresi, D., & Restivo, A. (1997). Two-dimensional languages. In G. Rozenberg & A. Salomaa (Eds.), *Handbook of formal languages (vol 3): Beyond words* (Vol. 3, pp. 215–268). Berlin: Springer.

Hopcroft, J. E., & Ullman, J. D. (1979). *Introduction to automata theory, languages, and computation*. Reading: Addison-Wesley.

Kanazawa, M. (2013). Monadic Quantifiers Recognized by Deterministic Pushdown Automata. In M. Aloni, M. Franke, & F. Roelofsen (Eds.), *Proceedings of the 19th Amsterdam Colloquium* (pp. 139–146).

Keenan, E. L. (1992). Beyond the Frege boundary. *Linguistics and Philosophy*, *15*(2), 199–221.

Keenan, E. L. (1996). Further beyond the Frege boundary. In J. van der Does & J. van Eijck (Eds.), *Quantifiers, logic, and language, volume 54 of CSLI Lecture Notes* (pp. 179–201). Stanford: CSLI Publications.

Lindström, P. (1966). First order predicate logic with generalized quantifiers. *Theoria*, *32*(3), 186–195.

McMillan, C. T., Clark, R., Moore, P., Devita, C., & Grossman, M. (2005). Neural basis for generalized quantifier comprehension. *Neuropsychologia*, *43*(12), 1729–1737. doi:10.1016/j.neuropsychologia.2005.02.012.

McNaughton, R., & Papert, S. A. (1971). *Counter-free automata, volume 65 of MIT Research Monographs*. Cambridge: The MIT Press.

McWhirter, S. (2014). An automata-theoretic perspective on polyadic quantification in natural language. Masters of logic thesis, Universiteit van Amsterdam.

Mostowski, A. (1957). On a generalization of quantifiers. *Fundamenta Mathematicae*, *44*(2), 12–36.

Mostowski, M. (1998). Computational semantics for monadic quantifiers. *Journal of Applied Non-classical Logics*, *8*(1–2), 107–121.

Peters, S., & Westerståhl, D. (2006). *Quantifiers in language and logic*. Oxford: Clarendon Press.

Sénizergues, G. (1997). The equivalence problem for deterministic pushdown automata is decidable. In P. Degano, R. Gorrieri, & A. Marchetti-Spaccamela (Eds.), *Automata, Languages and Programming, Volume 1256 of Lecture Notes in Computer Science* (pp. 671–681). Berlin: Springer.

Sénizergues, G. (2001). L(A) = L(B)? Decidability results from complete formal systems. *Theoretical Computer Science*, *251*(1–2), 1–166.

Sénizergues, G. (2002). L(A) = L(B)? A simplified decidability proof. *Theoretical Computer Science*, *281*(1–2), 555–608.

Steinert-Threlkeld, S. (2014). On the decidability of iterated languages. In Oleg, P. (Ed.), *Proceedings of philosophy, mathematics, linguistics: Aspects of interaction (PhML2014)*, (pp. 215–224).

Steinert-Threlkeld, S., & Icard III, T. F. (2013). Iterating semantic automata. *Linguistics and Philosophy*, *36*(2), 151–173. doi:10.1007/s10988-013-9132-6.

Szabolcsi, A. (2010). *Quantification*. Research surveys in linguistics. Cambridge: Cambridge University Press.

Szymanik, J., & Zajenkowski, M. (2010). Comprehension of simple quantifiers: Empirical evaluation of a computational model. *Cognitive Science*, *34*(3), 521–532. doi:10.1111/j.1551-6709.2009.01078.x.

Szymanik, J., Steinert-Threlkeld, S., Zajenkowski, M., Icard III, T. F. (2013). Automata and complexity in multiple-quantifier sentence verification. In *Proceedings of the 12th international conference on cognitive modeling*.

van Benthem, J. (1986). *Essays in logical semantics*. Dordrecht: D. Reidel Publishing Company.

van Benthem, J. (1989). Polyadic quantifiers. *Linguistics and Philosophy*, *12*(4), 437–464. doi:10.1007/BF00632472.

Yu, S., Zhuang, Q., & Salomaa, K. (1994). The state complexities of some basic operations on regular languages. *Theoretical Computer Science*, *125*(2), 315–328. doi:10.1016/0304-3975(92)00011-F.