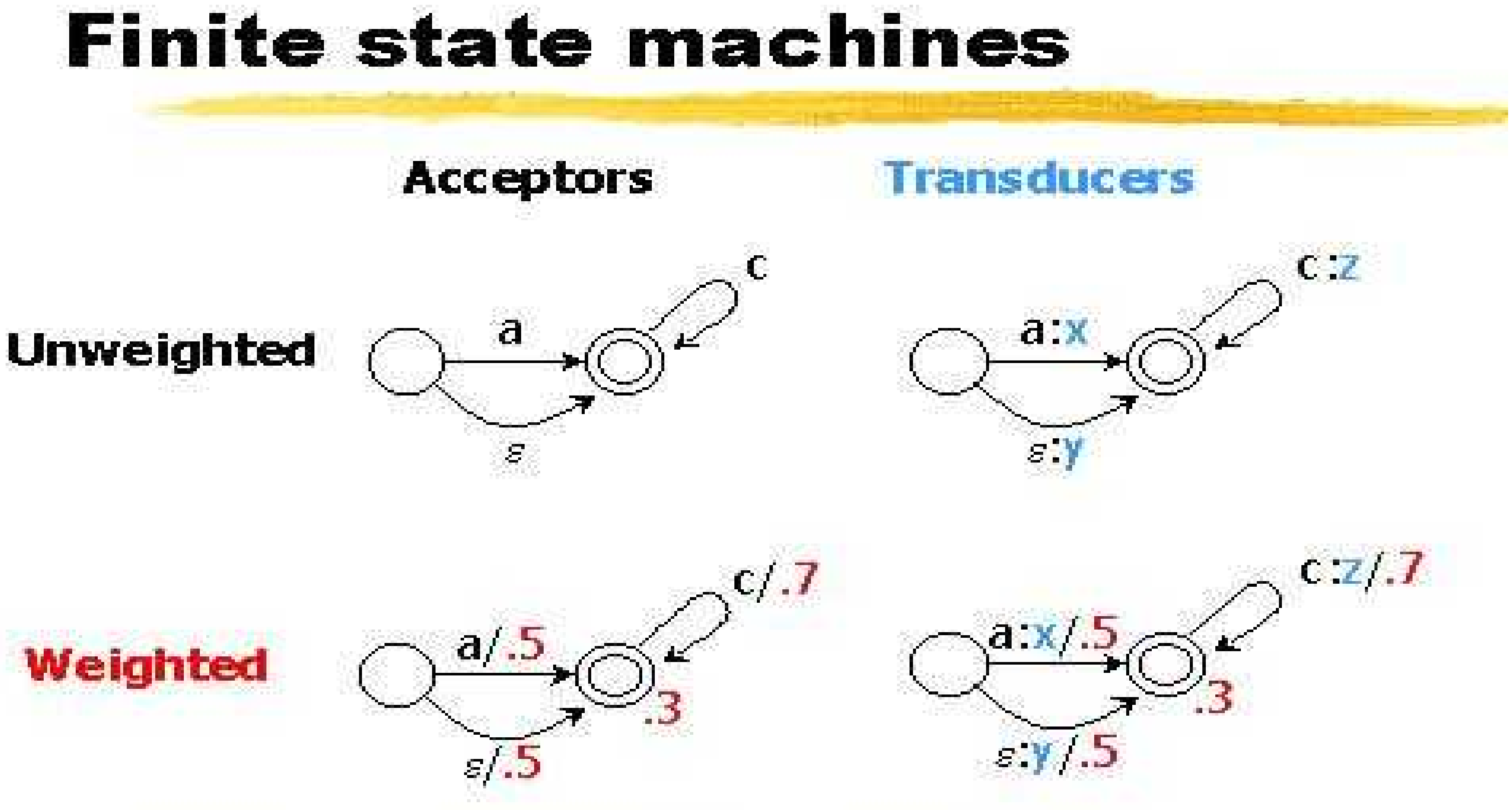
Finite-State Transducers: Applications in Natural Language Processing

Heli Uibo
Institute of Computer Science
University of Tartu
Heli.Uibo@ut.ee

Outline

- FSA and FST: operations, properties
- Natural languages vs. Chomsky's hierarchy
- FST-s: application areas in NLP
- Finite-state computational morphology
- Author's contribution: Estonian finite-state morphology
- Different morphology-based applications
- Conclusion

FSA-s and FST-s



Operations on FSTs

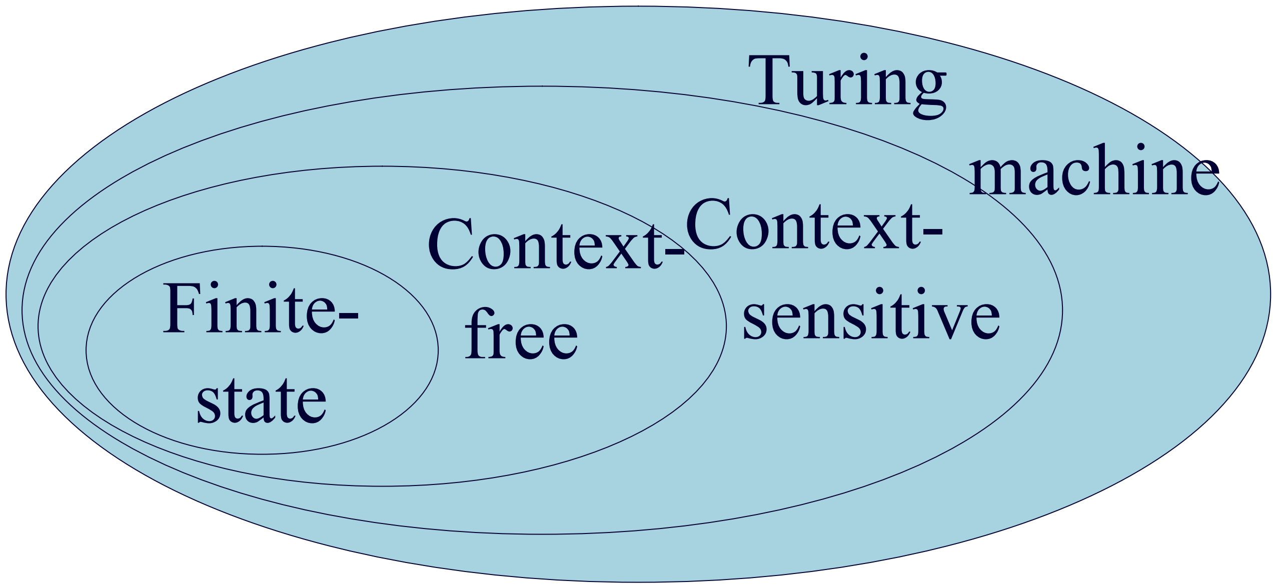
- concatenation
- union
- iteration (Kleene's star and plus)
- *complementation
- composition
- reverse, inverse
- *subtraction
- *intersection
- containment
- substitution
- cross-product
- projection

Algorithmic properties of FSTs

- epsilon-free
- deterministic
- minimized

Natural languages vs. Chomsky's hierarchy

- "*English is not a finite state language.*" (Chomsky "Syntactic structures" 1957)
- Chomsky's hierarchy:



Natural languages vs. Chomsky's hierarchy

- The Chomsky's claim was about syntax (sentence structure).
- Proved by (theoretically unbounded) recursive processes in syntax:
  - embedded subclauses

  I saw a dog, who chased a cat, who ate a rat, who …
  - adding of free adjuncts

  S $\rightarrow$ NP (AdvP)* VP (AdvP)*

Natural languages vs. Chomsky's hierarchy

→ Attempts to use more powerful formalisms

- Syntax: phrase structure grammars (PSG) and unification grammars (HPSG, LFG)
- Morphology: context-sensitive rewrite rules (not-reversible)

Natural languages vs. Chomsky's hierarchy

- Generative phonology by Chomsky&Halle (1968) used context-sensitive rewrite rules , applied in the certain order to convert the abstract phonological representation to the surface representation (wordform) through the intermediate representations.
- General form of rules: $x \rightarrow y \,/\, z \_ w$,

where x, y, z, w – arbitrary complex feature structures

Natural languages vs. Chomsky's hierarchy

- BUT: Writing large scale, practically usable context-sensitive grammars even for well-studied languages such as English turned out to be a very hard task.
- Finite-state devices have been "rediscovered" and widely used in language technology during last two decades.

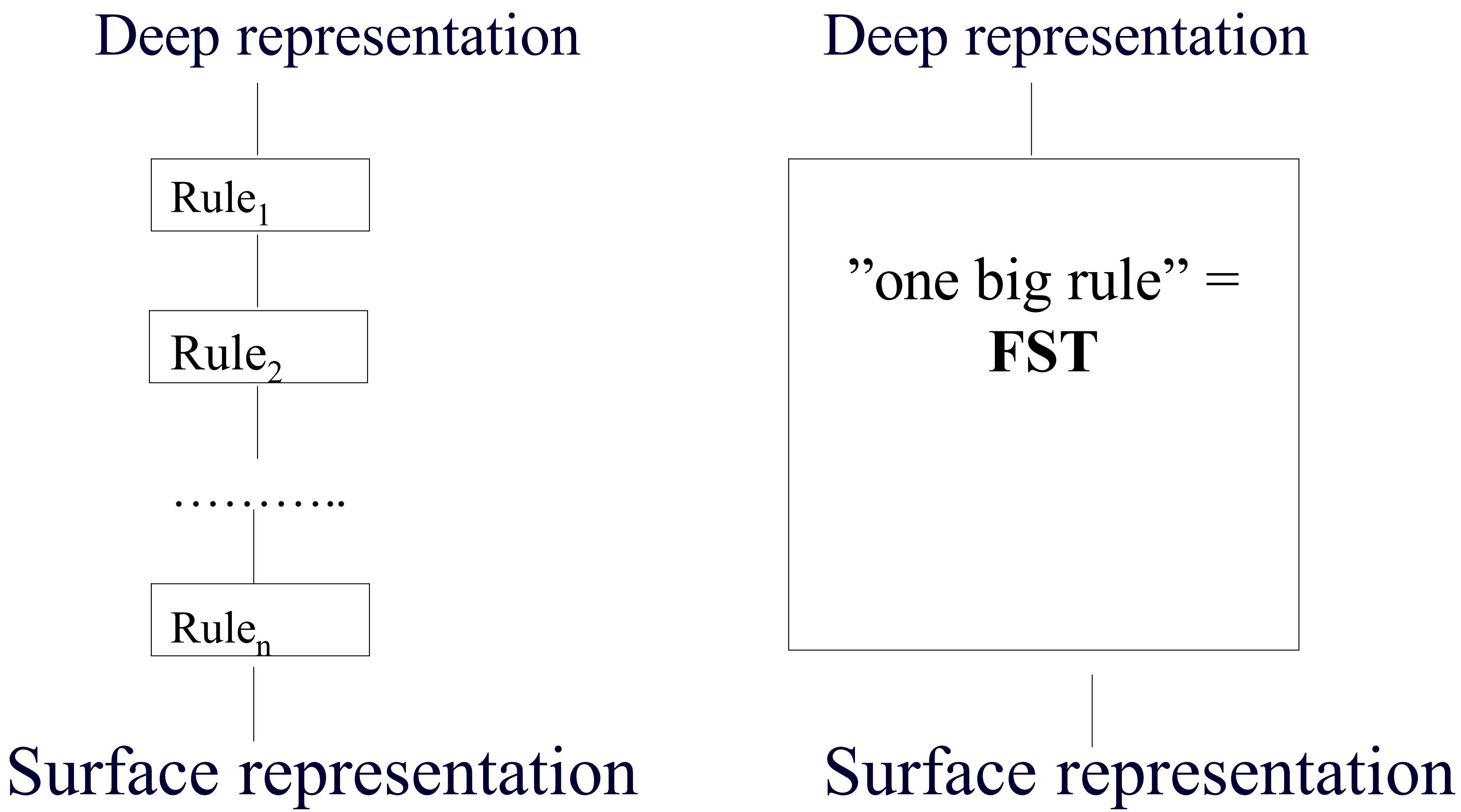
Natural languages vs. Chomsky's hierarchy

- Finite-state methods have been especially successful for describing morphology.
- The usability of FSA-s and FST-s in computational morphology relies on the following results:
- D. Johnson, 1972: Phonological rewrite rules are not context-sensitive in nature, but they can be represent as FST-s.
- Schützenberger, 1961: If we apply two FST-s sequentially, there exist a single FST, which is the composition of the two FST-s.

Natural languages vs. Chomsky's hierarchy

- Generalization to n FST-s: we manage without intermediate representations – deep representation is converted to surface representation by a single FST!
- 1980 – the result was rediscovered by R. Kaplan and M. Kay (Xerox PARC)

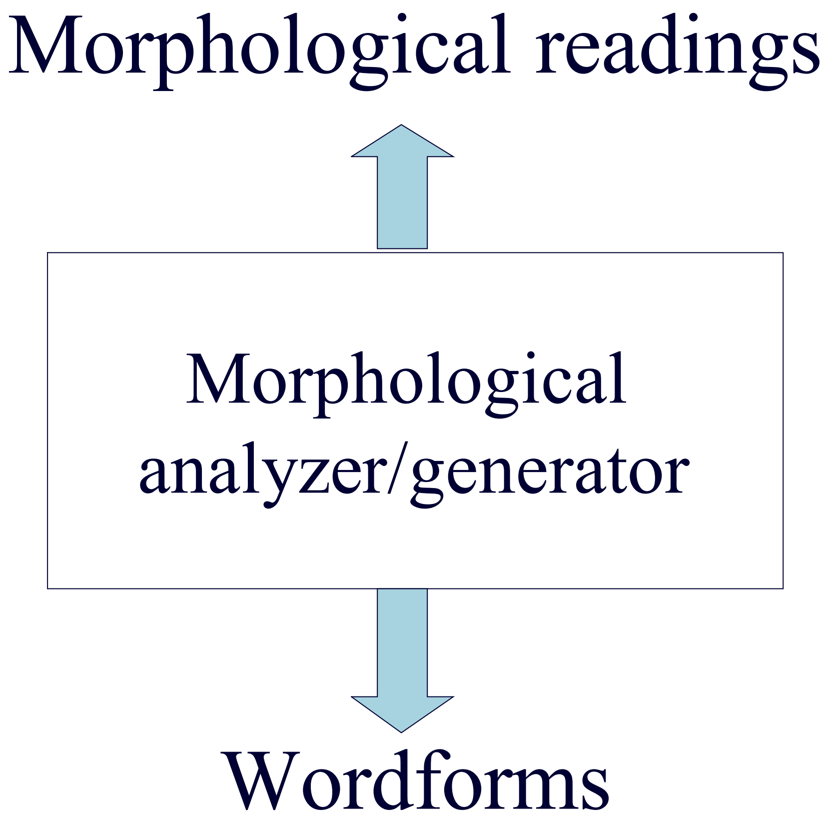
Natural languages vs. Chomsky's hierarchy



Applications of FSA-s and FST-s in NLP

- Lexicon (word list) as FSA – compression of data!
- Bilingual dictionary as lexical transducer
- Morphological transducer (may be combined with rule-transducer(s), e.g. Koskenniemi's two-level rules or Karttunen's replace rules – composition of transducers).
  - Each path from the initial state to a final state represents a mapping between a surface form and its lemma (lexical form).

Finite-state computational morphology



Morfological analysis by lexical transducer

Morphological analysis = *lookup*

- The paths in the lexical transducers are traversed, until one finds a path, where the concatenation of the lower labels of the arcs is equal to the given wordform.
- The output is the concatenation of the upper labels of the same path (lemma + grammatical information).
- If no path succeeds (transducer rejects the wordform), then the wordform does not belong to the language, described by the lexical transducer.

Morfological synthesis by lexical transducer

Morphological synthesis = *lookdown*

- The paths in the lexical transducers are traversed, until one finds a path, where the concatenation of the upper labels of the arcs is equal to the given lemma + grammatical information.
- The output is the concatenation of the lower labels of the same path (a wordform).
- If no path succeeds (transducer rejects the given lemma + grammatical information), then either the lexicon does not contain the lemma or the grammatical information is not correct.

Finite-state computational morphology

In morphology, one usually has to model two principally different processes:

1. Morphotactics (how to combine wordforms from morphemes)

   - prefixation and suffixation, compounding = concatenation

   - reduplication, infixation, interdigitation – non-concatenative processes

Finite-state computational morphology

2. Phonological/orthographical alternations

- assimilation (hind : hinna)
- insertion (jooksma : jooksev)
- deletion (number : numbri)
- gemination (tuba : tuppa)

All the listed morphological phenomena can be described by regular expressions.

Estonian finite-state morphology

In Estonian language different grammatical wordforms are built using

- stem flexion

tuba - singular nominative (*room*)

toa - singular genitive (*of the room*)

- suffixes (e.g. plural features and case endings)

tubadest - plural elative (*from the rooms*)

Estonian finite-state morphology

- productive derivation, using suffixes

kiire (*quick*) → kiiresti (*quickly*)

- compounding, using concatenation

piiri + valve + väe + osa = piirivalveväeosa

border(Gen) + *guarding*(Gen) + *force*(Gen) + *part* =

a troup of border guards

Estonian finite-state morphology

- Two-level model by K. Koskenniemi
- LexiconFST .o. RuleFST
- Three types of two-level rules: <=>, <=, => (formally regular expressions)
- e.g. two-level rule a:b => L _ R is equivalent to regular expression

  [~[[[?* L] a:b ?*] | [?* a:b ~[R ?*]]]
- Linguists are used to rules of type

  a → b || L _ R

Estonian finite-state morphology

- Phenomena handled by lexicons:
  - noun declination
  - verb conjugation
  - comparison of adjectives
  - derivation
  - compounding
  - stem end alternations ne-se, 0-da, 0-me etc.
  - choice of stem end vowel a, e, i, u

(noun declination, verb conjugation, comparison of adjectives, derivation:) Appropriate suffixes are added to a stem according to its inflection type

Estonian finite-state morphology

- Handled by rules:
  - stem flexion

  kägu : käo, hüpata : hüppan

  - phonotactics

  lumi : lumd* → lund

  - morphophonological distribution

  seis + da → seista

  - orthography

  kirj* → kiri, kristall + ne → kristalne

Estonian finite-state morphology

Problem with derivation from verbs with weakening stems: every stem occurs twice at the upper side of the lexicon

→ vaste of space!

LEXICON Verb
 lõika:lõiKa V2;
 ………..
LEXICON Verb-Deriv
 lõiga VD0;
 ………..
LEXICON VD0
 tud+A:tud #;
 tu+S:tu S1;
 nud+A:nud #;
 nu+S:nu S1;

Estonian finite-state morphology

- My own scientific contribution☺:

Solution to the problem of weak-grade verb derivatives: also primary form, belonging to the level of morphological information, has lexical (or deep) representation.

That is, two-levelness has been extended to the upper side of the lexical transducer (only for verbs).

LEXICON Verb

lõiKa:lõiKa V2;

....

No stem doubling for productively derived forms!

Estonian finite-state morphology

Result: The morphological transducer for Estonian is composed as follows:

$$((LexiconFST)^{-1} \quad RulesFST_1)^{-1} \quad RulesFST,$$

where $RulesFST_1$ RulesFST (subset of the whole rule set, containing grade alternation rules only)

Operations used: composition, inversion

Estonian finite-state morphology

- The experimental two-level morphology for Estonian has been implemented using the XEROX finite-state tools *lexc* and *twolc*.
- 45 two-level rules
- The root lexicons include ≈2000 word roots.
- Over 200 small lexicons describe the stem end alternations, conjugation, declination, derivation and compounding.

Estonian finite-state morphology

To-do list:

- avoid overgeneration of compound words

solution: compose the transducer with other transducers which constrain the generation process

- guess the analysis of unknown words (words not in the lexicon)

solution: use regexp in the lexicon which stand for any root, e.g. [Alpha*]

Language technological applications: requirements

- Different approaches of building the morphological transducer may be suitable for different language technological applications.
  - Speller – is the given wordform correct? (= accepted by the morphological transducer)

  Important to avoid overgeneration!
  - Improved information retrieval – find all the documents where the given keyword occurs in arbitrary form and sort the documents by relevance

  Weighted FST-s may be useful; morphological disambiguation also recommended; overgeneration not so big problem.

Full NLP with FST-s?

Description of a natural language = one big transducer