On the Decidability of Iterated Languages

Shane Steinert-Threlkeld
Department of Philosophy, Stanford University
450 Serra Mall, Stanford, CA 94305, USA
shanest@stanford.edu

Abstract: A special kind of substitution on languages called *iteration* is presented and studied. 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 language or a DCFL is an iteration of two languages. We also determine the state complexity of iteration of regular languages. Connections to the van Benthem / Keenan 'Frege Boundary' are discussed.

1. Introduction

(Mostowski, 1957) and (Lindström, 1966) introduced generalized quantifiers into mathematical logic. Starting with (Barwise & Cooper, 1981), these have been very fruitfully applied in the formal semantics of natural language. In particular, the meanings of determiners in sentences like

(1) Every student attends classes.

(2) Between five and ten people will attend.

(3) Most people enjoyed the show.

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

$$|people \cap enjoyed| > |people \setminus enjoyed|$$

This approach, however, has difficulty explaining sentences with quantified phrases in object position, such as:

(4) Every student takes at least three classes.

(5) Most professors teach two classes.

To model these sentences, one must *iterate* the unary quantifiers to express a property of the respective transitive verbs. These will be type $\langle 2\rangle$ quantifiers. For instance, (5) is true iff *teach* bears the property $It(most\_prof, two\_classes)$ iff

$$|profs \cap teach\_two\_classes| > |prof \setminus teach\_two\_classes|$$

A question then arises: are all type $\langle 2\rangle$ quantifiers realized in natural language iterations of unary quantifiers? The line dividing these quantifiers has been dubbed *the Frege Boundary* by (van Benthem, 1989) and studied extensively by (Keenan, 1992, 1996).

In those papers, Keenan provides an exact characterization of which type $\langle 2\rangle$ quantifiers are iterations of unary quantifiers. But it has remained unknown whether his characterization is *effective*: given a type $\langle 2\rangle$ quantifier, can we decide whether it's an iteration? In this paper,

we take steps towards answering that question by using the semantic automata framework pioneered by (van Benthem, 1986, ch. 6) which was recently extended to handle iterations by (Steinert-Threlkeld & Icard III., 2013).

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 $L^c = \mathcal{P}(\Sigma^*) \setminus L$ and $\boxdot$ is a fresh symbol not in Σ. Given $L_1, L_2 \subseteq \{0,1\}^*$, we are interested in languages of the form

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

where σ_{L_2} is extended to a substitution on the whole language L_1 in the usual way. We call this language *the iteration of* L_1 *and* L_2 and denote it by $L_1 \bullet L_2$. These languages are so named because they arise in the study of iterations of generalized quantifiers.[1] In particular, if L_1 and L_2 are languages associated with unary 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. In this note, we show that the star-free and regular languages are closed under iteration and that it is decidable whether a given regular language is an iteration of two languages. We conclude by discussing the prospects for extending the decidability result beyond the regular languages. In particular, the decidability proof carries over to the deterministic context-free case.

2. Preliminaries

We assume familiarity with the theory of regular languages and deterministic finite-state automata. We will use L, L_1, L_2 to denote languages and $\mathsf{M}, \mathsf{M}_1, \mathsf{M}_2$ to denote automata. 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 denote the components of an automaton by $Q(\mathsf{M})$, $\delta(\mathsf{M})$, et cetera. 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 $w \in \Sigma^*$. By $L(\mathsf{M})$ we denote the language accepted by automaton M. We write

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

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

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 automata; see (McNaughton & Papert, 1971). We will use another characterization of the star-free languages in terms of first-order definability.[2]

Given an alphabet Σ, we consider a standard first-order language with predicate symbols P_a for each $a \in \Sigma$ and a binary relation symbol $<$. We write $Form(\Sigma)$ and $Sent(\Sigma)$ for, respectively, the set of first-order formulas and sentences in the appropriate signature. We interpret formulas in this language in pairs $(w, (p_1, \ldots, p_n))$ of a word $w \in \Sigma^*$ and a sequence of positions p_i, where $0 \leq p_i \leq |w|$. The relevant semantic clauses are given by

$$(w, (p_1, \ldots, p_n)) \models x_i < x_j \text{ iff } p_i < p_j$$
$$(w, (p_1, \ldots, p_n)) \models P_a(x_i) \text{ iff } w_{p_i} = a$$

[1] See (Steinert-Threlkeld & Icard III., 2013) and references therein. Our definition of iteration is a more concise representation of their Definition 8.

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

A first-order sentence φ defines the language

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

The following theorem was mentioned above.

Theorem 1 ((McNaughton & Papert, 1971))**.** *A language L is star-free iff it is first-order definable.*

There are a few languages in $\{0, 1\}$ that will recur below. These are:

$$\begin{aligned} L_\forall &= \{w \mid \#_0(w) = 0\} \\ L_{\geq n} &= \{w \mid \#_1(w) \geq n\} \\ L_\exists &= L_{\geq 1} \\ L_{\leq n} &= \{w \mid \#_1(w) \leq n\} \end{aligned}$$

3. Closure

Proposition 1. *The following classes of languages are closed under iteration:*

- *Star-free*
- *Regular*

In other words, if L_1 and L_2 are star-free (resp. regular), then $L_1 \bullet L_2$ is star-free (resp. regular).

Proof. The regular language case follows immediately from the closure of those languages under complement and substitutions.

For the star-free case, let $\varphi_1(P_0, P_1)$ and $\varphi_2(P_0, P_1)$ be first-order sentences defining L_1 and L_2. We will use the $\boxdot$ symbol, with its corresponding predicate $P_\boxdot$ to convert predications in φ_1 into quantifications over words. We need the following string of defined symbols:

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

Inspection of these definitions shows that $(w, (p_j, p_k)) \models Word(x_j, x_k)$ iff the subword $w_{p_j} \cdots w_{p_k}$ 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

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

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

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

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

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

Proof. Put φ_1 in prenex normal form and generate $\tau\left(\varphi_1, \varphi_2\right)$ '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 $L_{\varphi_2}^c$ 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.

Note that our use of the separator symbol $\boxdot$ in the above proof is in a sense essential. This is because the star-free languages are not closed under substitution in general. As an example, let $\Sigma = \{1\}$ and consider $L_1 = \{1\}^*$, $L_2 = \{11\}$ and $\sigma\left(1\right) = L_2$. Then $\sigma\left[L_1\right] = \{w \mid \#_1\left(w\right) \text{ is even}\}$, which is quintessentially not star-free.

Example 1. $\varphi_\forall := \forall x P_1\left(x\right)$ defines $L_\forall$ and $\varphi_\exists := \exists x P_1\left(x\right)$ defines $L_\exists$. We have that

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

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

One may hope to prove closure for context-free languages. An immediate problem, however, arises since these languages are not closed under complement. But certain subclasses are. In particular, the deterministic context-free languages are. These, however, are not closed under substitution. Nevertheless, we will later see that they are still closed under iteration by using a novel construction on dterministic pushdown automata. Another subclass, however, is the two-letter permutation-closed CFLs:

Theorem 2 ((van Benthem, 1986))**.** *The permutation-closed context-free grammars 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:

Proposition 2. *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 2 and the closure of context-free languages under substitution.

4. 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, Zhuang, & Salomaa, 1994; Cui, Gao, Kari, & Yu, 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, we need one helper definition.

Definition 1. We will define the *one-step unraveling of* M, denoted M^+. If $sgn(\mathsf{M}) = 0$ or $\langle q_0, c, q_0\rangle \notin \delta(\mathsf{M})$ for all $c \in \Sigma$, then $\mathsf{M}^+ = \mathsf{M}$. Otherwise:

- $Q^+ = \{*\} \cup Q(\mathsf{M})$
- $q_0^+ = *$
- $F^+ = \{*\} \cup F(\mathsf{M})$
- $\delta^+ = \delta(\mathsf{M}) \cup \{\langle *, c, q\rangle \mid \langle q_0, c, q\rangle \in \delta(\mathsf{M})\}$

Essentially, if the start state is final and 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. $L(\mathsf{M}^+) = L(\mathsf{M})$

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:

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

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

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

Fact 1. $|\mathsf{It}(\mathsf{M}_1, \mathsf{M}_2)| = |\mathsf{M}_1| \cdot |\mathsf{M}_2^+|$

Example 2. Figure 1 shows the minimal automata for $L_\forall$, $L_\exists$. Figure 2 shows $\mathsf{It}(\mathsf{M}_\exists, \mathsf{M}_\forall)$. Figure 3 shows $\mathsf{It}(\mathsf{M}_\forall, \mathsf{M}_\exists)$. This machine illustrates the need for the one-step unraveling: without it, $0^* \subset L(\mathsf{It}(\mathsf{M}_\forall, \mathsf{M}_\exists))$, which we want to prevent. The states are labelled to enhance readability; the pair (a, b) is abbreviated ab.

Proposition 3. $L(\mathsf{It}(\mathsf{M}_1, \mathsf{M}_2)) = L(\mathsf{M}_1) \bullet L(\mathsf{M}_2)$

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

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

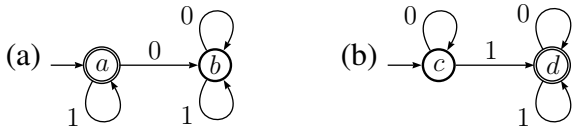

Figure 1: The minimal automaton for (a) $L_\forall$, (b) $L_\exists$.

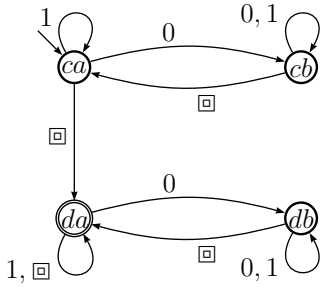

Figure 2: The automaton $\mathsf{It}\left(\min\left(L_\exists\right), \min\left(L_\forall\right)\right)$.

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 $sgn\left(\mathsf{M}_1\right) = 1$, i.e. iff $q_0\left(\mathsf{M}_1\right) \in F\left(\mathsf{M}_1\right)$. By the definition of the iteration automaton, this holds iff $q_0\left(\mathsf{It}\right) \in F\left(\mathsf{It}\right)$, i.e. iff $\varepsilon \in L\left(\mathsf{It}\right)$. Since $L_1^0 \bullet L_2 = \{\varepsilon\}$ iff $L_1^0 = \{\varepsilon\}$, this completes the base case.

Now, suppose that $L_1^n \bullet L_2 \subseteq L\left(\mathsf{It}\right)$ 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 $\delta\left(\mathsf{It}\right)$, 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 $F\left(\mathsf{It}\right)$ 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 $\delta\left(\mathsf{It}\right)$ and Lemma 1, reading w_{n+1} will lead to a state $\langle q, q_2\rangle$ such that $sgn\left(q_2, \mathsf{M}_2^+\right) = 1$ iff $w_{n+1} \in L_2$. In other words, $sgn\left(q_2, \mathsf{M}_2^+\right) = a_{n+1}$. Since, by assumption, $a_1 \ldots a_{n+1} \in L_1$, we know that $\langle q, a_{n+1}, q_1\rangle \in \delta\left(\mathsf{M}_1\right)$ for some $q_1 \in F\left(\mathsf{M}_1\right)$. Then clause (2) of the definition of $\delta\left(\mathsf{It}\right)$ 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 $F\left(\mathsf{It}\right)$, so $w \in L\left(\mathsf{It}\right)$, 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 $L_1 \bullet L_2 \subseteq L\left(\mathsf{It}\right)$.

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

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

where δ^* denotes the transitive closure of δ. Combined with the definition of $F\left(\mathsf{It}\right)$, this shows that $w \in L\left(\mathsf{It}\right) \Rightarrow w \in L_1 \bullet L_2$.

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.

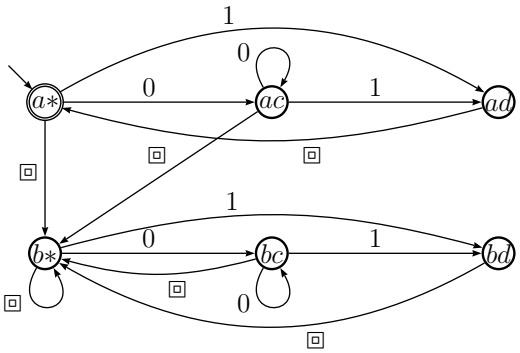

Figure 3: The automaton $\mathsf{lt}\left(\min\left(L_\forall\right), \min\left(L_\exists\right)\right)$.

Fact 2. The minimal automaton accepting $L_{\geq m} \bullet L_{\leq n-1}$ has $m \cdot (n+1)$ states, where the minimal automata accepting $L_{\geq m}$ and $L_{\leq n-1}$ have, respectively, m and n states.

We can summarize these results as:

Theorem 3. *Iteration has state complexity* $m \cdot (n+1)$.

5. Decidability

We now pursue the 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.

5.1. Regular Languages

Theorem 4 (Decidability)**.** *Let* $L \subseteq \{0, 1, \boxdot\}^*$ *be a regular language. Then it is decidable whether there are regular languages* L_1, L_2 *in* $\{0, 1\}$ *such that* $L = L_1 \bullet L_2$.

Proof. [3] 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 be the homomorphism given by

$$h(0) = h(1) = \varepsilon$$
$$h(\boxdot) = 0\boxdot$$

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 by

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

and let

$$L_1 = g^{-1}(L)$$
$$L_2 = \emptyset$$

[3] I am grateful to Makoto Kanazawa for suggesting this proof.

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

$$w = w_1 \boxdot \cdots w_n \boxdot$$
$$w' = w'_1 \boxdot \cdots w'_n \boxdot$$

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

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

Let f be the 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 $L'_2 = \{0,1\}^* - L_2$ 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.

5.2. Beyond

The main obstacle to using the above strategy to try and prove the decidability of iteration for context-free languages is that language equality is undecidable. This leads us to conjecture:

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

Some hope, however, comes from finding a large subclass of the context-free languages for which equality is decidable. That 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. It's famously known that language equality is decidable for determinstic context-free langauges (Sénizergues, 1997, 2001, 2002). Moreover, the deterministic context-free languages are in fact closed under complement; see, e.g., (Hopcroft & 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 $\boxdot$ to prove closure.

Proposition 4. *The deterministic context-free languages are closed under iteration.*

Proof Sketch. Given two DPDAs M_1 and M_2, the construction of a new DPDA for the iterated langauge proceeds very similarly to Definition 2. We must, however, also add ⧈ (or any fresh symbol) to the stack alphabet. Now, whenever a ⧈ is read on input, we erase the stack up to and including the first ⧈ and perform whatever stack manipulations M_1 specifies. Then, we push a ⧈ on top. This has the effect of maintaining M_1's stack "insulated below" the stacks needed for each time a copy of M_2 is run. Clearly this machine accepts the iterated language.

Corollary 1. It is decidable whether a given deterministic context-free language in $\{0, 1, ⧈\}$ is an iteration.

Proof. The proof of Theorem 4 carries over unchanged into the DCFL case since these languages are effectively closed under all the constructions used there.

References

Barwise, J., & Cooper, R. (1981). Generalized Quantifiers and Natural Language. *Linguistics and Philosophy*, *4*(2), 159–219.

Cui, B., Gao, Y., Kari, L., & Yu, S. (2012, June). State complexity of combined operations with two basic operations. *Theoretical Computer Science*, *437*, 82–102. Retrieved from `http://linkinghub.elsevier.com/retrieve/pii/S030439751200182X` doi: 10.1016/j.tcs.2012.02.030

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

Hopcroft, J. E., & Ullman, J. D. (1979). *Introduction to Automata Theory, Languages, and Computation*. Addison-Wesley.

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* (Vol. 54, pp. 179–201). Stanford: CSLI Publications.

Lindström, P. (1966, December). First order predicate logic with generalized quantifiers. *Theoria*, *32*, 186–195. Retrieved from `http://onlinelibrary.wiley.com/doi/10.1111/j.1755-2567.1966.tb00600.x/abstract`

McNaughton, R., & Papert, S. A. (1971). *Counter-free Automata* (Vol. 65). The MIT Press.

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

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* (Vol. 1256, pp. 671–681). Berlin: Springer Berlin Heidelberg.

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

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

Steinert-Threlkeld, S., & Icard III., T. F. (2013). Iterating semantic automata. *Linguistics and Philosophy*, *36*(2), 151–173.

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

van Benthem, J. (1989, August). Polyadic quantifiers. *Linguistics and Philosophy*, *12*(4), 437–464. Retrieved from http://www.springerlink.com/index/10.1007/BF00632472 doi: 10.1007/BF00632472

Yu, S., Zhuang, Q., & Salomaa, K. (1994, March). The state complexities of some basic operations on regular languages. *Theoretical Computer Science*, *125*(2), 315–328. Retrieved from http://linkinghub.elsevier.com/retrieve/pii/030439759200011F doi: 10.1016/0304-3975(92)00011-F