Disjunctive rule ordering in finite state morphology

Robert Malouf
San Diego State University

Chicago Linguistics Society
April 8, 2005

SAN DIEGO STATE UNIVERSITY

Finite state morphology

- Conventional finite state morphology (e.g., Beesley and Karttunen 2003)
- Finite state transducers (FSTs) for morphotactics:

cant	*are*+*Verb*:ε	*+1+Sg*:*o*
laud		*+2+Sg*:*as*
am		*+3+Sg*:*at*
		+3+Pl:*amus*
		+2+Pl:*atis*
		+3+Pl:*ant*

Finite state morphology

- FSTs for morphophonology
- Compose FSTs into a single FST mapping between lexical and surface forms

  Lexical: *cantare+Verb+1+Sg*

  Surface: *canto*

- Finite state techniques provide an efficient and well-founded framework for implementation
- Familiar 'Item and Arrangement' model of morphology

Word and Paradigm

- One well-known problem with this model is the treatment of discontinuous dependencies within words
- A solution to this and many other problems is offered by Word and Paradigm models of morphology (Robins, Matthews, Aronoff, Zwicky, Stump, et al.)
- In WP (aka *realizational* or *separationist*) models,
  - one set of rules build complete morphosyntactic representations
  - another set assigns a phonological form to a fully specified word

Word and Paradigm

- This eliminates a technical problem with FSTs
- Plenty of linguistic arguments that WP provides a more natural analysis of complex inflectional systems
- As Karttunen (2003) shows, WP finite state morphology requires only a slight change of perspective
  - Morphotactic rules construct lexical forms
  - Morphophonological rules build up the phonological realization of the given features
  - Final ‘clean-up’ transducer removes morphosyntactic features, leaving only the surface form

Word and Paradigm

- FSTs for morphotactics:

cantare	*+Verb*	*+1*	*+Sg*
laudare		*+2*	*+Pl*
amare		*+3*	

- Realizational rules:

  If *+Verb*, delete *-are* from stem

  If *+1+Sg*, add *-o* to stem
  If *+2+Sg*, add *-as* to stem

Disjunctive rule ordering

- Karttunen's (2003) implementation combines realization rules via serial composition
- *Conjunctive* rule ordering: every rule applies to every form
- However, an important property of realization rules is that they are *disjunctively* ordered
  - Within each rule block, only the first (or most specific) applicable rule applies
  - Earlier (or more specific) rules *block* the application of later (more general rules)

Disjunctive rule ordering

- Disjunctively ordered realizations for English *be*

is	3sg pres fin
am	1sg pres fin
are	pres fin
being	pres part
was	1sg or 3sg past fin
were	past fin
been	past part
be	elsewhere

- DATR/KATR (Finkel and Stump 2002)

Disjunctive rule ordering

- Compiler for rule blocks
  - construct “narrower than” relation
  - topological sort
  - disjunctive composition
- Implementation in prolog, using Gertjan van Noord’s FSA Utilities
- Follow general outline of Karttunen’s implementation

Realization rules

- A realization rule is pair $\langle \sigma, \phi \rangle$, where the *structural description* σ is a recognizer and the *structural change* ϕ is a transducer
- A *rule block* is an unordered set of realization rules
- For example:

  $\langle$ \$(pres fin), *be*:*are* $\rangle$
  $\langle$ \$(3sg pres fin), *be*:*is* $\rangle$
  $\langle$ \$(1sg pres fin), *be*:*am* $\rangle$
  $\langle$ \$(pres part), ε:*ing* $\rangle$

Realization rules

- Within a block, for any two rules either:
  - they are mutually inconsistent

    Two rules $\langle \sigma_1, \phi_1 \rangle$ and $\langle \sigma_2, \phi_2 \rangle$ are *mutually inconsistent* iff $\sigma_1 \,\&\, \sigma_2 = \varnothing$

  - or one must be narrower than the other

    A rule $\langle \sigma_1, \phi_1 \rangle$ is *narrower than* a rule $\langle \sigma_2, \phi_2 \rangle$ iff $\sigma_1 - \sigma_2 = \varnothing$

- We check every pair of rules in a block and construct the "narrower than" partial order

Disjunctive composition

- Given the “narrower than” partial order, we perform a topological sort on the rules within a block
- The result is a list of rules such that no rule is before a rule which it is narrower than
- For example:

⟨ $(3sg pres fin), *be*:*is* ⟩
⟨ $(1sg pres fin), *be*:*am* ⟩
⟨ $(pres fin), *be*:*are* ⟩
⟨ $(pres part), ε:*ing* ⟩

Disjunctive composition

- Next, we modify the structural description of each rule so that it excludes words which earlier rules would apply to

  $\langle \sigma_i, \phi_i \rangle$ becomes $\langle \sigma'_i, \phi_i \rangle$, where $\sigma'_i = \sigma_i - \cup_{j=1,i-1} \sigma_j$

- For example:

  $\langle$ \$(3sg pres fin), *be*:*is* $\rangle$
  $\langle$ \$(1sg pres fin) – \$(3sg pres fin), *be*:*am* $\rangle$
  $\langle$ \$(pres fin) – \$((3sg ∨ 1sg) pres fin), *be*:*are* $\rangle$
  $\langle$ \$(pres part) – \$(pres fin), ε:*ing* $\rangle$

- Cf. Erjavac's (1994) *ordered disjunction* and Karttunen's (1998) *priority union*

Disjunctive composition

- Finally, rules are converted to transducers using a *conditional replacement* operator
- The result is a set of mutually exclusive realization rules
- These FSTs can be combined into a single FST using normal composition
- They also may be combined with conjunctively ordered rules, parallel rules, etc.
- Implementation available with FSA utilities

http://odur.let.rug.nl/~vannoord/Fsa/

Conclusion

- Word and Paradigm morphology with disjunctively ordered realization rules fits comfortably within a finite state model of lexical processing
- This fact is of benefit to both theoretical linguistics and language engineers
  - Finite state models provide a robust, efficient platform to allow linguists to explore the consequences of their analyses
  - Realizational morphology provides an elaborated theoretical framework for language engineers to describe complex inflectional systems