Finite State Morphology

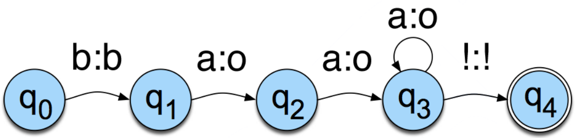
Roark & Sproat (2007)

Dan Wells

27 March 2015

Outline

Finite State Transducers



FST transducing sheep language to ghost language

- $Q = \{q_0, q_1, \ldots, q_{N-1}\}$: finite set of N states
- Σ: finite alphabet of input symbols
- Δ: finite alphabet of output symbols
- $q_0 \in Q$: start state
- $F \subseteq Q$: set of final states
- $\delta(q, i)$: transition function between states
- $\sigma(q, i)$: output function for a given state

Why use FSTs?

Advantages

- Efficient processing – linear in the length of the string for deterministic FSTs
- Closed under concatenation, union, Kleene closure, inversion and composition

Disadvantages

- Relatively limited generative power

Composition of FSTs

Composition: If a transducer T_1 maps I_1 to O_1 and T_2 maps I_2 to O_2, then $T_1 \circ T_2$ maps I_1 directly onto O_2

Roark & Sproat argue that composition of FSTs can be used to describe (nearly) all kinds of morphological processes.

Basic example: English plurals

For simple concatenative morphology, we might like to combine a stem A with an affix β to derive a new form Γ through string concatenation:

$$\Gamma = A \cdot \beta$$

However, it is often the case that stems and affixes undergo changes upon combination.

Example

Plural marker in English surfaces as:

- [s] after unvoiced segments e.g. ‘cats’
- [z] after voiced segments e.g. ‘dogs’
- [ɨz] following apical fricatives and affricates e.g. ‘horses’, ‘churches’

Basic example: English plurals

Instead, think in terms of a function β' that takes a string as input and returns that string concatenated with β:

$$\beta' = \Sigma^*[\epsilon : \beta]$$

This function defines a set of regular relations, and therefore also an FST, and so we can reframe the process using composition:

$$\Gamma = A \circ \beta'$$

Basic example: English plurals

We can define a transducer T which encodes the alternations in the English plural marker. Then we can produce a plural form Π from an English stem S and the plural suffix σ as follows:

$$\Pi = [S \cdot \sigma] \circ T$$

Which we can refactor as before:

$$\Pi = S \circ [\Sigma^*[\epsilon : \sigma]] \circ T$$

If we then define a function σ' as:

$$\sigma' = [\Sigma^*[\epsilon : \sigma]] \circ T$$

Then our final derivation is:

$$\Pi = S \circ \sigma'$$

Prosodic circumscription

Sometimes the domains of morphological processes are prosodically specified e.g. infixation in Tagalog:

tawag	$\rightarrow$	*t<u>um</u>awag*
'call'		'call (perfective)'

The infix *-um-* attaches as a prefix to the remainder of a word following any initial onset.

Prosodic circumscription

Prosodic circumscription formalises the definition of prosodic entities in these rules.

A base B can be decomposed into a prosodically defined unit B: and a residue $B/$ which are concatenated in some order:

$$B = B{:} \ \cdot \ B/$$

Morphological operations can then be defined as applying to either of these entities:

$$O{:} \ = O(B{:}) \ \cdot \ B/$$

$$O/ = B{:} \ \cdot \ O(B/)$$

Tagalog infixation

The definition of a prosodic unit can be implemented using an FST which inserts a marker (e.g. $>$) at the appropriate point in a string.

For our Tagalog example, a transducer M can be defined as:

$$M = C?[\epsilon :>]V\Sigma^*$$

Another transducer ι rewrites this marker to the infix *-um-* and appends a perfective marker [+be] to the resulting word form:

$$\iota = \Sigma^*[>: \mathsf{um}]\Sigma^*[\epsilon : +\mathsf{be}]$$

The whole operation can then be applied to a stem A as:

$$\Gamma = A \circ M \circ \iota$$

Arabic templatic morphology

Verb stems in Arabic are derived under a non-concatenative 'root-and-pattern' system, with consonantal roots (e.g. *ktb* 'notion of writing') being combined with characteristic vocalic patterns:

Pattern	Template	Verb stem	Gloss
I	$C_1aC_2aC_3$	*katab*	'wrote'
II	$C_1aC_2C_2aC_3$	*kattab*	'caused to write'
III	$C_1aaC_2aC_3$	*kaatab*	'corresponded'
IV	$aC_1C_2aC_3$	*aktab*	'caused to write'
VI	$taC_1aaC_2aC_3$	*takaatab*	'wrote to each other'
VII	$nC_1aC_2aC_3$	*nkatab*	'subscribed'
VIII	$C_1taC_2aC_3$	*ktatab*	'copied'
X	$staC_1C_2aC_3$	*staktab*	'caused to write'

Arabic templatic morphology

For a finite state account of this kind of morphological system, we can begin by defining a root

$$P = ktb$$

and a set of CV templates

$$\begin{aligned} patterns = \{\ & \tau_{\text{I}} = CaCaC, \\ & \tau_{\text{II}} = CaCCaC, \\ & \ldots \\ & \tau_{\text{X}} = [\epsilon : sta]CCaC\ \} \end{aligned}$$

Arabic templatic morphology

We are then able to define a transducer corresponding to all of these templates by taking the union:

$$\tau = \bigcup_{p \in patterns} \tau_p$$

We also need a transducer linking roots to templates. This has two components:

- A transducer introducing optional vowels between consonants:

$$\lambda_1 = C[\epsilon : V]^* C[\epsilon : V]^* C$$

- A transducer encoding a consonant doubling rule as in τ_{II}:

$$\lambda_2 = C_i \to C_i C_i$$

Composing these linking transducers gives us $\lambda = \lambda_1 \circ \lambda_2$, and finally we can derive the entire set of related verb stems from the consonantal root *ktb* by composing everything together:

$$\Gamma = P \circ \lambda \circ \tau$$

Reduplication

Reduplication is problematic for finite state models because it involves copying strings, and FSTs are not equipped to handle unbounded copying.

It is possible, however, to account for *bounded* copying through exhaustive enumeration of strings within the domain of the copying operation.

- So we can do it, but it's messy

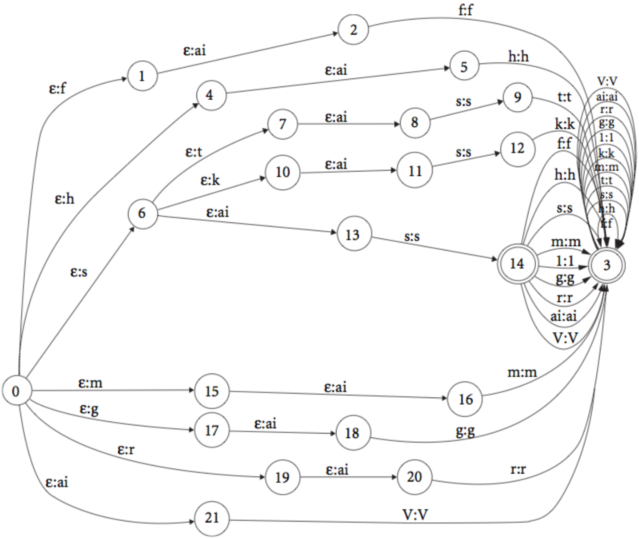
Bounded reduplication in Gothic

Infinitive	Gloss	Preterite
falþan	‘fold’	faífalþ
haldan	‘hold’	haíhald
ga-staldan	‘possess’	ga-staístald
af-áikan	‘deny’	af-aíáik
máitan	‘cut’	maímáit
skáidan	‘divide’	skaískáiþ

- Prefix a syllable of the form *(A)Caí* to the stem
- Copy any onset of the stem to the *C* position and any pre-onset appendix to the *(A)* position
- Closed class of verbs ⇒ bounded reduplication

Bounded reduplication in Gothic

But...



Unbounded reduplication in Bambara

wulu dog	*o* MARKER	*wulu* dog	‘whichever dog’
wulu-nyinina dog searcher	*o* MARKER	*wulu-nyinina* dog searcher	‘whichever dog searcher’
malo-nyinina-filèla rice searcher watcher	*o* MARKER	*malo-nyinina-filèla* rice searcher watcher	‘whichever rice searcher watcher’

Where any number of compounds could serve as input to this reduplication process, it becomes impossible to precompile all possible copies as we did for Gothic.

Dealing with (bounded) reduplication

Roark & Sproat break the problem down into two components:

- Model prosodic constraints on base and reduplicated portion e.g. for Gothic that reduplicated portion is of the form *(A)Caí*
- Construct a copying component which verifies that the reduplicated portion appropriately matches the base

Dealing with (bounded) reduplication

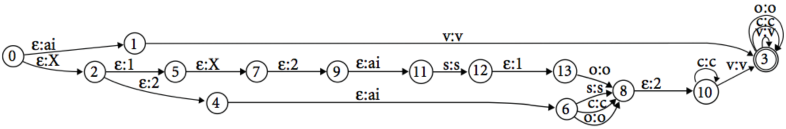
Prosodic constraint

Assume a transducer R which when composed with a base β returns a prefixed version of β, and which also adds indices to the elements in β which should match co-indexed elements in the reduplicated prefix:

$$\alpha = \beta \circ R = (A_1)C_2\text{aí}\beta'$$

Here β' is the indexed version of β.

Example: skáiþ $\circ R = X_1 X_2$aí$\text{s}_1\text{k}_2$áiþ



Dealing with (bounded) reduplication

Copy filter

Checking the identity of co-indexed arcs can be achieved by implementing a set of finite state filters, one for each index. For bounded reduplication we can define a filter as below:

$$\bigcup_{i \in indices} \bigcup_{s \in segments} \overline{[\Sigma^* s_i \Sigma^* \overline{s_i} \Sigma^*]}$$

Summary

- If the problem allows it, FSTs provide very efficient processing
- But limited generative power restricts the kinds of structures and patterns able to be recognised
- Applications for morphological parsing and text normalisation in speech synthesis

Any questions?