Phonology **35** (2018). Supplementary materials

Input Strictly Local opaque maps

Jane Chandlee
Haverford College

Jeffrey Heinz
Stony Brook University

Adam Jardine
Rutgers University

Supplementary materials

Appendix A: Abstract characterisation of ISL functions

In §2.2 of the paper, we define ISL functions in terms of finite-state transducers. Here we provide an alternative, abstract characterisation, grounded in formal language theory. This characterisation is abstract in the sense that it identifies a property of the EXTENSION of an ISL function, meaning the set of string pairs. This particular characterisation proves useful for showing that a given function is *not* ISL. We will take advantage of this usefulness to prove that the ISL functions (as defined in Chandlee 2014) are not closed under composition.

First, some preliminary notation must be explained. The length of a string w is $|w|$. Recall that the empty string of length zero is denoted λ. If $w = uv$ let $u^{-1}w = v$ (so v is the string obtained by stripping u from the beginning of w). The set of prefixes of w, $\mathrm{Pref}(w)$, is $\{p \in \Sigma^* \mid (\exists s \in \Sigma^*)[w = ps]\}$, and the set of suffixes of w, $\mathrm{Suff}(w)$, is $\{s \in \Sigma^* \mid (\exists p \in \Sigma^*)[w = ps]\}$. The longest common prefix of a set of strings S, denoted $\mathrm{lcp}(S)$, is $p \in \bigcap_{w \in S}\mathrm{Pref}(w)$ such that $\forall p' \in \bigcap_{w \in S}\mathrm{Pref}(w)$, $|p'| < |p|$. Note for any set S the lcp is unique and may be λ.

With the lcp, we can now define the language-theoretic concept of 'tails' of a function f as follows: $\mathrm{tails}_f(x) = \{(y, v) \mid f(xy) = uv \wedge u = \mathrm{lcp}(f(x\Sigma^*))\}$. Informally, the tails of a string x given a function f is the function which maps strings y to v where v is the output of $f(xy)$ after the longest common prefix of $f(x\Sigma^*)$ has been factored out.

Finally, we can define ISL functions abstractly in terms of the qualities of their tails. A function f is ISL if there is an integer k such that for all u_1, $u_2 \in \Sigma^*$, if $\mathrm{Suff}^{k-1}(u_1) = \mathrm{Suff}^{k-1}(u_2)$, then $\mathrm{tails}_f(u_1) = \mathrm{tails}_f(u_2)$. In other words, if two strings share a $(k-1)$-long suffix, they must have the same tails.

Using this characterisation of ISL, we can now show that the ISL functions are not closed under composition. The counterexample in Fig. 5 serves as proof of this fact (we thank Rémi Eyraud for this example).

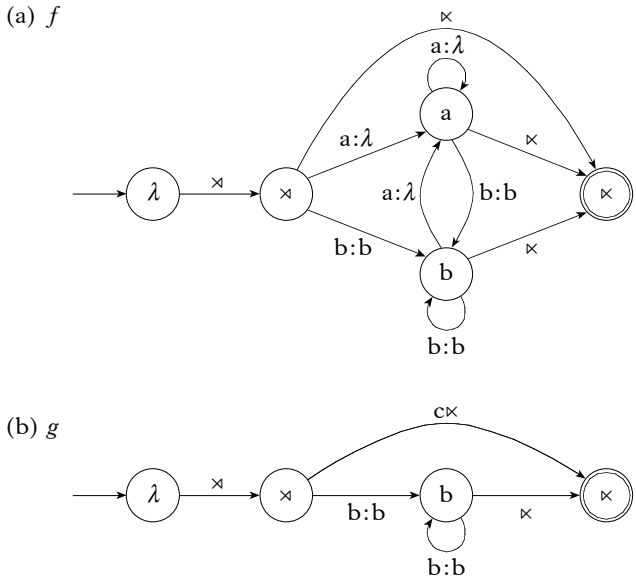

Figure 5
Counterexample to ISL closure under composition.

The 2-ISL function f on the left side of Fig. 5 maps the inputs a^k and ba^k to λ and b respectively, for all values of k. If these outputs are given as inputs to the 2-ISL function g on the right side of the figure, it would map them to c and b respectively. This means $g \circ f(a^k) = c$ and $g \circ f(ba^k) = b$. Clearly, ba^k and a^k share a suffix of length $k-1$, so if $g \circ f$ were a k-ISL function then they should also share the same tails. But they do not, since (λ, c) is in the tails of a^k and (λ, λ) is in the tails of ba^k. Thus this function is not k-ISL, and since k here is arbitrary, we conclude that the function is not ISL for any value of k.

Appendix B: Additional analyses

Here we provide ISL analyses for three additional opaque generalisations, to add to the evidence presented in § 3 that opaque interactions preserve the ISL property.

1 Counterfeeding on environment in Bedouin Arabic

McCarthy (1999) analyses two generalisations in Bedouin Arabic as counterfeeding on environment and counterfeeding on focus. Counterfeeding on environment is exemplified in Bedouin Arabic in (20a). The two processes that interact are shown in (20b). (In McCarthy 1999, the raising rule is actually presented as: a → i / __ σ. We use CV for σ, to simplify the exposition. Modelling the more general version of the rule in which the syllable that follows the target vowel could take more than one shape would still be an ISL map, though possibly for a higher value of k.)

(20) a. /badw/ ↦ [badu] 'Bedouin' b. i. a → i / __ CV
ii. G → V / C __ #

This is a case of apparent underapplication: (20b.i) does not raise the /a/ in [badu], because it isn't until the application of (b.ii) that it falls into the raising environment.

Another way of stating this is that an /a/ is raised before an input /CV/ sequence, but not before an input /CG/ sequence. Our scanning window then only needs to pick out an /aCV/ sequence to determine whether or not to raise /a/, making this map 3-ISL. Table IX illustrates this. First, as shown in row (a), an input /a/ is output as λ, regardless of the preceding 2-suffix (X here is a variable representing any symbol). As before, this is because it is necessary to see what comes next before deciding whether the output of this /a/ should be [a] or [i]. Similarly, as in row (b), an input /C/ following an /a/ must also be output as λ, as its output also cannot yet be determined.

	2-suffix	input	output		2-suffix	input	output		2-suffix	input	output
a.	XX	a	λ	e.	aC	G	aC	i.	Xa	G	aG
b.	Xa	C	λ	f.	aC	a	aC	j.	CG	C	GC
c.	aC	V	iCV	g.	XC	G	λ	k.	CG	V	GV
d.	aC	C	aCC	h.	XV	G	G	l.	CG	⋉	V

Table IX
Abbreviated input/output table for counterfeeding on environment in Bedouin Arabic.

The output can be determined when the 2-suffix is /aC/. When the following input is a /V/, completing the target sequence /aCV/ for raising, the corresponding output is [iCV], as shown in row (c). When the input is anything else, such as a /C/ or a /G/, the /a/ of the /aC/ sequence is output without being raised. Note that for an input /a/ or /G/ only the 2-suffix itself is output; this is because the /a/ and /G/ in turn need to be withheld in order to see what their corresponding output is.

The other process in this interaction is simple. An input /G/ is output as λ when following a /C/ (row (g)), because again its actual output cannot yet be determined. It is output faithfully after a V, since it is clearly not in the environment for vocalisation (row (h)). It is also output faithfully after /a/ (row (i)), along with /a/ itself, which is now known to not be in the environment for raising.

The output for a /G/ following a C is then determined by the following input symbol. When an input /C/ follows the 2-suffix /CG/ (row (j)), the output is [GC], as the glide is not word-final, and therefore should not be vocalised. Likewise for an input /V/ following /CG/ (row (k)). In contrast, the output of a /⋉/ following an input /CG/ results in the /G/ being realised as [V] (row (l)).

To sum up, this interaction of processes is a 3-ISL map; the relationship between the input and the output can be determined by looking at the 2-suffix of each input symbol. The derivations in (21) show how the above input/output tables obtain the correct outputs for /CaCV/ (in which raising occurs), /CVCG/ (in which vocalisation occurs) and /CaCG/ (in which vocalisation, but not raising, occurs – this corresponds to /badw/ in (20a)), using a scanning window of size 3.

(21) a. /CaCV/

	⋊ C a C V ⋉
	⋊ C
(a)	⋊ C a C V ⋉
	⋊ C λ
(b)	⋊ C a C V ⋉
	⋊ C λ λ
(c)	⋊ C a C V ⋉
	⋊ C λ λ iCV
	⋊ C a C V ⋉
	⋊ C λ λ iCV ⋉
	[CiCV]

b. /CVCG/

	⋊ C V C G ⋉
	⋊ C
	⋊ C V C G ⋉
	⋊ C V
	⋊ C V C G ⋉
	⋊ C V C
(g)	⋊ C V C G ⋉
	⋊ C V C λ
(l)	⋊ C V C G ⋉
	⋊ C V C λ V ⋉
	[CVCV]

c. /CaCG/

	⋊ C a C G ⋉
	⋊ C
(a)	⋊ C a C G ⋉
	⋊ C λ
(b)	⋊ C a C G ⋉
	⋊ C λ λ
(e)	⋊ C a C G ⋉
	⋊ C λ λ aC
(l)	⋊ C a C G ⋉
	⋊ C λ λ aC V ⋉
	[CaCV]

2 Counterfeeding on focus in Bedouin Arabic

Bedouin Arabic also provides an example of counterfeeding on focus. An [i] created by the application of the raising rule in (22b.ii) (repeated from (20b.i)) escapes deletion by the preceding syncope rule in (22a).

(22) a. /katab/ ↦ [kitab] 'he wrote'

b. i. i → Ø / __ CV
ii. a → i / __ CV

This map is also 3-ISL. The crucial input sequences are /iCV/ and /aCV/, which are output as [CV] and [iCV] respectively. The fact that ISL maps focus entirely on the input allows them to easily distinguish input /i/'s, which are subject to deletion, from the [i]'s derived by the raising rule, which survive in the output. Using the alphabet {i, a, u, C}, we represent this map in Table X.

	2-suffix	input	output		2-suffix	input	output		2-suffix	input	output
a.	⋊	a	λ	k.	uC	i	λ	t.	iC	C	iCC
b.	⋊C	a	λ	l.	Cu	i	λ	u.	Ca	u	au
c.	uC	a	λ	m.	Ca	i	a	v.	Ci	u	iu
d.	Cu	a	λ	n.	Ci	i	i	w.	aC	u	iCu
e.	Ca	a	a	o.	aC	i	iC	x.	iC	u	Cu
f.	Ci	a	i	p.	iC	i	C	y.	Xa	⋉	a⋉
g.	aC	a	iC	q.	Ca	C	λ	z.	Xi	⋉	i⋉
h.	iC	a	C	r.	Ci	C	λ	aa.	aC	⋉	aC⋉
i.	⋊	i	λ	s.	aC	C	aCC	ab.	iC	⋉	iC⋉
j.	⋊C	i	λ								

Table X

Abbreviated input/output table for counterfeeding on focus in Bedouin Arabic.

Rows (a–h) show the outputs for input /a/: (a–d) show that the output of /a/ is λ when it is word-initial (row (a)), follows an initial /C/ or /uC/ sequence (rows (b–c)) or follows the vowel /u/ (row (d)), while (e–h) show the non-empty outputs for input /a/. When the 2-suffix is /Ca/, corresponding to the input sequence /Ca/, the output is [a]. This is because the first /a/ can safely be output as [a], because it is not in the raising context. The second /a/, however, is now withheld, since its fate cannot yet be determined. When the 2-suffix is /Ci/, the output is [i] because the /i/ was being withheld but is now known to not be in the context for deletion (because it is followed by /a/ instead of C). The /a/ itself, however, is now withheld. When the 2-suffix is /aC/, the output reflects raising, because the /a/ in the 2-suffix is now known to be in the raising context (i.e. /aCa/). But again the input /a/ is withheld. Lastly, when the 2-suffix is /iC/, the output reflects deletion, because the /i/ in the 2-suffix is now known to be in the deletion context (i.e. /iCa/) (and again the input /a/ is withheld). Rows (i–p) show the outputs when the input is /i/, which shows an identical pattern to /a/.

The remaining rows show the outputs for inputs C, u and ⋉. Recall that when the suffix is /Ca/ or /Ci/ the /a/ or /i/ is in a state of being withheld until it can be determined whether to raise or delete respectively. An input /C/ that follows one of these suffixes (rows (q–r)) must also be withheld, since the output cannot be decided until it is determined whether the next symbol is a vowel. If a /C/ input instead follows either of the 2-suffixes /aC/ or /iC/ (rows (s–t)), what was being withheld is returned unchanged, along

with the input /C/ itself. When an input /u/ follows /Ca/ or /Ci/ (rows (u–v)), those vowels are returned unchanged (as we now know neither is in its respective context for being changed), followed by [u] itself. An /u/ following /aC/ or /iC/ (rows (w–x)), however, completes the environment for raising or deletion, and so the outputs reflect those changes: [iCu] and [Cu] respectively.

As for ⋉ inputs, when ⋉ is read at a point at which some input is being withheld, the withheld string is the output of ⋉. One way to think of this is that the substring that comprises the rule's structural description has been partially found. If the string ends before it is found in its entirety (i.e. in this example if the string ends in one of /a, i, aC, iC/), then the output of ⋉ 'returns' the withheld portion. In rows (y–z), /Xa/ and /Xi/ abbreviate all 2-suffixes in which any symbol is followed by /a/ and /i/ respectively (e.g. /Xa/ = {Ca, ua, aa, ia}).

(23) provides example derivations for Bedouin raising and deletion.

(23)	a.	/CaCu/							b.	/CiCu/							c.	/CaCi/	
		⋉	C	a	C	u	⋉			⋉	C	i	C	u	⋉			⋉	C	a	C	i	⋉
		⋉	C							⋉	C							⋉	C				
	(b)	⋉	C	a	C	u	⋉		(j)	⋉	C	i	C	u	⋉		(b)	⋉	C	a	C	i	⋉
		⋉	C	λ						⋉	C	λ						⋉	C	λ			
	(q)	⋉	C	a	C	u	⋉		(r)	⋉	C	i	C	u	⋉		(q)	⋉	C	a	C	i	⋉
		⋉	C	λ	λ					⋉	C	λ	λ					⋉	C	λ	λ		
	(w)	⋉	C	a	C	u	⋉		(x)	⋉	C	i	C	u	⋉		(o)	⋉	C	a	C	i	⋉
		⋉	C	λ	λ	iCu				⋉	C	λ	λ	Cu				⋉	C	λ	λ	iC	
		⋉	C	a	C	u	⋉			⋉	C	i	C	u	⋉		(z)	⋉	C	a	C	i	⋉
		⋉	C	λ	λ	iCu	⋉			⋉	C	λ	λ	Cu	⋉			⋉	C	λ	λ	iCi	⋉
		[CiCu]								[CCu]								[CiCi]					

3 Self-destructive feeding in Turkish

Finally, we look at an example from Turkish (Sprouse 1997) of self-destructive feeding, in which one process feeds another, and the environment of the first is partially destroyed by the application of the second. This results in apparent overapplication of the first process. In Turkish, epenthesis (24b.i) occurs between two consonants at the end of the word. Additionally, an intervocalic /k/ deletes before a morpheme boundary (24b.ii).

(24) a. /bebek+n/ ↦ [bebein] 'your baby'

b. i. $\emptyset \rightarrow i \,/\, C\_\_C\#$

ii. $k \rightarrow \emptyset \,/\, V\_\_+V$

As shown in (24a), epenthesis feeds deletion, but the input /k/ that triggers the epenthesis is itself deleted, obscuring the reason epenthesis occurred.

This map is 5-ISL. In the following, K represents dorsal consonants, C represents all other consonants and V represents vowels. The alphabet also includes + for morpheme boundaries, which are a necessary part of the deletion environment. If needed, morpheme boundaries can be removed from the output in a context-free manner by outputting all input +'s as λ no matter what the 4-suffix, as indicated in row (a) of Table XI.

	4-suffix	input	output		4-suffix	input	output		4-suffix	input	output
a.	XXXX	+	λ	h.	XXXK	C	λ	o.	XXCC	⋉	iC⋉
b.	XXXV	K	λ	i.	XXC+	C	λ	p.	XC+C	⋉	iC⋉
c.	XXVK	V	KV	j.	XXK+	C	λ	q.	VK+C	⋉	iC⋉
d.	XXVK	C	KC	k.	XXXC	K	λ	r.	XXCK	⋉	iK⋉
e.	XXVK	K	K	l.	XXXK	K	λ	s.	XC+K	⋉	iK⋉
f.	XXVK	⋉	K⋉	m.	XXC+	K	λ	t.	VK+K	⋉	iK⋉
g.	XXXC	C	λ	n.	XXK+	K	λ				

Table XI

Abbreviated input/output table for self-destructive feeding in Turkish.

An input K following a vowel is output as λ, as it may potentially be in the deletion environment (row (b)). If anything besides + follows a K, then the K can be output, as it is not in the deletion environment (which, recall, explicitly includes the morpheme boundary). This is illustrated in rows (c–f).

Any C or K following another C or K is output as λ, as shown in rows (g–n). (Note that this occurs regardless of whether or not a morpheme boundary intervenes.) If the following input symbol is ⋉, then we have the environment for epenthesis, and [iC] or [iK] is output accordingly, as shown in rows (o–t). Note in particular rows (q) and (t), in which the 4-suffix includes a postvocalic K that precedes a morpheme boundary followed by a consonant. The output for ⋉ in these rows reflects both epenthesis and deletion (recall that the postvocalic K is being withheld, so its absence from the output of ⋉ achieves the deletion).

The derivation /CVK+C/ → [CViC] (corresponding to example (24a)) is given in (25). Note that the scanning window is of size 5, which is necessary to identify the situation in which deletion and epenthesis interact, namely a /VK+C⋉/ sequence. Thus the Turkish self-destructive feeding map is 5-ISL.

(25) /CVK+C/

⋊ C V K + C ⋉
⋊ C

⋊ C V K + C ⋉
⋊ C V

(b) ⋊ C V K + C ⋉
⋊ C V λ

(a) ⋊ C V K + C ⋉
⋊ C V λ λ

(j) ⋊ C V K + C ⋉
⋊ C V λ λ λ

(q) ⋊ C V K + C ⋉
⋊ C V λ λ λ iC⋉

[CViC]