Strictly Local Multi-Tape Transducers for Templatic Morphology

Hossep Dolatian and Jonathan Rawski
hossep.dolatian,jonathan.rawski@stonybrook.edu

Stony Brook University

Department of Linguistics, Stony Brook University

Introduction: computation

Objectives & Introduction

Goals: Want to compute templatic morphology such that...

1. **Model** it as productive process
2. **Understand** its computational properties
3. **Limit** its expressive power

Contribution: Use multi-tape finite-state transducers with local subclasses

Templatic morphology

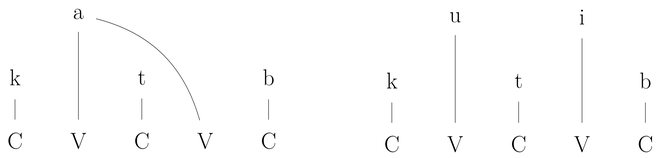
Most languages use concatenative morphology

- English progressive: *hold* → *hold-ing*

Semitic has templatic morphology (McCarthy, 1981)

Active verbs
katab 'it wrote'

1. Inflectional vowels V: *a*
2. Root consonants C: *ktb*
3. Template T: *CV.CVC*



Passive verbs
kutib 'it was written'

1. Inflectional vowels V: *ui*
2. Root consonants C: *ktb*
3. Template T: *CV.CVC*

Locality in Natural language

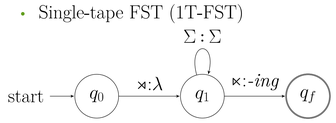
Most natural language processes are local
= uses *bounded* finite window (Chandlee, 2014)
= computed with N-grams
Our question:

- Are Semitic templates local?

Computing morphology

Computing concatenation

- *hold* → *hold-ing*
- Easy to compute
  - Single-tape FST (1T-FST)



- Computationally local
  - Input-Strictly Local (ISL)

Templatic morphology

- ***kutib*** 'to be written'
- How to compute?
  - 1T-FSTs aren't for non-linearity
  - State-explosion = large finite language
  - Modifications redirect state-explosion (Bird and Ellison, 1994; Beesley and Karttunen, 2003)
- Is it local?
  - *seems* non-local because of discontinuity
  - Unknown locality

Multi-tape FSTs for templates

MT-FSTs are elegant formalism for template-filling

1. **Modeling**: models productivity of Arabic templates
2. **Locality**: template-filling is local over multiple tapes
3. **Expressivity**: expressivity is constrained by locality + morphology

Single vs. Multi-tape FSTs

Single-tape FST (1T-FST)

1. read input as linear string
2. Most common formalism in CL + NLP (Mohri, 1997)
3. Advances on only one tape

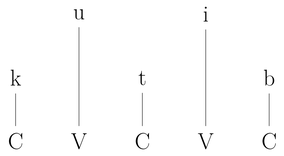
Multi-tape FST (MT-FST)

1. Input is multiple items that are read together
2. Early and intuitive model for Semitic (Kay, 1987; Kiraz, 2001)
3. Advance on every input tape either
   - **synchronously** *or* (= 1T FST)
   - **asynchronously** (» 1T FST)

What does an Arabic MT-FST look like?

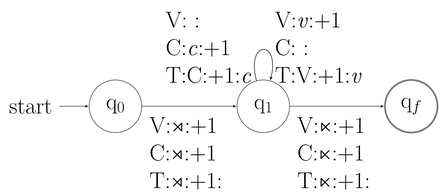
Morphology has 3 input items

1. Inflectional V: *ui*
2. Root C: *ktb*
3. Template T: *CV.CVC*



MT-FST has 4

1. Read input on each input tape (3)
2. Advance on each input tape
3. Create one output tape (1)



Watch it!

	Current State	C-tape		V-tape		T-tape		Output Symbol	Output String
1.	q_0	⋊ktb⋉		⋊ui⋉		⋊CVCVC⋉			
2.	q_1	⋊ktb⋉	C:⋊:+1	⋊ui⋉	V:⋊:+1	⋊CVCVC⋉	T:⋊:+1	ϵ	
3.	q_1	⋊ktb⋉	C:*k*:+1	⋊ui⋉	V:ϵ:0	⋊CVCVC⋉	T:*C*:+1	*k*	*k*
4.	q_1	⋊ktb⋉	C:ϵ:0	⋊ui⋉	V:*u*:+1	⋊CVCVC⋉	T:*V*:+1	*u*	*ku*
5.	q_1	⋊ktb⋉	C:*t*:+1	⋊ui⋉	V:ϵ:0	⋊CVCVC⋉	T:*t*:+1	*t*	*kut*
6.	q_1	⋊ktb⋉	C:ϵ:0	⋊ui⋉,	V:*i*:+1	⋊CVCVC⋉	T:*V*:+1	*i*	*kuti*
7.	q_1	⋊ktb⋉,	C:*b*:+1	⋊ui⋉,	V:ϵ:0	⋊CVCVC⋉	T:*C*:+1	*b*	*kutib*
8.	q_1	⋊ktb⋉	C:⋉:+1	⋊ui⋉	C:⋉:+1	⋊CVCVC⋉	T:⋉:+1	ϵ	*kutib*

Computing locality

Concatenation (=suffixation) is computationally weak = *k-Input Strictly Local* subclass
= keep track of **ONLY** last k segments in input
Suffixation is 1-ISL:
= 1-ISL states keep track of last *k-1* (=0) seen input (*empty string*)
Template-filling (1-1 match) is 1-ISL *over* multiple tapes
= Change is based on *current* input symbol on *two* tapes

Locality across template types

Locality in template-filling depends on # of vowels, consonants, and slots

Matching	Input			Output	Power
1-1 Matching	*ktb*	*ui*	*CVCVC*	*ku.tib*	MT-1-ISL
Final spread	*ktb*	***a***	*CVCVC*	*k**a**.t**a**b*	MT-2-ISL
Gemination	*ktb*	*ui*	*CVC.**G**VC*	*ka**t**.t**a**b*	MT-2-ISL
Pre-association	*ksb*	*a*	*C**t**VCVC*	*k**t**a.sab*	MT-1-ISL
Partial copying	*brd*	*a*	*CVC.**F**VC*	*bar.**b**ad*	2-way
Total copying	*zl*	*ia*	*CVC.CVC*	***zil.zal***	2-way
Edge-in	*ktb*	*ua**i***	*mV-tV-CVC.CVC*	*mu-ta-kas.s**i**b*	unclear...
C-spreading	*trʒm*	*ui*	*CVC.CVC*	*tar.ʒam*	MT-3-ISL
	ktb	*ui*	*CVC.CVC*	*kut.ʈib*	MT-3-ISL

Discussion

Finiteness

- Arabic has at most 2 syllable templates
- Could model Arabic with just large finite language-language mapping
- But finiteness costs you generalizations (Savitch, 1993)

Phonological emergence

- MT-FST took *template T* as morphological input
- Contemporary theories argue there is *NO* template T, it's phonologically emergent
- But the phonology still needs to create a template (= MT-FST)

Expressivity

- Expressivity of MT-FST depends on how you encode your input
- With the right encoding, majority rules is MT-ISL
- Problem is curtailed by letting morphology dictate your encoding

Conclusions

Semitic templates are

- Typologically rare
- Theoretically cool
- Computationally local ... with the right representation = MT-FSTs

References

Beesley, K. and L. Karttunen (2003). Finite-state morphology: Xerox tools and techniques. CSLI Publications.
Bird, S. and T. M. Ellison (1994). One-level phonology: Autosegmental representations and rules as finite automata. Computational Linguistics 20(1), 55–90.
Chandlee, J. (2014). Strictly Local Phonological Processes. Ph. D. thesis, University of Delaware, Newark, DE.
Kay, M. (1987). Nonconcatenative finite-state morphology. In Third Conference of the European Chapter of the Association for Computational Linguistics.
Kiraz, G. A. (2001). Computational nonlinear morphology: with emphasis on Semitic languages. Cambridge University Press.
McCarthy, J. J. (1981). A prosodic theory of nonconcatenative morphology. Linguistic inquiry 12(3), 373–418.
Mohri, M. (1997). Finite-state transducers in language and speech processing. Computational Linguistics 23(2), 269–311.
Savitch, W. J. (1993). Why it might pay to assume that languages are infinite. Annals of Mathematics and Artificial Intelligence 8(1-2), 17–25.

iACS INSTITUTE FOR ADVANCED COMPUTATIONAL SCIENCE