FST Trimming: Ending Dictionary Redundancy in Apertium

Francis Tyers[0] Matthew Marting[1] Kevin Unhammer[2]

[0]UiT Norgga árktalaš universitehta
Romssa, Norga
ftyers@prompsit.com

[1]St. David's School
Raleigh, NC.
∅

[2]Kaldera språkteknologi
Stavanger, Noreg
unhammer+apertium@mm.st

27th May 2014

Outline of talk

Introduction and background

Implementation of `lt-trim`

Ending Dictionary Redundancy

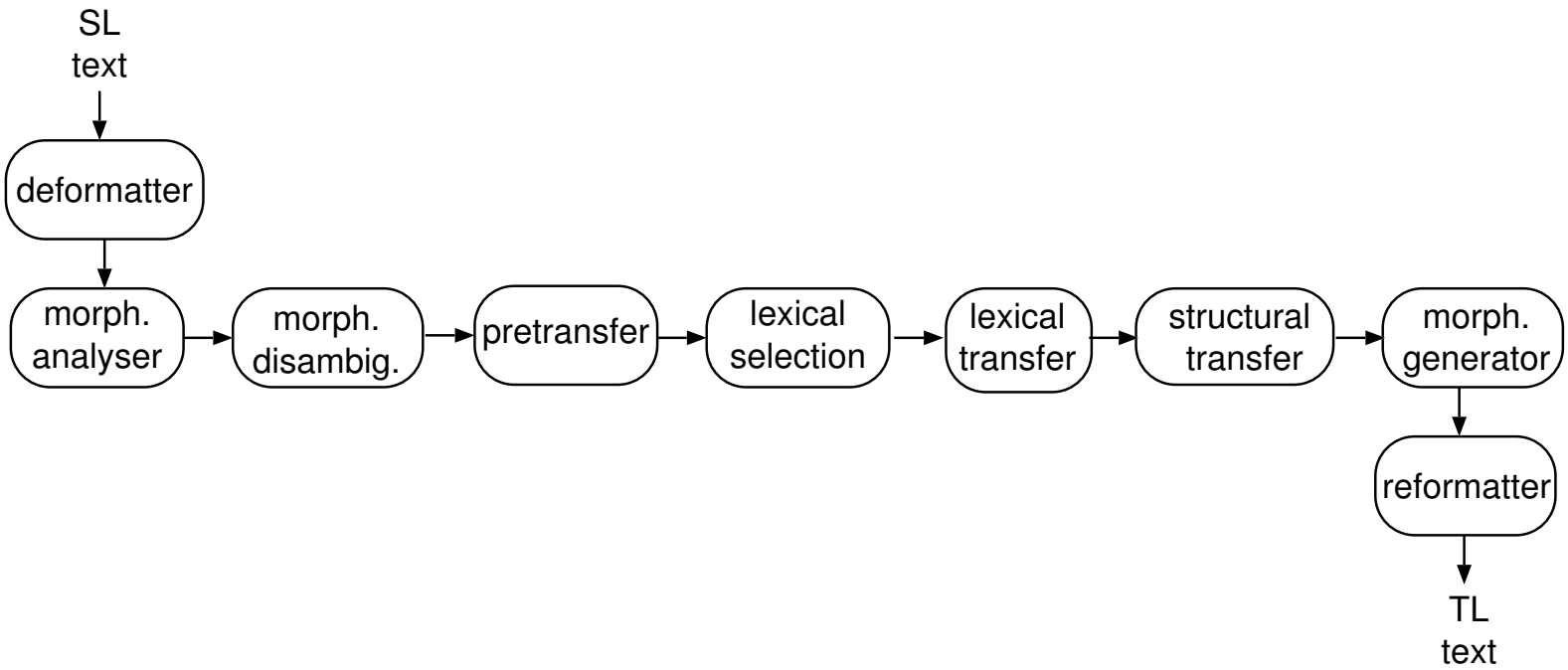
Conclusion

Introduction and background

- Apertium: Free/Open Source, Rule-based Machine Translation platform
- Uses lttoolbox Finite State Transducers for:
  - morph. analysis: ‘fishes’ to `fish<n><pl>/fish<vblex><pres>`
  - lex. transfer: `fish<n><pl>` to `fisk<n><m><pl><DD>`
  - morph. generation: `fisk<n><m><pl><def>` to ‘fiskane’

Francis Tyers,
Matthew Marting,
Kevin Unhammer

Apertium pipeline architecture



Francis Tyers,
Matthew Marting,
Kevin Unhammer

Multiword support

lttoolbox FST's support a variety of multiwords

An lttoolbox "lexical unit" is one token, and can be:

- simple non-multi-words: 'fish'
- simple space-separated words: 'hairy frogfish' as a single token
- multiwords with **inner inflection**: 'takes out', analysed as `take<vblex><pri><p3><sg># out`, converted to `take# out<vblex><pri><p3><sg>` before lexical transfer

Francis Tyers,
Matthew Marting,
Kevin Unhammer

Multiword support

- **joined** multiwords: ‘they’ll’;
  analysed as single token
  `prpers<prn><subj><p3><mf><pl>+will<vaux><inf>`,
  then split into two tokens
  `prpers<prn><subj><p3><mf><pl>` and
  `will<vaux><inf>` before lexical transfer

Multiword support

- combinations of the 3 multiword types: ‘creure-ho que’, analysed as single token `creure<vblex><inf>+ho<prn><enc><p3><nt># que`, then moved and split into two tokens `creure# que<vblex><inf>` and `ho<prn><enc><p3><nt>` before lexical transfer

Francis Tyers,
Matthew Marting,
Kevin Unhammer

The Problem: Redundant data

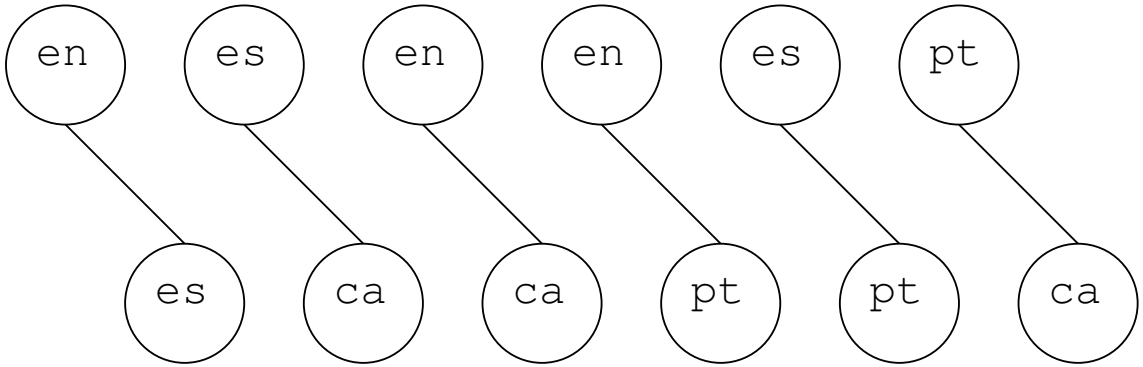


Figure: Current number of monodixes with pairs of four languages

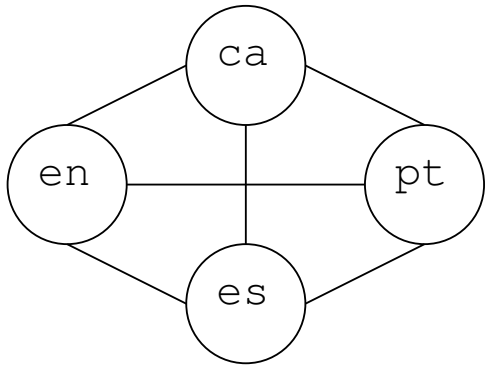


Figure: Ideal number of monodixes with four languages

A Solution: Intersection

Implementation of `lt-trim`

Preprocessing the bilingual dictionary

Prefixing the bilingual dictionary

Moving uninflected lemma parts

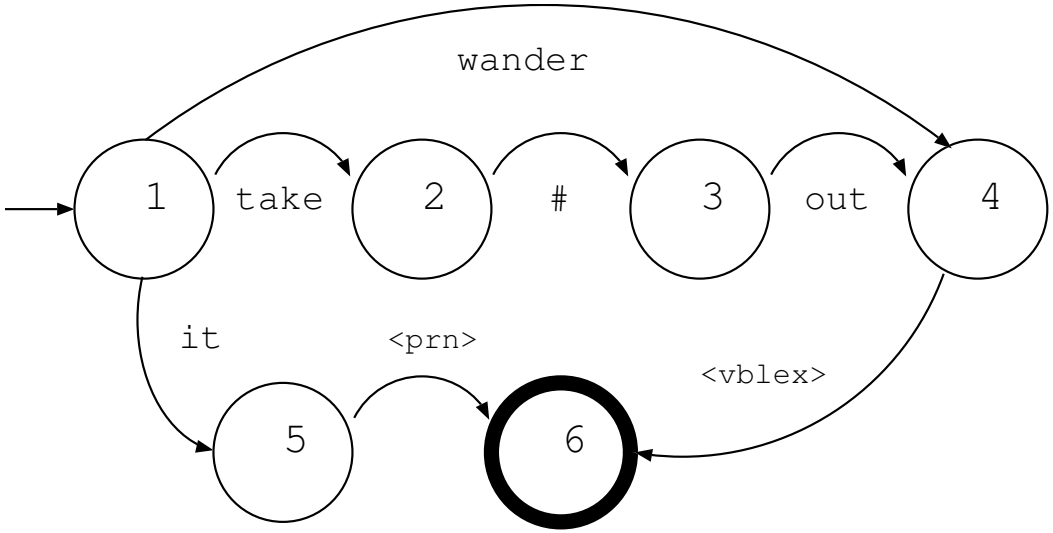
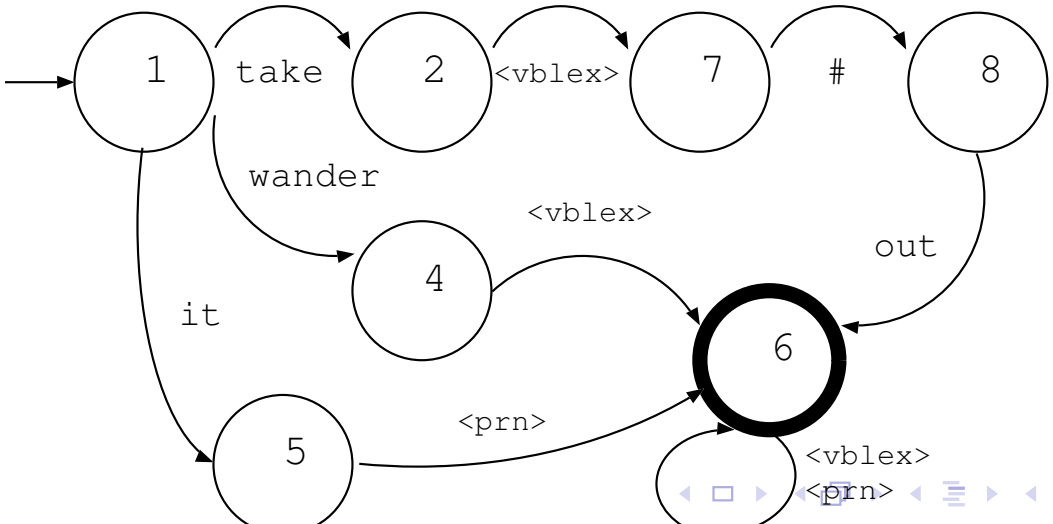


Figure: Input bilingual FST (letter transitions compressed to single arcs)



Intersection

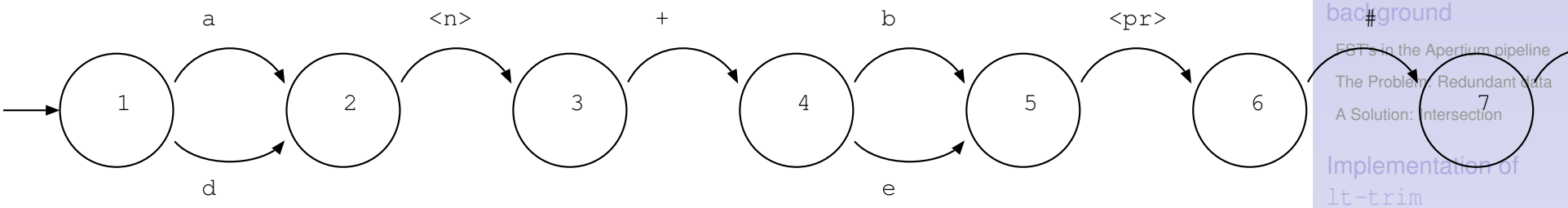


Figure: Input monolingual FST

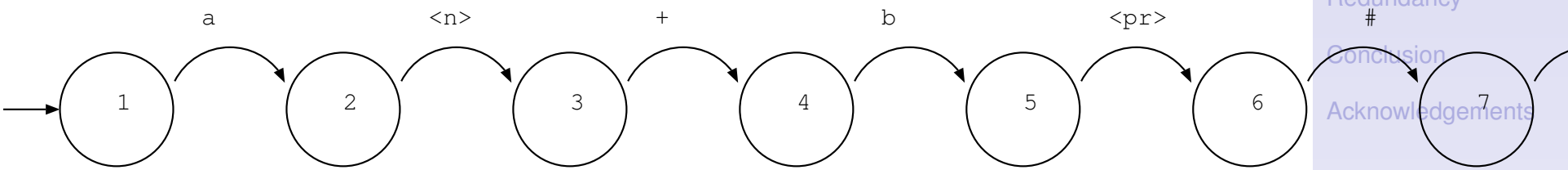


Figure: Trimmed monolingual FST

lt-trim in use

Ending Dictionary Redundancy

Conclusion

Acknowledgements