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

Introduction and background
FST's in the Apertium pipeline
The Problem: Redundant data
A Solution: trim on compile
Implementation of lt-trim
Preprocessing the bilingual dictionary
Intersection
lt-trim in use
Ending Dictionary Redundancy
Conclusion

Outline of talk

Introduction and background

Implementation of `lt-trim`

Ending Dictionary Redundancy

Conclusion

Introduction and background
FST's in the Apertium pipeline
The Problem: Redundant data
A Solution: trim on compile
Implementation of `lt-trim`
Preprocessing the bilingual dictionary
Intersection
`lt-trim` in use
Ending Dictionary Redundancy
Conclusion

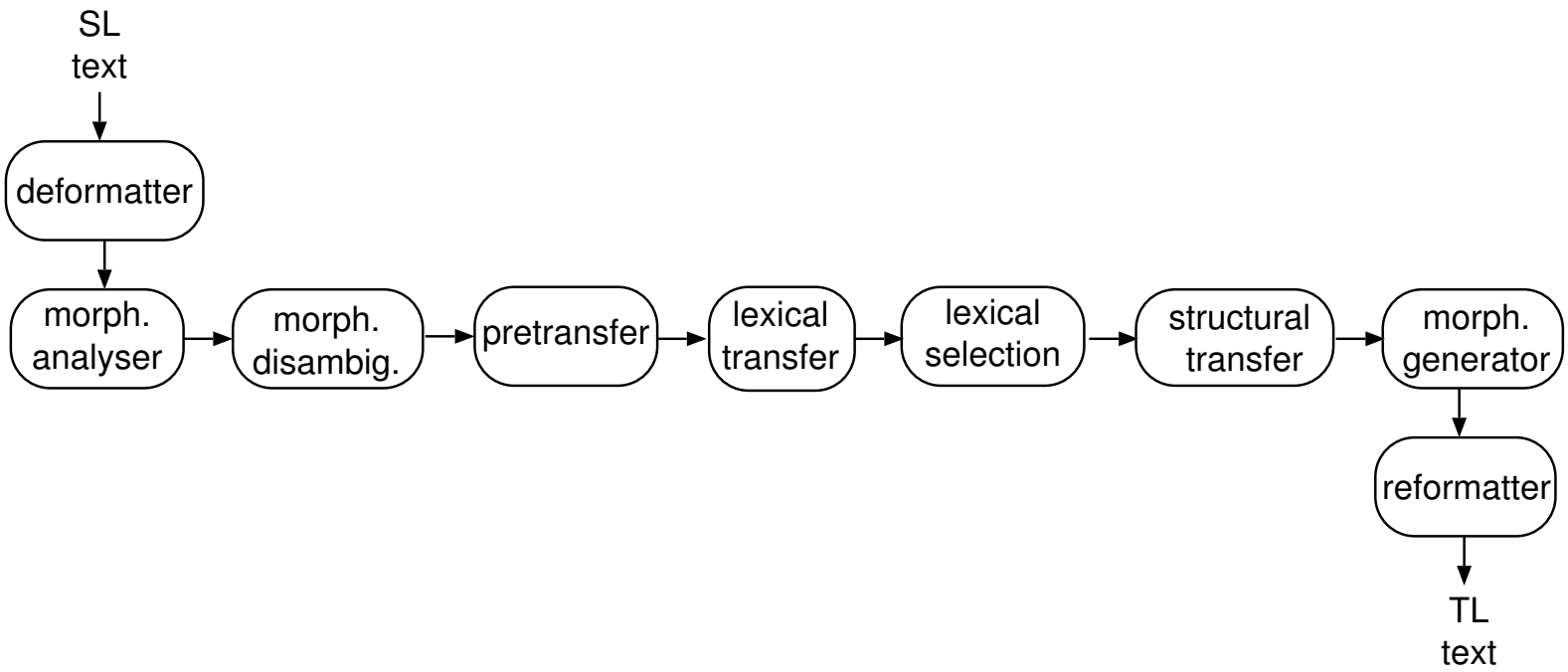
Introduction and background

- Apertium: Free/Open Source, Rule-based Machine Translation platform
- Goals include:
  - supporting lesser-resourced languages
  - wide coverage
  - post-editable output
  - reusable resources
- Language data (dictionaries, etc.) typically organised in language *pairs* (Catalan-Spanish, Portuguese-Spanish, etc.)
  - historically: each with its own copy of monolingual data

Introduction and background
FST's in the Apertium pipeline
The Problem: Redundant data
A Solution: trim on compile
Implementation of lt-trim
Preprocessing the bilingual dictionary
Intersection
lt-trim in use
Ending Dictionary Redundancy
Conclusion

Apertium pipeline architecture

- lttoolbox Finite State Transducers used for, among others:
  - morph. analysis: ‘fishes’ to `fish<n><pl>/fish<vblex><pres>`
  - lex. transfer: `fish<n><pl>` to `fisk<n><m><pl>`
  - morph. generation: `fisk<n><m><pl><def>` to ‘fiskane’



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

Introduction and background
FST's in the Apertium pipeline
The Problem: Redundant data
A Solution: trim on compile
Implementation of `lt-trim`
Preprocessing the bilingual dictionary
Intersection
`lt-trim` in use
Ending Dictionary Redundancy
Conclusion

Multiword support

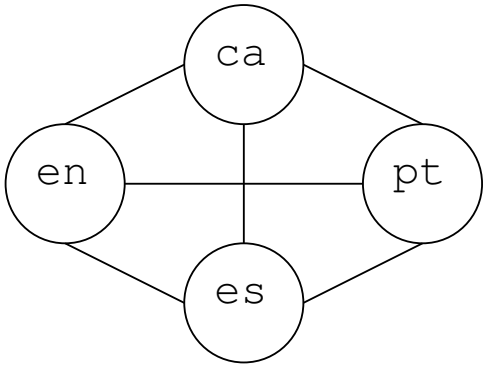
- **joined** multiwords: ‘they’ll’;
  analysed as single token
  `prpers<prn><subj><p3><mf><pl>+will<vaux><inf>`,
  then split into two tokens
  `prpers<prn><subj><p3><mf><pl>` and
  `will<vaux><inf>` before lexical transfer
- **compounds**: ‘frogfish’;
  analysed as single token `frog<n><sg>+fish<n>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

Introduction and background
FST’s in the Apertium pipeline
The Problem: Redundant data
A Solution: trim on compile
Implementation of `lt-trim`
Preprocessing the bilingual dictionary
Intersection
`lt-trim` in use
Ending Dictionary Redundancy
Conclusion

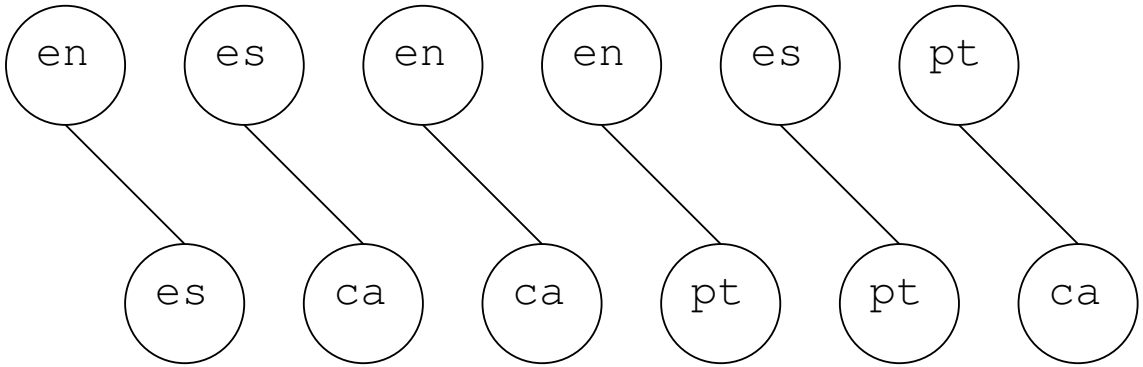
Multiword support

- combinations (space-separated + joined + inner inflection): ‘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

The Problem: Redundant data



Ideal number of monodixes with four languages



Current number of monodixes with pairs of four languages

Introduction and background
FST's in the Apertium pipeline
The Problem: Redundant data
A Solution: trim on compile
Implementation of lt-trim
Preprocessing the bilingual dictionary
Intersection
lt-trim in use
Ending Dictionary Redundancy
Conclusion

Words in analyser but missing from lexical transfer can be problematic:

- ‘fishes’ to ‘@fish’: loses the inflection
- ‘gikk til hundene’ “went to the dogs” to ‘went to @hund’ “went to dog”: losing the inflection hides the idiomatic meaning
- ‘öldürmedi’ “did not kill” to ‘@öl’ “kill”: loses the *negation*
- lexical transfer is also tag transfer – structural transfer thus needs exceptions for half-translated tags

But, most importantly, multiword tokenisation means that

‘He takes out the trash’ translates to ‘Han @take out søpla’ *even though both ‘take’-‘ta’ and ‘out’-‘ut’ are in the bilingual dictionary*.

Adding more words makes the translator worse!

A Solution: trim on compile

Compile a *trimmed* analyser-FST containing only those entries from original analyser FST that would pass through bilingual FST.

- FSA's closed under intersection: FSA1 ∩ FSA2 = FSA3
- Similarly, we can compose-intersect FST's:
  output-side of FST1 ∩ input-side of FST2 = FST3

Goal: One big monolingual source dictionary, trimmed during compile to language-pair specific analysers.
Dictionaries in HFST instead of lttoolbox can trim already (but it breaks with compounds!).
Most Apertium dictionaries use lttoolbox.

Introduction and background
FST's in the Apertium pipeline
The Problem: Redundant data
A Solution: trim on compile
Implementation of lt-trim
Preprocessing the bilingual dictionary
Intersection
lt-trim in use
Ending Dictionary Redundancy
Conclusion

Our tool needs some exceptions to compose-intersect:

- Append `.*` (any-symbol loop) to bilingual FST
  - Lexical transfer only needs a match on the start of the string
- Reorder `#`-multiwords in bilingual FST
  - so they look like analyser (else they won't match)
- Let `+` in analyser mean transition-to-start in bilingual FST
  - since single token `a+b` in analyser is split into two tokens `a b` before lexical transfer

Francis Tyers,
Matthew Marting,
Kevin Unhammer

Introduction and background
FST's in the Apertium pipeline
The Problem: Redundant data
A Solution: trim on compile
Implementation of `lt-trim`
Preprocessing the bilingual dictionary
Intersection
`lt-trim` in use
Ending Dictionary Redundancy
Conclusion

Digression: Some Apertium dictionaries use HFST instead of lttoolbox.

Trimming with standard HFST commands works for most but not all such dictionaries.

Again multiword issues: compounds.

Introduction and background
FST's in the Apertium pipeline
The Problem: Redundant data
A Solution: trim on compile
Implementation of lt-trim
Preprocessing the bilingual dictionary
Intersection
lt-trim in use
Ending Dictionary Redundancy
Conclusion

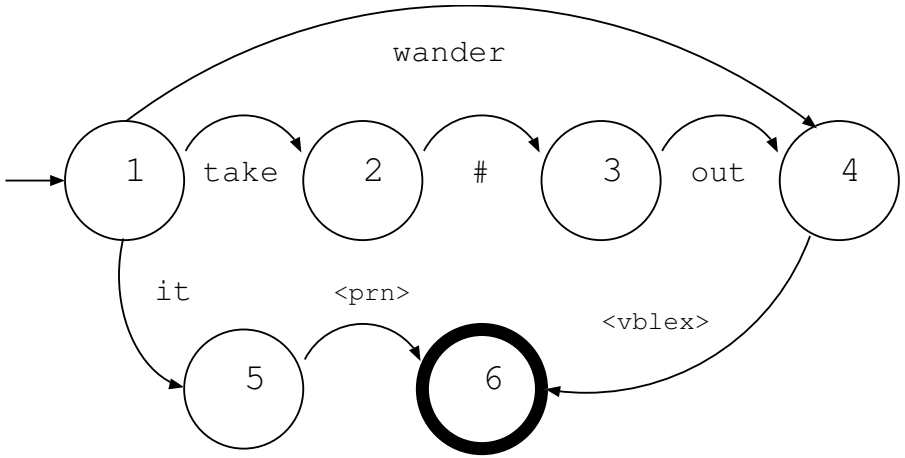
Implementation of `lt-trim`

Tool takes two compiled FST's, produces a new, *trimmed* FST

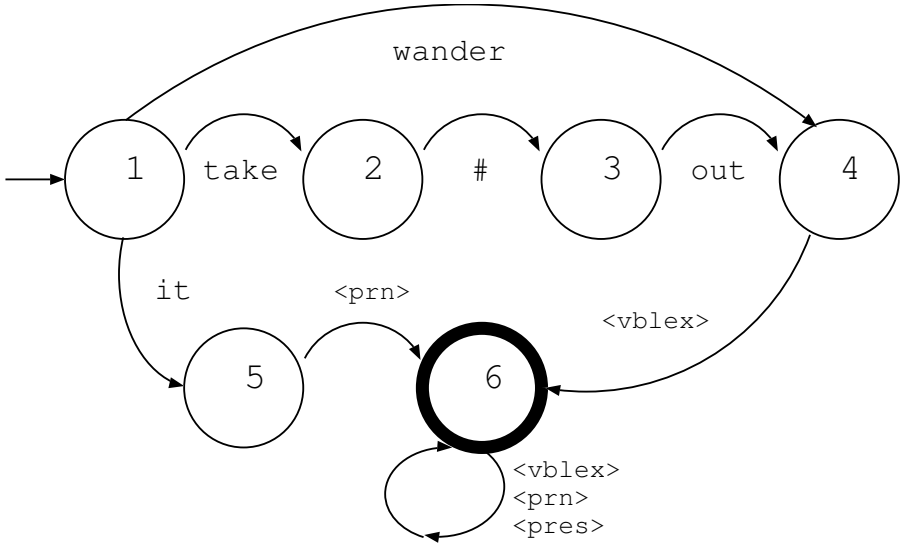
1. Preprocess bilingual FST
   1.1 "Prefixing": Append any-symbol loop
   1.2 Reorder #-multiwords
2. Depth-first intersection of output-side of analyser with input-side of bilingual FST
   - with an exception on seeing +

Introduction and background
FST's in the Apertium pipeline
The Problem: Redundant data
A Solution: trim on compile
Implementation of `lt-trim`
Preprocessing the bilingual dictionary
Intersection
`lt-trim` in use
Ending Dictionary Redundancy
Conclusion

Prefixing bilingual FST



Input bilingual FST (letter transitions compressed to single arcs)



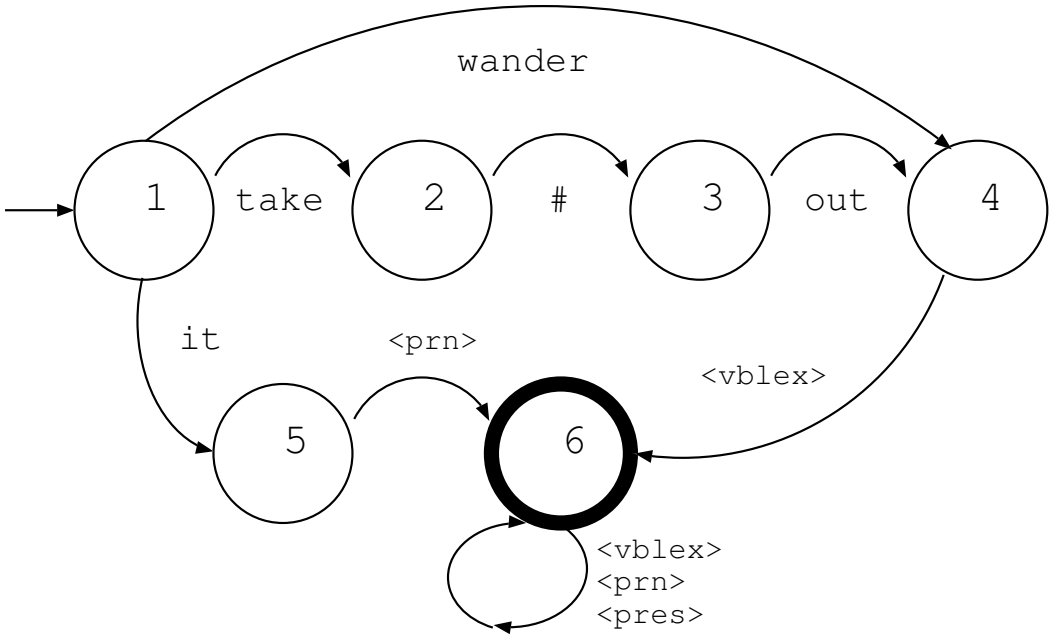
“Prefixed” bilingual FST (any-symbol loop appended)

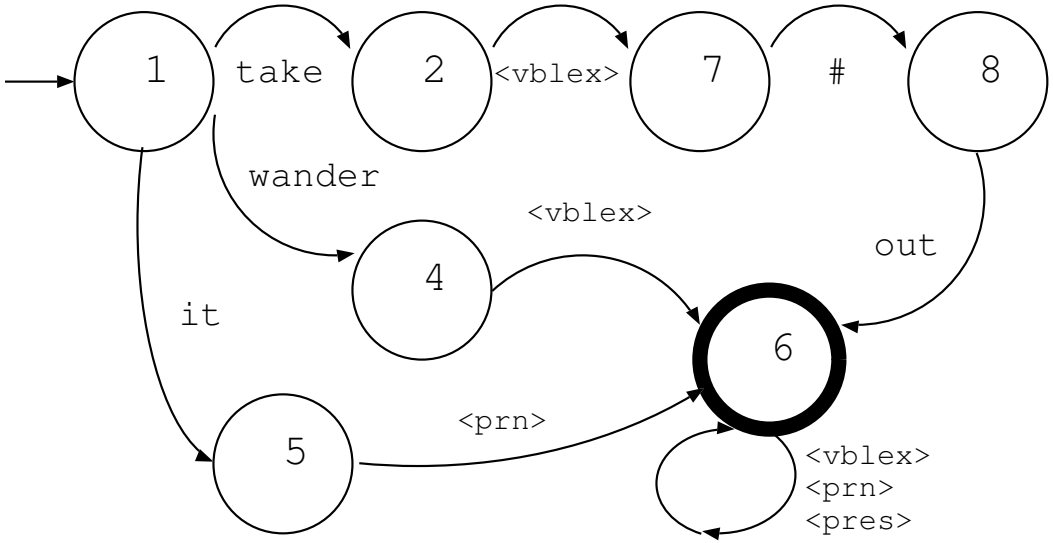
Moving uninflected lemma parts in bilingual FST

Want `take# out<vblex>` to become `take<vblex># out`, so

1. Depth-first traverse bilingual FST
2. On seeing a #, replace the transition *t* with results of $\mathrm{copyWithTagsFirst(t)}$
3. Function $\mathrm{copyWithTagsFirst(t)}$ builds a new partial FST where any tag sequence and uninflected lemma parts have swapped places

Moving uninflected lemma parts in bilingual FST

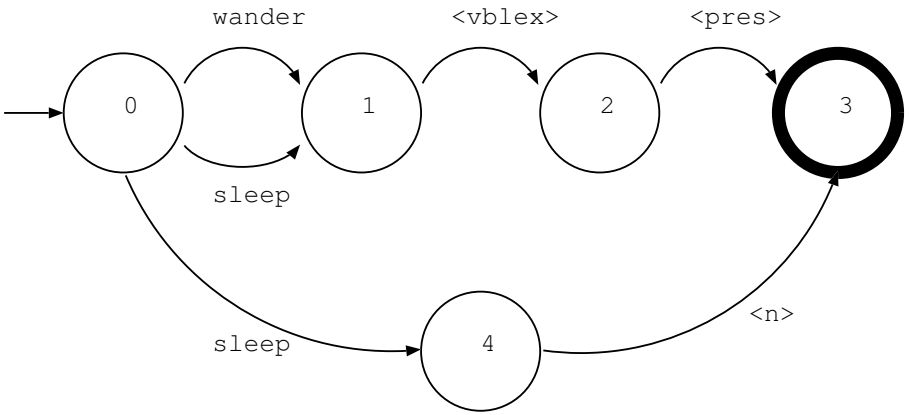




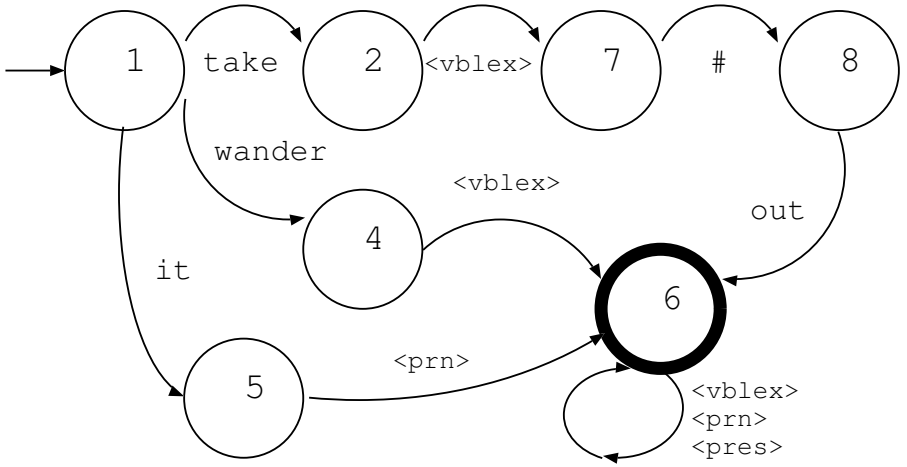
Francis Tyers,
Matthew Marting,
Kevin Unhammer

Introduction and background
FST's in the Apertium pipeline
The Problem: Redundant data
A Solution: trim on compile
Implementation of lt-trim
Preprocessing the bilingual dictionary
Intersection
lt-trim in use
Ending Dictionary Redundancy
Conclusion

Intersection



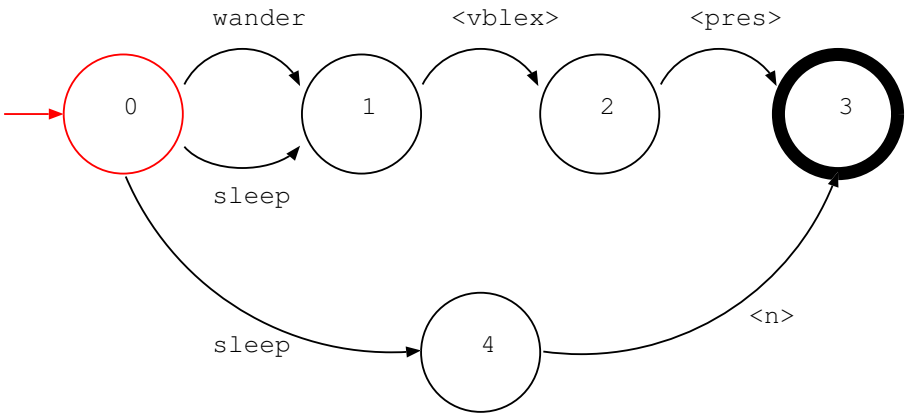
Input analyser

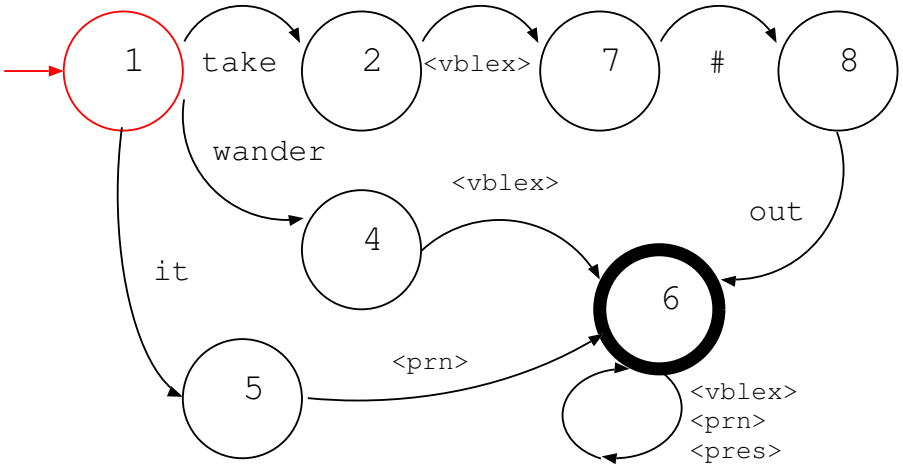


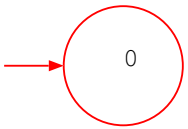
Preprocessed bilingual FST

Francis Tyers,
Matthew Marting,
Kevin Unhammer

Introduction and background
FST's in the Apertium pipeline
The Problem: Redundant data
A Solution: trim on compile
Implementation of lt-trim
Preprocessing the bilingual dictionary
Intersection
lt-trim in use
Ending Dictionary Redundancy
Conclusion

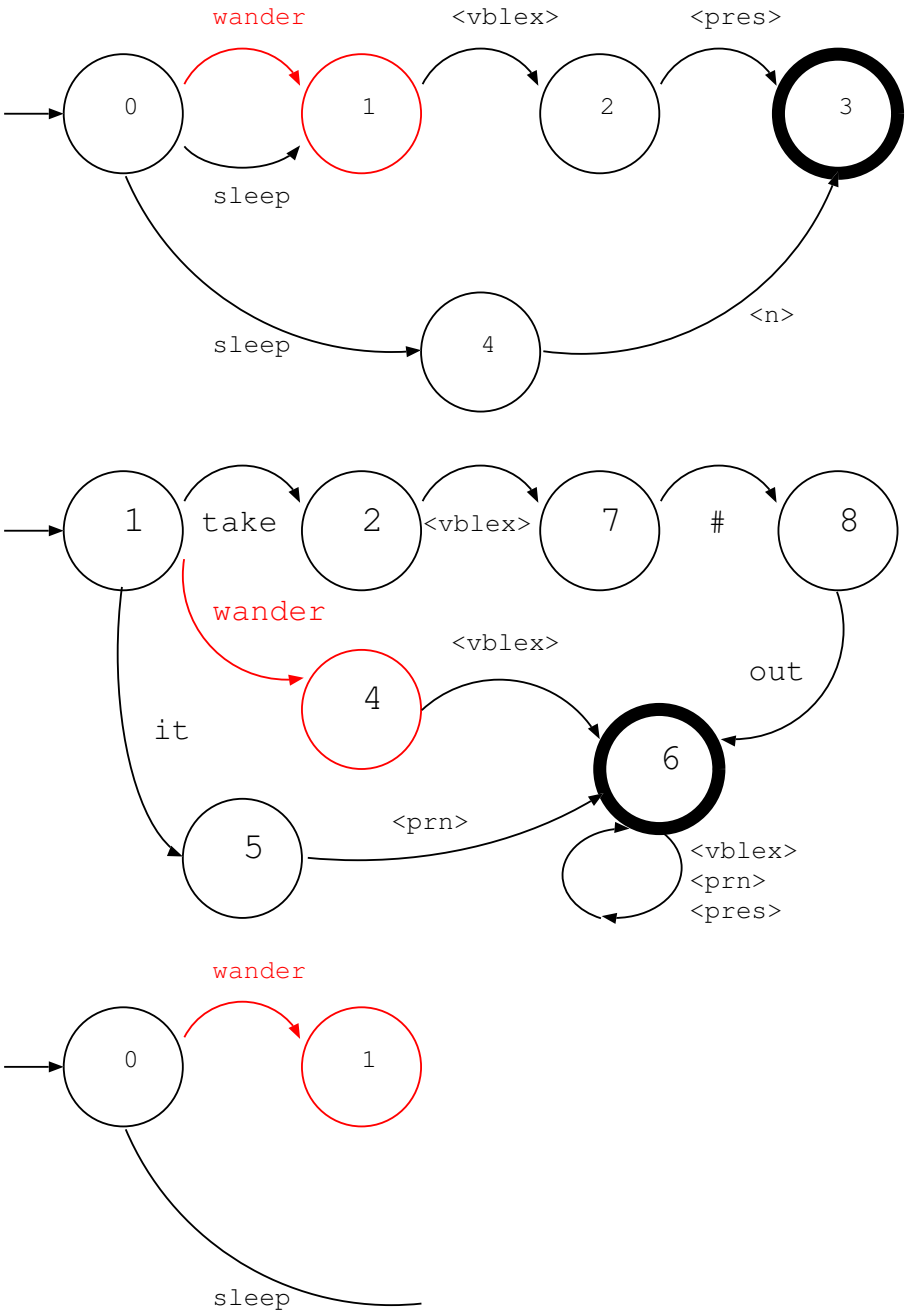


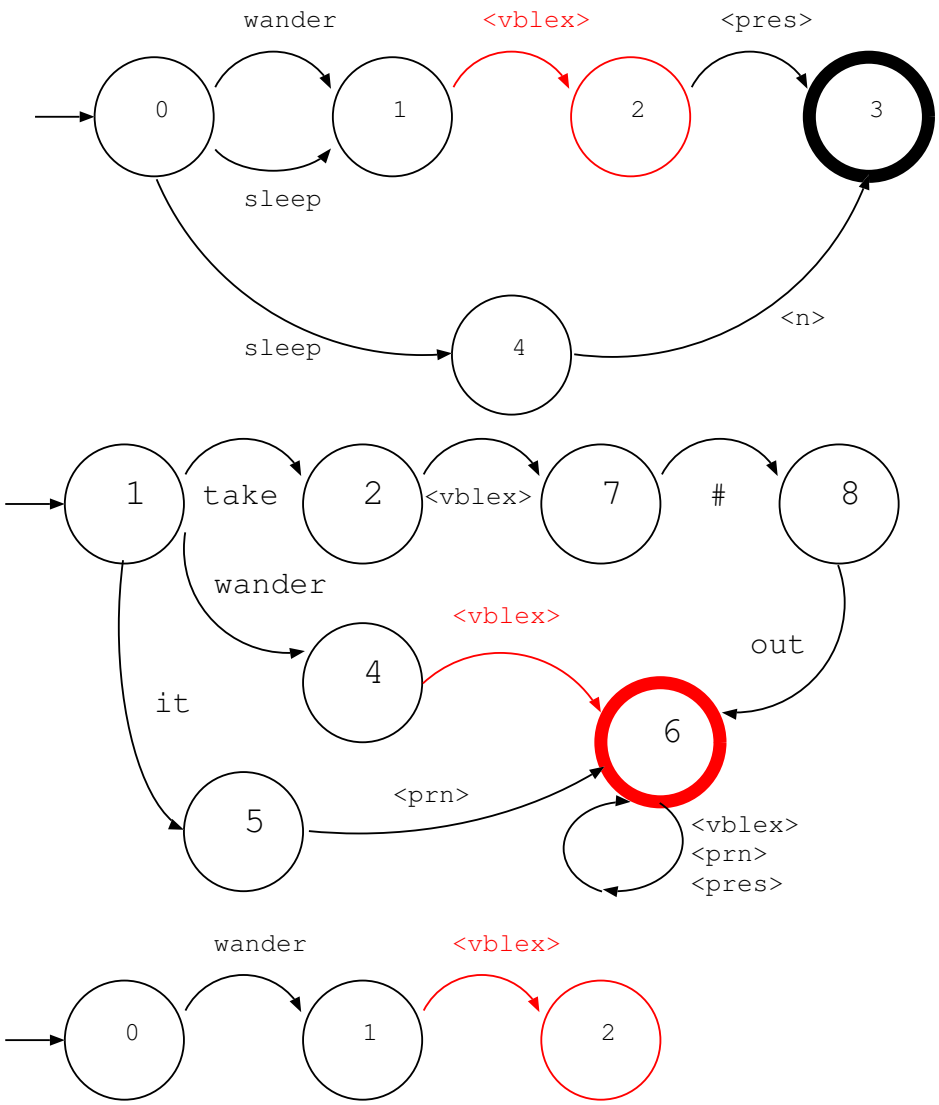


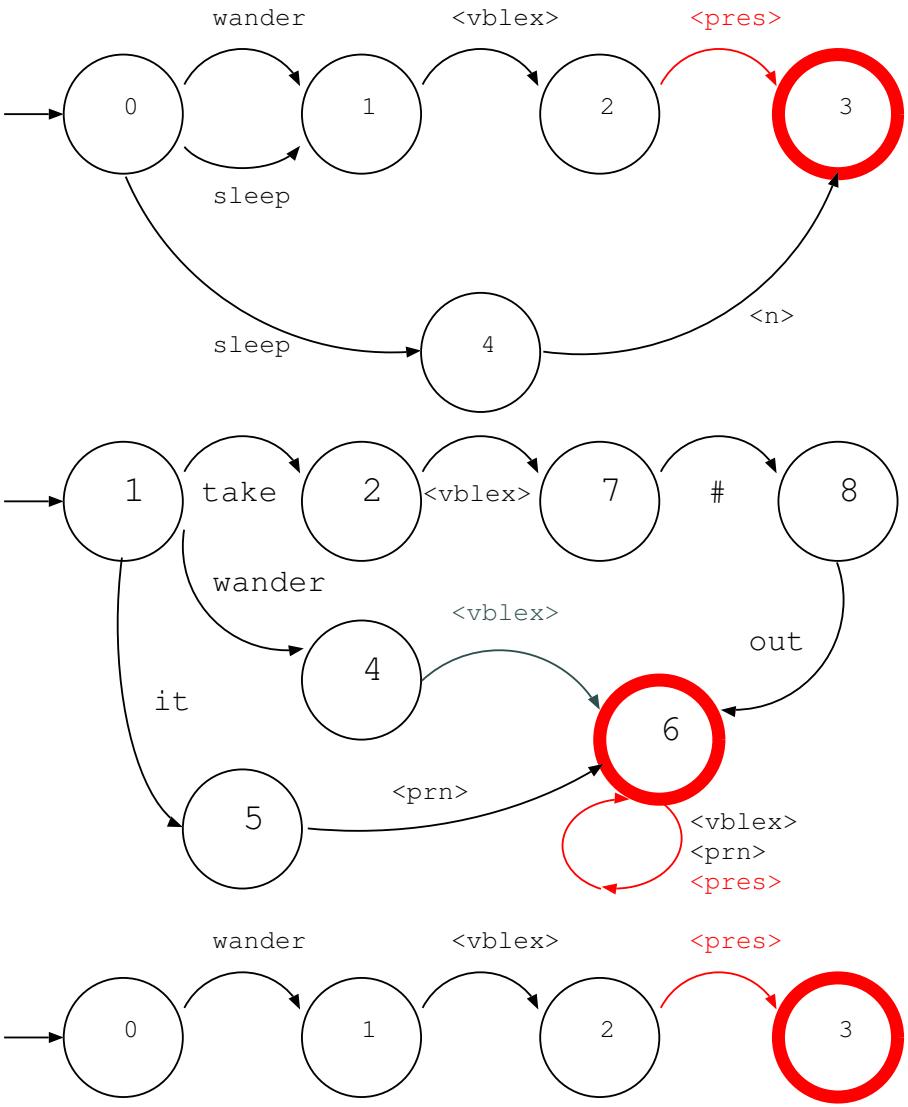


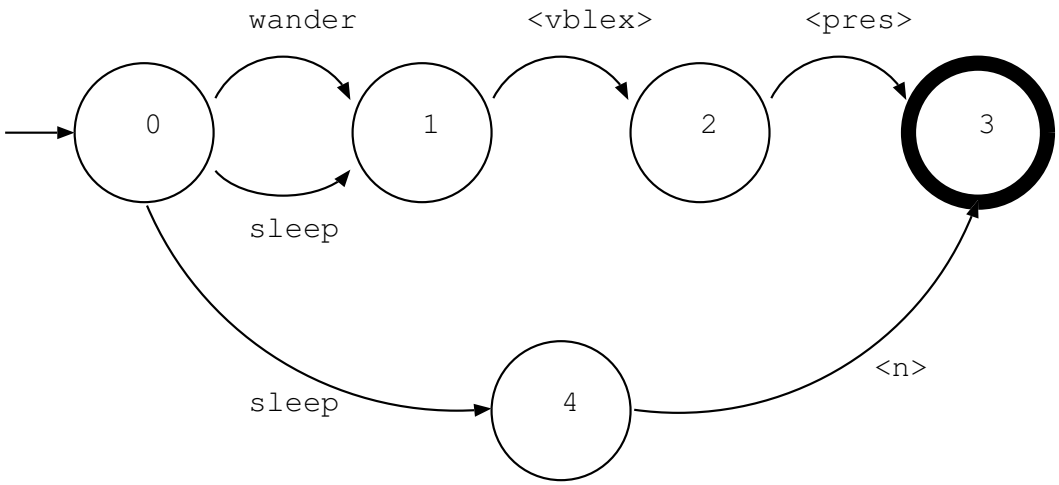
Francis Tyers,
Matthew Marting,
Kevin Unhammer

Introduction and background
FST's in the Apertium pipeline
The Problem: Redundant data
A Solution: trim on compile
Implementation of lt-trim
Preprocessing the bilingual dictionary
Intersection
lt-trim in use
Ending Dictionary Redundancy
Conclusion

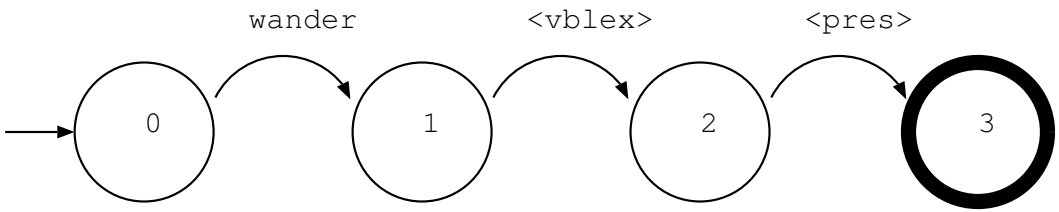

wander
<vblex>
<pres>
0
1
2
3
sleep
sleep
4
<n>
1
take
2
<vblex>
7
#
8
wander
<vblex>
4
out
it
6
5
<prn>
<vblex>
<prn>
<pres>
wander
0
1
sleep


wander
<vblex>
<pres>
0
1
2
3
sleep
sleep
4
<n>
1
take
2
<vblex>
7
#
8
wander
<vblex>
out
4
it
6
5
<prn>
<vblex>
<prn>
<pres>
wander
<vblex>
0
1
2


wander
<vblex>
<pres>
0
1
2
3
sleep
sleep
4
<n>
1
take
2
<vblex>
7
#
8
wander
<vblex>
4
out
it
6
5
<prn>
<vblex>
<prn>
<pres>
wander
<vblex>
<pres>
0
1
2
3

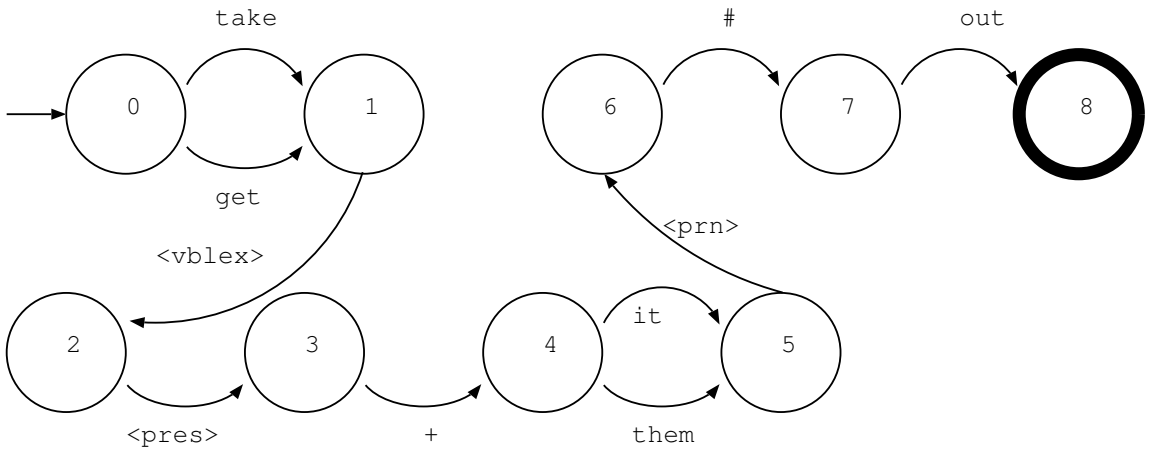


Input analyser

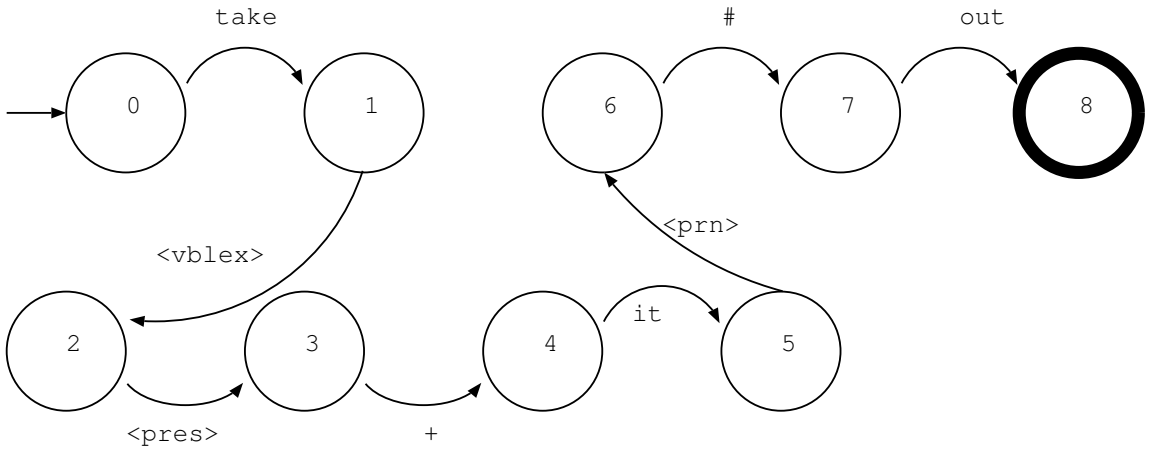


Fully trimmed analyser

Intersection with multiwords



Input analyser



Trimmed analyser

Francis Tyers,
Matthew Marting,
Kevin Unhammer

Introduction and background
FST's in the Apertium pipeline
The Problem: Redundant data
A Solution: trim on compile
Implementation of lt-trim
Preprocessing the bilingual dictionary
Intersection
lt-trim in use
Ending Dictionary Redundancy
Conclusion

`lt-trim` in use

```
$ lt-trim full-ana.bin bi.bin trimmed-ana.bin
```

(But language pair developers typically just type “make”.)

Speed and memory usage is comparable to regular lttoolbox (`lt-comp`) compiling.

Introduction and background
FST's in the Apertium pipeline
The Problem: Redundant data
A Solution: trim on compile
Implementation of `lt-trim`
Preprocessing the bilingual dictionary
Intersection
`lt-trim` in use
Ending Dictionary Redundancy
Conclusion

Ending Dictionary Redundancy

- New Autotools rules let us formally depend on monolingual data packages
- All new languages added to Apertium use this system – no monolingual data redundancy
- But: Implementing trimming in old/well-developed language pairs means manual merging – divergent dictionaries problematic
  - Merging Norwegian Bokmål between sme-nob and nno-nob: about 3 hrs work
  - dan-nob added further 3 hrs

Introduction and background
FST's in the Apertium pipeline
The Problem: Redundant data
A Solution: trim on compile
Implementation of `lt-trim`
Preprocessing the bilingual dictionary
Intersection
`lt-trim` in use
Ending Dictionary Redundancy
Conclusion

Conclusion

- `lt-trim` works
- Monolingual data now in a single /languages/ SVN module, easier for other projects to find and use our data
- Next up: special-purpose HFST trimming tool to get around compounding problem?
- Still much manual merge work to be done

Introduction and background
FST's in the Apertium pipeline
The Problem: Redundant data
A Solution: trim on compile
Implementation of `lt-trim`
Preprocessing the bilingual dictionary
Intersection
`lt-trim` in use
Ending Dictionary Redundancy
Conclusion