- Jurafsky, D. and Martin, J. H. (2009): Speech and Language Processing. An Introduction to Natural Language Processing, Computational Linguistics and Speech Recognition. Second Edition. Pearson: New Jersey: Chapter 3.2
- Carstensen, K.U., Ebert, Ch., Endriss, C., Jekat, S., Klabunde, R. and Langer, H. (Editors) (2004): Computerlinguistik und Sprachtechnologie. Eine Einführung. 2. Auflage. Spektrum: Heidelberg, pages 198-205
- G. Heyer, U. Quasthoff, T. Wittig (Eds.) (2006): Wissensrohstoff Text, Bochum, w3L. Kapitel 3.4

Transducers, Compact Patricia Tries and DAWGs

FINITE STATE MORPHOLOGY

MORPHOLOGY WITH FSAS

Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

- Morphology works fairly regular, so FSAs are an appropriate machinery for morphological analysis
- Tasks for automated morphology:
  - analyze a given word into its morphemes
  - generate a full form from a base form + morphological information

Surface	Lexical
runs	run+Verb+Present+3sg run+Noun+Pl
largest	large+Adj+Sup
better	good+Adj+Comp

Surface	Lexical
Boote	boot+Nomen+Plural
verlangsamte	verlangsam+Verb +Imperf+3sg verlangsamt+Adj +NomAkk

- Plain word lists are a possibility, but redundancies are not utilized and access can be slow. Further, no generalization properties: cannot utilize regularities from inflection classes, cannot guess for unseen words

FINITE STATE TRANSDUCER

Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

A **finite state transducer** is a 6-tuple **FST**=($\Phi,\Sigma,\Gamma,\delta,S,F$) and consists of

- set of states Φ
- input alphabet Σ, disjunct with Φ
- output alphabet Γ, disjunct with Φ
- set of start states $S \subset \Phi$
- set of final states $F \subset \Phi$
- transition function $\delta \subseteq \Phi \times (\Sigma \cup \{\varepsilon\}) \times (\Gamma \cup \{\varepsilon\}) \times \Phi$

An **FST** is essentially an FSA with two tapes. It is useful to think about them as input tape and output tape, or upper tape and lower tape.

An FST transduces an input string x to an output string y if there is a sequence of transitions that starts with a start state and ends with a final state and has x as its input and y as its output string.

FSTs accept **regular relations**.

REGULAR RELATIONS, CLOSURE

Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

The set of **regular relations** is defined as follows:

- For all $(x,y) \in \Sigma \times \Gamma$, $\{(x, y)\}$ is a regular relation
- The empty set $\varnothing$ is a regular relation
- If Q, R are regular relations, so are $Q \bullet R = \{(x_1x_2, y_1y_2) \mid (x_1,y_1) \in Q, (x_2,y_2) \in R\}$, $Q \cup R$ and Q^*.
- Nothing else is a regular relation.

Like regular languages, regular relations are closed under

- union
- concatenation
- Kleene closure

Unlike regular languages, regular relations are NOT closed under

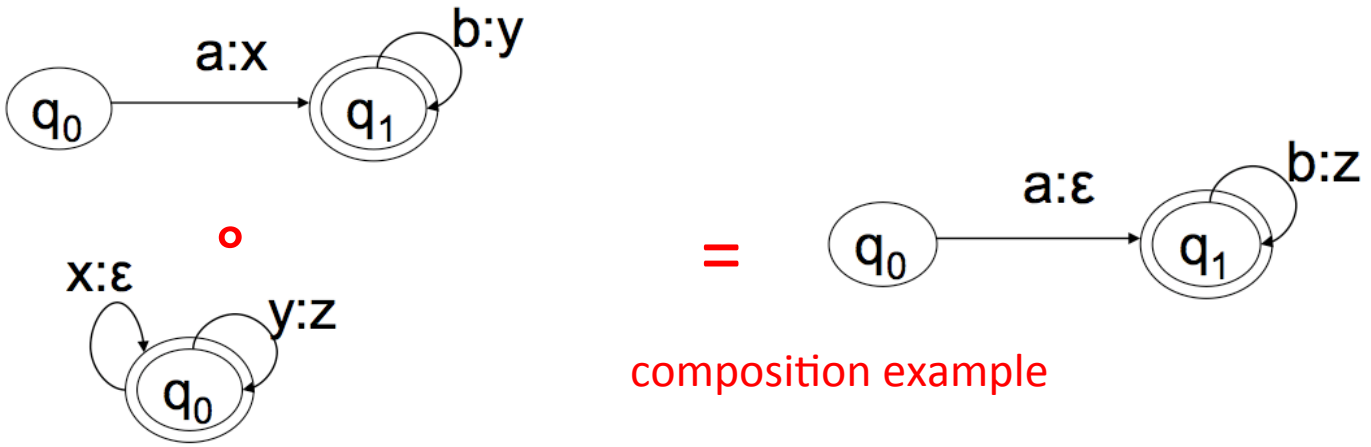
- intersection
- difference
- complementation

CLOSURE OF REGULAR RELATIONS (CTD.)

Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

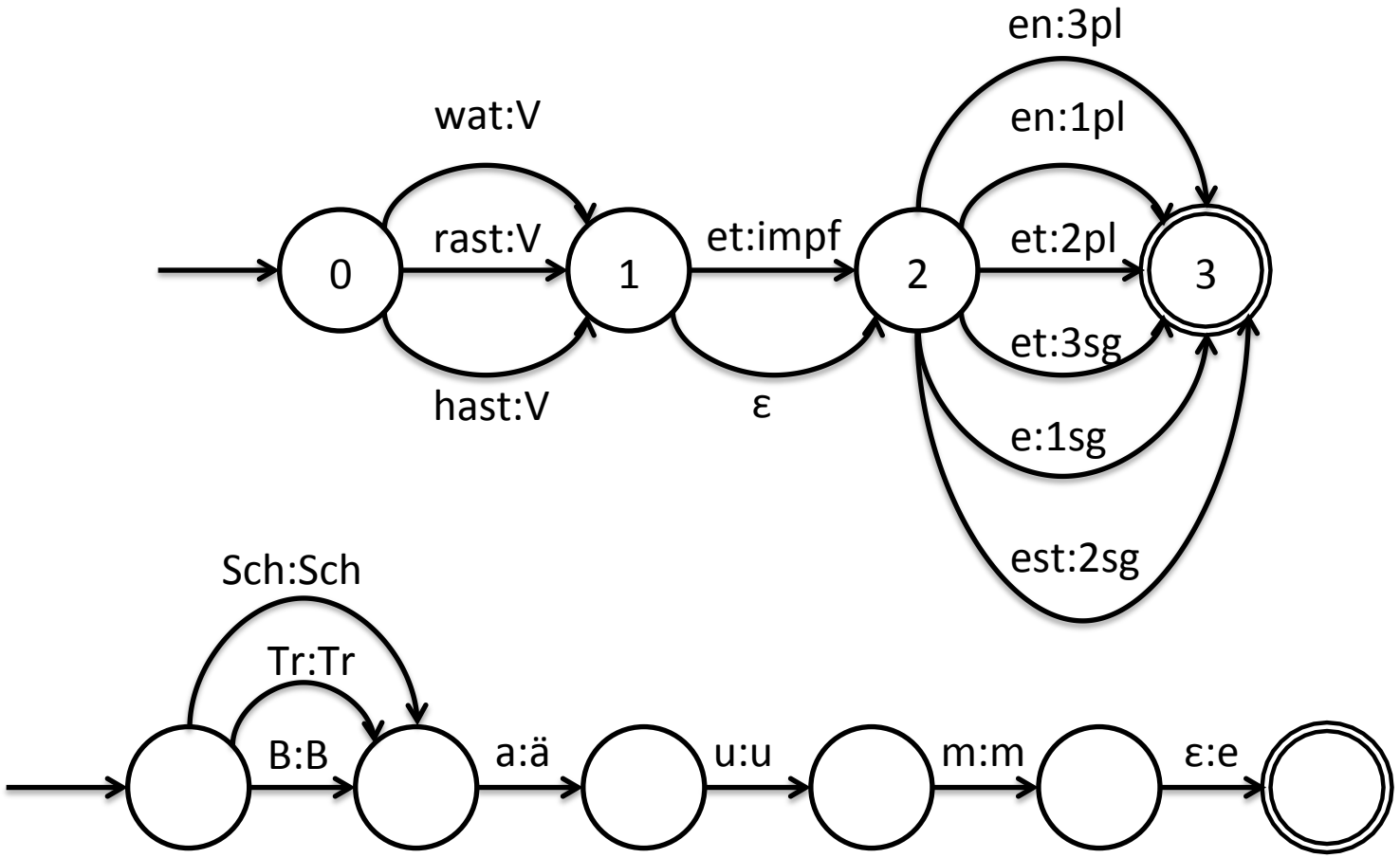
New operations for regular relations:

- Composition: Q°R: {(x,z) | ∃y: (x,y) ∈Q and (y,z) ∈R}
- Projection: {x | ∃y, (x,y)∈R}
- Inversion: {(y,x) | (x,y)∈R}
- From regular language L to identity regular relation: {(x,x) | x∈L}
- From two regular languages L and M, create the cross product relation: {(x,y) | x∈L, y∈M}



composition example

EXAMPLES FOR MORPHOLOGY FSTS

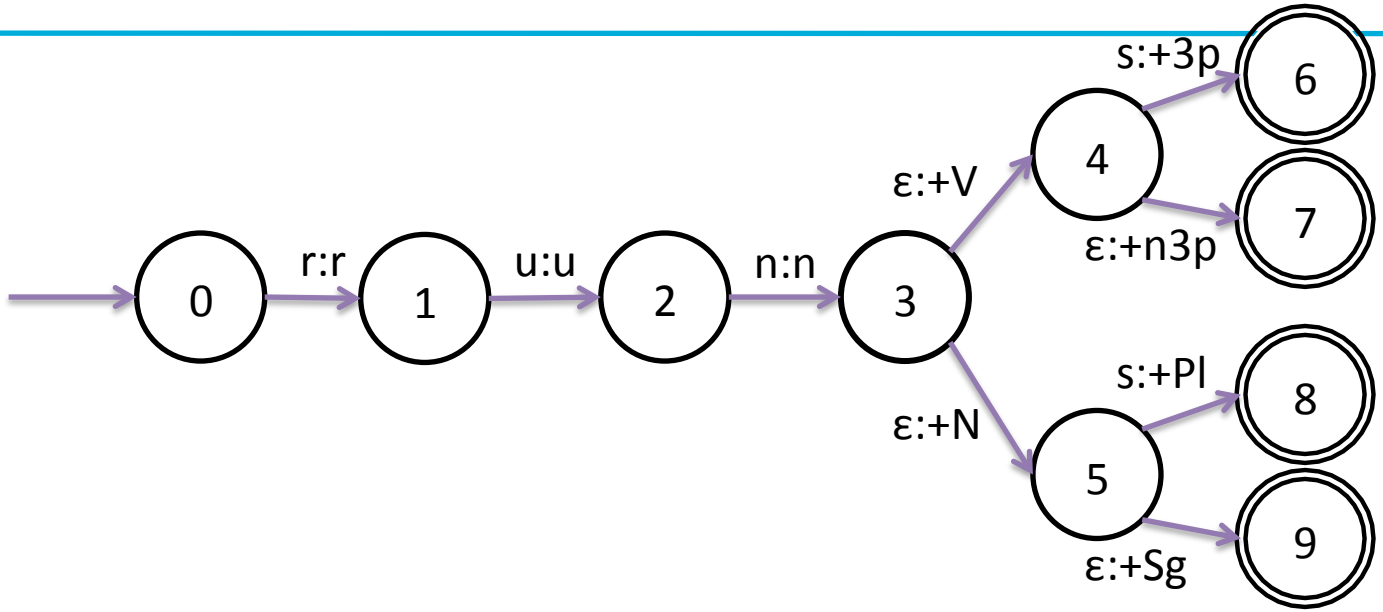


Note that FSTs can be non-deterministic and can have ε-transitions.

HANDLING NONDETERMINISM AND AMBIGUITIES

- Since language is ambiguous on many levels, we embrace nondeterminism as a mechanism to reflect that
- As long as we do not know how to resolve ambiguities, we carry along several possibilities
- Nondeterminism for FSA: we don't know which path we took
- Nondeterminism for FST: different paths produce different output strings
- Nondeterminism requires to keep track of a **set** of current states
- A nondeterministic automaton accepts if there is at least one path to a final state

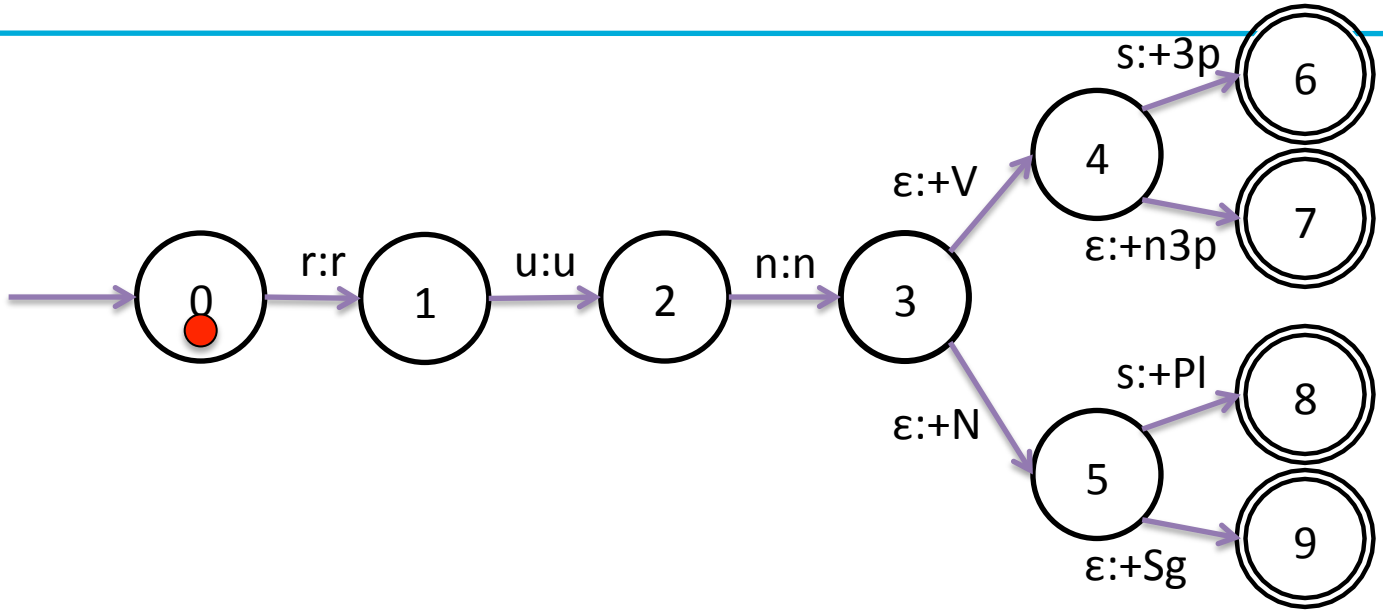
RUNNING EXAMPLE

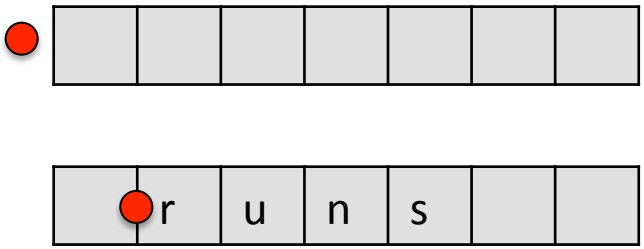


input string

	r	u	n	s		

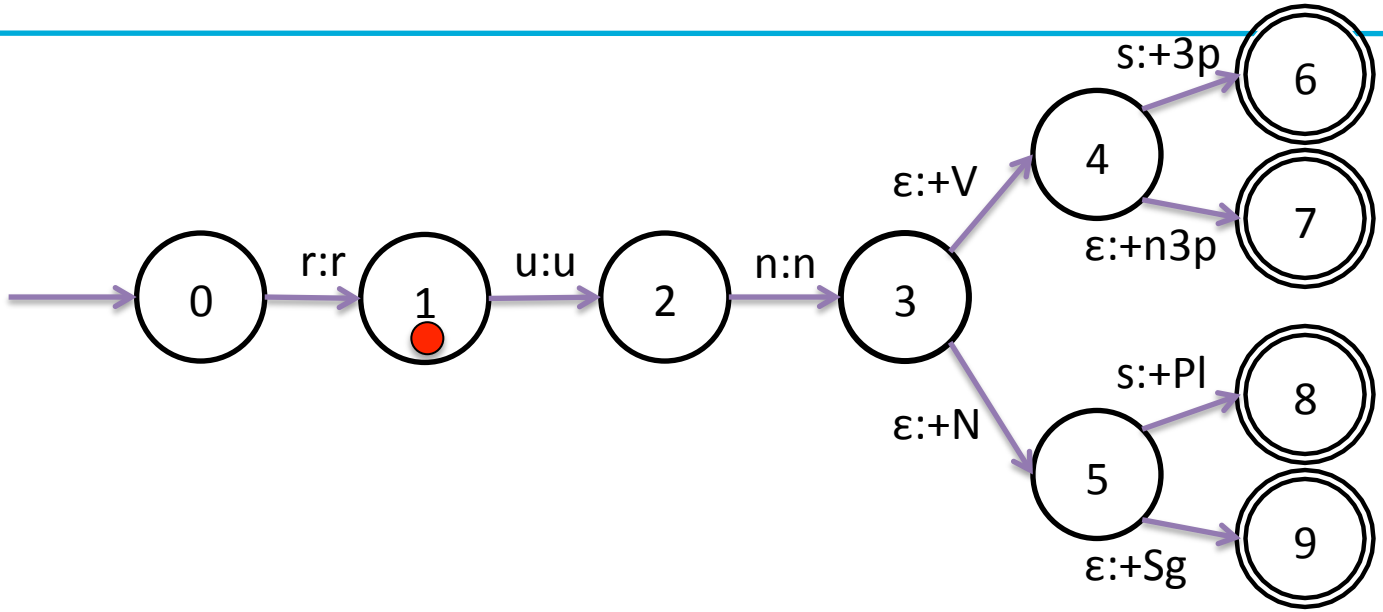
RUNNING EXAMPLE

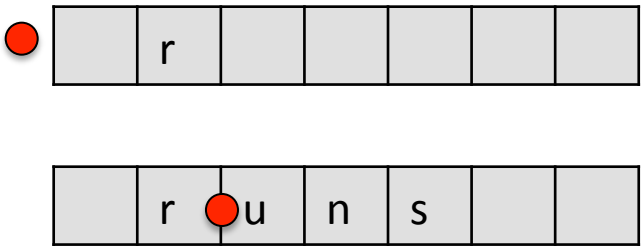




Dots: Keep track current state and output generated so far.

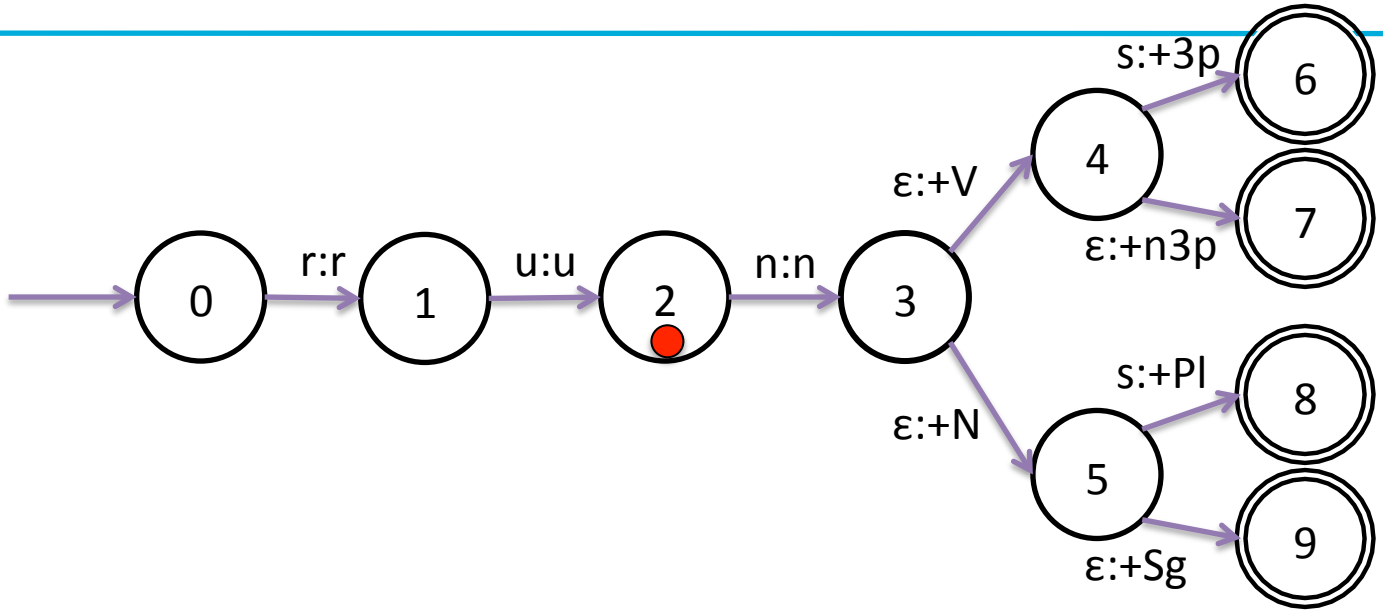
RUNNING EXAMPLE

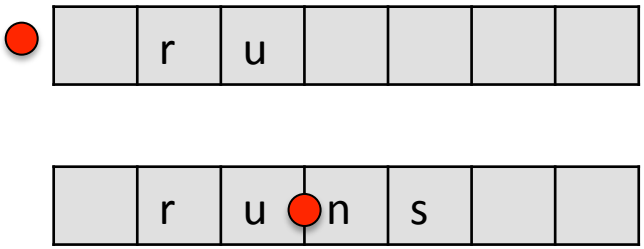




Transition: dot moves on input tape and to next state, generating output

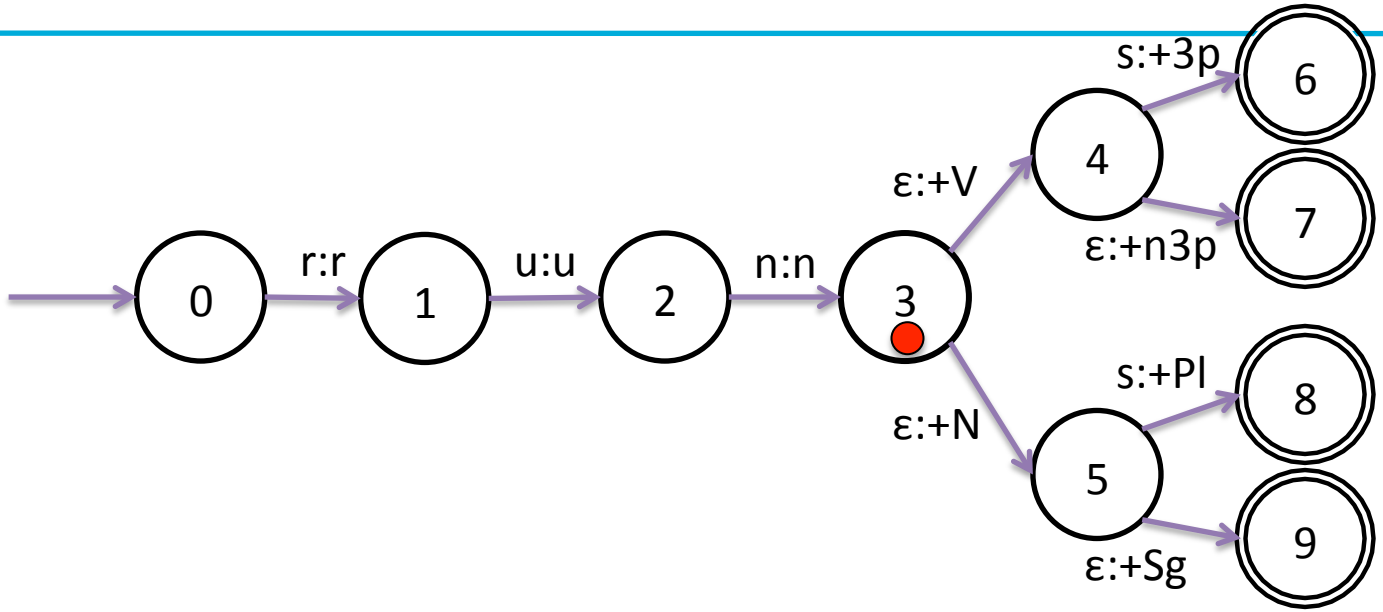
RUNNING EXAMPLE

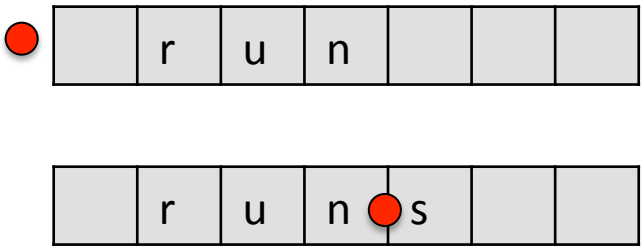




Transition: dot moves on input tape and to next state, generating output

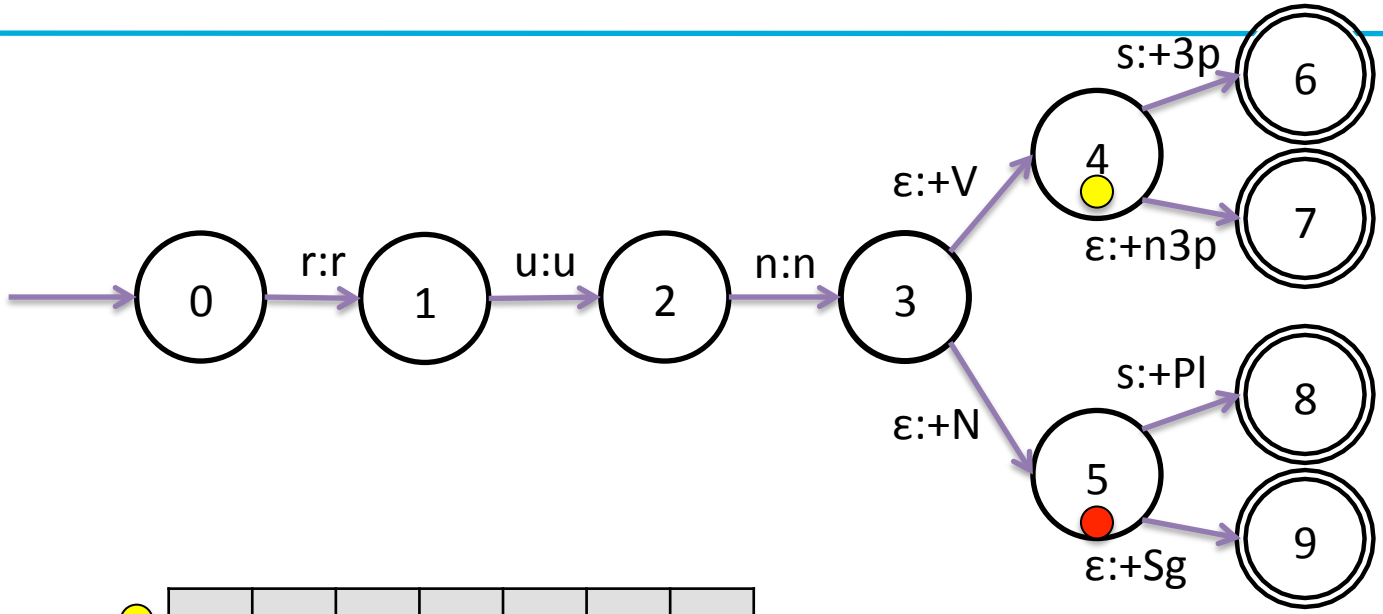
RUNNING EXAMPLE

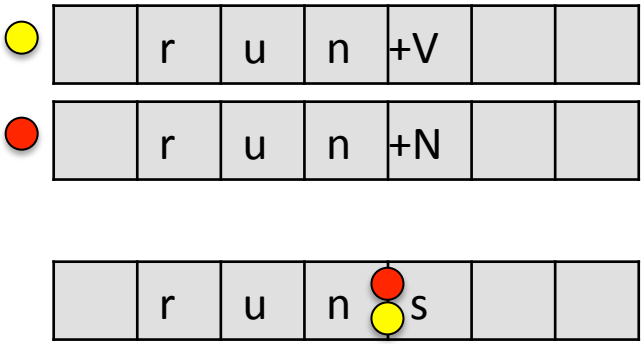




Transition: dot moves on input tape and to next state, generating output

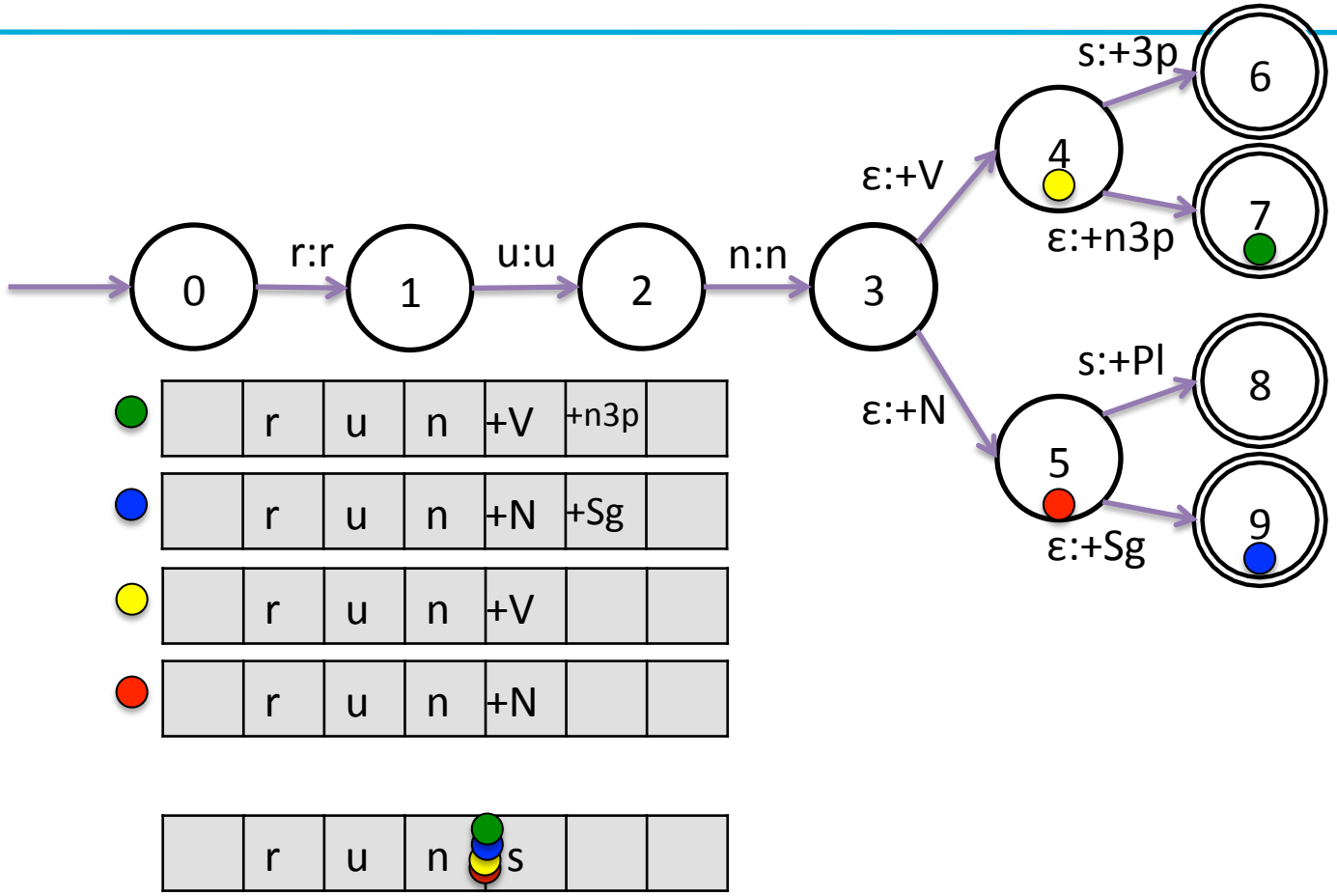
RUNNING EXAMPLE





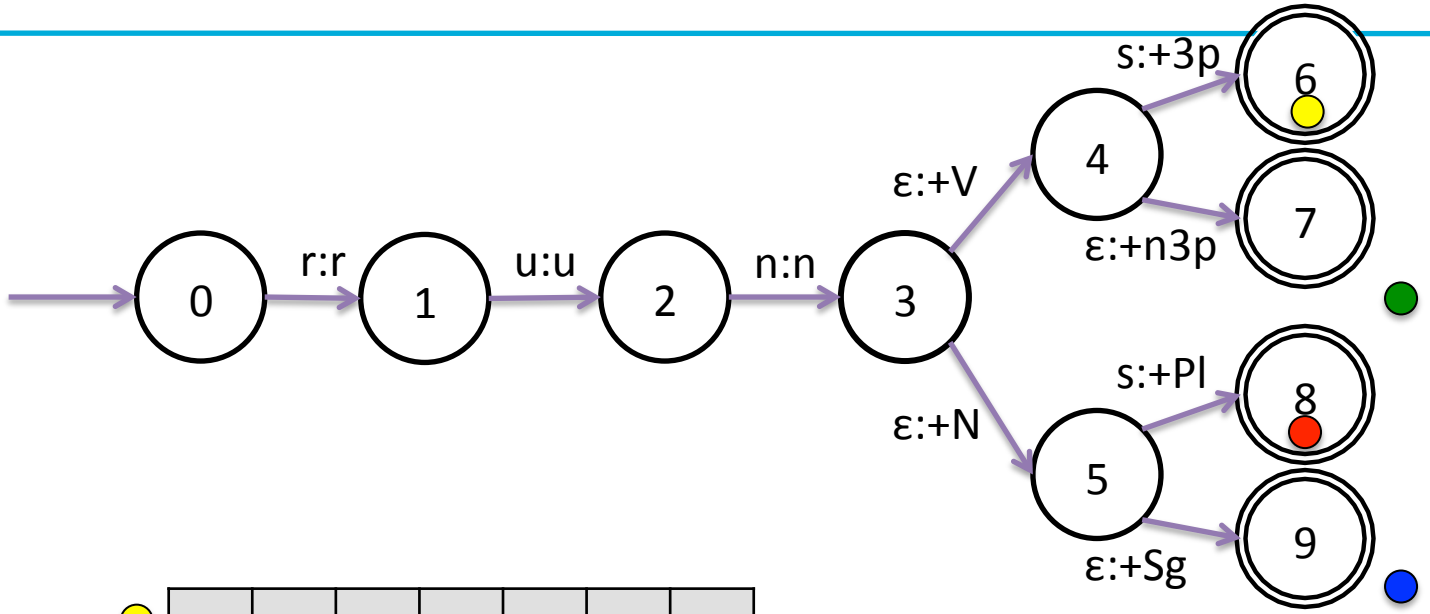
Non-determinism: Dot splits. Output tape is copied.

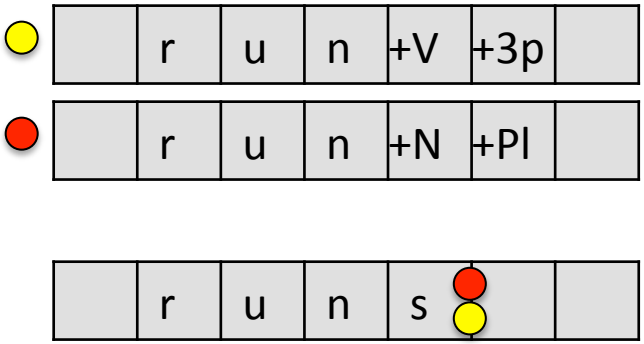
RUNNING EXAMPLE



ε-transitions are also non-determinisms.

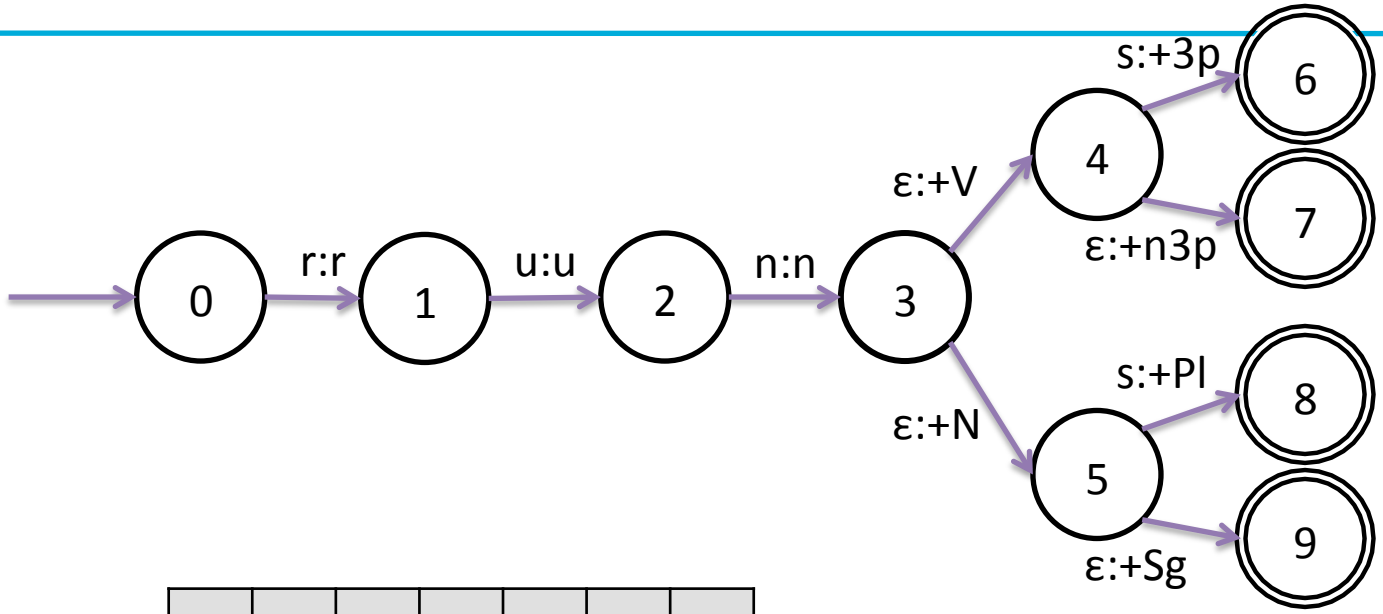
RUNNING EXAMPLE





Dots that do not have a follow-up state are abandoned.

RUNNING EXAMPLE



output strings

	r	u	n	+V	+3p	
	r	u	n	+N	+Pl	

	r	u	n	s		

End of input is reached. All dots at final states have successfully transduced.

TWO LEVEL MORPHOLOGY

Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

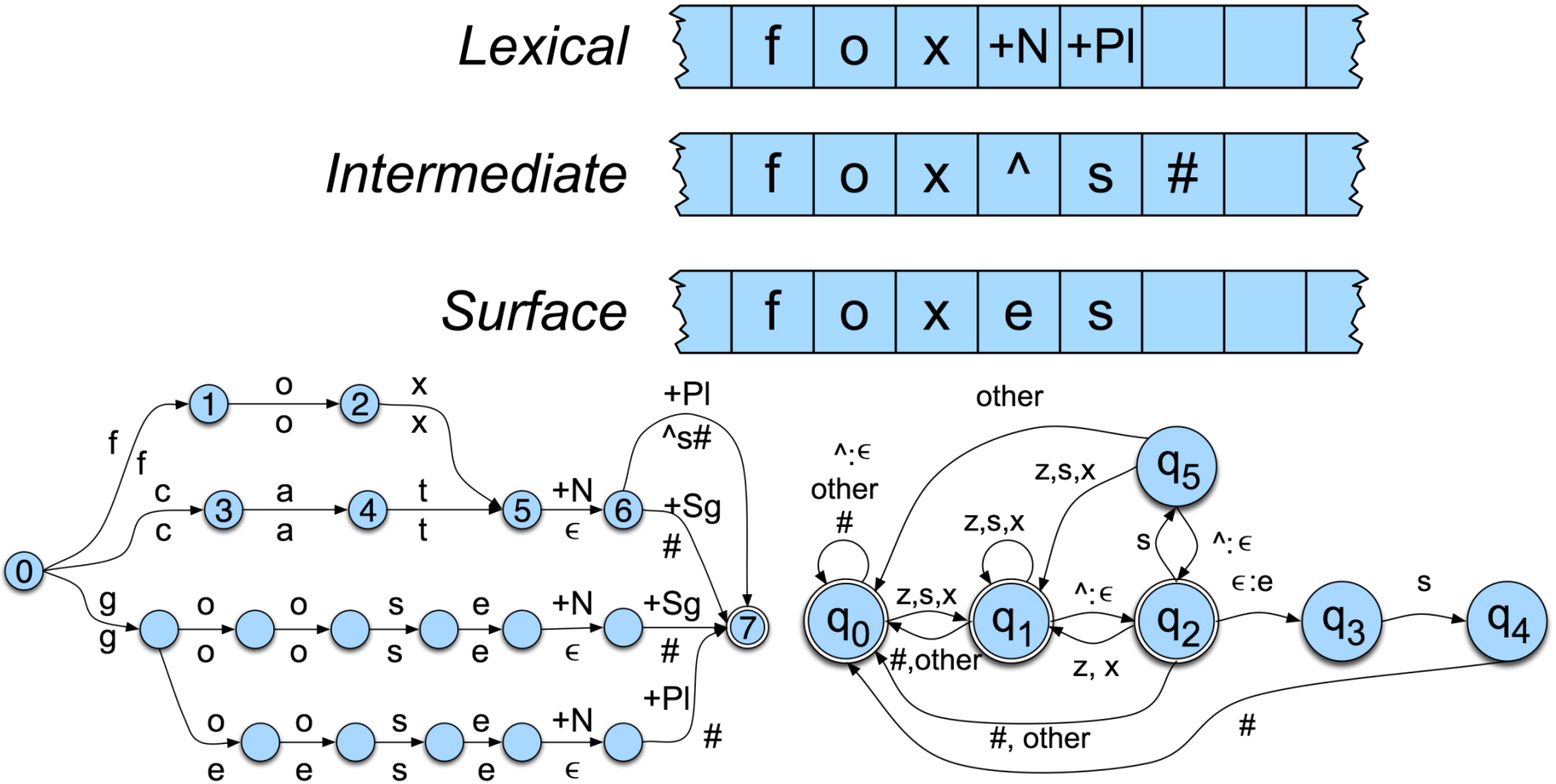
- Single morphology FSTs get very complex when accommodating large word lists in a large number of different flexion classes
- need to express word lists and spelling rules separately: use concatenation
- two-level morphology works by introducing an intermediate level: use composition and intersection
  - surface to intermediate level: from surface form to morphemes
  - intermediate to lexical level: from morphemes to morphological analysis

The introduction of levels is here guided by linguistic intuition and merely a way to make writing and maintaining of FST morphological components simpler. In practice, all together is compiled into one big FST.

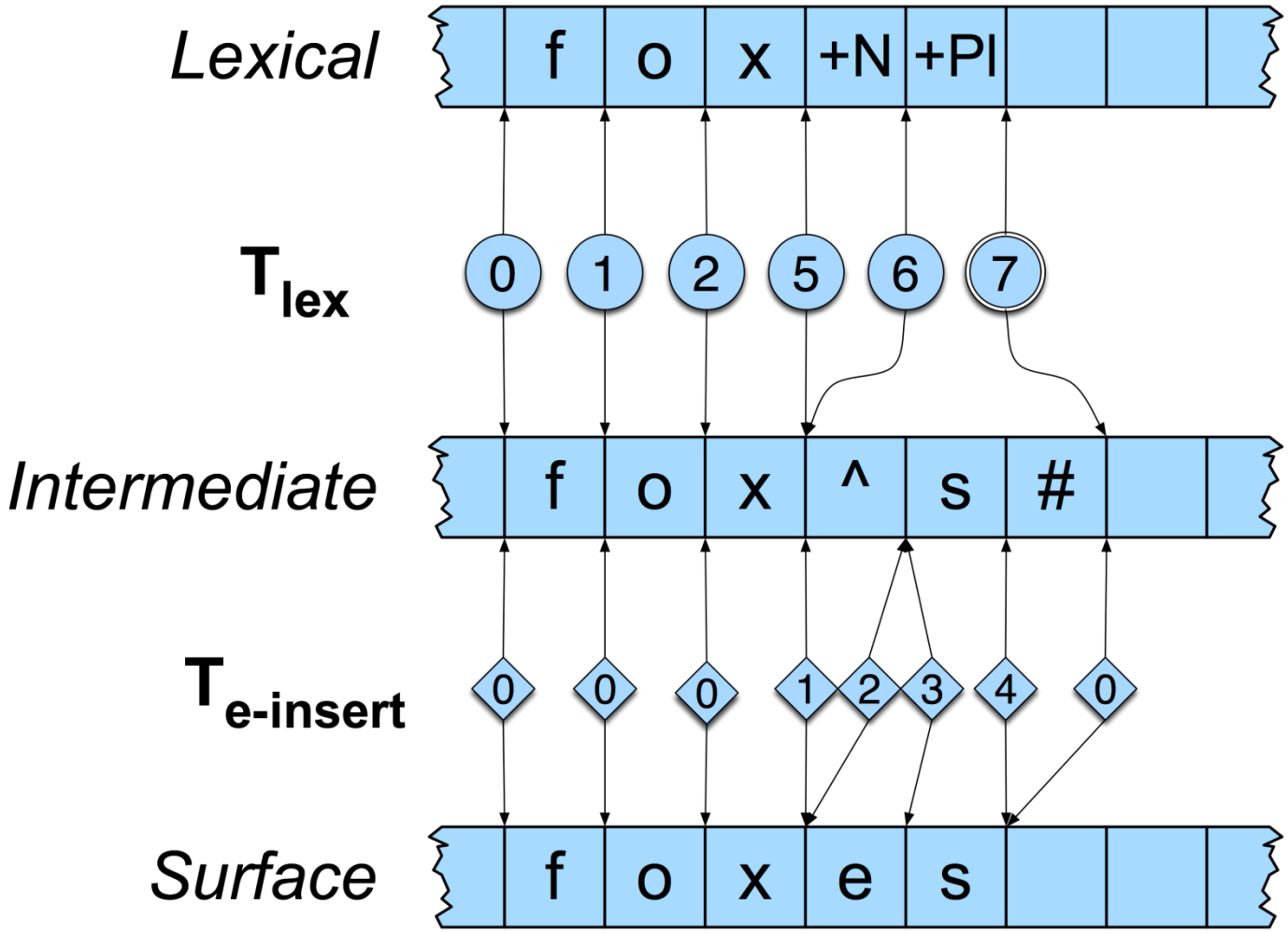
THE FOXES EXAMPLE (I)

Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

Synthesis/Analysis of “foxes”:



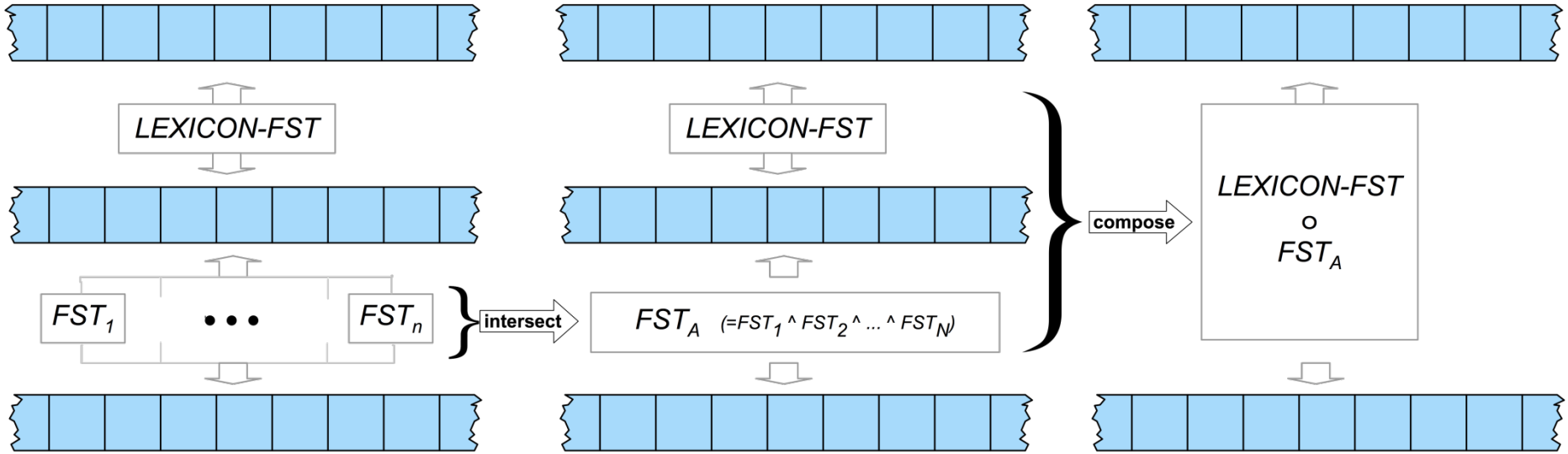
THE FOXES EXAMPLE (II)



OVERALL SCHEME

Why intersection?

- spelling rules are constraints, capturing each some phenomenon of spelling while not constraining cases where they do not apply
- spelling is correct if all constrains are satisfied
- intersection handles the parallel checking if all constraints are satisfied, i.e. no spelling rule is violated



declaration intersection of spelling rules composition

- For intersection, the rules have to be modified to treat ε as part of the alphabet to ensure equal length

APPLICATIONS OF FSTS IN LANGUAGE TECHNOLOGY

Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

- Lexicon data structure for e.g. speller
- Morphology analysis and synthesis
- Segmentation
- Tokenization
- Sentence boundary detection
- Chunk parsing (cascaded)
- decoding in speech recognition

MOTIVATION FOR SEARCH TREES

Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

Tasks:

- memory-efficient storage of word lists
- classification on the word level, e.g. lemmatization
- generalization capabilities: e.g. lemmatize "`googled` / `googelte`" even it it is not in the list of known/given words

In applications, full FSTs are too complex. Simpler structures: Tries and DAWGs

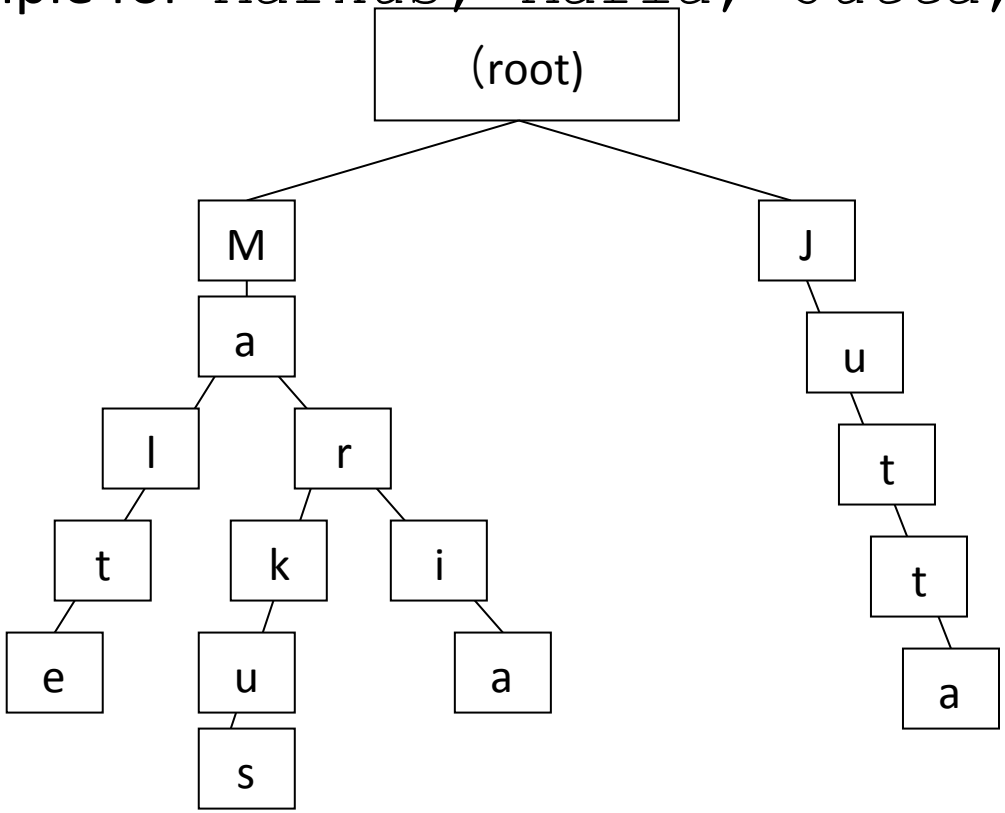
- deterministic: only one path per input
- no output tape
- compressing word lists
- generalization capabilities

Prerequisite: Search Trees

TRIES (A.K.A. PREFIX TREE): COMBINE COMMON PREFIXES

Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

- A trie is a tree structure. The nodes have 0 to N daughters (N number of possible characters in alphabet).
- Example for `Markus, Maria, Jutta, Malte`

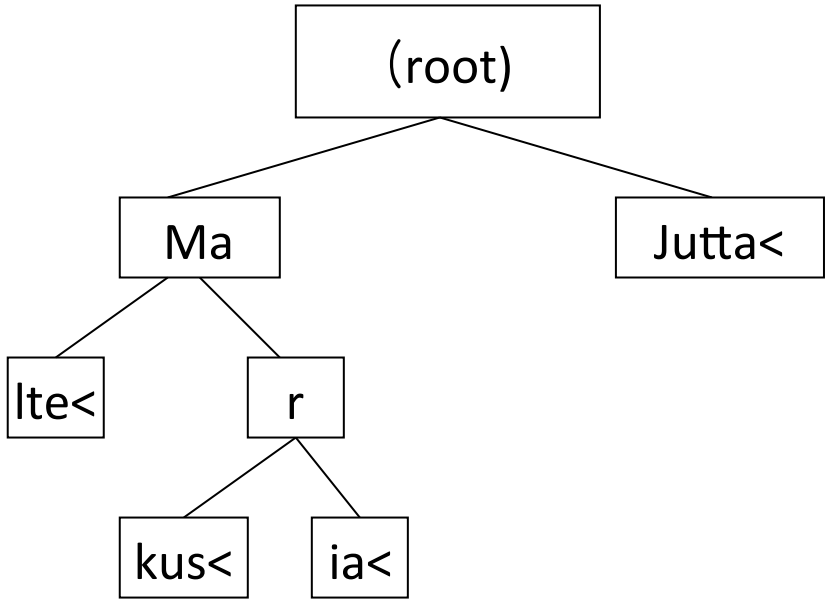


17 nodes with
16 characters,
16 edges

PATRICIA TRIE (PT)
(A.K.A. RADIX TREE)

Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

- Decrease number of edges by putting several characters in one node
- Example for `Markus, Maria, Jutta, Malte`

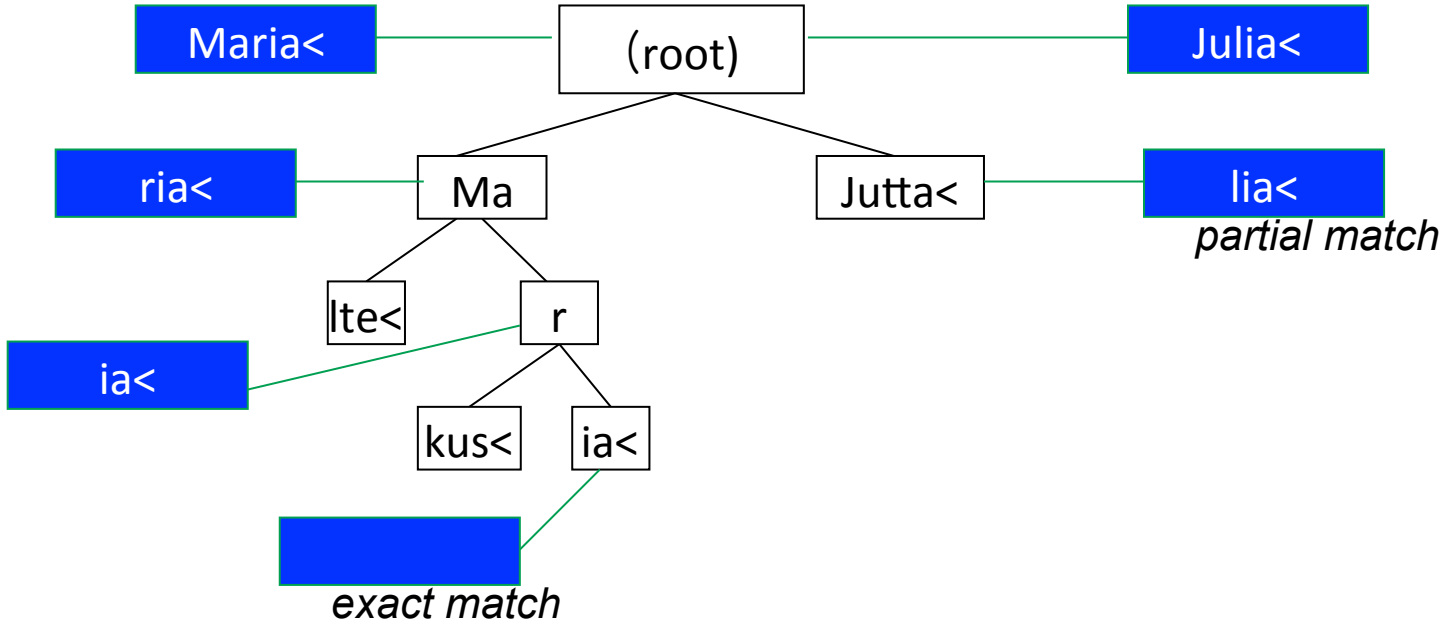


7 nodes,
16 characters,
6 edges.

"<" designates end-of-word

SEARCH IN PTS

- Recursively walk down, search word gets eaten up
- Return last reached node.
- If remaining search word is empty: *exact match*, otherwise *partial match*



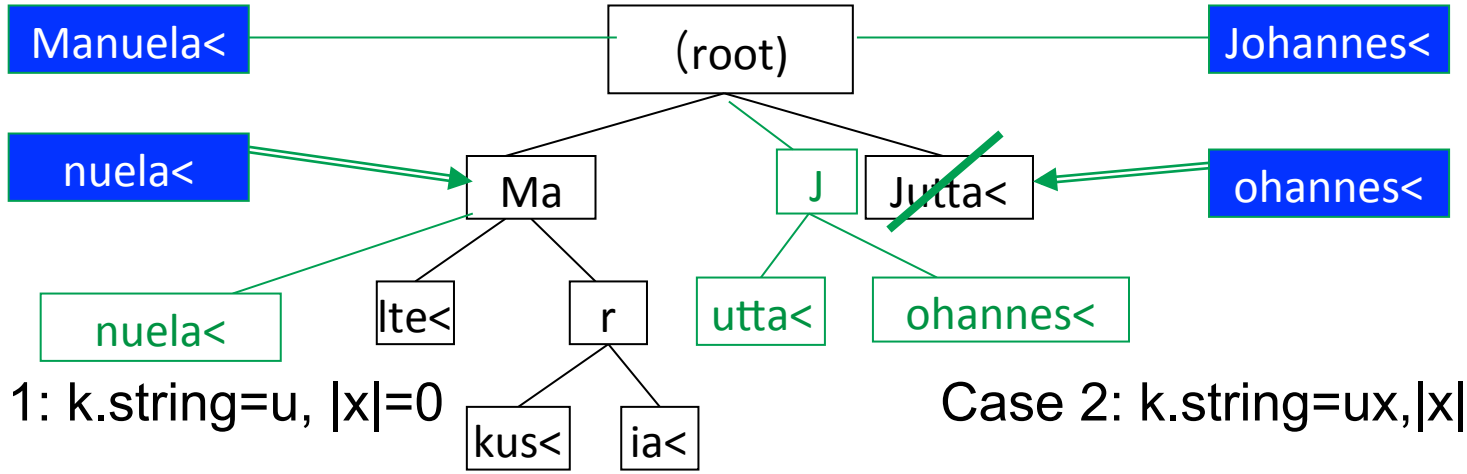
INSERT IN PTS

Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

Insert of w:

- Search for w returns appropriate node k
- if *exact match*: Word was in PT already
- if *partial match*: Split string contained in k, attach daughter nodes.

In k holds: k: w=uv, k.string=ux



Case 1: k.string=u, |x|=0

Insert one node with string v as daughter of k

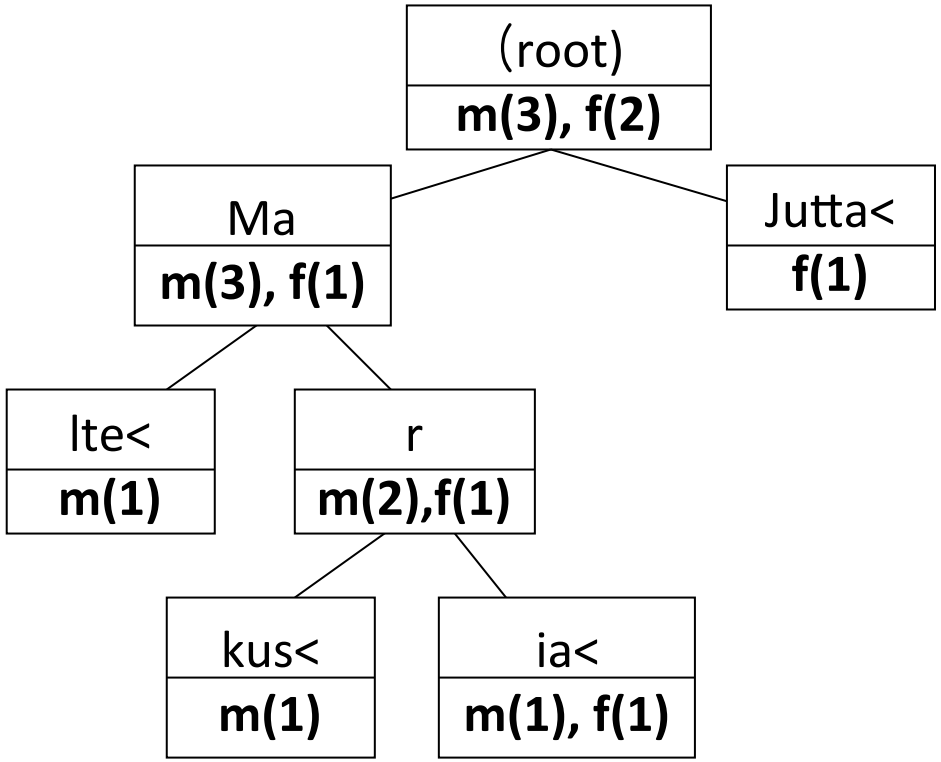
Case 2: k.string=ux,|x|>0

Insert two nodes with strings v and x as daughters of k

STORING ADDITIONAL INFORMATION IN PTS

Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

- Nodes are extended: An additional field stores some information
- Example: Storing the gender of names:



The classes can be found in the leaves.

In intermediate nodes, the additional field stores the sum of the classes in the subtree.

APPLICATION: BASE FORM REDUCTION (LEMMATIZATION)

Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

- Given: List of words with reduction rules

```
Haus                0
Hauses              2
Häuser              5aus
Maus                0
Mäuse               4aus
Bau                 0
Baus                1
Aus                 0
```

- Reduction: integer *n* and (possible empty) string *x*.
- read: cut *n* characters (bytes) from behind and attach *x*.
- ambiguous cases: multiple instructions:

```
Fang                    0; 0en
```

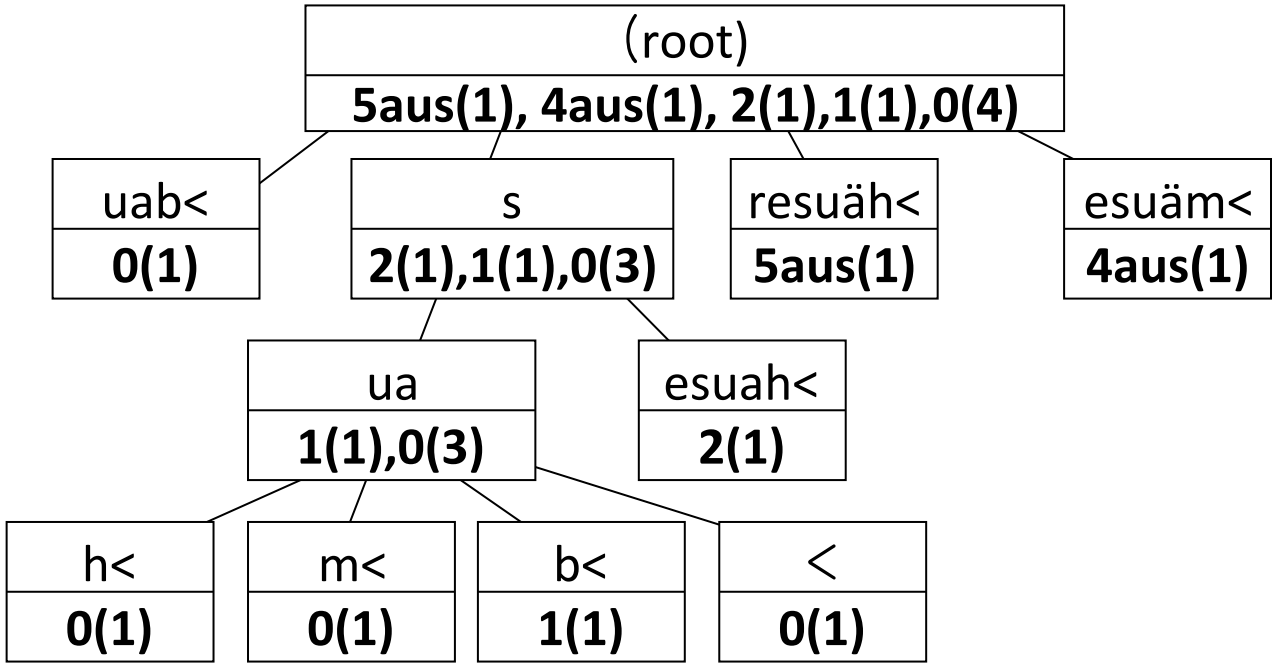
- inflection removal for verbs (German-specific): remove first string occurrence after operator #

```
geschienen          5einen#ge
```

BASE FORM REDUCTION II

Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

PT is built from reversed words, the reduction rules are stored in the nodes. "<" denotes start-of-word.

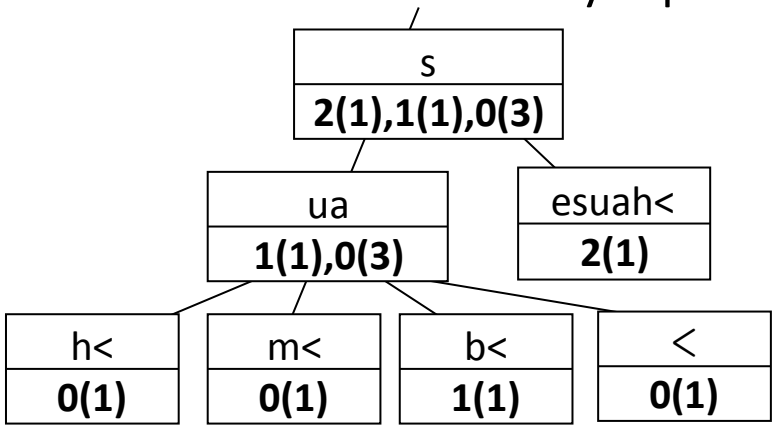



```
Haus     0
Hauses   2
Häuser   5aus
Maus     0
Mäuse    4aus
Bau      0
Baus     1
Aus      0
```

BASE FORM REDUCTION III

Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

- For base form reduction, a search with the reversed word is performed, this returns some node (leaf or intermediate node).
- The rule in this node will be applied. If there are several rules, take the one with the highest score and above some threshold
- Under the threshold, return '*undecided*'
- Unknown words receive a morphologically motivated guess
- all known words are fully represented: 100% correct on training



Hochhaus → 0

Spass → 0

Unterbaus → 1

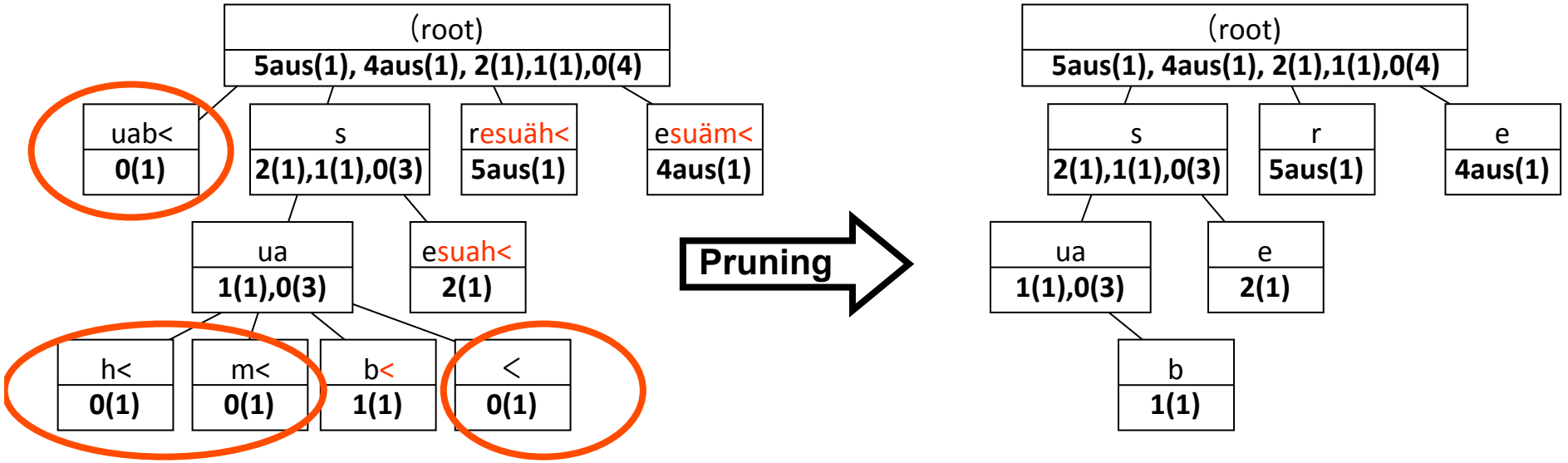
PRUNING TO CPT: MEMORY REDUCTION

Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

If the PT serves merely as classifier and not for storing word lists:

- class-redundant subtrees can be deleted.
- strings in the remaining leaves can be cut to length 1

A pruned PT is called **Compact Patricia Trie (CPT)**



SUMMARY OF CPTS

Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

Properties:

- Fully reproducing a training set, yet perform educated guesses
- compact data structure for string-based classification
- Trained from training data: word+class
- Insertion and deletion possible (before pruning) without reorganization

Applications:

- morphological classification
- base form reduction
- compound noun decomposition: `Bauhaus` -> `Bau - haus`
- context-independent word class assignment, e.g. Noun, Verb, etc.
- autocomplete / type-ahead

DIRECTED ACYCLIC WORD GRAPH (DAWG)

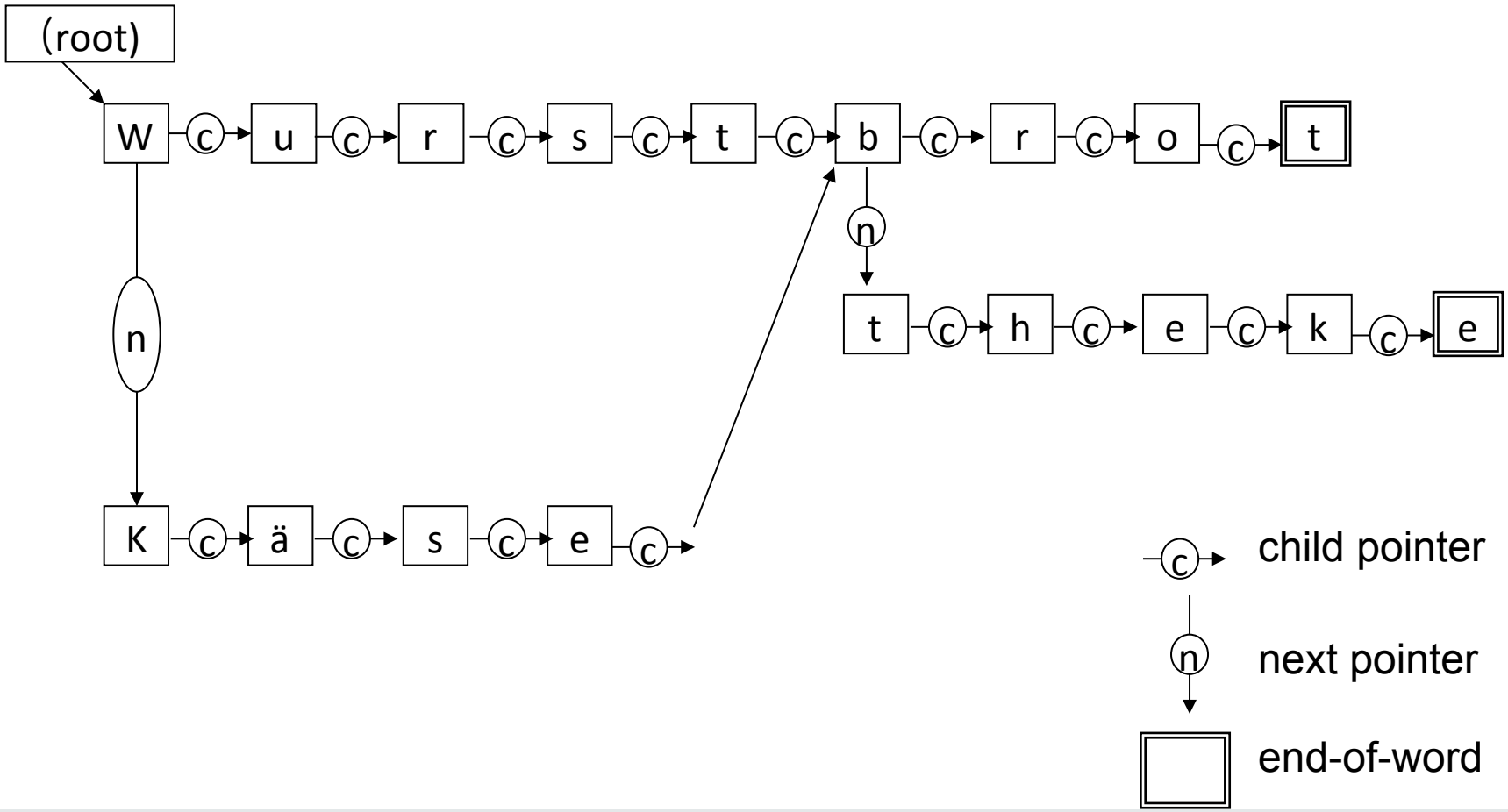
Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

- A DAWG is a directed acyclic graph, built from the letters of words
- In a DAWG, similar prefixes AND similar suffixes are used for compression
- There are two kinds of edges:
  - child pointer
  - next pointer
- DAWGs merely store the existence of a word and cannot be used for classification tasks
- DAWGs need in average less memory than CPTs

EXAMPLE DAWG

Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

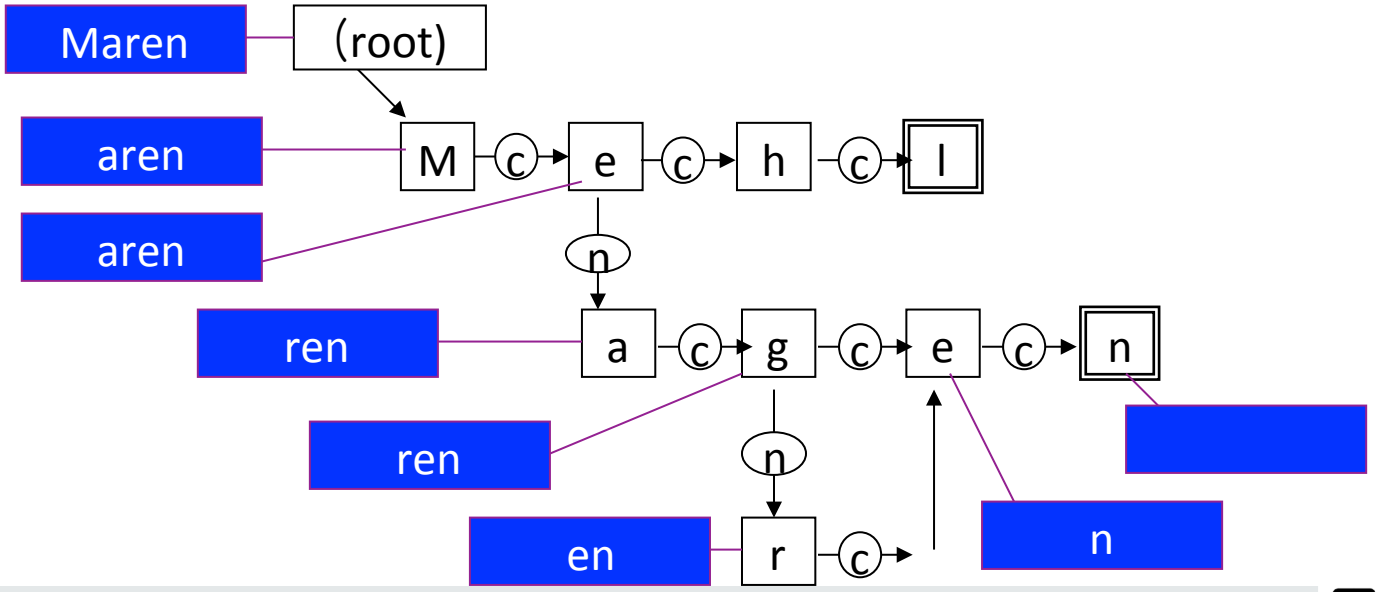
example for `Wurst`*`brot`*`, Wurst`*`theke`*`, Käse`*`brot`*`, Käse`*`theke`*



SEARCHING A DAWG

Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

- if there is a path from the root to a node marked as EOW, the word is contained in the DAWG
- child pointer eats the search word‘s characters up
- next pointer serves for choosing alternatives.

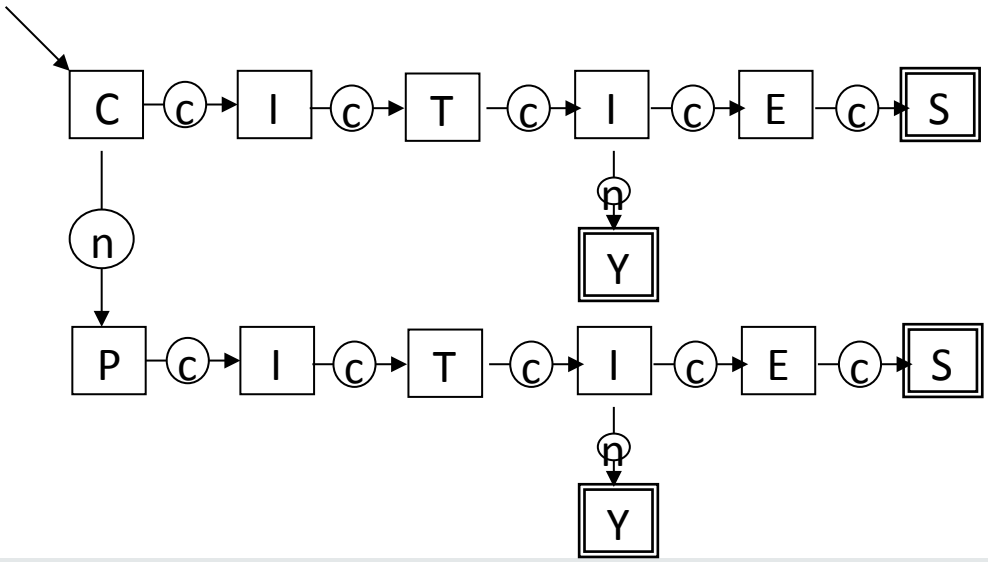


CONSTRUCTION OF DAWGS

Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

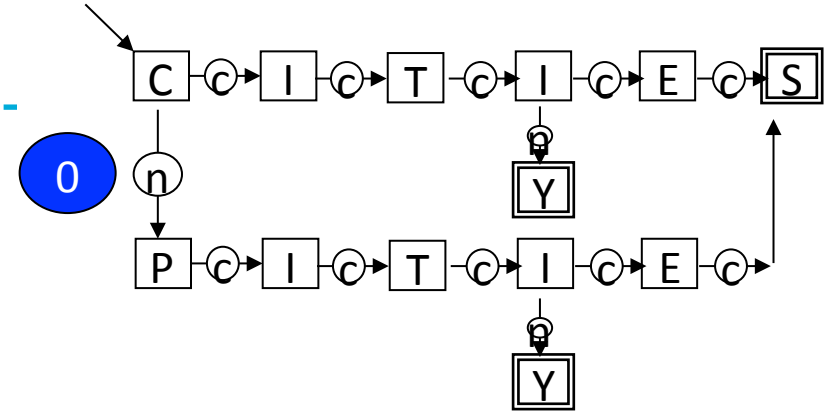
1. Construction of a tree with child-next pointers (isomorphic to trie)
   - Search until no more match
   - Insert rest of search word
2. Combine similar subtrees, starting with subtree depth 0 and increase depth.

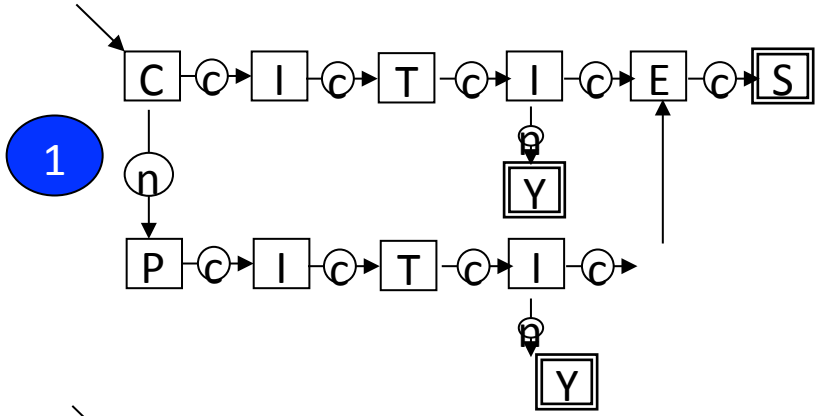
Example for `CITIES, CITY, PITIES, PITY`

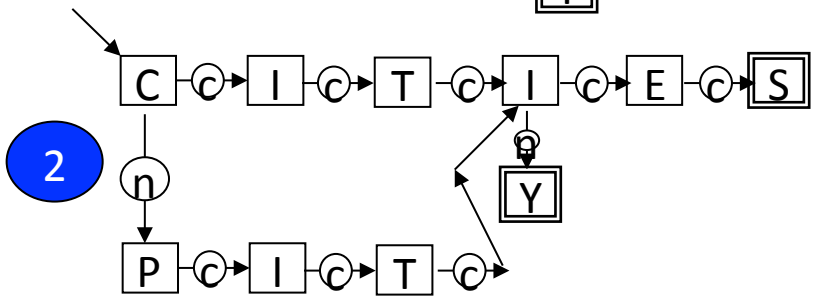


After step 1 we only used common prefixes for compression

CONSTRUCTION OF DAWGS II





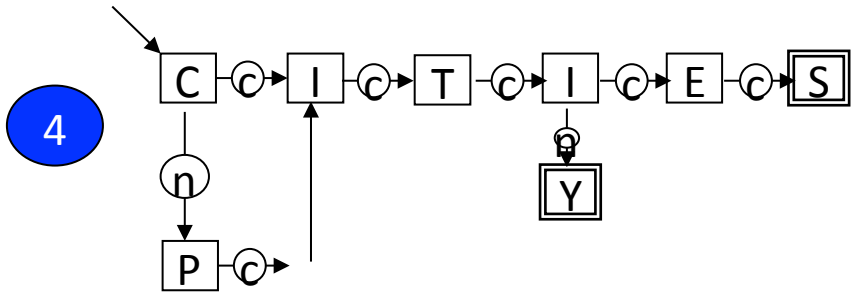


At first combine leaves (depth 0) with identical content

Note that only subtrees under child-pointers are combined: therefore the Y's stay separated.

Then combine identical subtrees with depth 1

Iterate for increasing depth, until nothing can be combined any more.



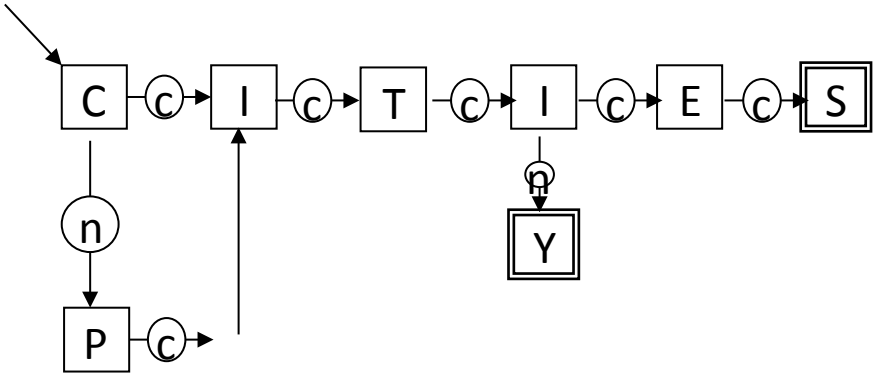
FILE FORMAT DAWG

Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

Every node is represented by 4 bytes (ASCII-DAWG)

- 2 bytes for child pointer
- 1 byte for flags: "1"- for End-of-word, "2" of End-of-List
- 1 byte for character

nr	child		flg	chr	
00	00	00	03	00	(root)
01	00	03	00	43	C
02	00	03	02	50	P
03	00	04	02	49	I
04	00	05	02	54	T
05	00	07	00	49	I
06	00	00	03	59	Y
07	00	08	02	45	E
08	00	00	03	53	S



next pointer: next line, if not end-of-list or some child pointer points to the next line

SUMMARY ON DAWGS

Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

- DAWGs are a very compact form to store word lists, e.g. for dictionaries
- DAWGs are efficient in deciding whether a word is in the word list or not
- DAWGs are not suitable for classification on string sequences

Applications

- Scrabble
- edit distance
- indexing
- auto-completion

SIGN UP IN MOODLE – ANYONE MISSING?

Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

- https://lernen.min.uni-hamburg.de/enrol/instances.php?id=73
- The required enrolment key is: **SMOLT2019**

- Manning, C. D. and Schütze, H. (1999): Foundations of Statistical Natural Language Processing. MIT Press: Cambridge, Massachusetts. Chapters 2.1, 2.2, 6.
- Bengio, Y., Ducharme, R., Vincent, P., Jauvin, C. (2013): A Neural Probabilistic Language Model. Journal of Machine Learning Research 3 (2003):1137–1155
- Mikolov, T., Karafiát, M., Burget, L., Če… ký, J., Khudan… S. (2010) …rrent neural network based language model. Proceedings of Inte… …1045-1048

coming up next

Entropy, Perplexity, Maximum Likelih… …-off, Neural LMs

LANGUAGE MODELS