# Nanjing FST

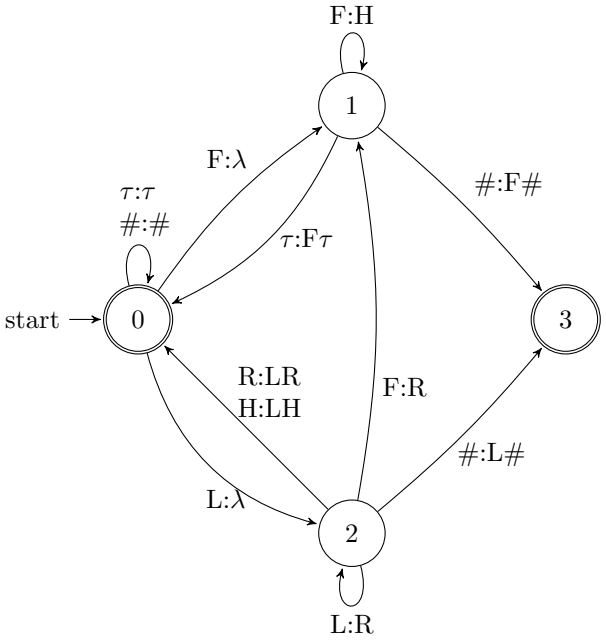
Chris Oakden

July 11, 2018

## 1 FST on syllable-level representation

Below is an FST that captures tone sandhi processes in the Nanjing dialect (as ISL functions). Alternations involving checked tones are omitted, and syllable-level representation is used.

(1)




Word (or domain) boundaries are included in the representation, as I haven't been able to figure out how to get the "right-dominant" nature of the patterns within some reference to word boundaries. Transductions on relevant inputs (specifically, lines 1,2,3 and 6 from previous handout (8)) follow.

(2)

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | F | | F | | F | | # | |
| 0 | ⇒ | 1 | ⇒ | 1 | ⇒ | 1 | ⇒ | (3) |
| | λ | | H | | H | | F# | |

For the string FFF, the machine begins in state 0, taking the F:λ transition to state 1 upon reading a F in the input. In intuitive turns, this state is akin to: I have seen a F, if I see another F, I will output a H, otherwise a F. The machine will output transductions on inputs /F$^n$/ for $n$ of any size, as well as string-medial /F$^n$/ sequences. This is guaranteed by the transition τ:Fτ between states 1 and 0 and the looping τ:τ transition on state 0 (where τ represents either R or H).

The following two example transductions illustrate how the machine incorporates the disyllabic sandhi rules L → R / __ F and L → R / __ L.

(3)
| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | L | | F | | F | | # | | | |
| 0 | ⇒ | 2 | ⇒ | 1 | ⇒ | 1 | ⇒ | (3) | | |
| | λ | | R | | H | | F# | | | |

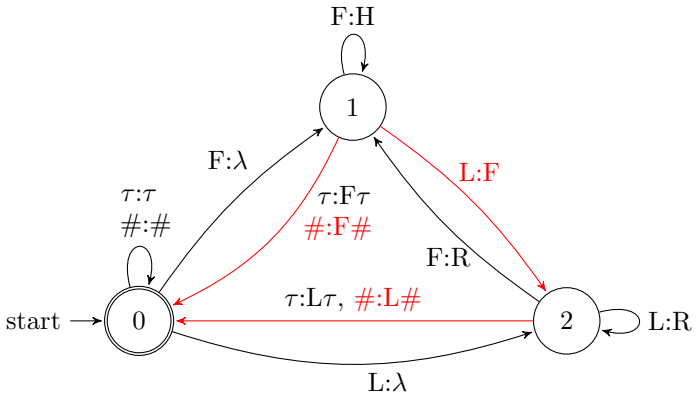
(4)
| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | L | | L | | F | | # | | | |
| 0 | ⇒ | 2 | ⇒ | 2 | ⇒ | 1 | ⇒ | (3) | | |
| | λ | | R | | R | | F# | | | |

(5)
| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | L | | L | | L | | # | | | |
| 0 | ⇒ | 2 | ⇒ | 2 | ⇒ | 2 | ⇒ | (3) | | |
| | λ | | R | | R | | L# | | | |

Transductions (3) - (5) show that state 2 is L's analog of H's state 1; once a L has been read in the input, the machine waits to read the following symbol to see whether to output a L or R. The cyclic transition L:R captures the canonical "Mandarin 3rd Tone Sandhi" cases, while the transition F:R to state 1 represents the rule L → R / __ F.

This is a good first run, but it's not perfect. For one, we do not yet have a transition which can account accurately for /$\tau^n$FLL$\tau^n$/ sequences. Our current machine yields *FLL, since there is no transition from state 1 to state 2. Additionally, state 3 and its transitions are (potentially) superfluous. Since state 0 is an accepting state, we can collapse the #:F# and #:L# transitions into those which depart from states 1 and 2 (respectively) and terminate in state 0. The FST below shows these changes, highlighted in red.

(6)




This machine produces the same transductions as (2) - (5), with the added benefit of accurate transductions on /$\tau^n$FLL$\tau^n$/ sequences (as well as being less redundant), as shown below:

(7)
| | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | H | | H | | F | | L | | L | | # | |
| 0 | ⇒ | 0 | ⇒ | 0 | ⇒ | 1 | ⇒ | 2 | ⇒ | 2 | ⇒ | (0) |
| | λ | | H | | H | | F | | R | | L# | |

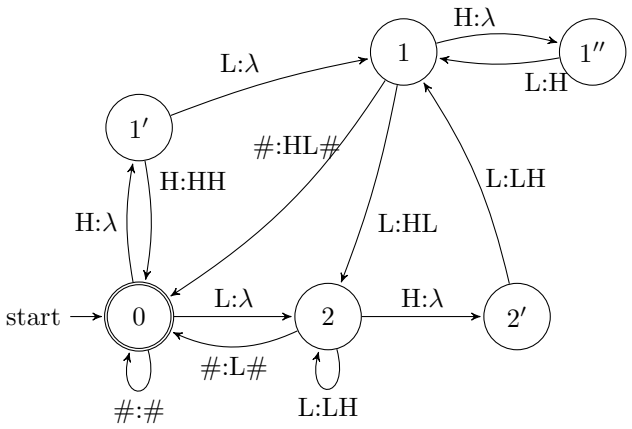
In the next section, we construct at an FST using melody-tier representations. Will it work?

## 2 FST on melodic-tier strings

Intuitively, translating the FST in (6) into one that reads symbols on the melodic tier should be straightforward; some intermediate states will be necessary to capture the fact that two symbols on the melodic tier

correspond to a single symbol in syllable-level representation (HL ≡ F; LH ≡ R). The FST below retains the initial state numbers from (6) and many of its transitions, but adds extra states and transitions with such equivalences in mind.

(8)




This machine produces outputs from input strings for trisyllabic sandhi forms using only H and L symbols, and does so accurately (9 - 12):

(9) /HLHLHL/ ↦ [HHHL] ≡ /FFF/ ↦ [HHF]

| | H | | L | | H | | L | | H | | L | | # | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | ⇒ | 1′ | ⇒ | 1 | ⇒ | 1″ | ⇒ | 1 | ⇒ | 1″ | ⇒ | 1 | ⇒ | ◎ 0 |
| | | λ | | λ | | λ | | H | | λ | | H | | HL# |

(10) /LHLHL/ ↦ [LHHHL] ≡ /LFF/ ↦ [RHF]

| | L | | H | | L | | H | | L | | # | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | ⇒ | 2 | ⇒ | 2′ | ⇒ | 1 | ⇒ | 1″ | ⇒ | 1 | ⇒ | ◎ 0 |
| | | λ | | λ | | LH | | λ | | H | | HL# |

(11) /LLHL/ ↦ [LHLHHL] ≡ /LLF/ ↦ [RRF]

| | L | | L | | H | | L | | # | |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | ⇒ | 2 | ⇒ | 2 | ⇒ | 2′ | ⇒ | 1 | ⇒ | ◎ 0 |
| | | λ | | LH | | λ | | LH | | HL# |

(12) /LLL/ ↦ [LHLHL] ≡ /LLL/ ↦ [RRL]

| | L | | L | | L | | # | |
|---|---|---|---|---|---|---|---|---|
| 0 | ⇒ | 2 | ⇒ | 2 | ⇒ | 2 | ⇒ | ◎ 0 |
| | | λ | | LH | | LH | | L# |

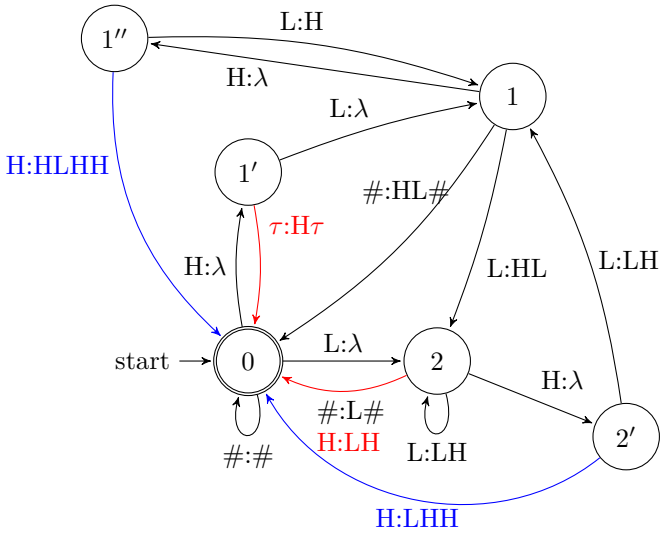
A crucial difference between this machine and the machine which accepts syllable-level symbols is that the latter operates over single syllables. Syllable boundaries are thus inherent in single symbols; the number of symbols in a given string will always be equivalent to the number of syllables represented by the string. The same cannot be said of the melody-tier FST given that syllables are represented by either one (H/L) or two (HL/LH) symbols. A given string may have multiple syllabifications, and since the current FST is deterministic, it is bound to produce transductions which do not reflect surface generalizations.

(13)

| Input | `cntr(w)` | Output | Observed Output |
|---|---|---|---|
| /F.F.F/ | HLHLHL | HHHL | HHHL (≡ HHF) |
| /H.L.H.L.H.L/ | HLHLHL | *HHHL | HLHLHL |
| /L.F/ | LHL | LHHL | LHHL (≡ RF) |
| /R.L/ | LHL | *LHHL | LHL |

This is reflective of the pitfalls of illicit substructure grammar characterizations of sandhi patterns on the melodic tier. Without adding syllable boundaries to the representation, it is possible to change the transitions on the existing FST to ensure that transductions on the strings above are possible, but the machine loses its determinism as a result (I don't know if we want that). These additional transitions are indicated in red below. Transitions added to create a total function are indicated in blue.

(14)




With these changes, transductions on inputs /H.L.H.L.H.L/ and /R.L/ are no longer problematic.

(15) /HLHLHL/ ↦ [HLHLHL]

| | H | | L | | H | | L | | H | | L | | # | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | ⇒ | 1′ | ⇒ | 0 | ⇒ | 1′ | ⇒ | 0 | ⇒ | 1′ | ⇒ | 0 | ⇒ | ◎0 |
| | λ | | HL | | λ | | HL | | λ | | HL | | # | |

(16) /LHL/ ↦ [LHL]

| | L | | H | | L | | # | |
|---|---|---|---|---|---|---|---|---|
| 0 | ⇒ | 2 | ⇒ | 0 | ⇒ | 2 | ⇒ | ◎0 |
| | λ | | LH | | λ | | L# | |