Finite-State Methods in Natural-Language Processing: 1—Motivation

Ronald M. Kaplan

and

Martin Kay

Finite-State Methods in Language Processing

The Application of a branch of mathematics

- **The regular branch of automata theory**

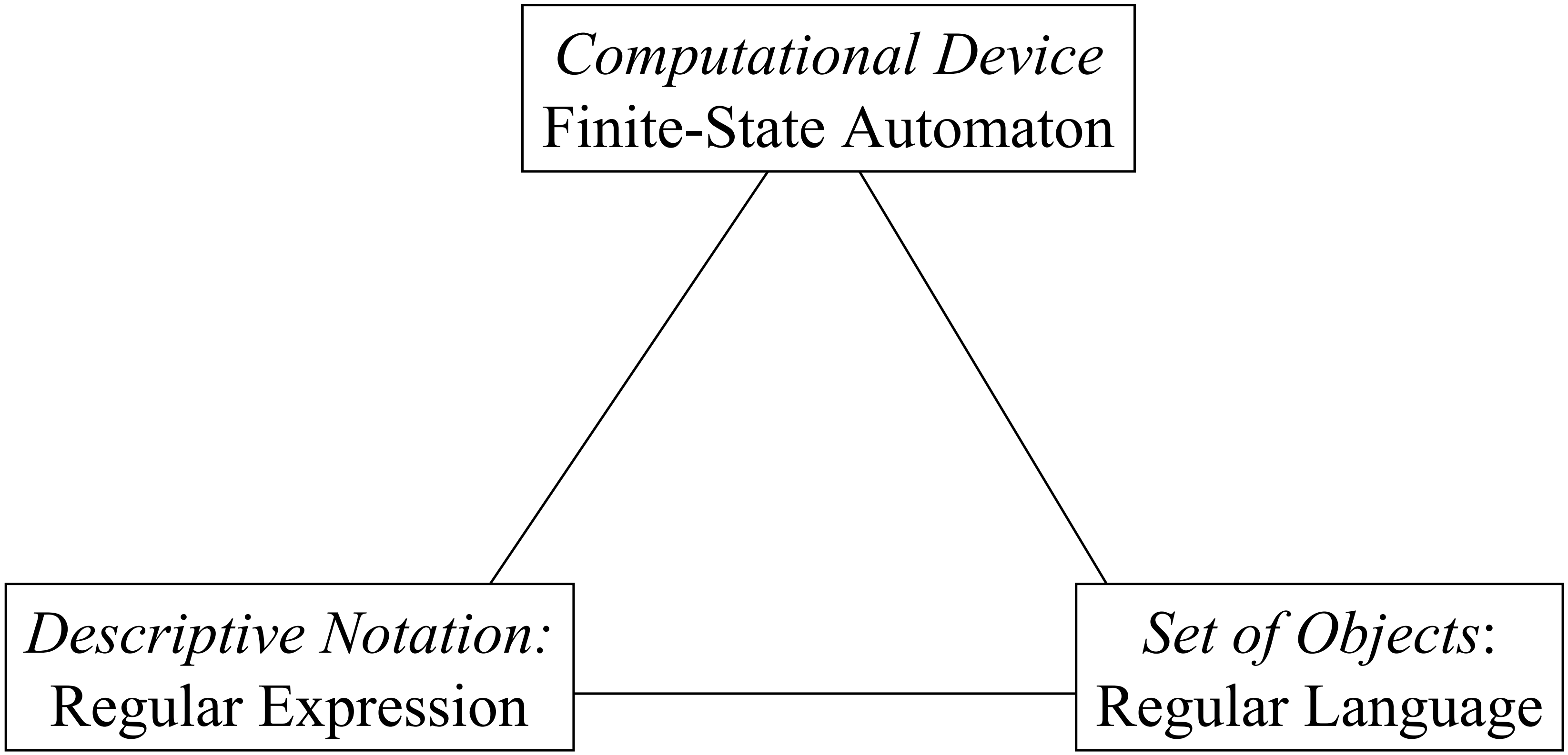
to a branch of computational linguistics in which what is crucial is (or can be reduced to)

- **Properties of string sets and string relations with**
- **A notion of bounded dependency**

Applications

- **Finite Languges**
  - — Dictionaries
  - — Compression
- **Phenomena involving bounded dependency**
  - — Morpholgy
    - Spelling
    - Hyphenation
    - Tokenization
    - Morphological Analysis
  - — Phonology
- **Approximations to phenomena involving *mostly* bounded dependency**
  - — Syntax
- **Phenomena that can be translated into the realm of strings with bounded dependency**
  - — Syntax

Correspondences

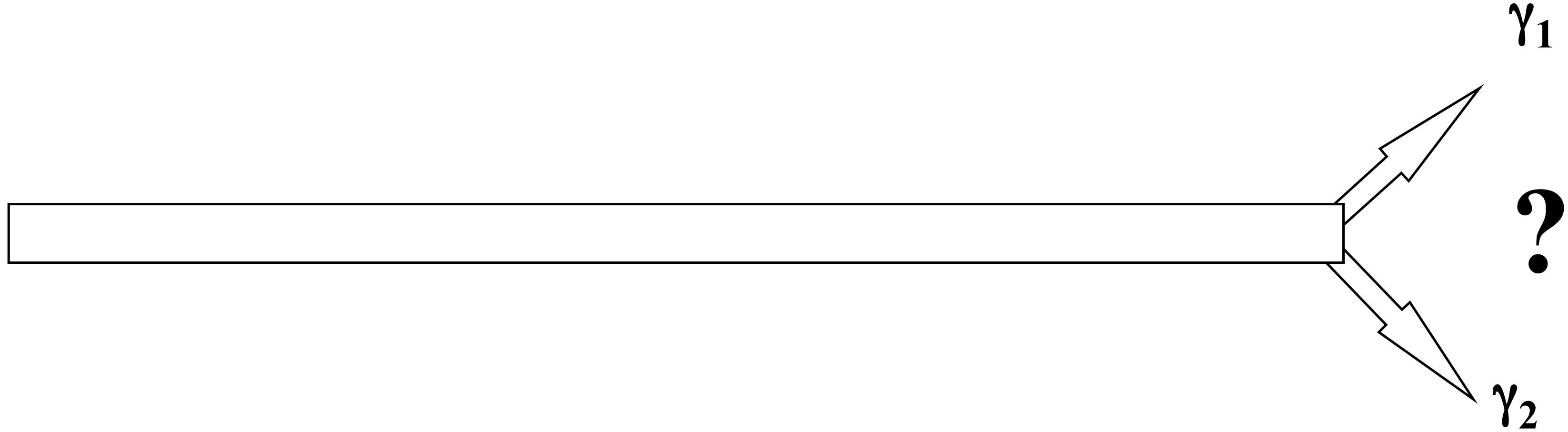


The Basic Idea

- **At any given moment,an automaton is in one of a finite number of states**
- **A transition from one state to another is possible when the automaton contains a corresponding *transition.***
- **The process can stop only when the automaton is in one of a subset of the states, called *final*.**
- **Transitions are labeled with symbols so that a sequence of transitions corresponds to a sequence of symbols.**

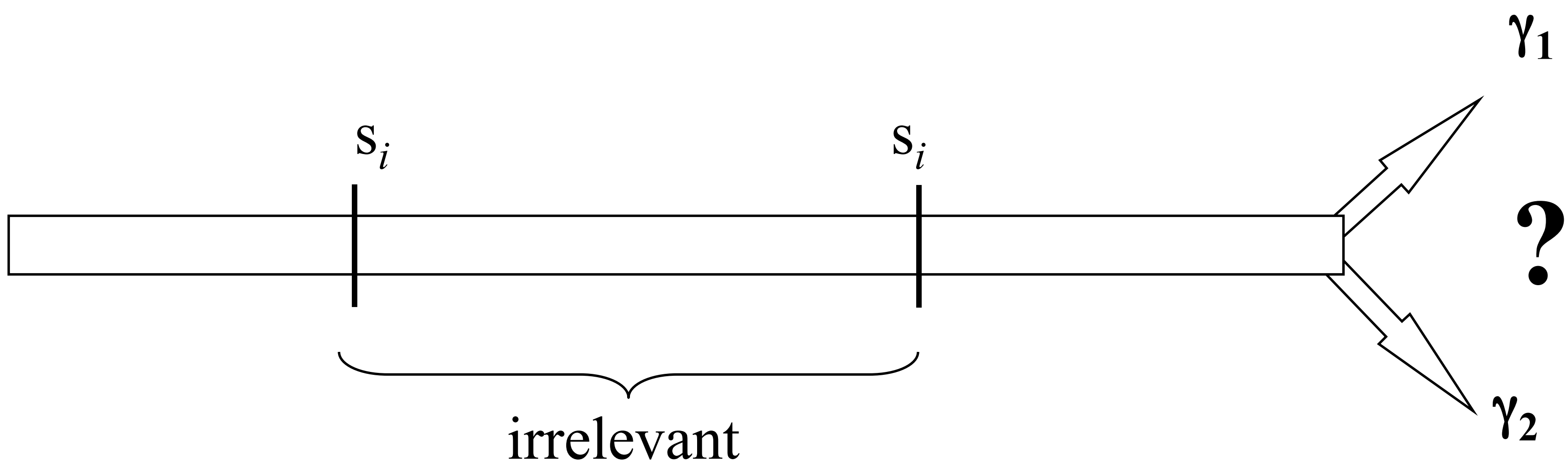
Bounded Dependency

The choice between γ_1 and γ_2 depends on a bounded number of preceding symbols.



Bounded Dependency

The choice between γ_1 and γ_2 depends on a bounded number of preceding symbols.



Closure Properties and Operations

- **By definition**
  - —Union
  - —Concatenation
  - —Iteration
- **By deduction**
  - —Intersection
  - —Complementation
  - —Substitution
  - —Reversal
  - — ...

Operations on Languages and Automata

For the set-theoretic operations on languages there are corresponding operations on automata.

$$M(L_1 \otimes L_2) = M(L_1) \oplus M(L_2)$$

***M*(*L*) is a machine that characterizes the language *L*.**

We will use the same symbols for corresponding operations

Automata-based Calculus

- **Closure gives:**
  - —Complementation → Universal quantification
  - —Intersection → Combinations of constraints
- **Machines give:**
  - —Finite representations for (potentially) infinite sets
  - —Practical implementation
- **Combination gives:**
  - —Coherence
  - —Robustness
  - —Reasonable machine transformations

Quantification

$\Sigma^* xy \Sigma^*$ There is an x followed by a y in the string

$\overline{\Sigma^* xy \Sigma^*}$ There is no xy sequence in the string

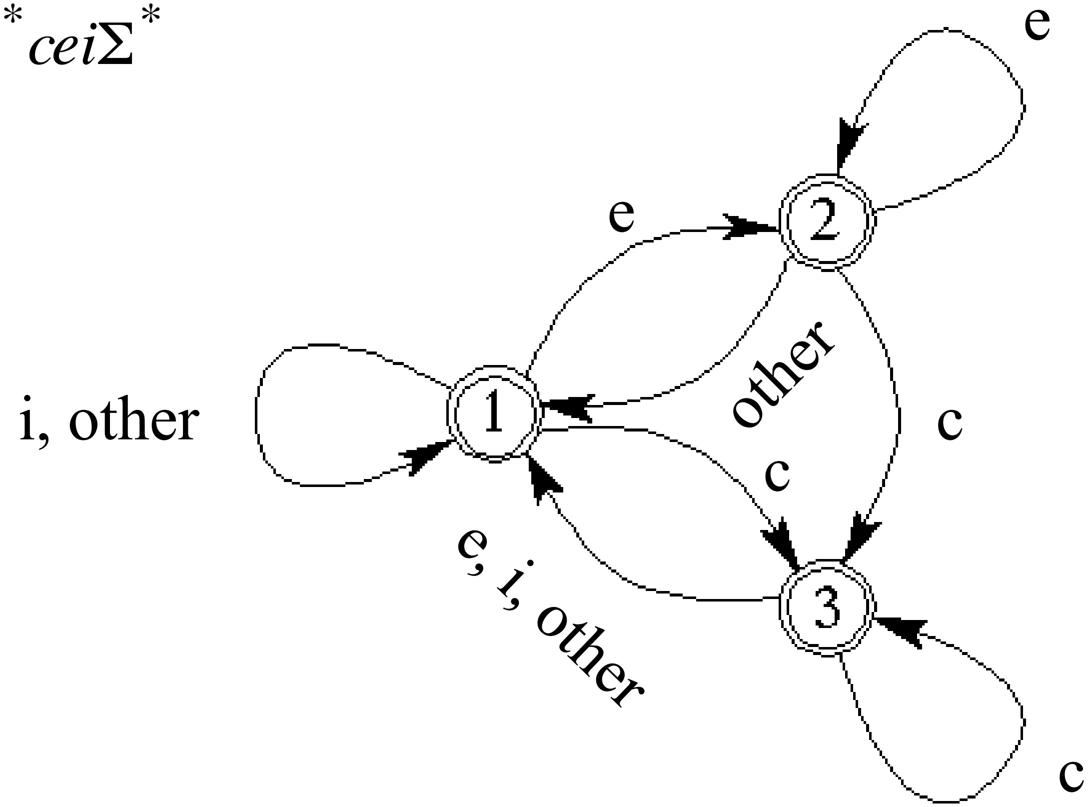
$\overline{\Sigma^*} xy \Sigma^*$ There is a y preceded by something that is not an x

$\overline{\overline{\Sigma^*} xy \Sigma^*}$ Every y is preceded by an x.

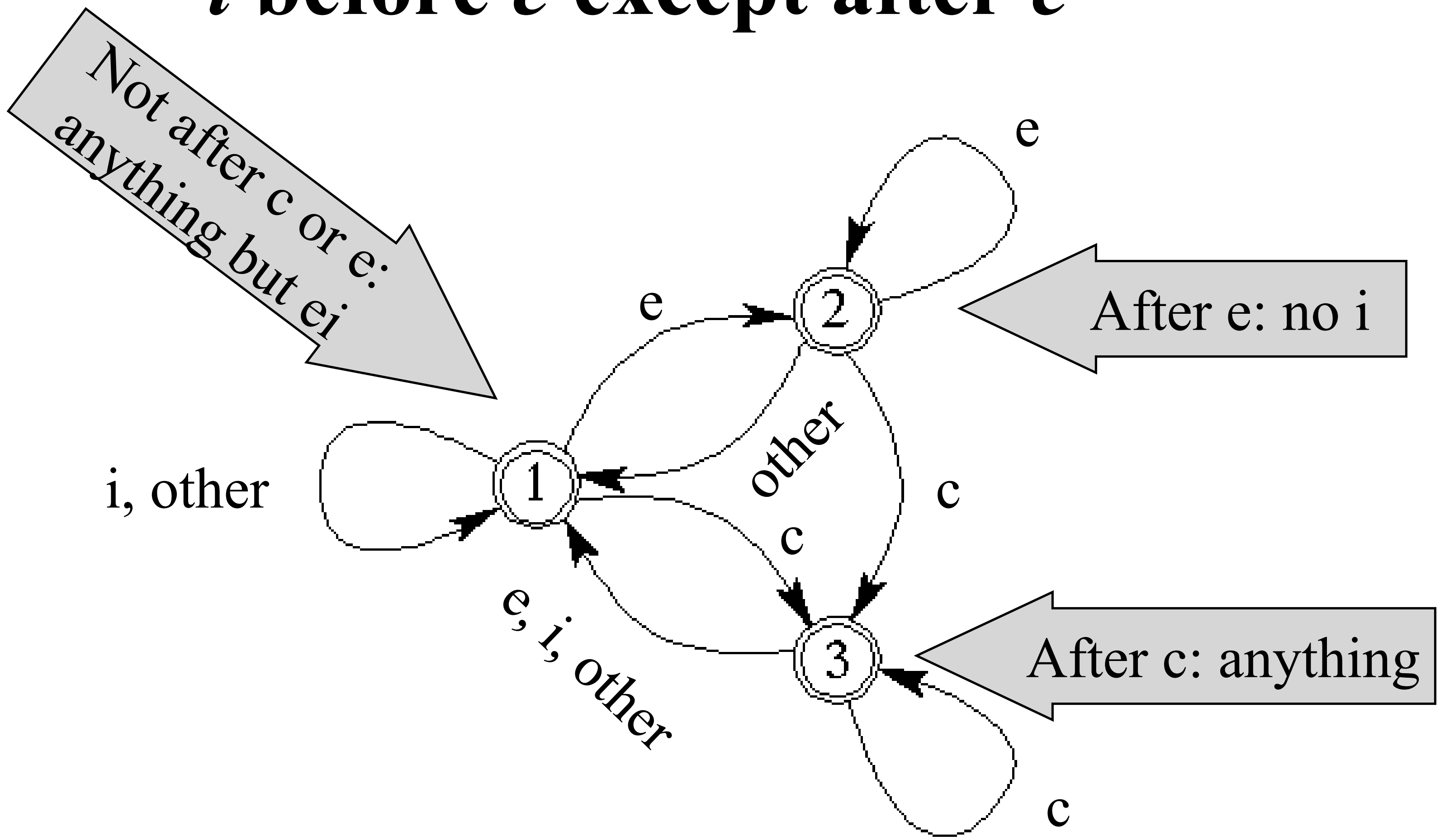
$$\overline{\exists y. \overline{\exists x. precedes(x, y)}}$$

Universal Quantification—
i before *e* except after *c*

$\overline{\overline{\Sigma^*}cei\Sigma^*}$

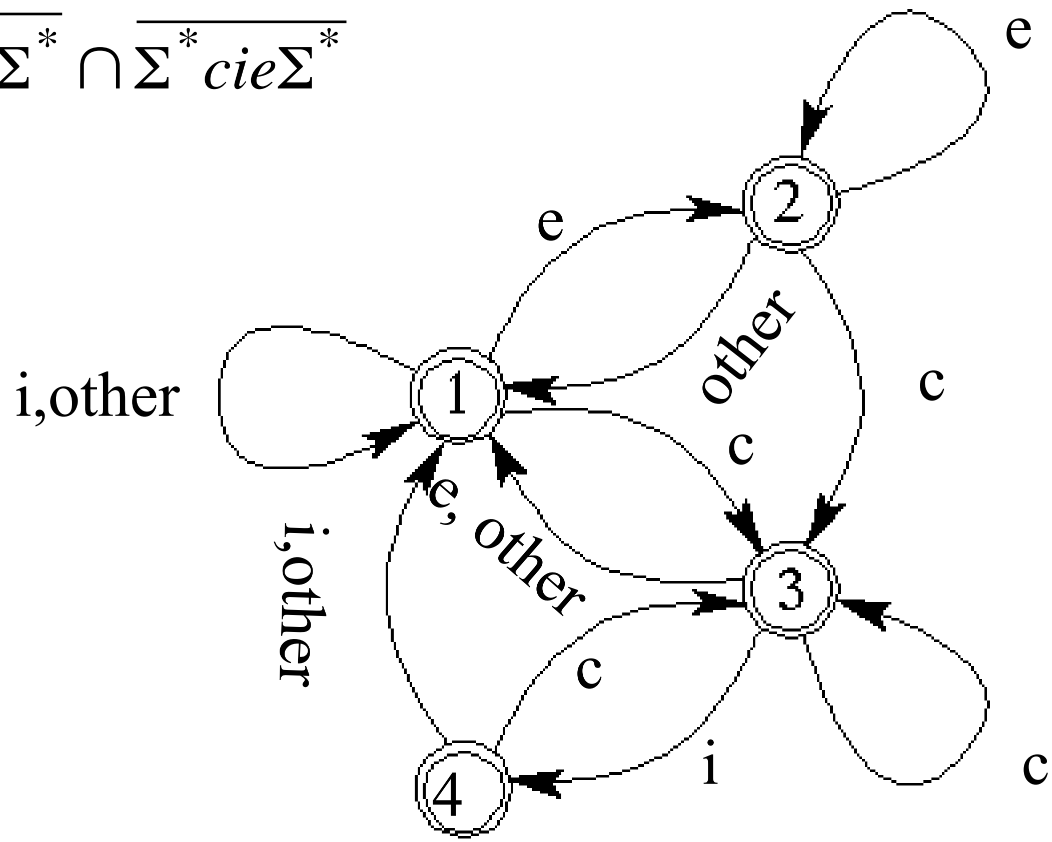


Universal Quantification—
i before *e* except after *c*

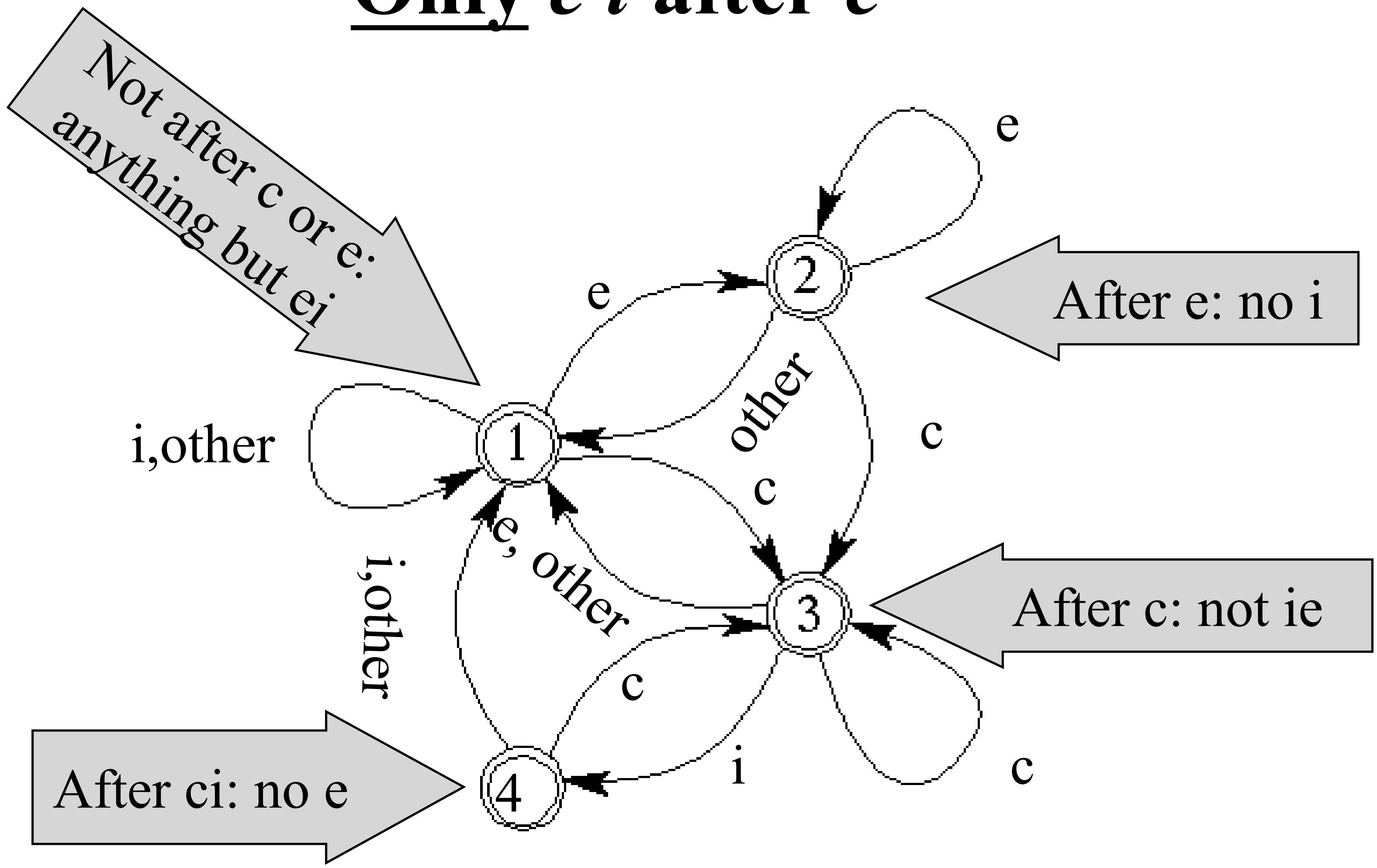


<u>Only</u> *e i* after *c*

$$\overline{\overline{\Sigma^*}cei\Sigma^*} \cap \overline{\Sigma^* cie\Sigma^*}$$



<u>Only</u> *e i* after *c*



Alternative Notations

Closure ⇒ Recursive Formalisms ⇒

Higher-level Constructs

$$L_1 \leftarrow L_2 \quad \equiv \quad \overline{\overline{L_1} L_2}$$

$$\overline{\overline{\Sigma^*} cei\Sigma^*} \quad \equiv \quad \Sigma^* c \leftarrow ei\Sigma^*$$

Choose notation for theoretical significance and practical convenience.

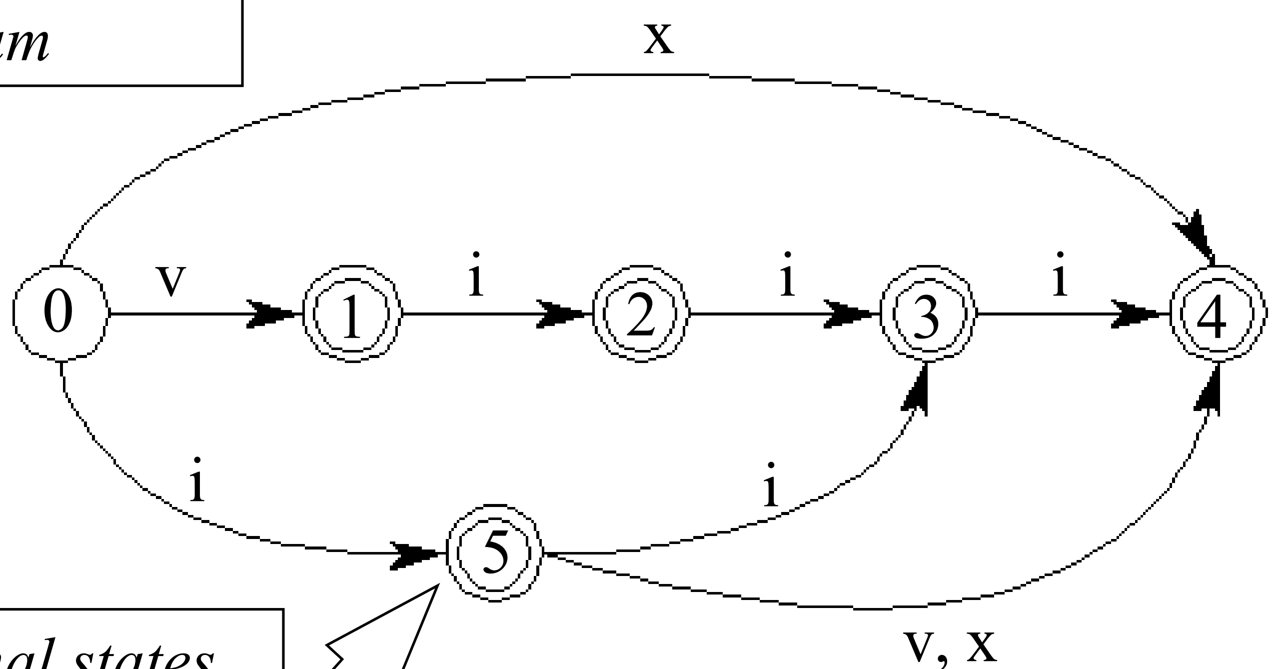
What is a Finite-State Automaton?

- **An alphabet of *symbols,***
- **A finite set of *states,***
- **A *transition function* from states and symbols to states,**
- **A distinguished member of the set of states called the *start state*, and**
- **A distinguished subset of the set of states called *final states*.**

***Pace* terminology, same definition as for directed graphs with labeled edges, plus initial and final states.**

i to x

Unless otherwise marked, the start state is usually the leftmost in the diagram



We draw final states with a double circle

Regular Languages

- Languages — sets of strings
- Regular languages — a subset of languages
- Closed under concatenation, union, and iteration
- Every regular language is chracterized by (at least) one finite-state automaton
- Languages may contain infinitely many strings but automata are finite

Regular Expressions

- **Formulae with operators that denote**
  - **union**
  - **concatenation**
  - **iteration**

```
a* [b | c]
```

Any number of *a*'s followed by either *b* or *c*.

Some Motivations

- **Word Recognition**
- **Dictionary Lookup**
- **Spelling Conventions**

Word Recognition

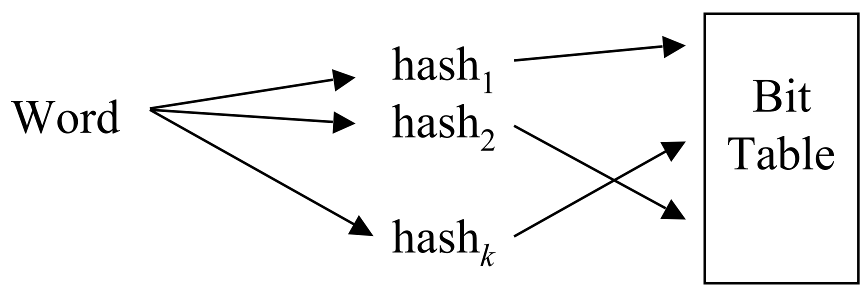
A word recognizer takes a string of characters as input and returns "yes" or "no" according as the word is or is not in a given set.

Solves the *membership* problem.

e.g. Spell Checking, Scrabble

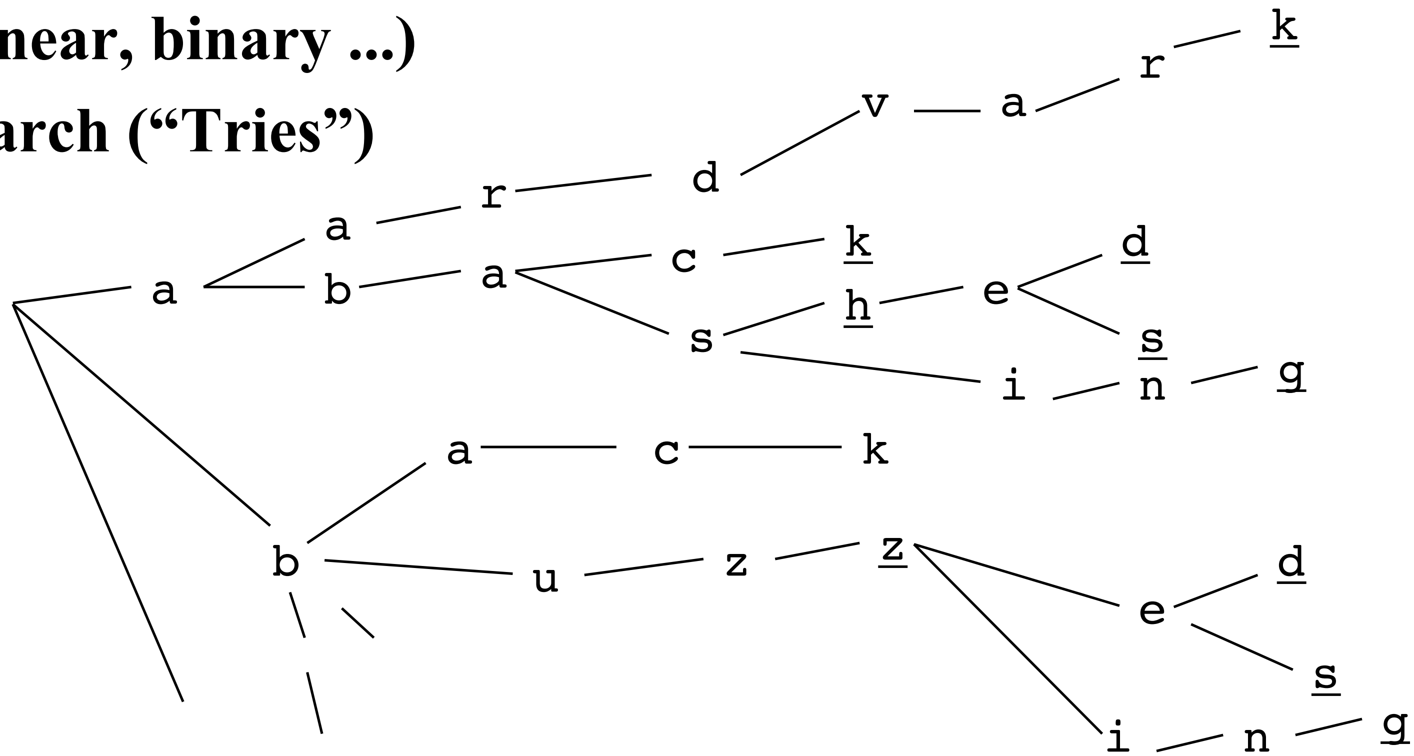
Approximate methods

- **Has right set of letters (any order).**
- **Has right sounds (Soundex).**
- **Random (suprimposed) coding (Unix Spell)**



Exact Methods

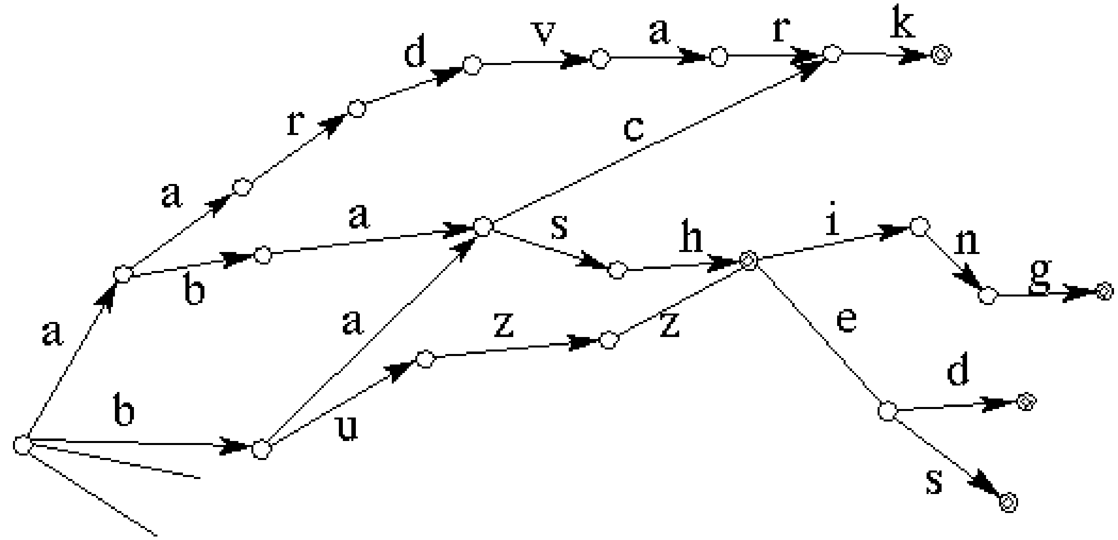
- **Hashing**
- **Search (linear, binary ...)**
- **Digital search (“Tries”)**



Folds together common prefixes

Exact Methods (continued)

- **Finite-state automata**



Folds together common prefixes and suffixes

Enumeration vs. Description

- **Enumeration**
  - —Representation includes an item for each object.

    Size = f(Items)
- **Description**
  - —Representation provides a characterization of the set of all items.

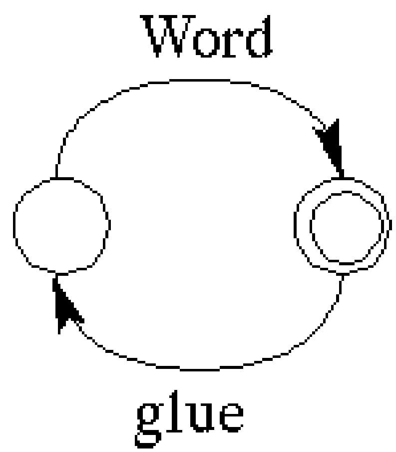
    Size = g(Common properties, Exceptions)
  - —Adding item can decrease size.

Classification

	Exact	Approximate
Enumeration	Hash table Binary search	Soundex
Description	Trie FSM	Unix Spell Right letter

FSM Extends to Infinite Sets

Productive compounding



Kindergartensgeselschaft

Statistics

	English	Portuguese
Vocabulary		
Words	81,142	206,786
KBytes	858	2,389
PKPAK	313	683
PKZIP	253	602
FSM		
States	29,317	17,267
Transitions	67,709	45,838
KBytes	203	124

From Lucchese and Kowaltowski (1993)

Dictionary Lookup

Dictionary lookup takes a string of characters as input and returns "yes" or "no" according as the word is or is not in a given set *and returns information about the word*.

Lookup Methods

Approximate — guess the information

If it ends in “ed”, it’s a past-tense verb.

Exact — store the information for finitely many words

Table Lookup

- Hash
- Search
- Trie —store at word-endings.

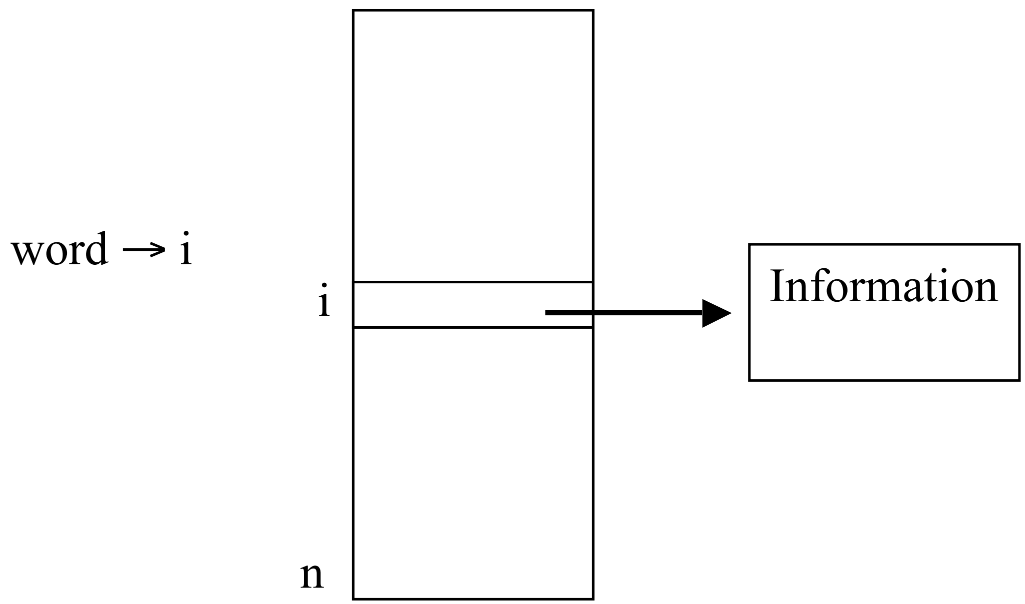
FSM

- Store at final states?

  No suffix collapse — reverts to Trie.

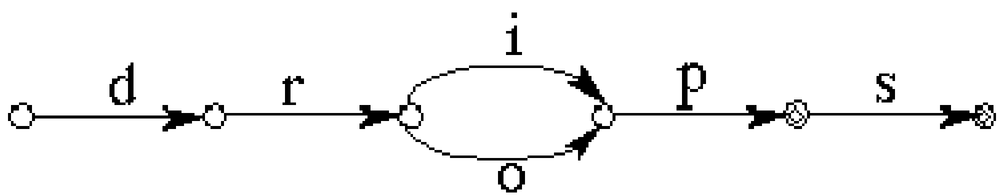
Word Identifiers

Associate a unique, useful, identifier with each of *n* words, e.g. an integer from 1 to *n*. This can be used to index a vector of dictionary information.



Pre-order Walk

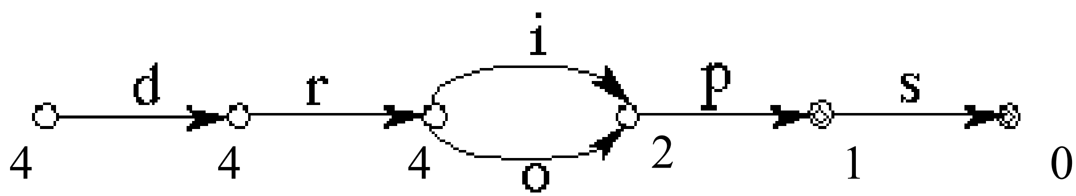
A pre-order walk of an *n*-word FSM, counting final states, assigns such integers, even if suffixes are collapsed ⇒ Linear Search.



drip →	1
drips →	2
drop →	3
drops →	4

Suffix Counts

- **Store with each state the size of its suffix set**
- **Skip irrelevant transitions, incrementing count by destination suffix sizes.**



drip →	1
drips →	2
drop →	3
drops →	4

- **Minimal Perfect Hash (Lucchesi and Kowaltowski)**
- **Word-number mapping (Kaplan and Kay, 1985)**

Spelling Conventions

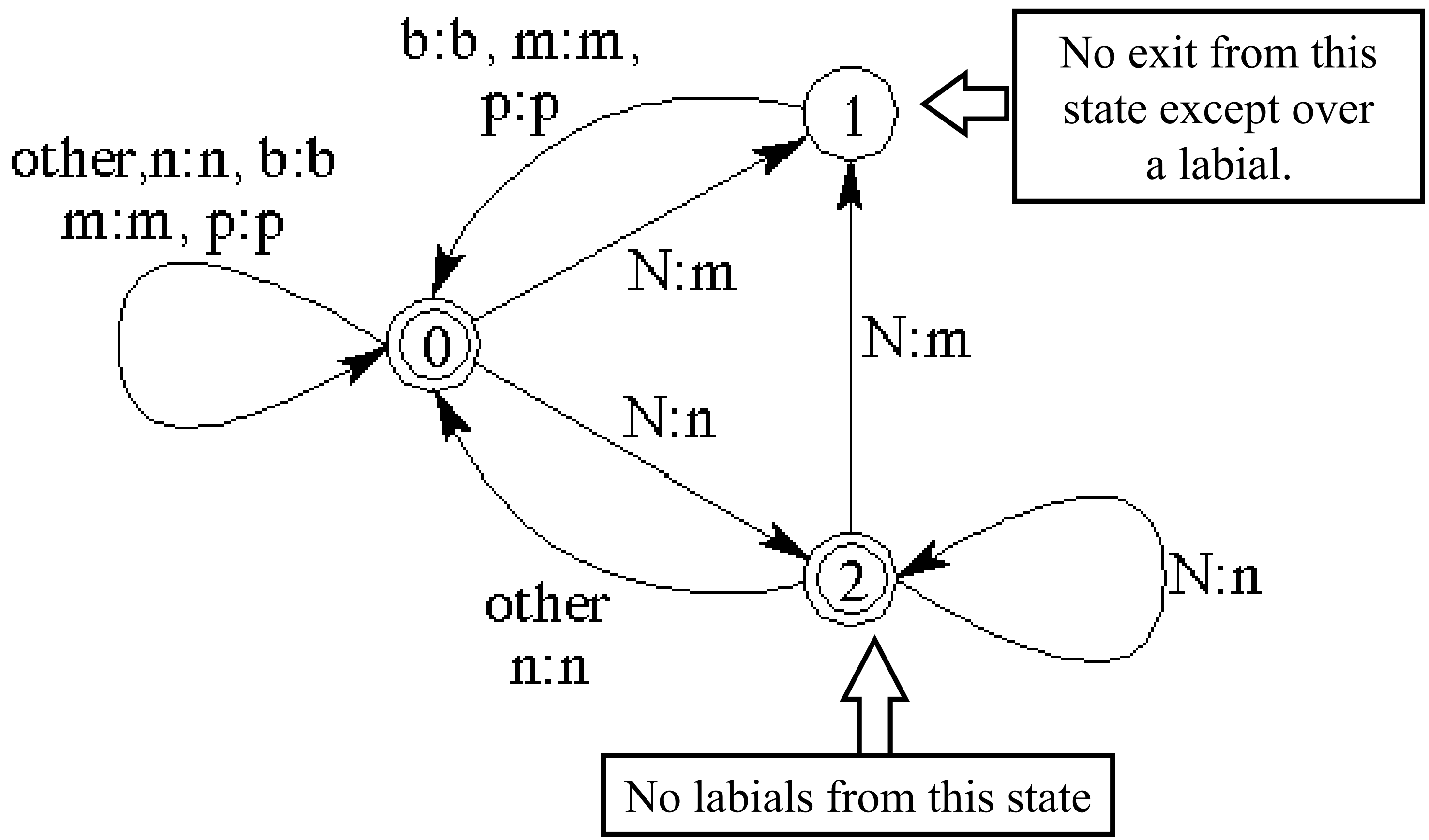
iN+tractable → intractable

iN+practical → impractical

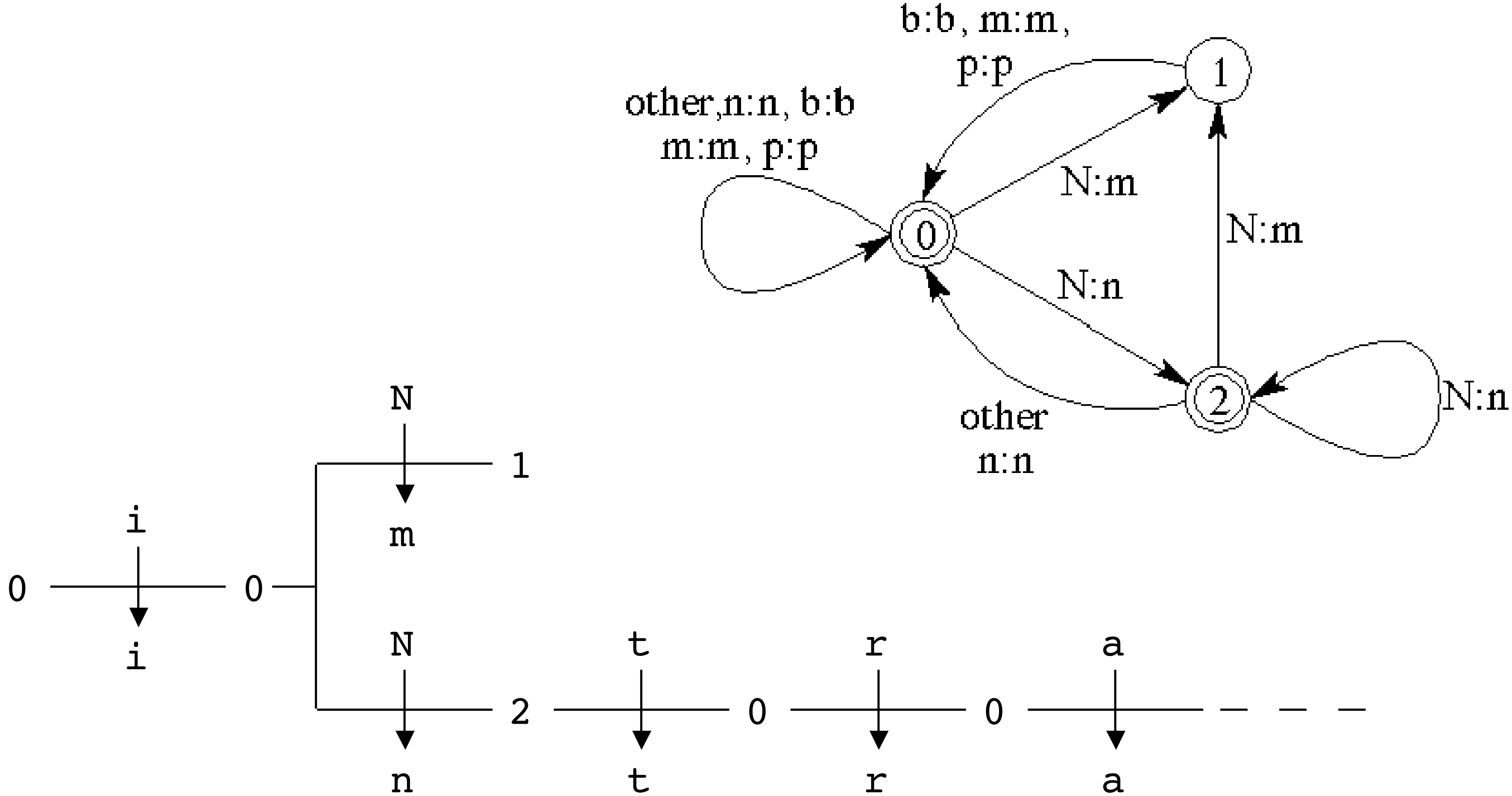
iN is the common negative prefix
— im before labial
— in otherwise

c.f. input → input

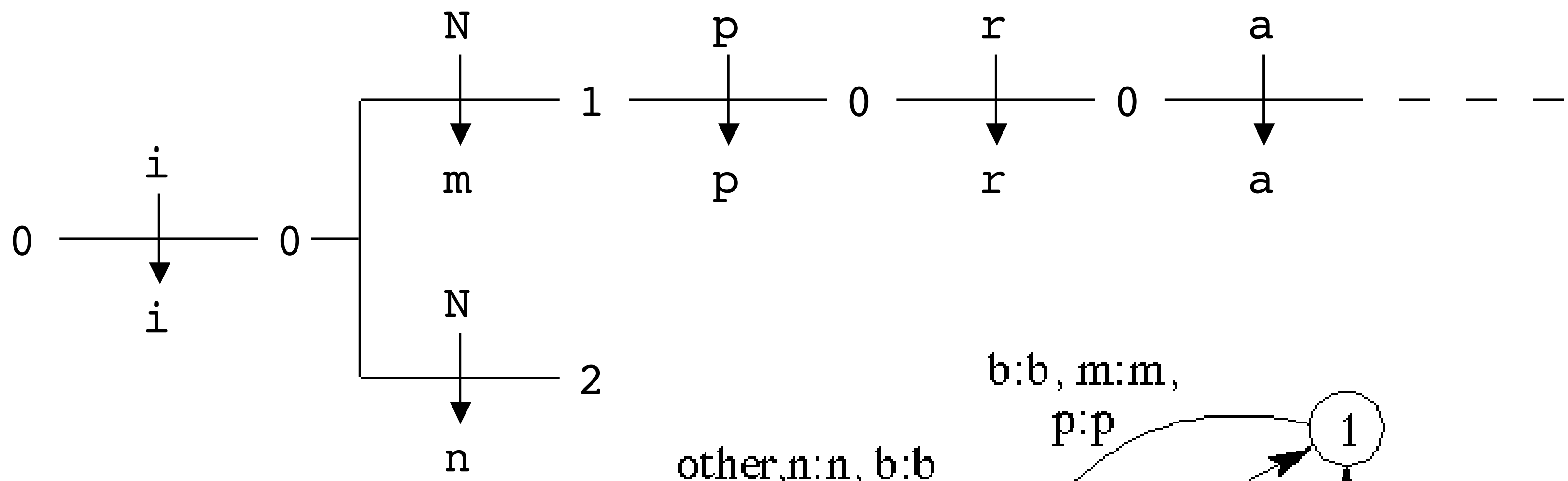
An in/im Transducer

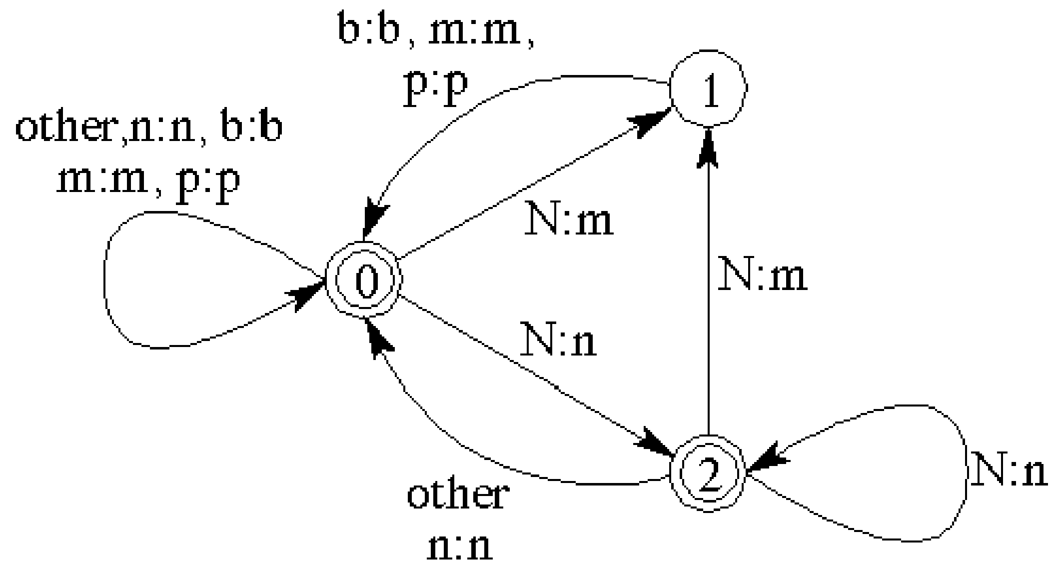


Generation — “intractable”

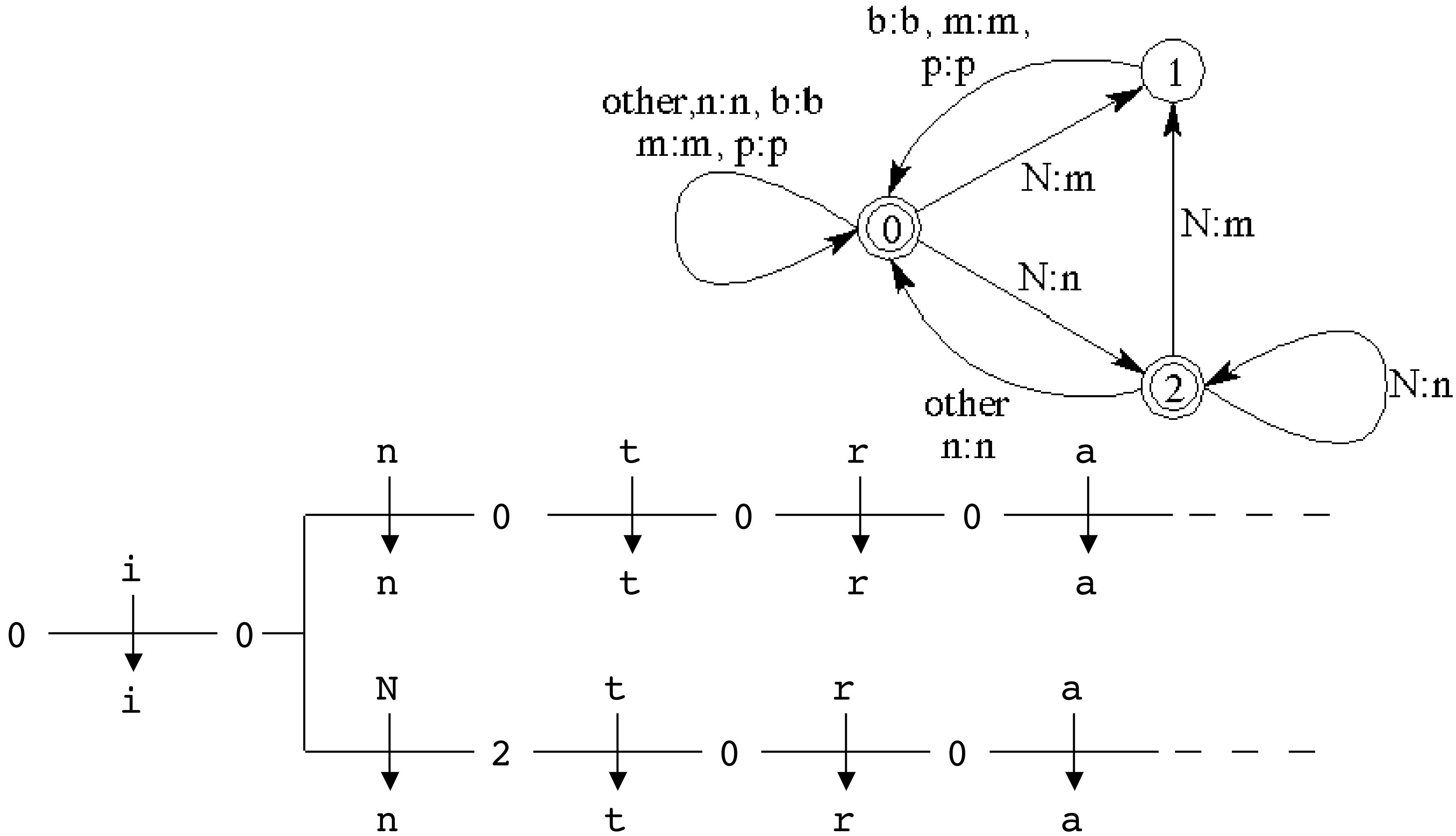


Generation — "impractical"

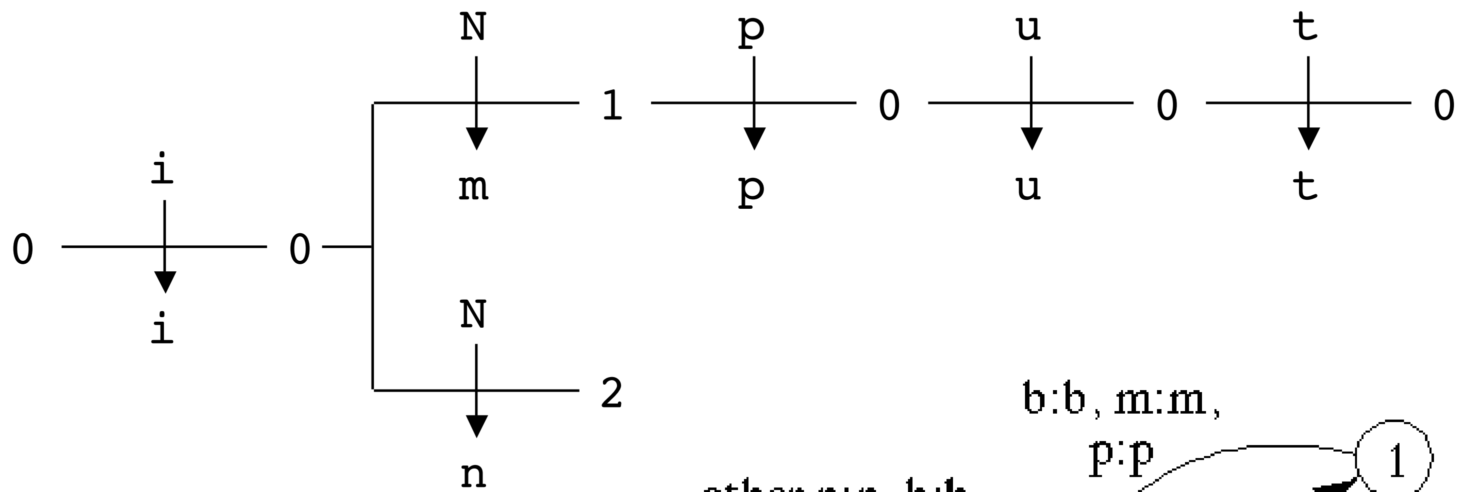


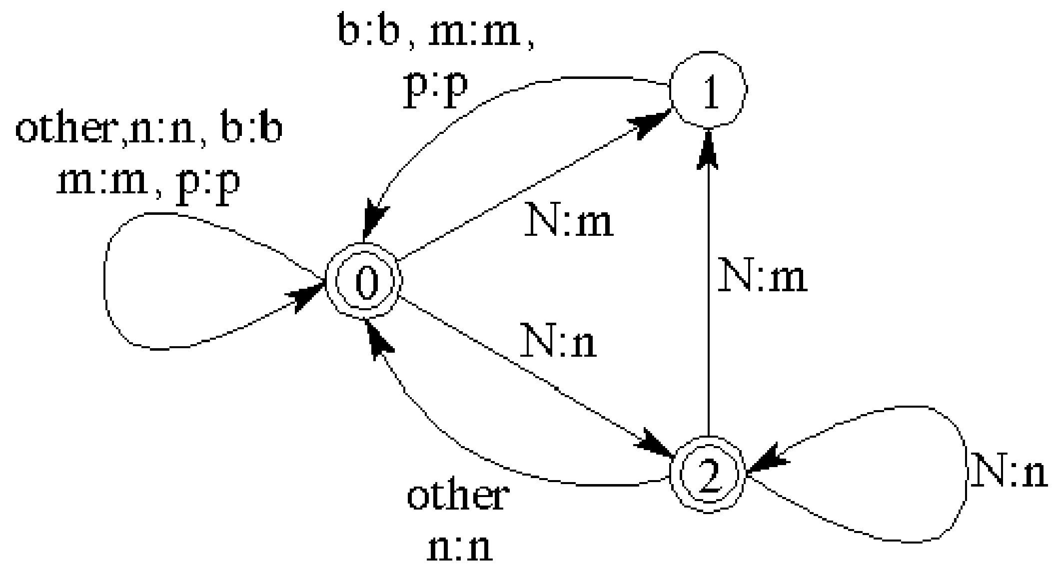


Recognition — "intractable"

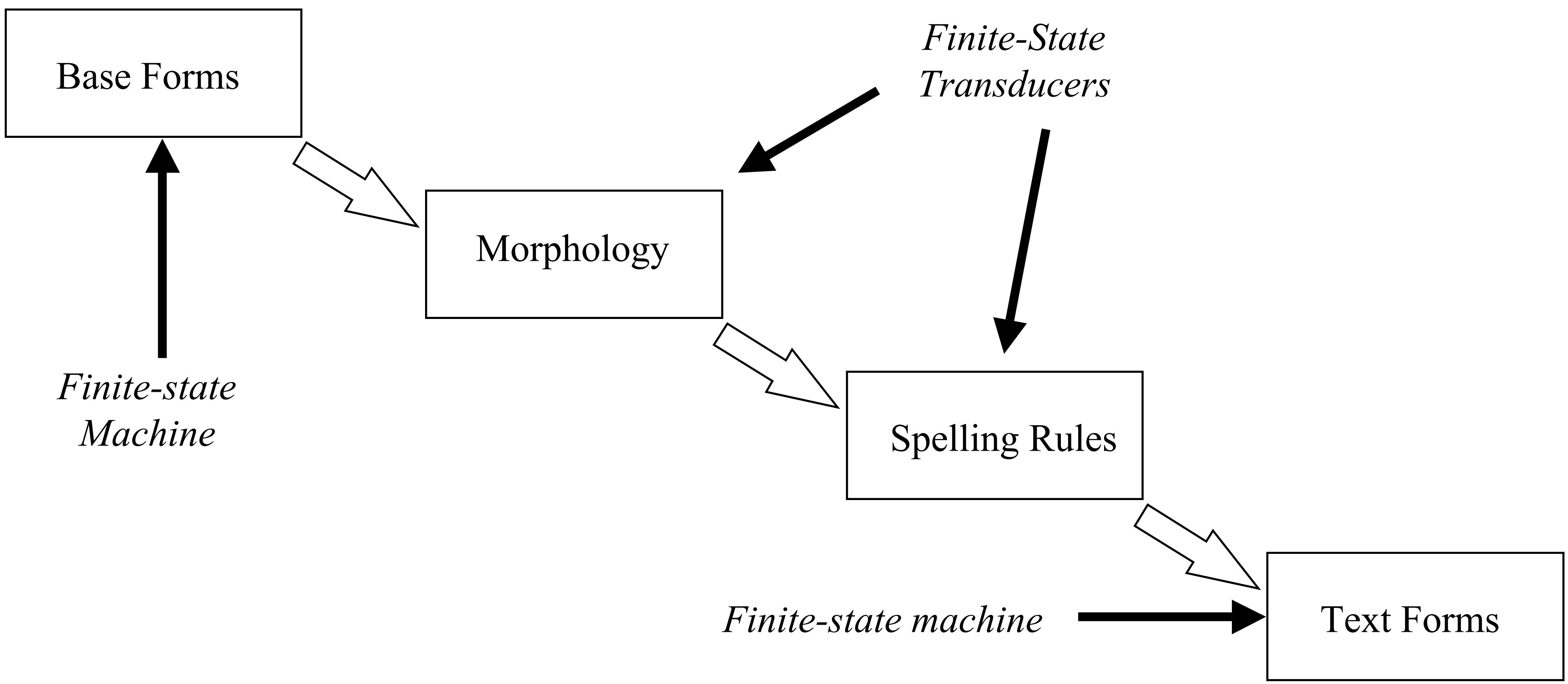


Generation — “input”





A Word Transducer



Bibliography

Kaplan, Ronald M. and Martin Kay. "Regular models of phololgical rule systems." *Computational Linguistics*, 23:3, September 1994.

Hopcroft, John E. and Jeffrey D. Ullman. *Introduction to Automata Theory, Languages and Computation*. Addison Wesley, 1979. Chapters 2 and 3.

Partee, Barbara H., Alice ter Meulen and Robert E. Wall. *Mathematical Methods in Linguistics*. Kluwer, 1990. Chapters 16 and 17.

Lucchesi, Cláudio L. and Tomasz Kowaltowski. "Applications of Finite Automata Representing Large Vocabularies". *Software Practice and Esperience*. 23:1, January 1993.