Creating LRs with FSTs Part I

Overview

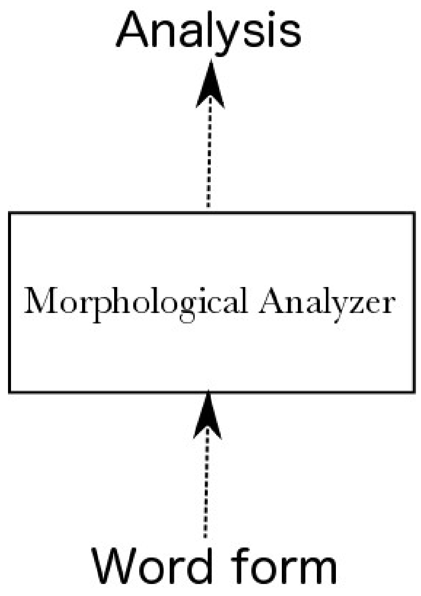
Mans Hulden
(University of Helsinki)
Iñaki Alegria
(University of The Basque Country)

Grand outline of tutorial

(1) Morphological analysis with finite-state technology
(2) Tools for compiling automata and transducers
(3) Specifying the lexicon descriptions (lexc)
(4) Compiling grammars with lexc & rewrite rules
(5) Advanced morphotactics
(6) Applications I: spell checking, spelling correction, etc.
(7) Applications II: surface syntactic parsing

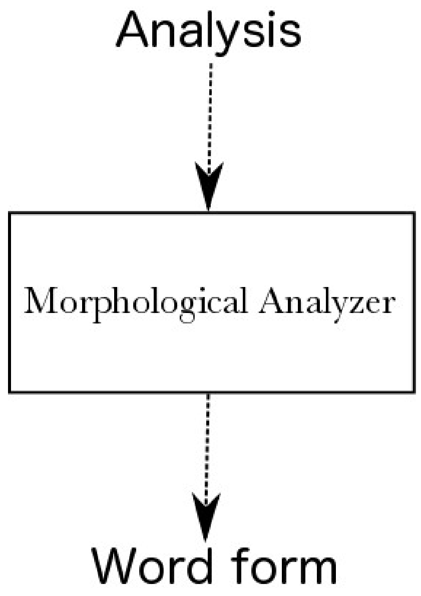
Morphological analysis...

- String-to-string translation of word forms to analyses...



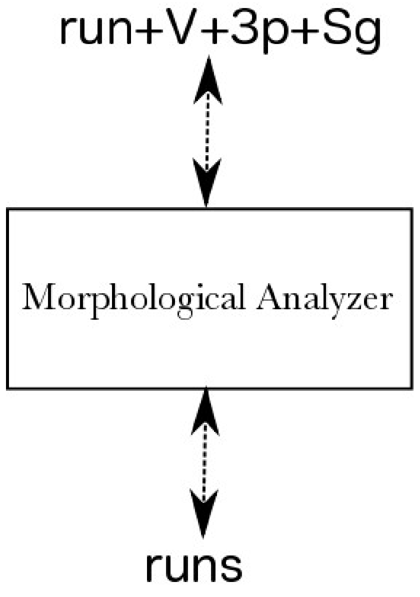
...and generation

- String-to-string translation of analyses to word forms...



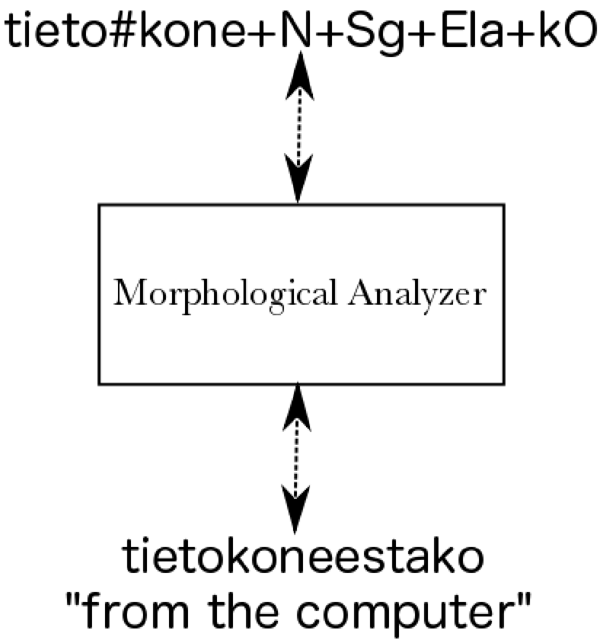
Morphological analysis

- English example (simple)



Morphological analysis

- Finnish example...
- “tietokoneestako”
- compound noun tieto + kone
- singular
- elative case
- question particle



Real-life example: Basque analyzer

```
foma[0]: load basque-whole-MI.fst
47.5 MB. 2915595 states, 3109378 arcs, Cyclic.
foma[1]: up etxeak
etxe[[Sarrera_etxe--0][KAT_IZE][AZP_ARR][BIZ_-]]+ak[[Sarrera_ak--1][KAT_DEK]
[KAS_ABS][NUM_P][MUG_M][FSL_[FS1_@OBJ][FS2_@PRED][FS3_@SUBJ]]]
etxe[[Sarrera_etxe--0][KAT_IZE][AZP_ARR][BIZ_-]]+ak[[Sarrera_ak--2][KAT_DEK]
[KAS_ERG][NUM_S][MUG_M][FSL_[FS1_@SUBJ]]]
```

etxe[[Sarrera_etxe--0][KAT_IZE][AZP_ARR][BIZ_-]]
(etxe-house, NOUN, COMMON, NOT ANIMATE)
+ak[[Sarrera_ak--1][KAT_DEK][KAS_ABS][NUM_P][MUG_M][FSL_[FS1_@OBJ][FS2_@PRED][FS3_@SUBJ]]]
(+ak (1), DECLEN, ABSOLUT, PLURAL, DETER, SYN_F OBJECT PREDICATE SUBJECT)

etxe[[Sarrera_etxe--0][KAT_IZE][AZP_ARR][BIZ_-]]
(etxe-house, NOUN, COMMON, NOT ANIMATE)
+ak[[Sarrera_ak--2][KAT_DEK][KAS_ERG][NUM_S][MUG_M][FSL_[FS1_@SUBJ]]]
(+ak (2), DECLEN, ERGATIVE, SING, DETERM, SYN_F SUBJECT)

Finite-state technology ...

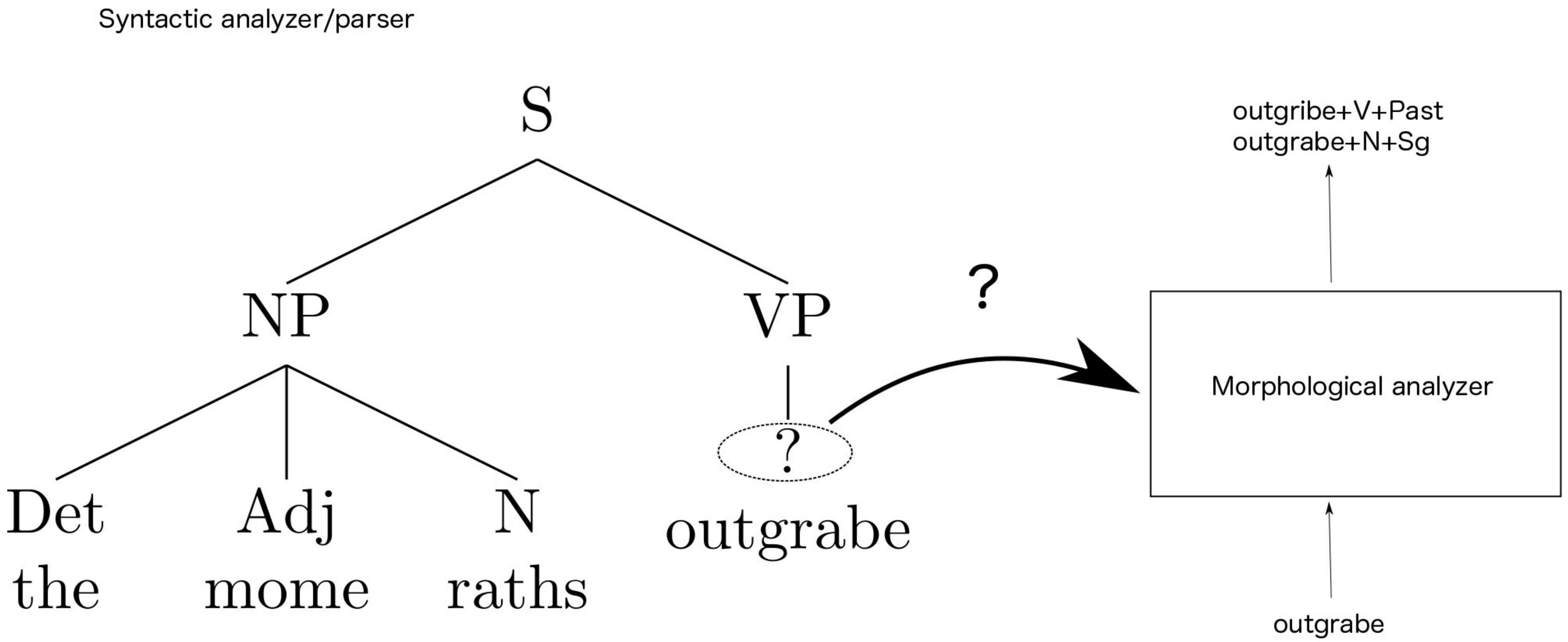
- Solves the problem ("...is a solved problem" [LK, 2003])
- Research in FST morphology since the early 1980s
- Some different formalisms around
  - Two-level rules (two-level morphology)
  - Composed rewrite rules (generative SPE rules)
- A variety of tools and applications around:
  - Xerox xfst (commercial, for composed rewrite rules)
  - Xerox twolc (commercial, for two-level rules)
  - foma (GPL license, for composed rewrite rules)
  - hfst-twolc (GPL license, for two-level rules)
- Freely available advanced tools fairly recent
  - foma (since 2009, GPL license)
  - HFST (since 2009, GPL license)

Applications that require morphological processing

- POS Tagger
- Shallow parser (chunker)
- Syntactic parser
- Information extraction
- Text-to-speech
- Machine translation

Example: syntactic parsing

- Generally consults a separate morphological analyzer



Direct derivatives

With a finite-state morphological analyzer for a language, one can with very little effort produce a:

- spell checker
- spelling corrector (for various types of errors)
- lemmatizer
- verb conjugator
- CALL tools
- electronic dictionary tools

Practical goals

- Give an overview of finite-state technology
- Focus on morphological analysis
- Learn how to write a morphological analyzer/generator with freely available tools
- Learn how to create derivative tools once a morphological grammar is built: spell checker, spelling correctors, and more
- Recognize other potential targets for finite-state technology: linguistic research (phonology and morphology), syntactic parser