Computational Linguistics at CIS - from basic research to applications in Digital Humanities

Stefan Gerdjikov, Klaus U. Schulz

CIS - LMU Munich

Including joint work with Stoyan Mihov and Petar Mitankin (Sofia)

Survey of talk

Excursion 1

Fast approximate search in large lexica

- Interactive postcorrection of OCRed historical texts
- Advanced text index structures

Excursion 2

Advanced text index structures

- Modernization/normalization of historical language
- Novel ways of querying and exploring corpora and text collections

Loose ends...

Ontologies & finding topics in texts (TopicZoom)
Wittgenstein search (Max Hadersbeck)
Latin (Uwe Springmann, Helmut Schmidt)

Excursion 1

Fast approximate search in large lexica

Interactive postcorrection of
OCRed historical texts

Advanced text index structures

Approximate search in lexica

Intuition

Given a large lexicon L and possibly erroneous input tokens, for each input token V efficiently find the „most similar“ entries W of L.

Levenshtein distance d

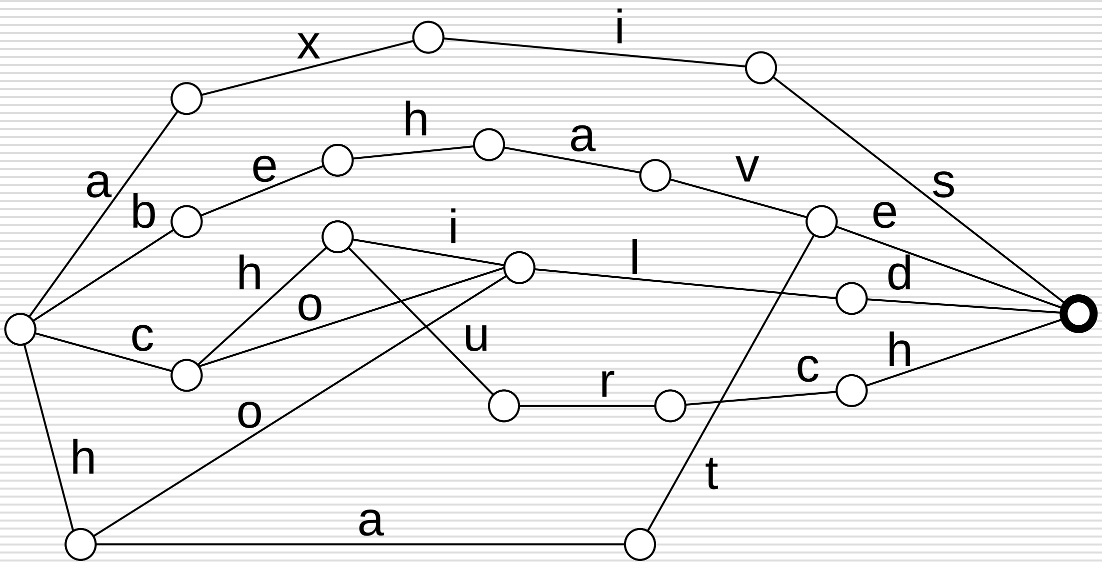
The Levenshtein distance between two words V, W is the minimal number of edit operations (deletion, insertion of a symbol or substitution of one symbol by another) needed to rewrite V into W.

Formalized problem

Given a lexicon L, an input word („pattern“) V and a bound b (=0,1,2,..), compute all entries W of L such such $d(V,W) \leq b$.

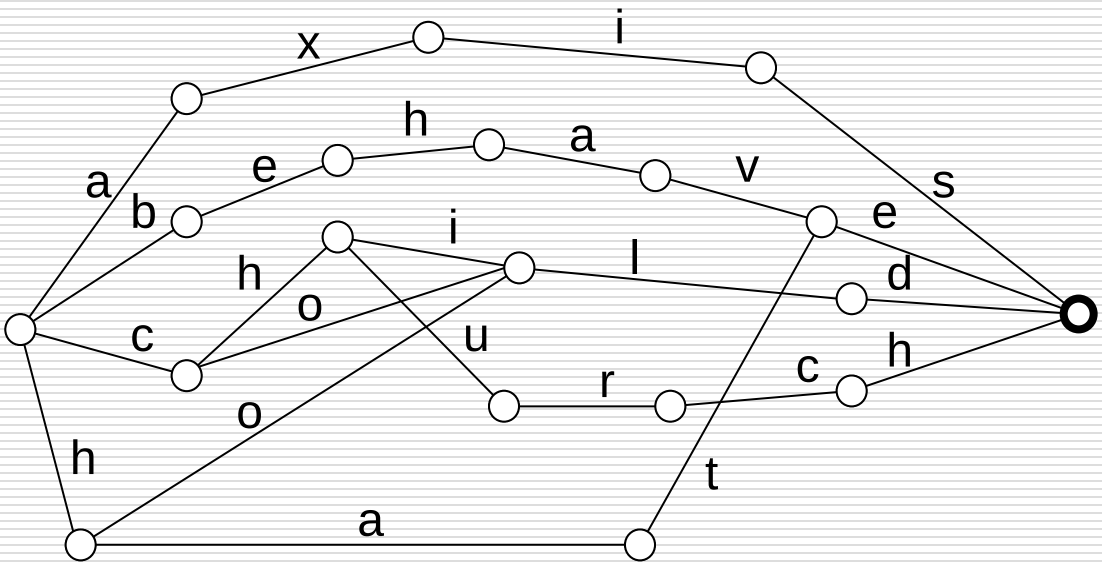
Lexicon representation as minimal deterministic finite-state automaton

Example lexicon: axis,behave,child,church,cold,hate,hold



Lexicon representation as minimal deterministic finite-state automaton

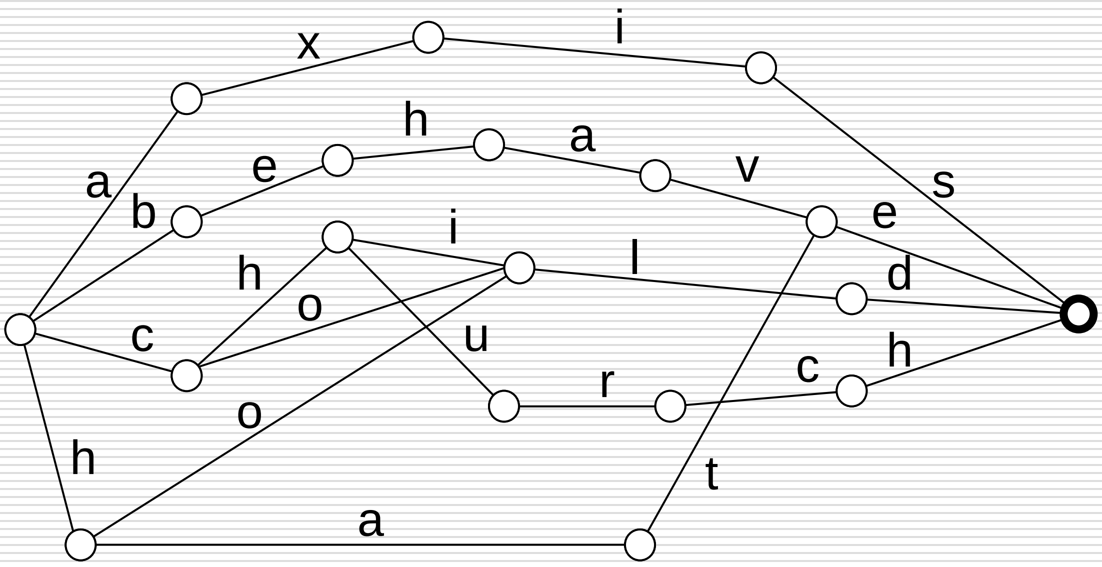
Example lexicon: axis,behave,child,church,cold,hate,hold



- Fast access

Lexicon representation as minimal deterministic finite-state automaton

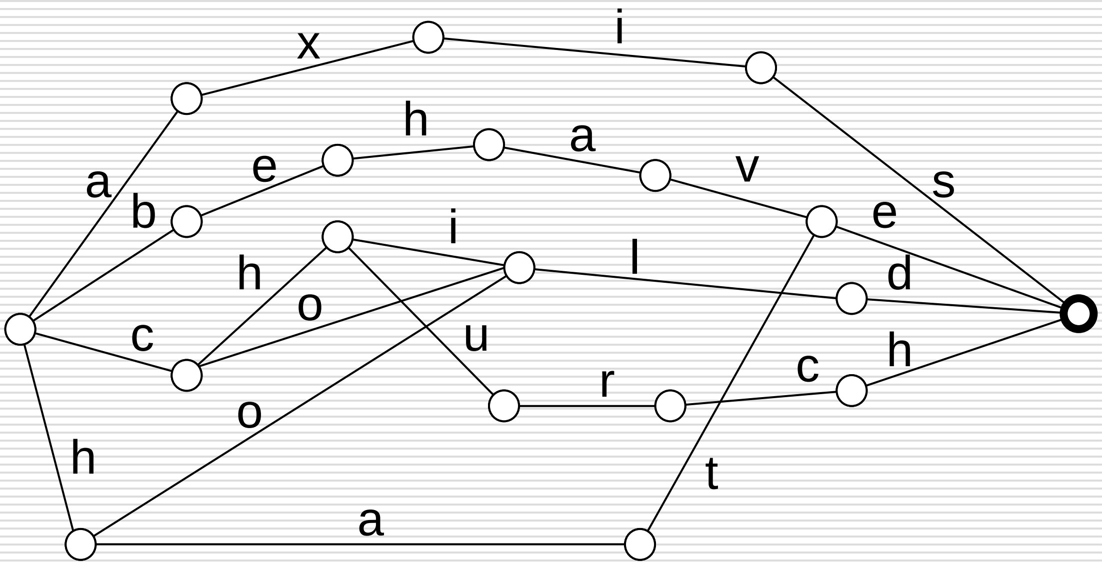
Example lexicon: axis,behave,child,church,cold,hate,hold



- Fast access
- Size: each prefix of a lexicon word only represented once. Similar sharing of suffixes!

Lexicon representation as minimal deterministic finite-state automaton

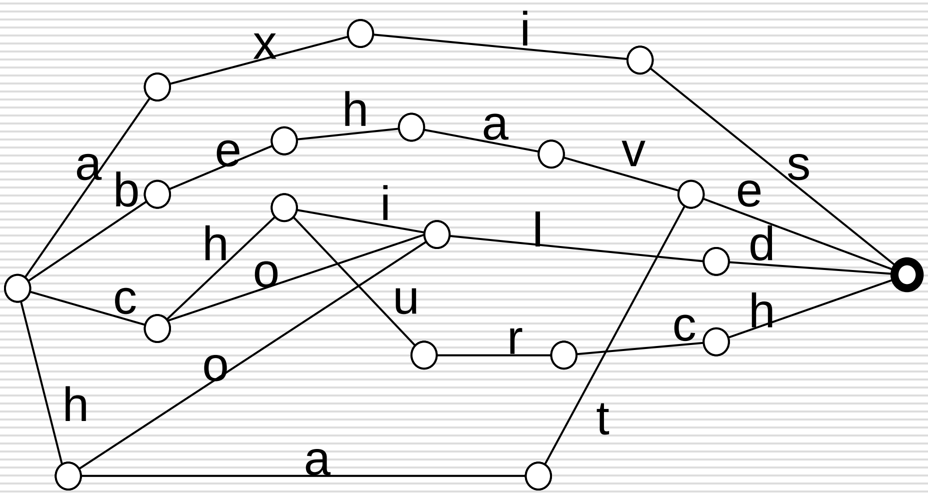
Example lexicon: axis,behave,child,church,cold,hate,hold



- Fast access
- Size: each prefix of a lexicon word only represented once. Similar sharing of suffixes!
- Using the lexicon: for many „processing tasks“ strings occurring in several words only processed once.

Approximate search in lexicon using dynamic programming (Oflazer)

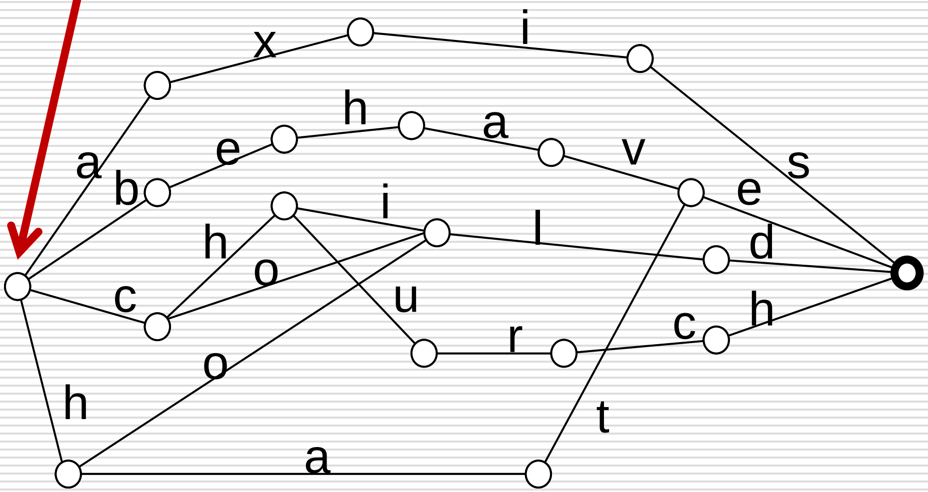
Search for „chiid“, maximal Levenshtein distance 1



Approximate search in lexicon using dynamic programming (Oflazer)

Search for „chiid“, maximal Levenshtein distance 1

„Complete“ traversal of lexicon automaton

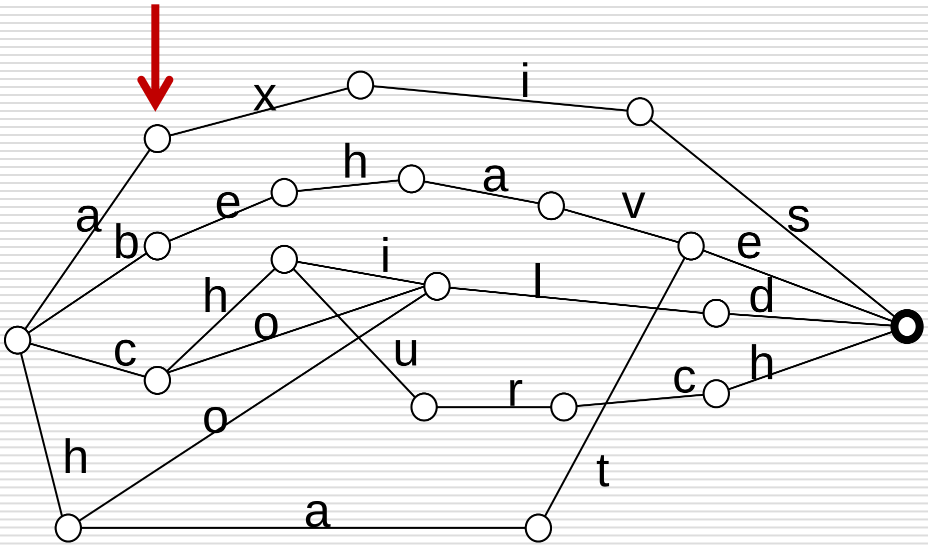


	0
c	1
h	2
i	3
i	4
d	5

Parallel control

Oflazer approximate search

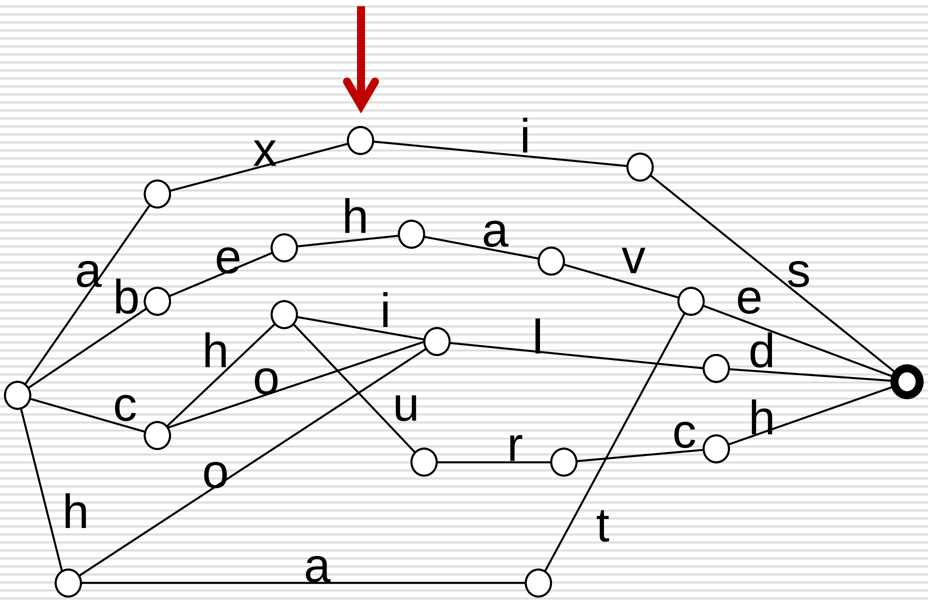
Search for „chiid“, maximal Levenshtein distance 1



		a
	0	1
c	1	1
h	2	2
i	3	3
i	4	4
d	5	5

Oflazer approximate search

Search for „chiid“, maximal Levenshtein distance 1

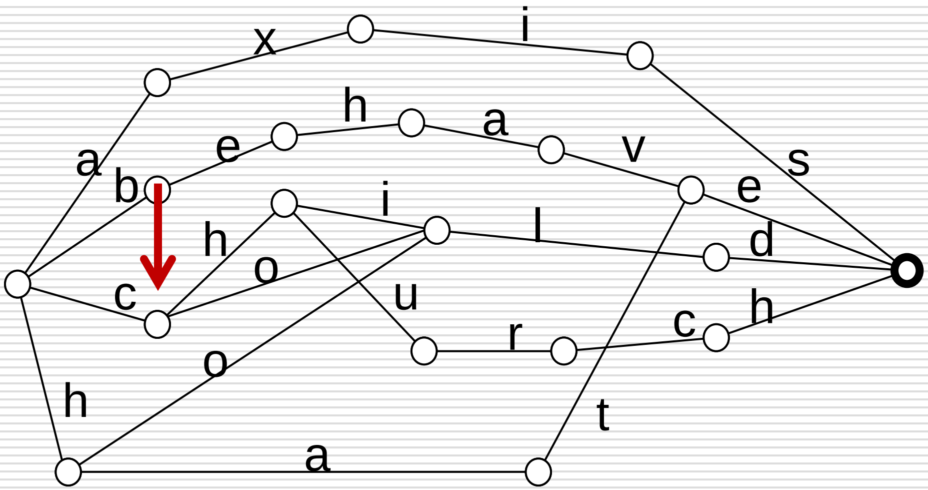


		a	x
	0	1	2
c	1	1	2
h	2	2	2
i	3	3	3
i	4	4	4
d	5	5	5

Blind path, backtracking!

Oflazer approximate search

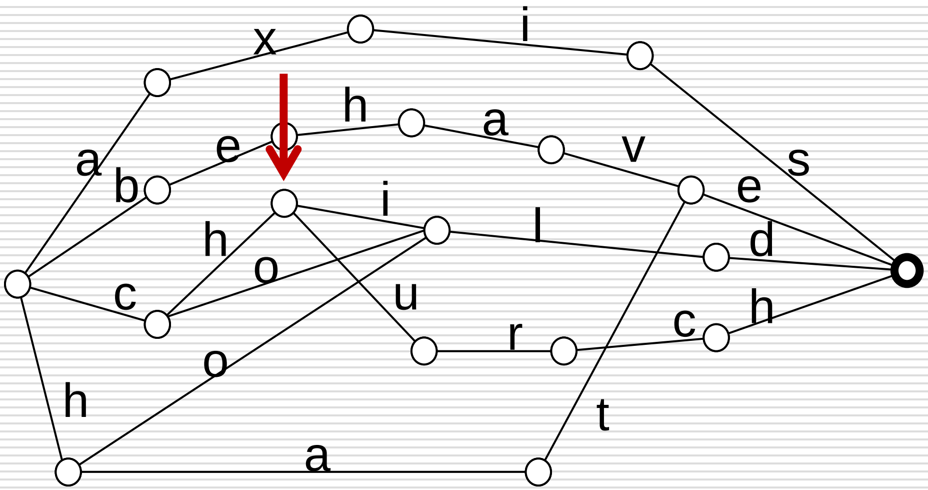
Search for „chiid“, maximal Levenshtein distance 1



		c
	0	1
c	1	0
h	2	1
i	3	2
i	4	3
d	5	4

Oflazer approximate search

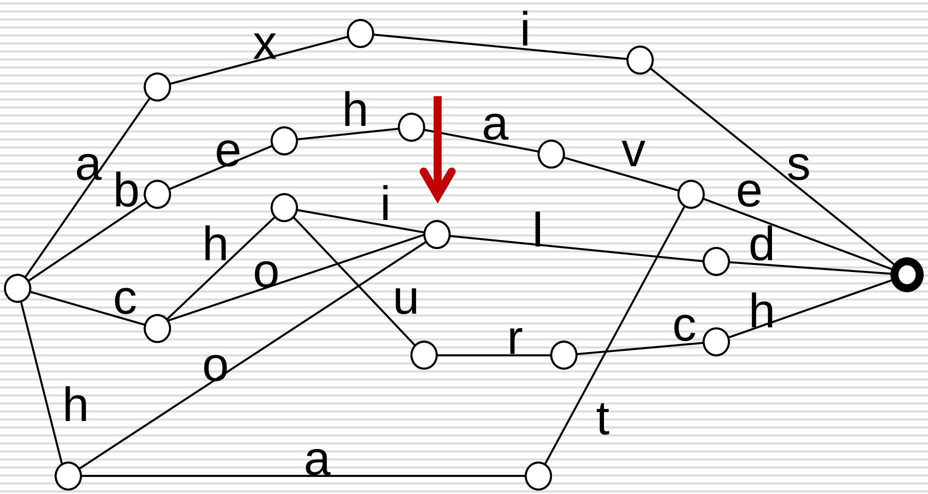
Search for „chiid“, maximal Levenshtein distance 1



		c	h
	0	1	2
c	1	0	1
h	2	1	0
i	3	2	1
i	4	3	2
d	5	4	3

Oflazer approximate search

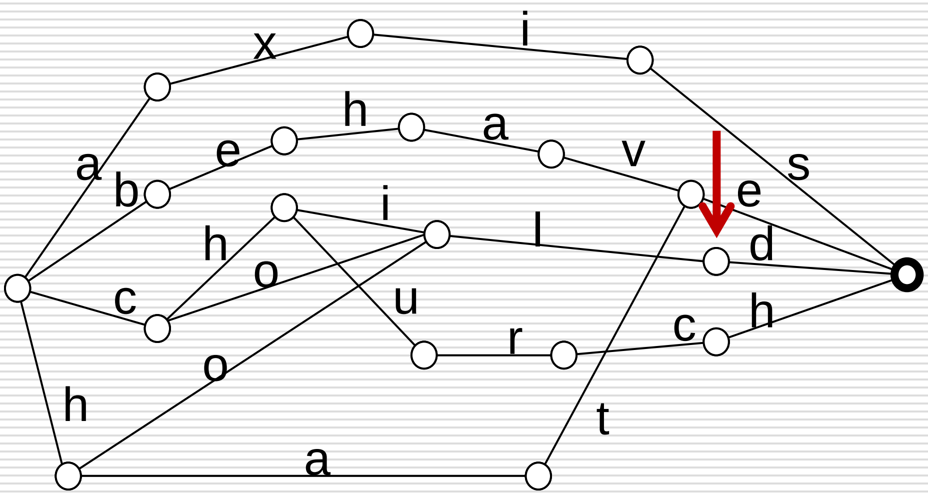
Search for „chiid“, maximal Levenshtein distance 1



		c	h	i
	0	1	2	3
c	1	0	1	2
h	2	1	0	1
i	3	2	1	0
i	4	3	2	1
d	5	4	3	2

Oflazer approximate search

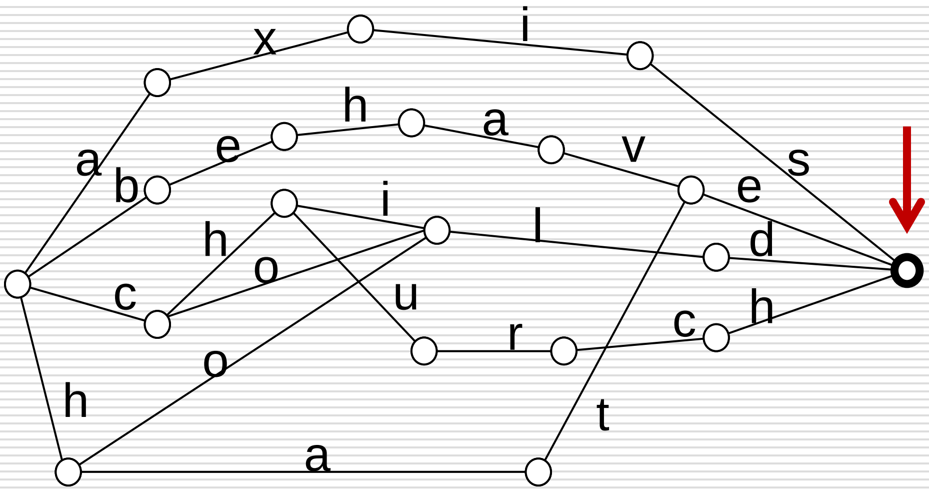
Search for „chiid“, maximal Levenshtein distance 1



		c	h	i	l
	0	1	2	3	4
c	1	0	1	2	3
h	2	1	0	1	2
i	3	2	1	0	1
i	4	3	2	1	1
d	5	4	3	2	2

Oflazer approximate search

Search for „chiid“, maximal Levenshtein distance 1



		c	h	i	l	d
	0	1	2	3	4	5
c	1	0	1	2	3	4
h	2	1	0	1	2	3
i	3	2	1	0	1	2
i	4	3	2	1	1	2
d	5	4	3	2	2	1

match: child

Oflazer approximate search

For given pattern,

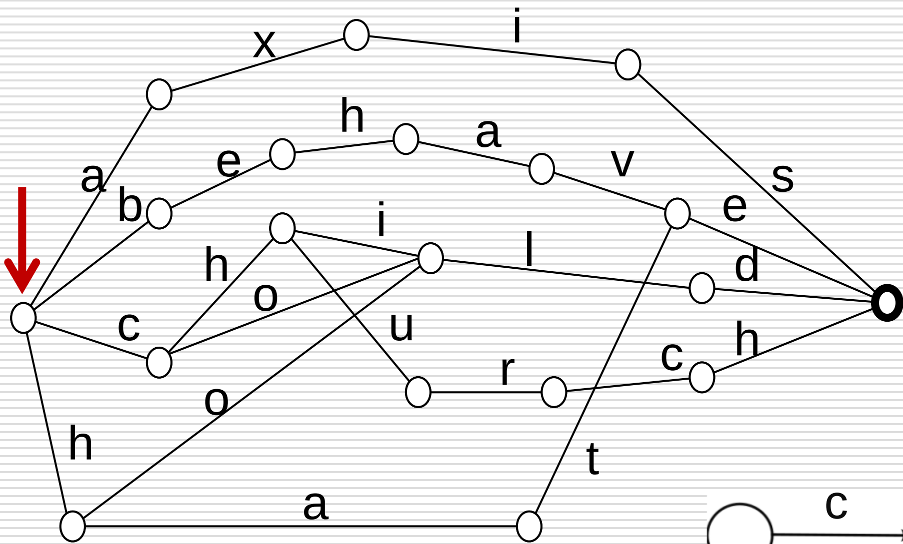
- „blind“ paths in automaton recognized as early as possible,
- only „small“ part of the automaton visited

But

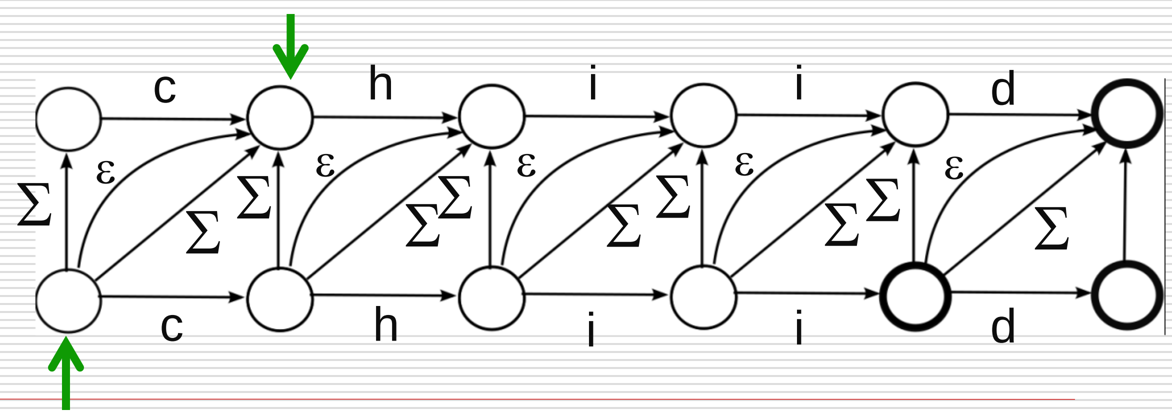
- Computing matrix columns at each step expensive!

Approximate search using non-deterministic Levenshtein automata

Search for „chiid“, maximal Levenshtein distance 1

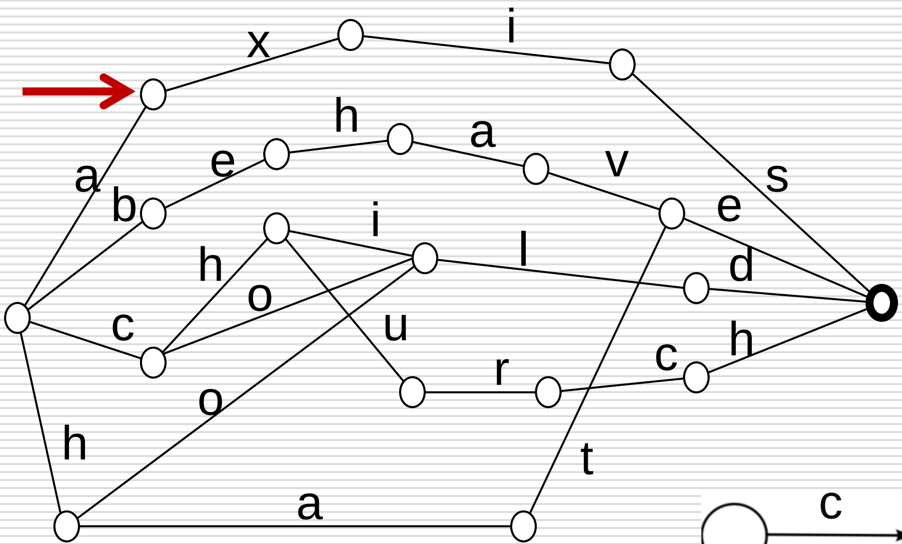


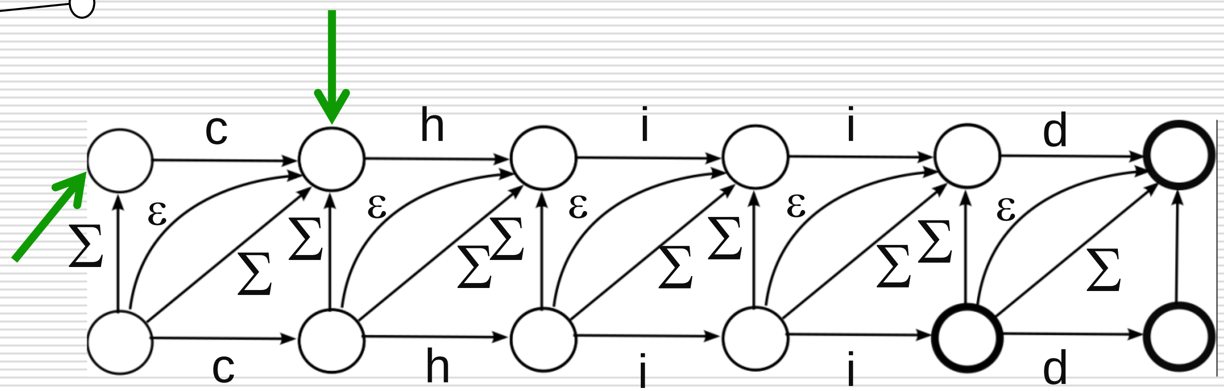
active states



Approximate search using non-deterministic Levenshtein automata

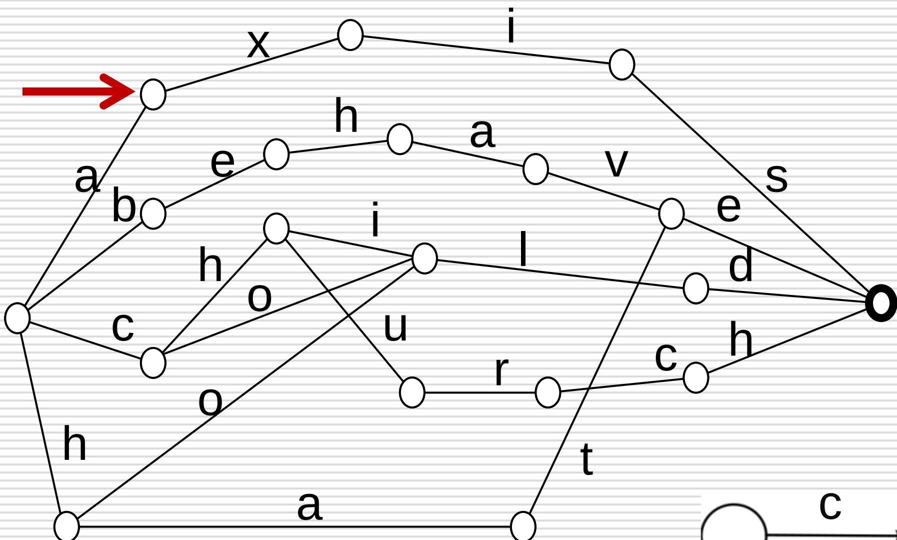
Search for „chiid“, maximal Levenshtein distance 1



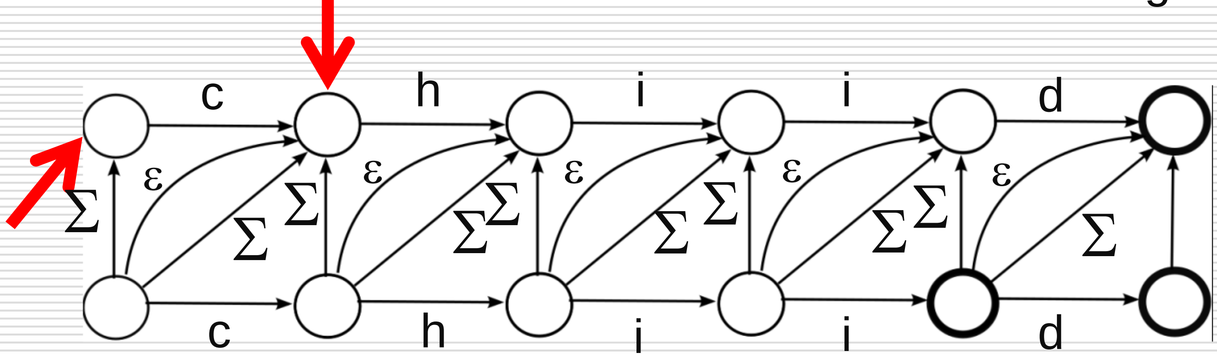


Approximate search using non-deterministic Levenshtein automata

Search for „chiid“, maximal Levenshtein distance 1

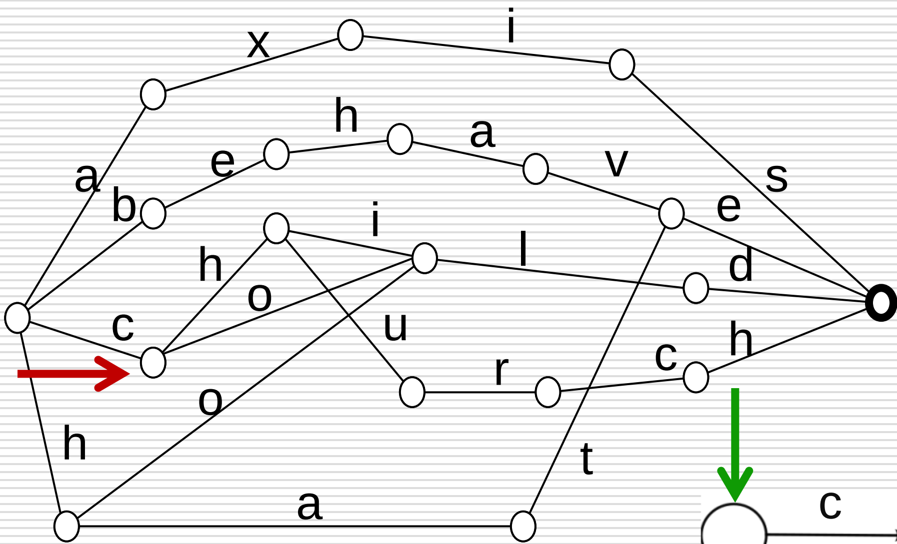


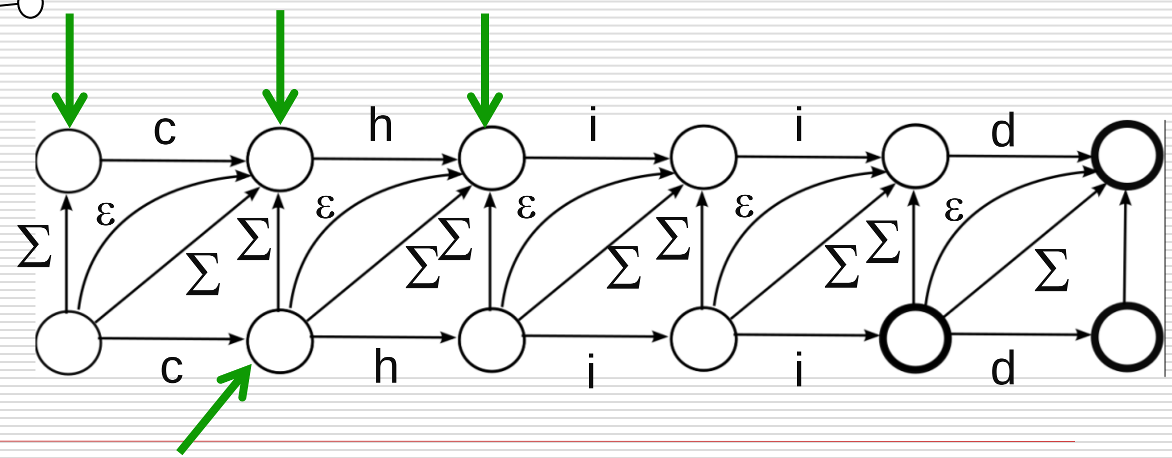
Transition with x fails - backtracking



Approximate search using non-deterministic Levenshtein automata

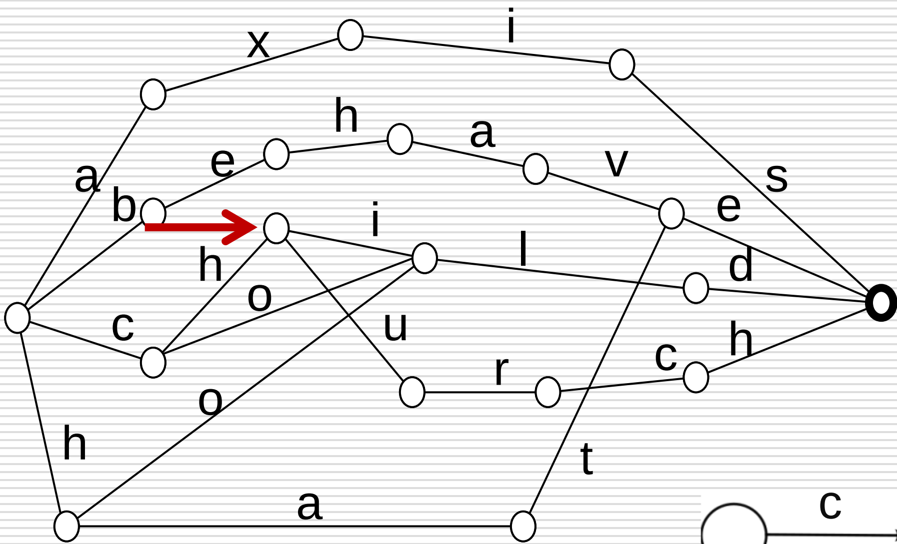
Search for „chiid“, maximal Levenshtein distance 1

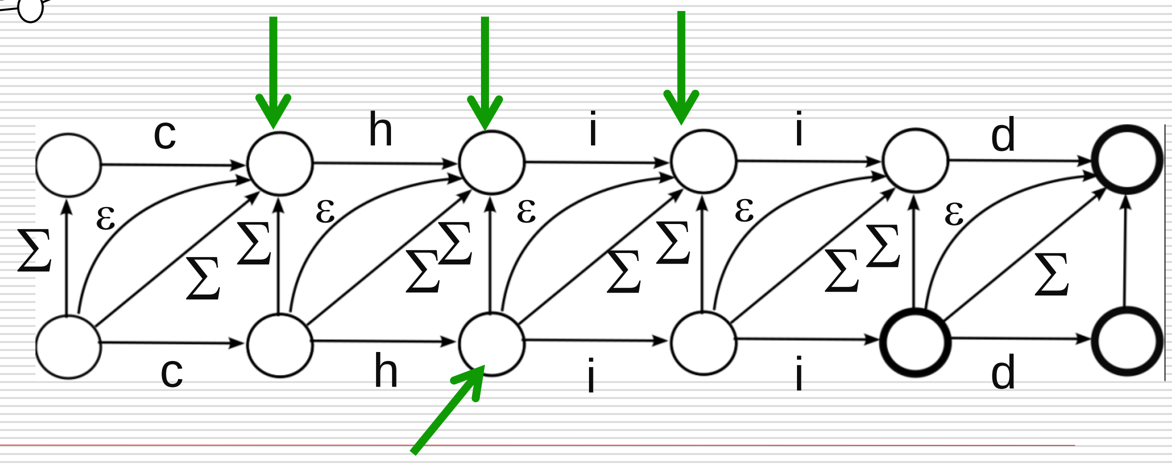




Approximate search using non-deterministic Levenshtein automata

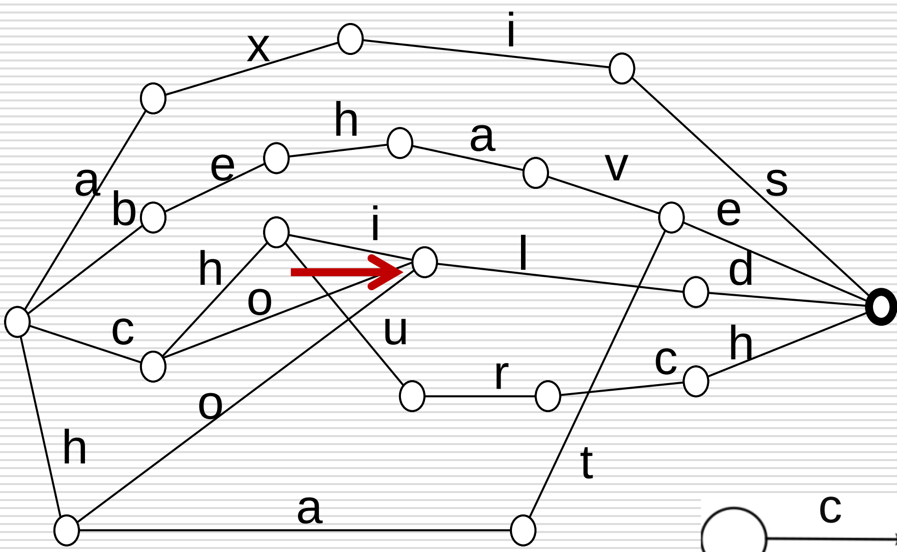
Search for „chiid“, maximal Levenshtein distance 1

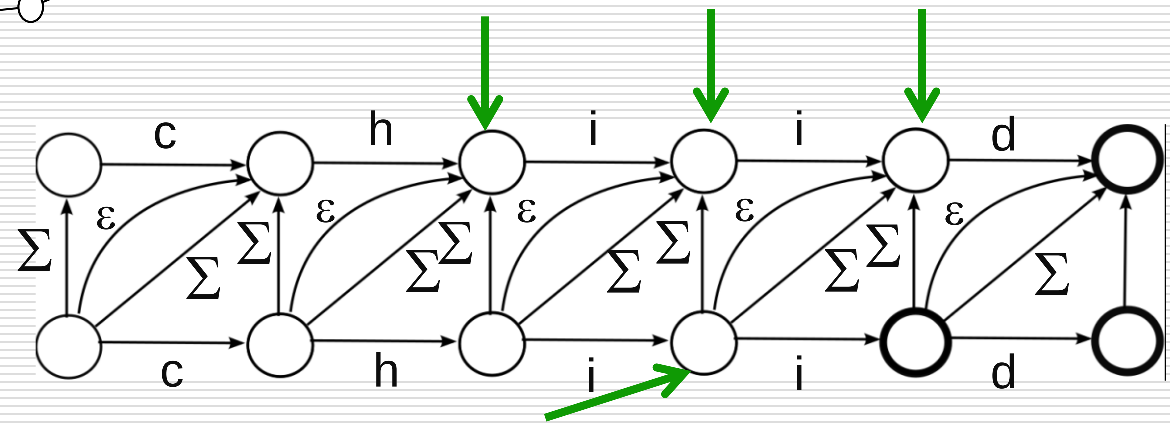




Approximate search using non-deterministic Levenshtein automata

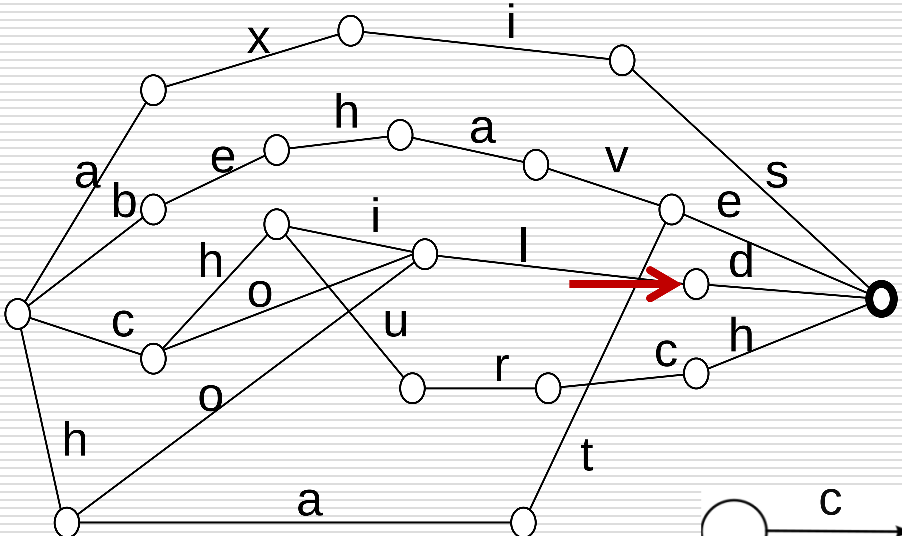
Search for „chiid“, maximal Levenshtein distance 1

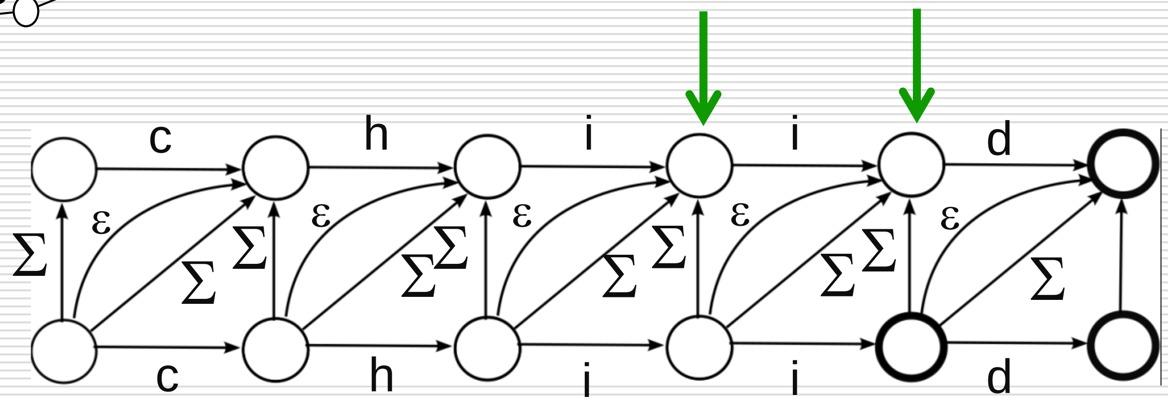




Approximate search using non-deterministic Levenshtein automata

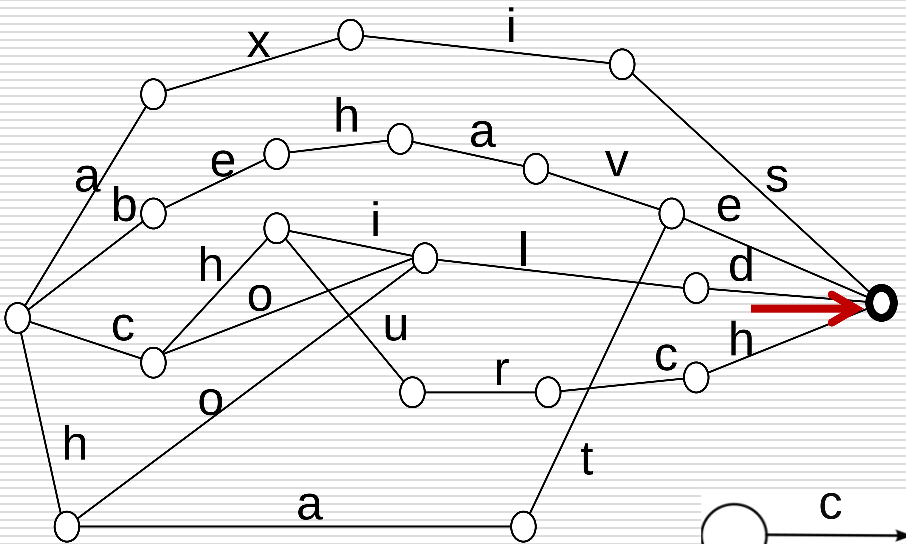
Search for „chiid“, maximal Levenshtein distance 1



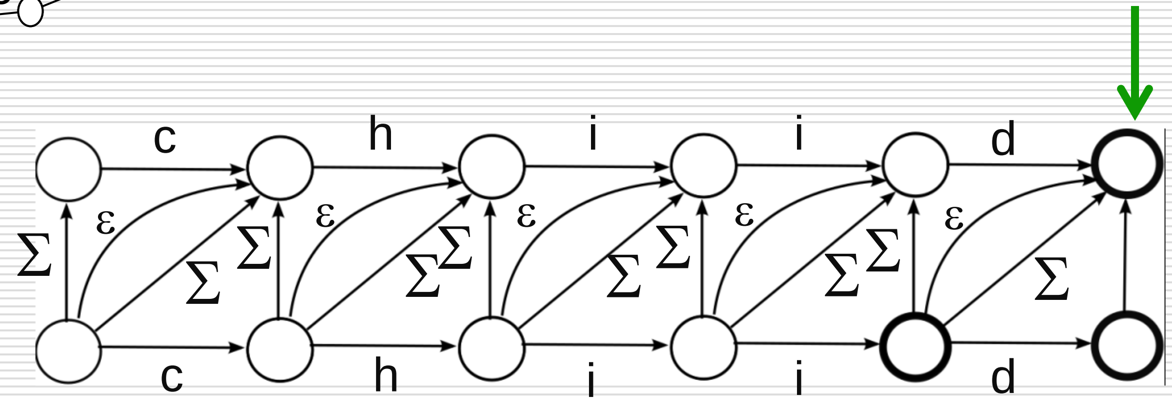


Approximate search using non-deterministic Levenshtein automata

Search for „chiid“, maximal Levenshtein distance 1



match: child



Approximate search using non-deterministic Levenshtein automata

Same strengths

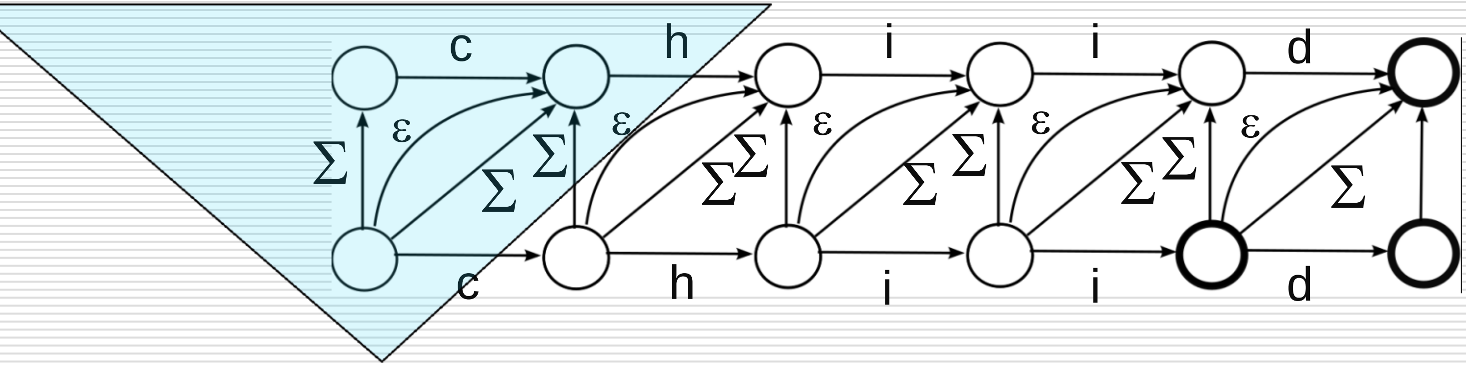
But similar weaknesses:

- Specific automaton for each pattern,
- Propagation of active states computationally expensive.

Approximate search using non-deterministic Levenshtein automata

Observation 1

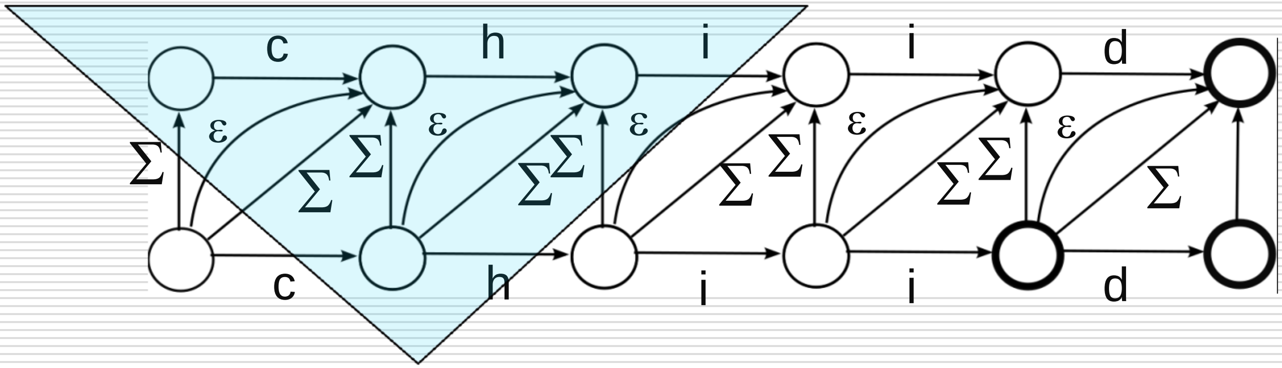
The set of active states after step n always restricted to a „triangular area“.



Approximate search using non-deterministic Levenshtein automata

Observation 1

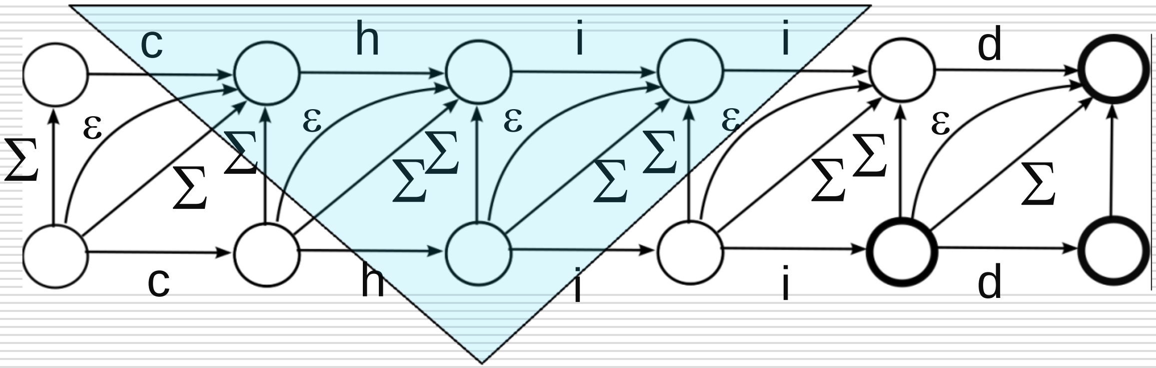
The set of active states after step n always restricted to a „triangular area“.



Approximate search using non-deterministic Levenshtein automata

Observation 1

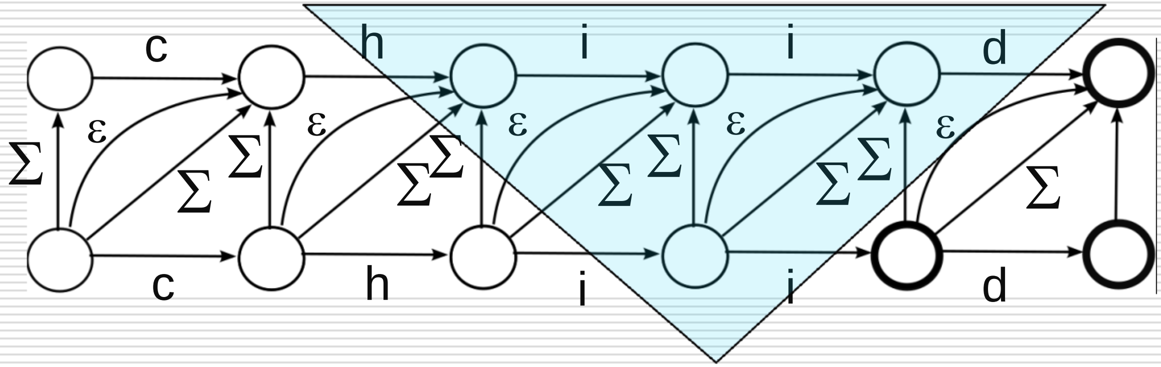
The set of active states after step n always restricted to a „triangular area".



Approximate search using non-deterministic Levenshtein automata

Observation 1

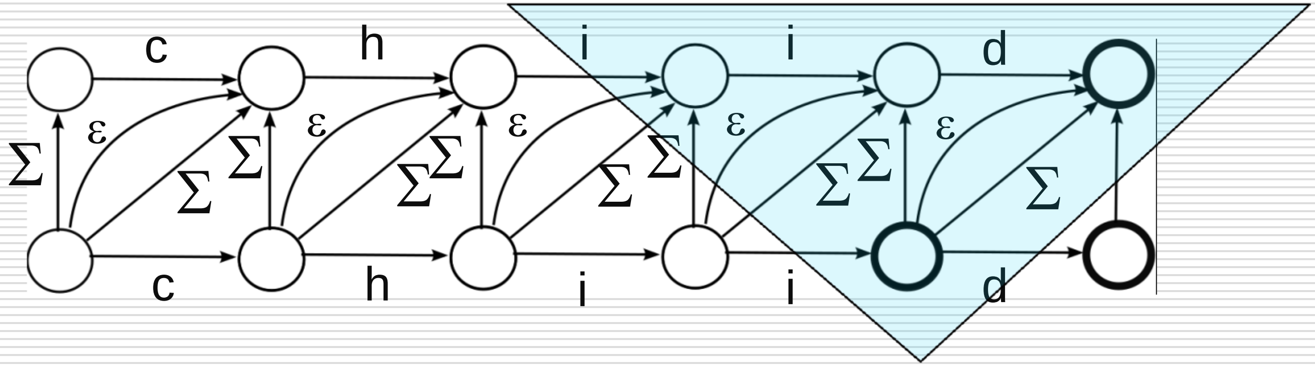
The set of active states after step n always restricted to a „triangular area“.



Approximate search using non-deterministic Levenshtein automata

Observation 1

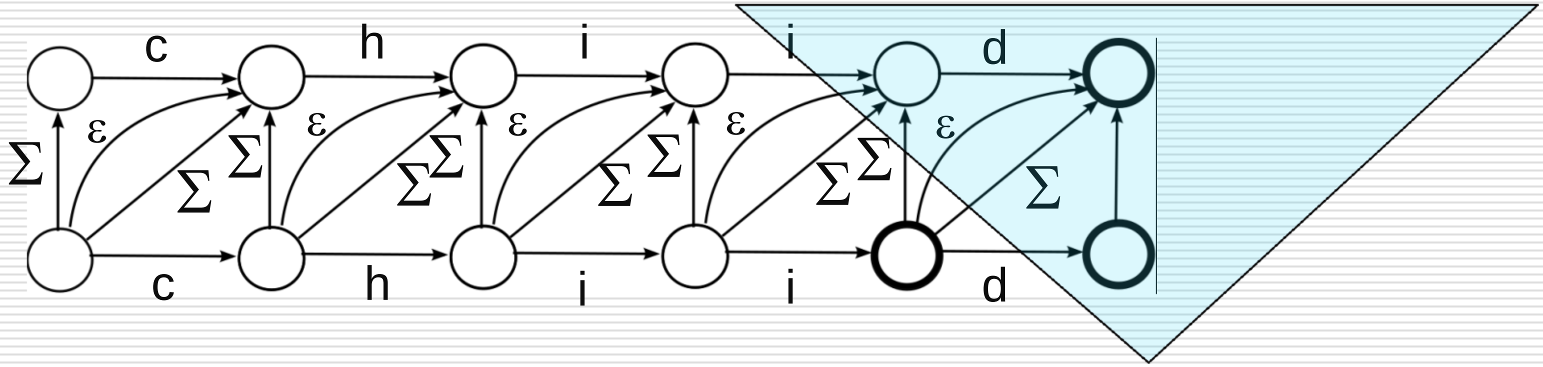
The set of active states after step n always restricted to a „triangular area“.



Approximate search using non-deterministic Levenshtein automata

Observation 1

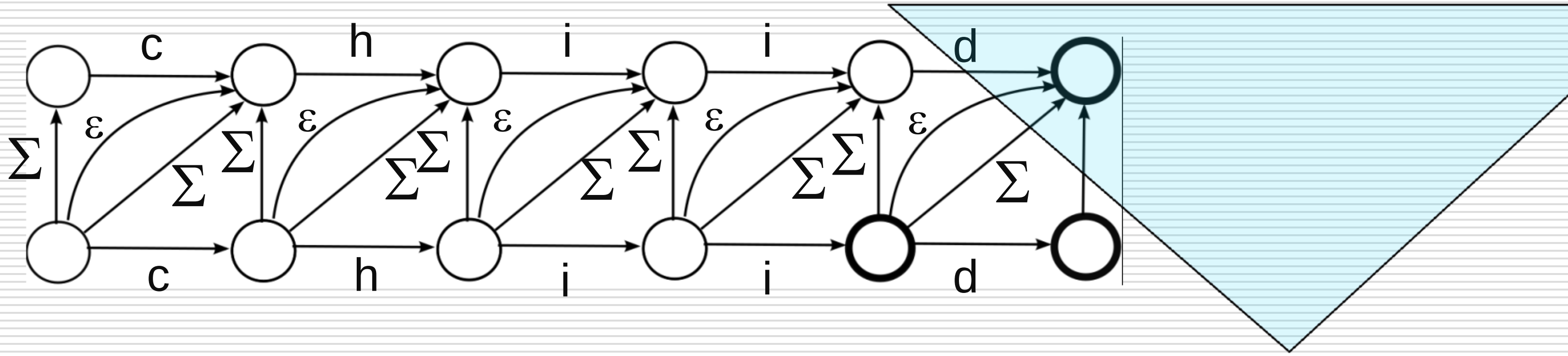
The set of active states after step n always restricted to a „triangular area“.



Approximate search using non-deterministic Levenshtein automata

Observation 1

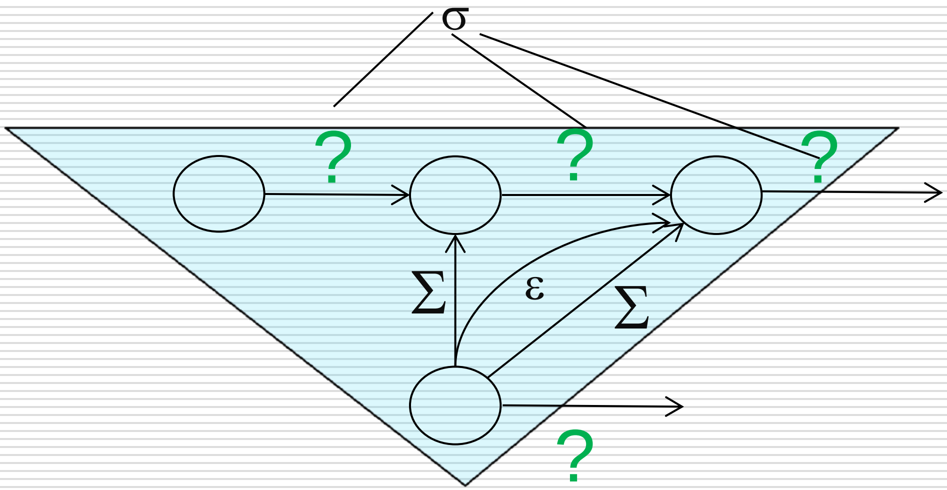
The set of active states after step n always restricted to a „triangular area“.



Approximate search using non-deterministic Levenshtein automata

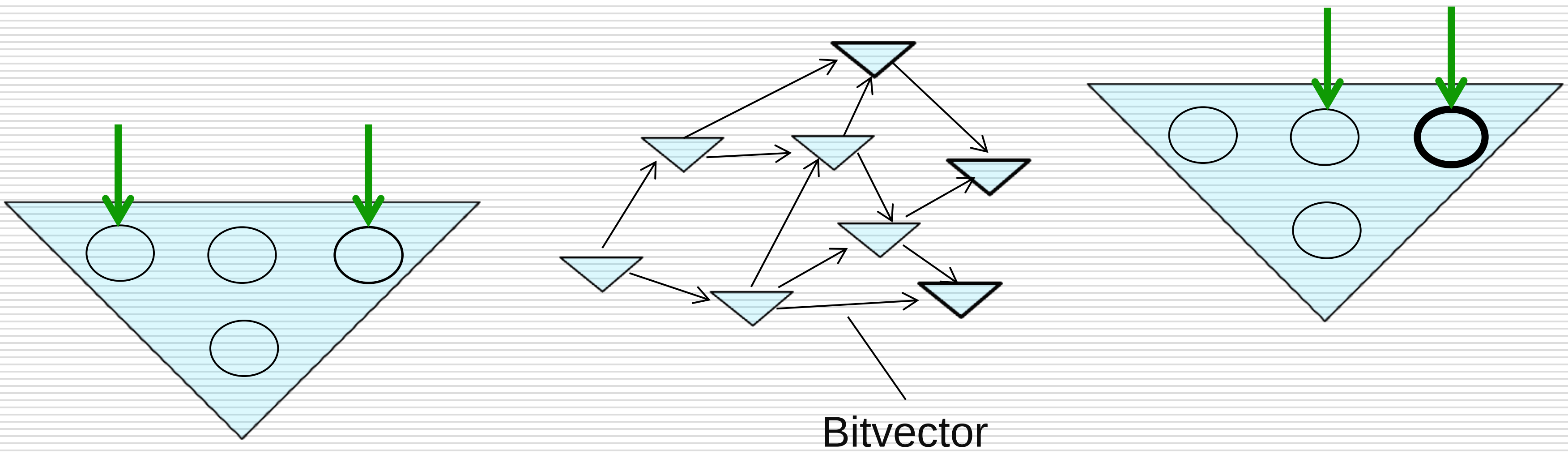
Observation 2

Given any subset of active states in the triangular area and an input symbol σ, the next set of active states only depends on the question: which labels of horizontal transitions in the triangular area are identical to σ?

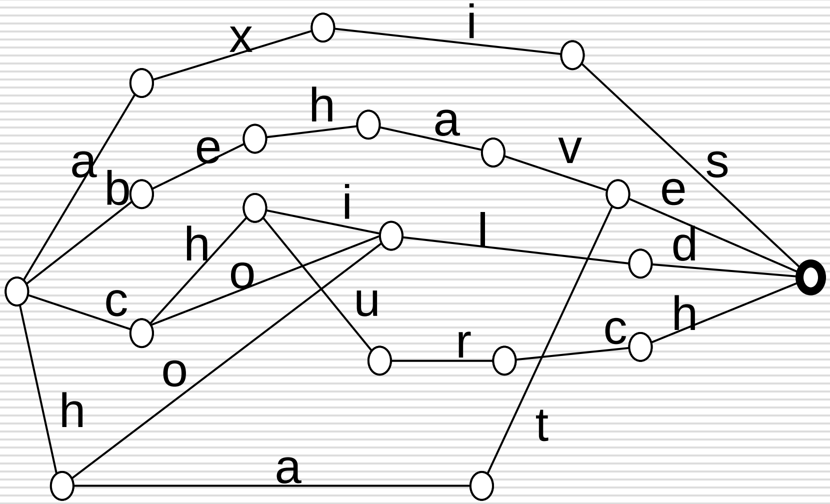


Determinization / Universal Automata

Use “abstracted“ triangular areas with distributions of active states (without any reference to an input string) as states of a universal deterministic automaton. Bitvectors used as input encode transition info.
„Universal deterministic Levenshtein automaton“ (here: bound b=1)



Example: input "chiid", b=1



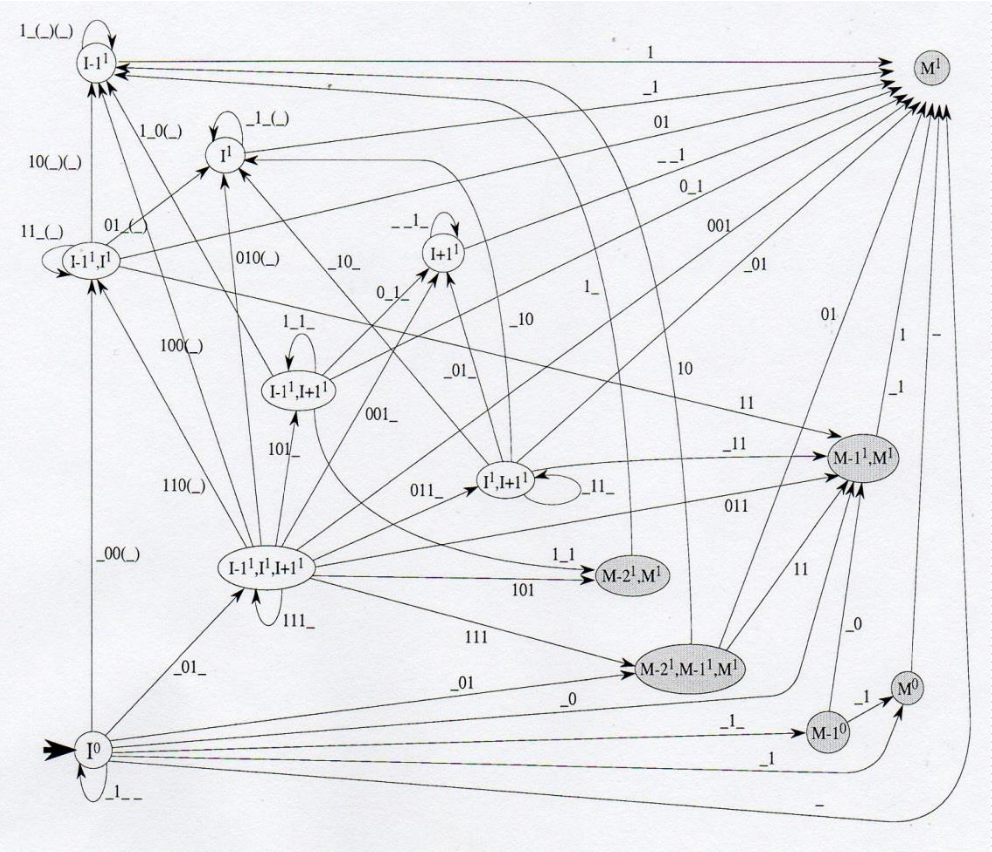
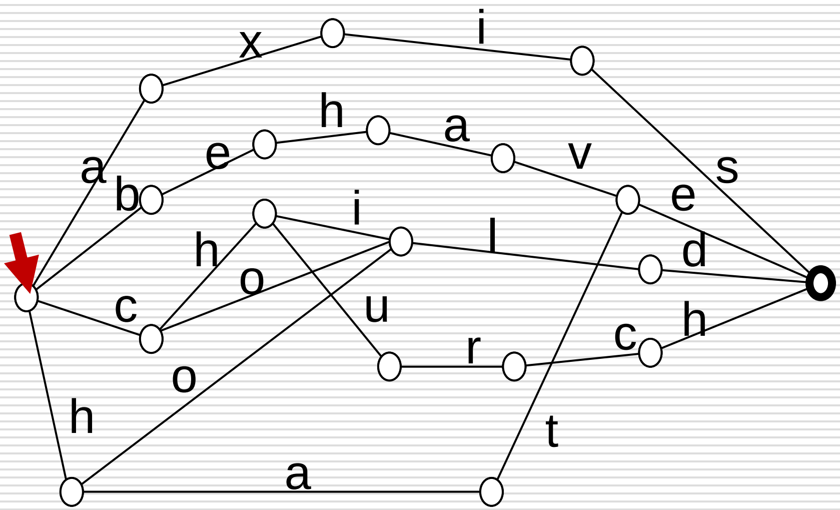


Figure 6
The universal deterministic Levenshtein automaton $A^{\forall}(1)$. See Example 4 for notation.

Example: input “chiid”, b=1



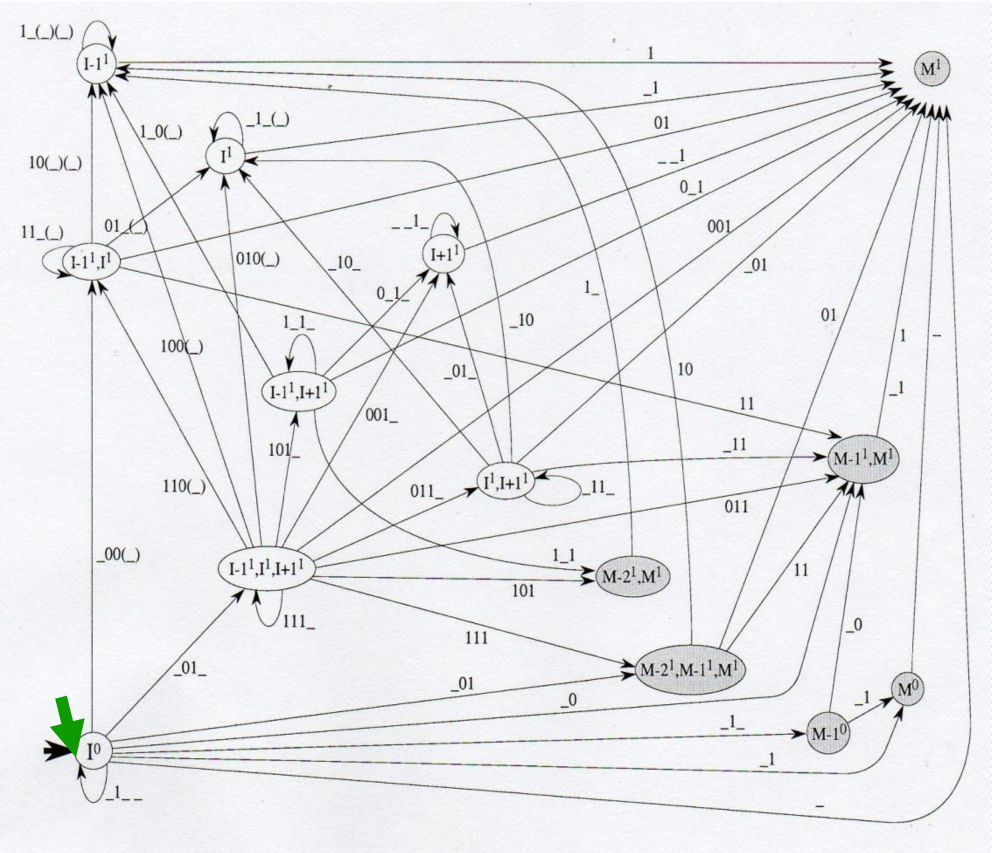
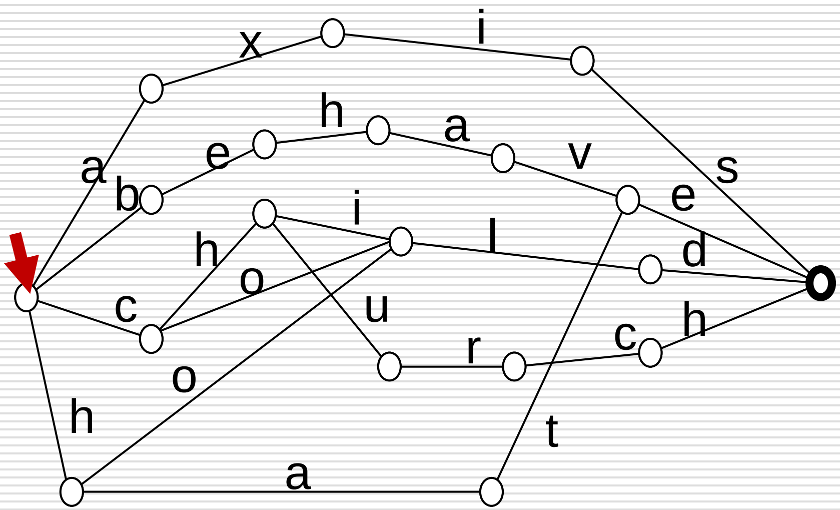


Figure 6
The universal deterministic Levenshtein automaton $A^{\forall}(1)$. See Example 4 for notation.

Example: input “chiid”, b=1



χ(a,$chi)=0000

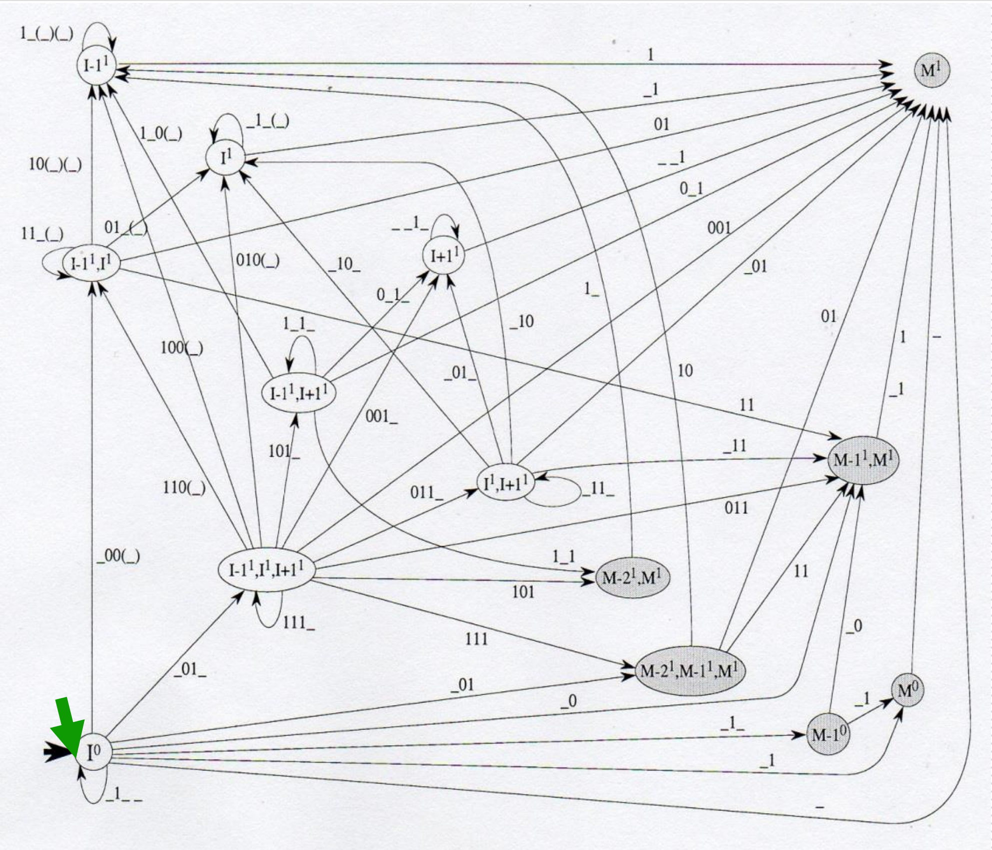
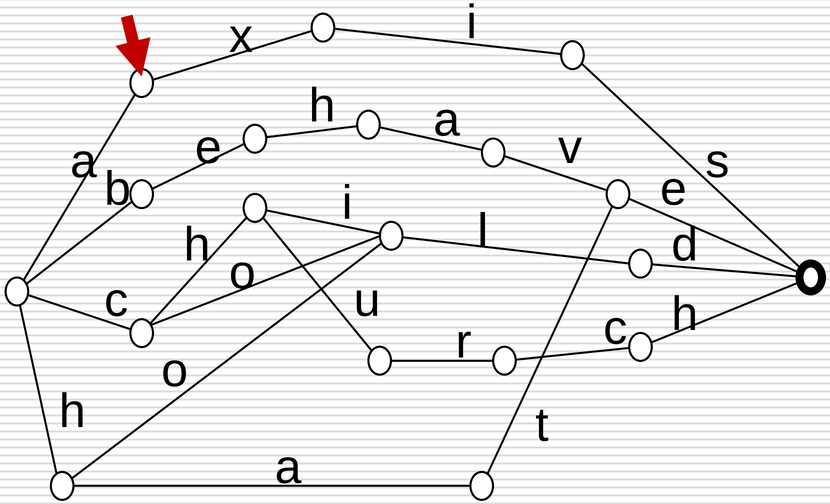


Figure 6
The universal deterministic Levenshtein automaton $A^{\forall}(1)$. See Example 4 for notation.

Example: input “chiid”, b=1



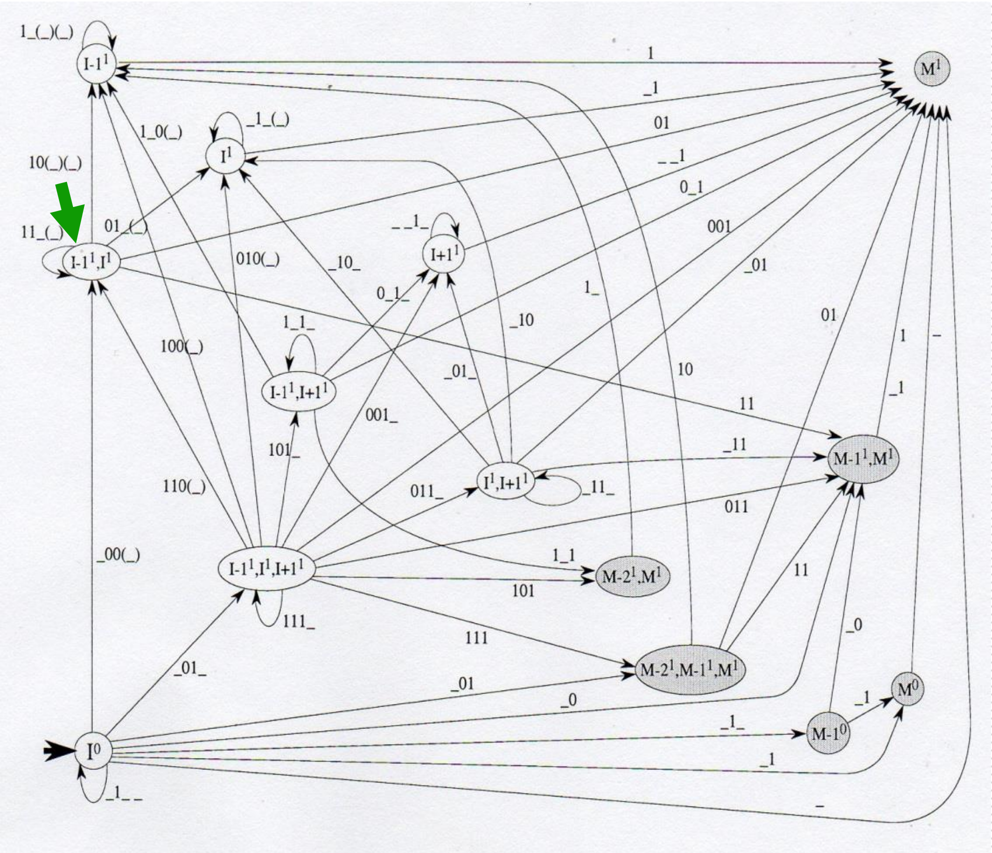
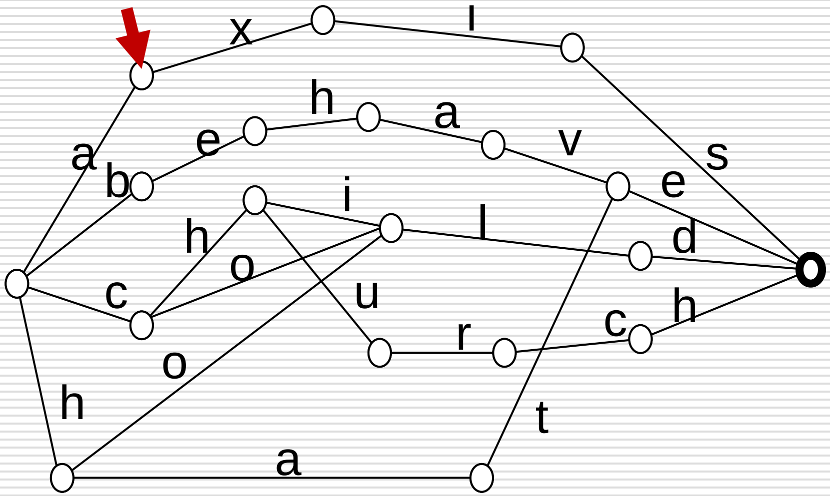


Figure 6
The universal deterministic Levenshtein automaton $A^{\forall}(1)$. See Example 4 for notation.

Example: input “chiid”, b=1



χ(x, chii)=0000

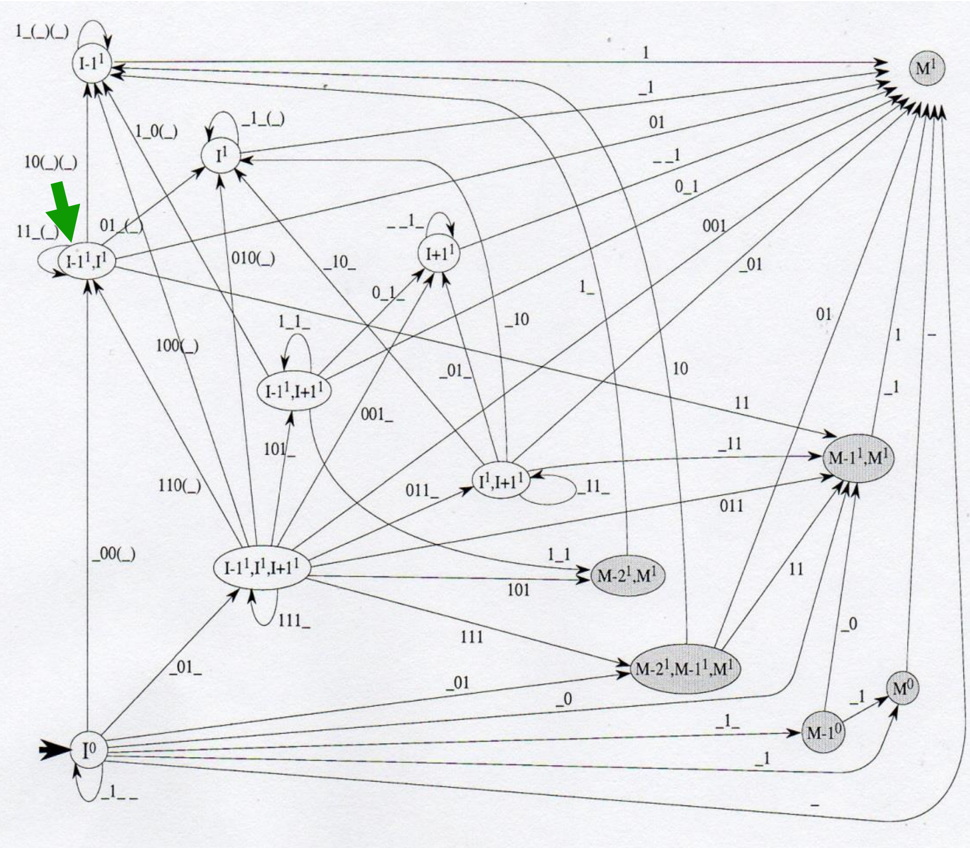
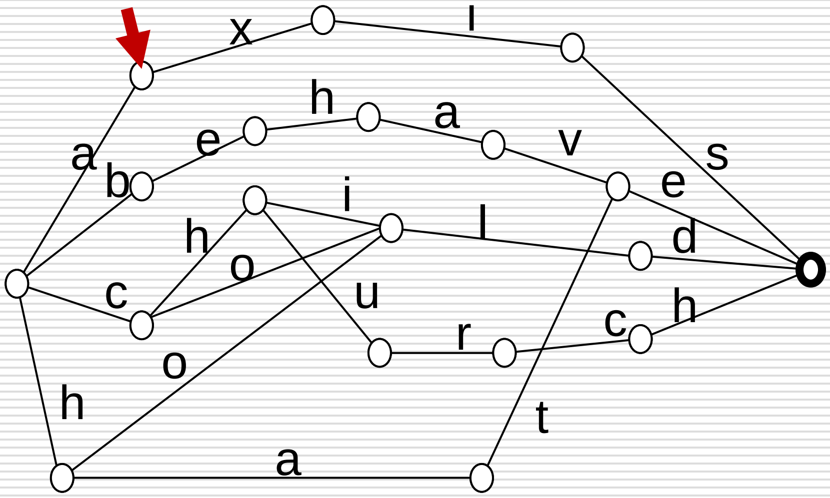


Figure 6
The universal deterministic Levenshtein automaton $A^{\forall}(1)$. See Example 4 for notation.

Example: input “chiid”, b=1



χ(x, chii)=0000

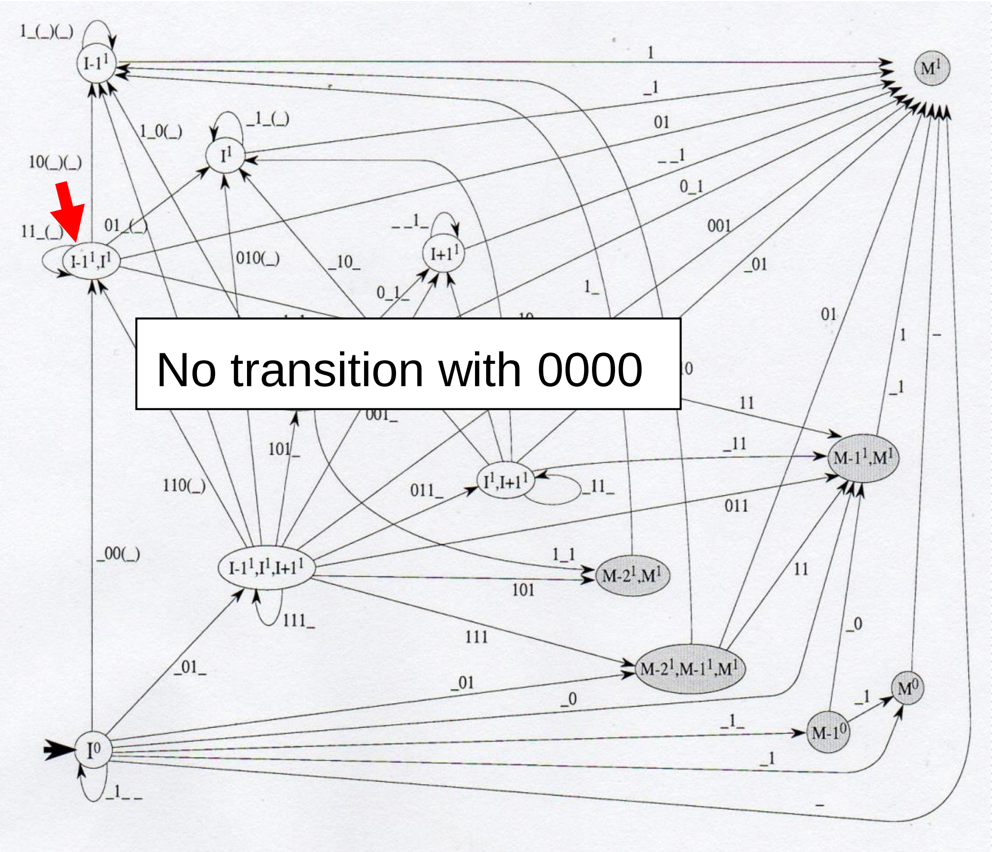
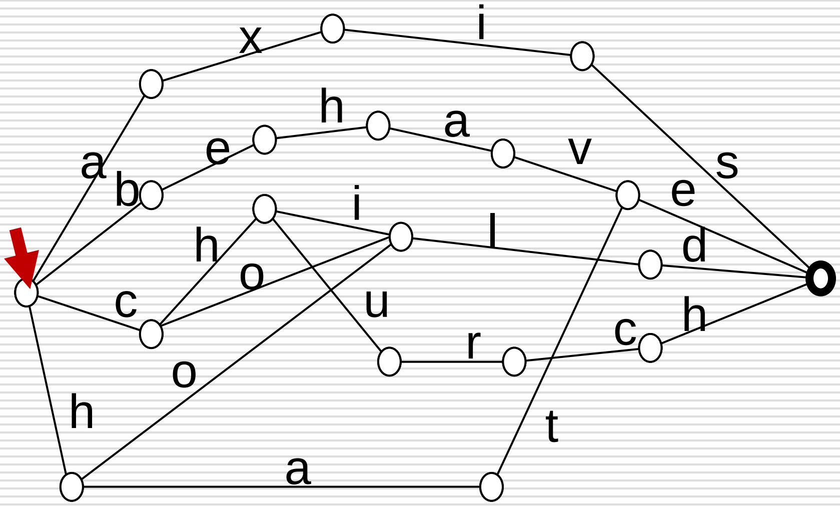


Figure 6
The universal deterministic Levenshtein automaton $A^{\forall}(1)$. See Example 4 for notation.

Example: input “chiid”, b=1



χ(c,$chi)=0100

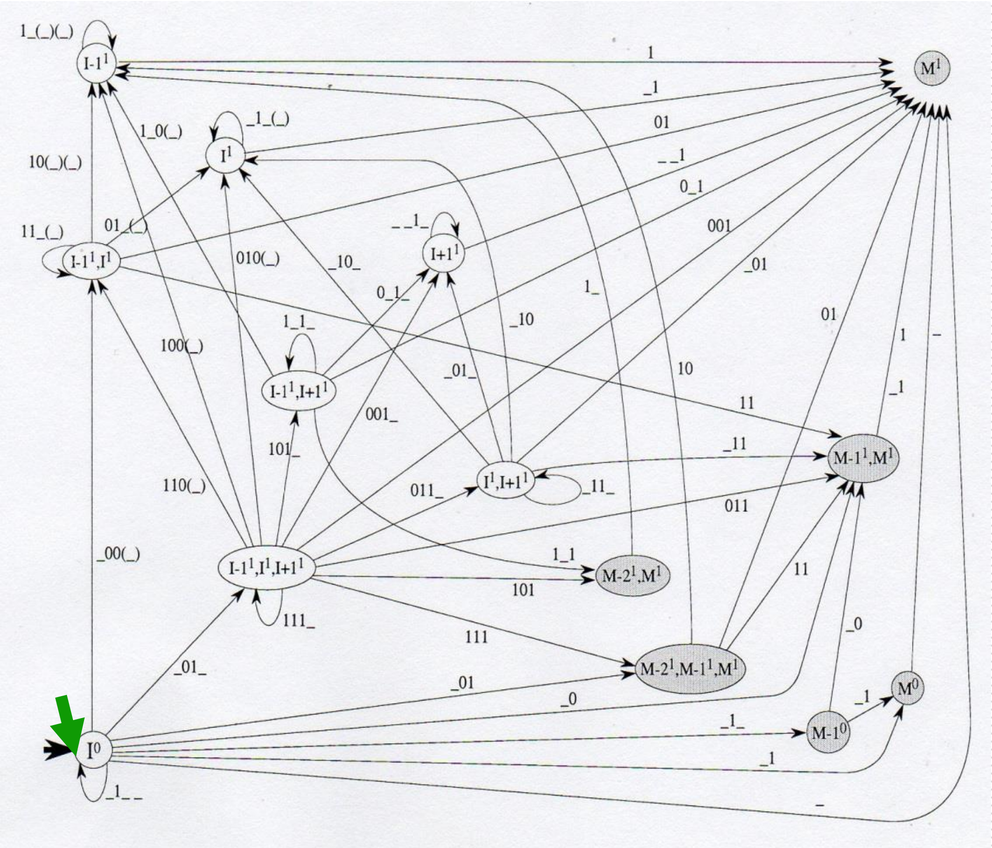
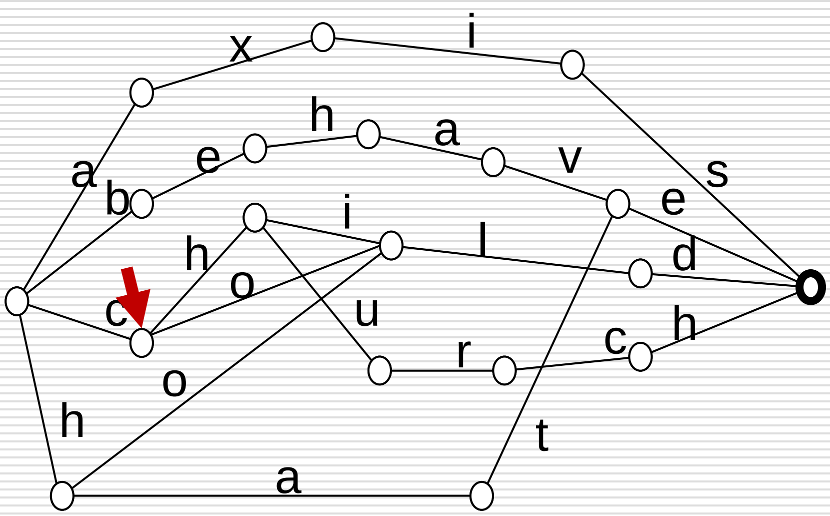


Figure 6
The universal deterministic Levenshtein automaton $A^{\forall}(1)$. See Example 4 for notation.

Example: input “chiid”, b=1



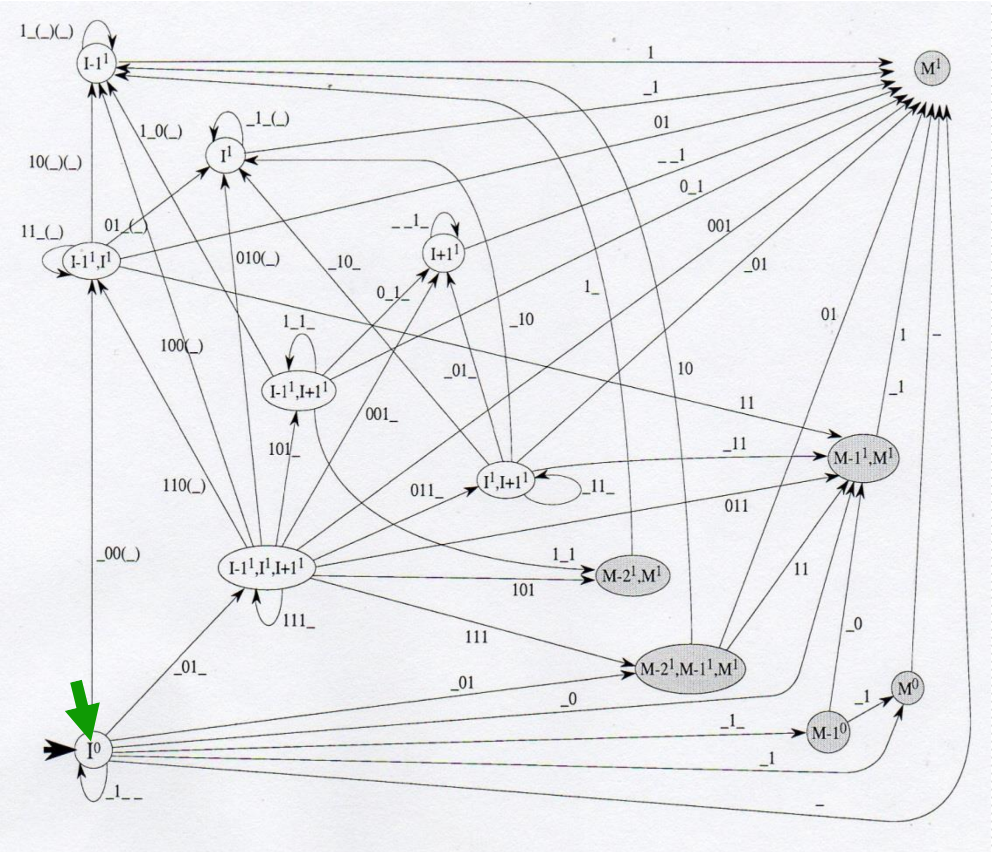
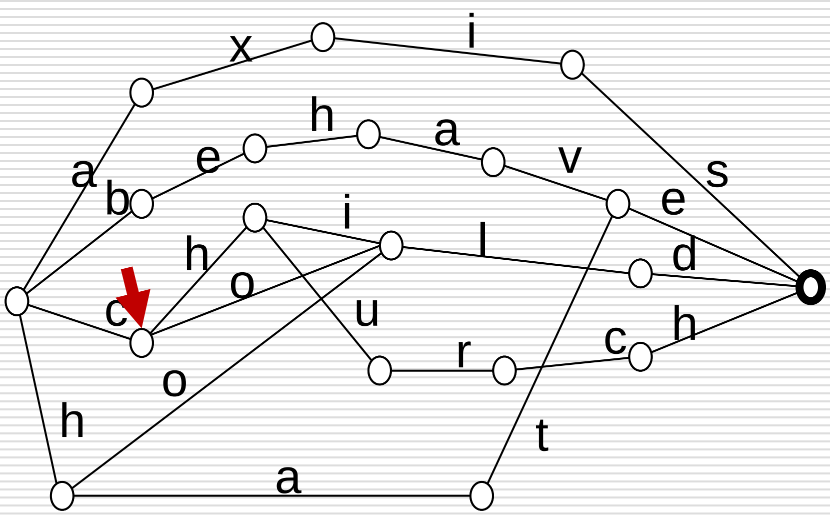


Figure 6
The universal deterministic Levenshtein automaton $A^{\forall}(1)$. See Example 4 for notation.

Example: input “chiid”, b=1



χ(h,chii)=0100

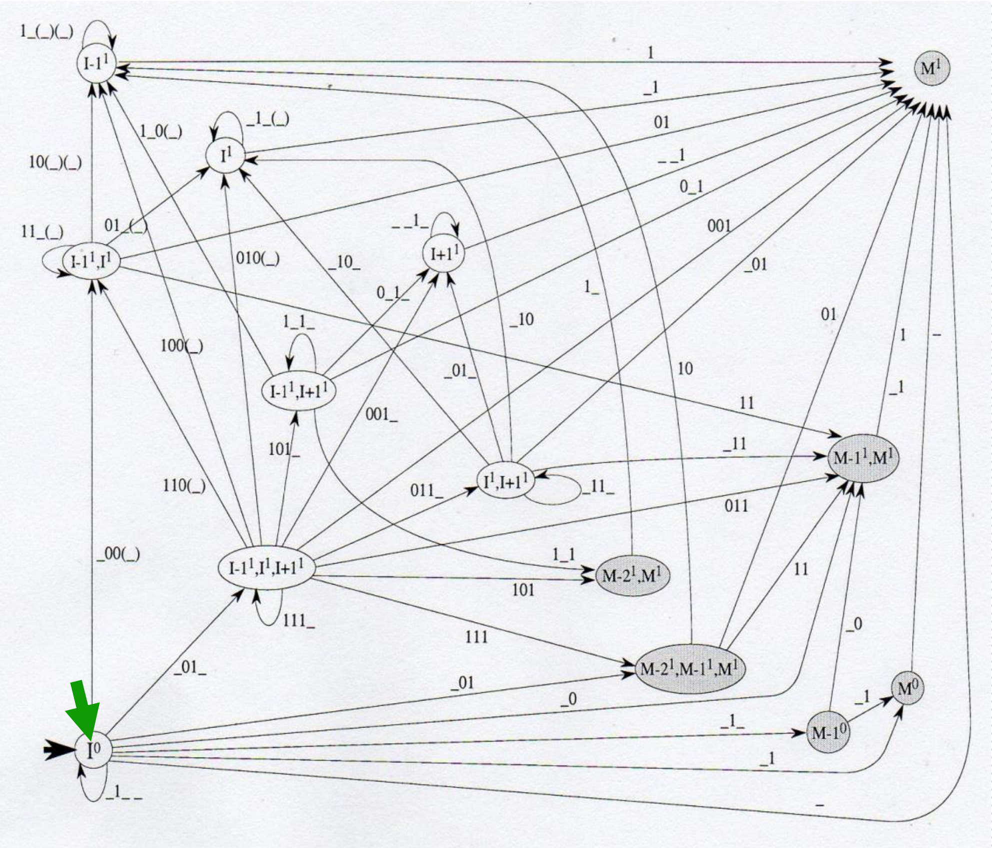
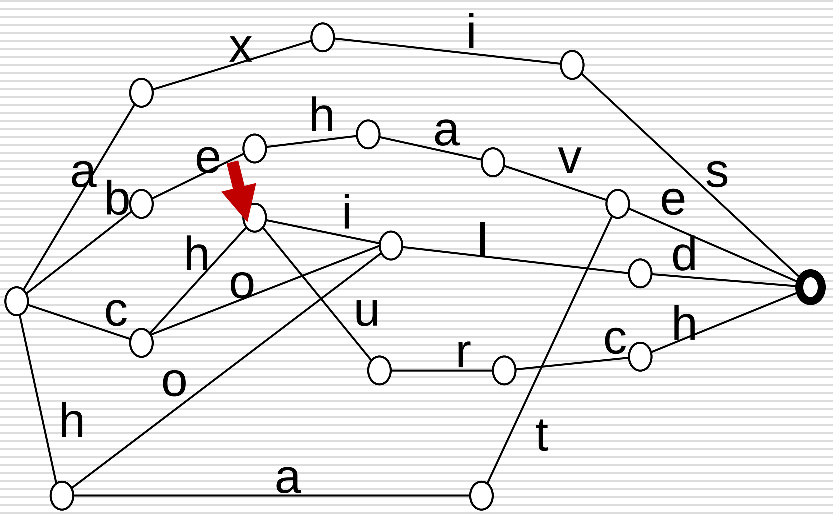


Figure 6
The universal deterministic Levenshtein automaton $A^{\forall}(1)$. See Example 4 for notation.

Example: input “chiid”, b=1



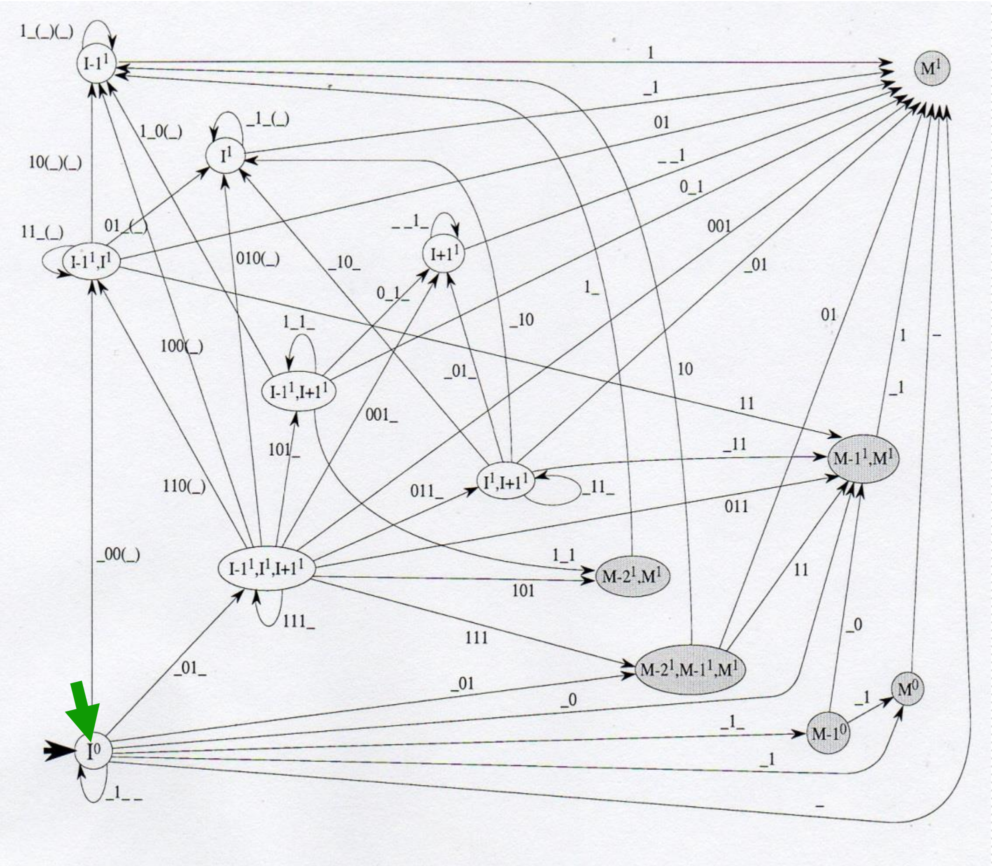
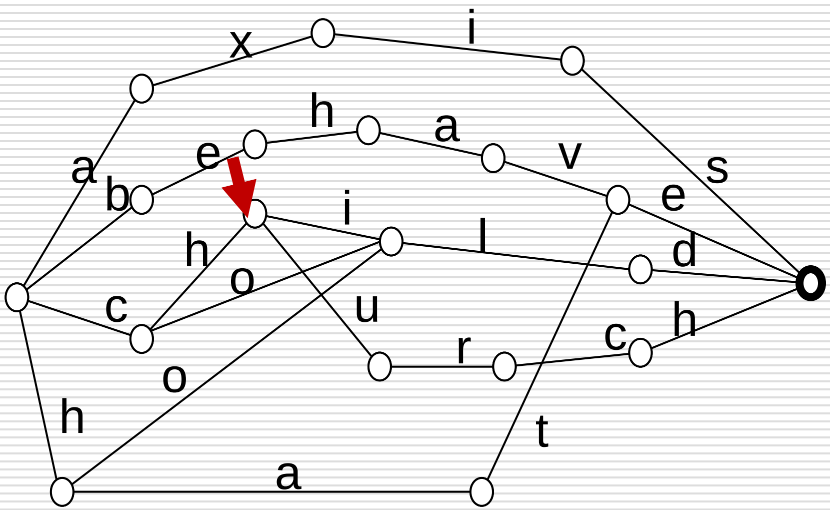


Figure 6
The universal deterministic Levenshtein automaton $A^{\forall}(1)$. See Example 4 for notation.

Example: input “chiid”, b=1



χ(i, hiid)=0100

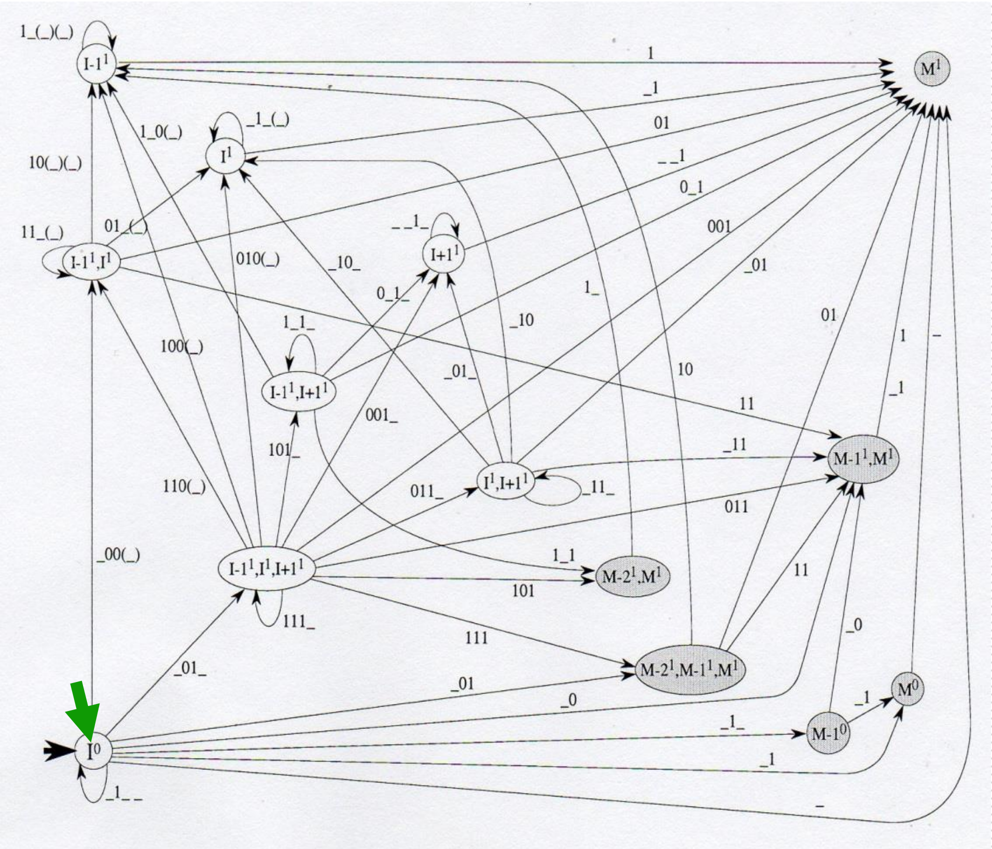
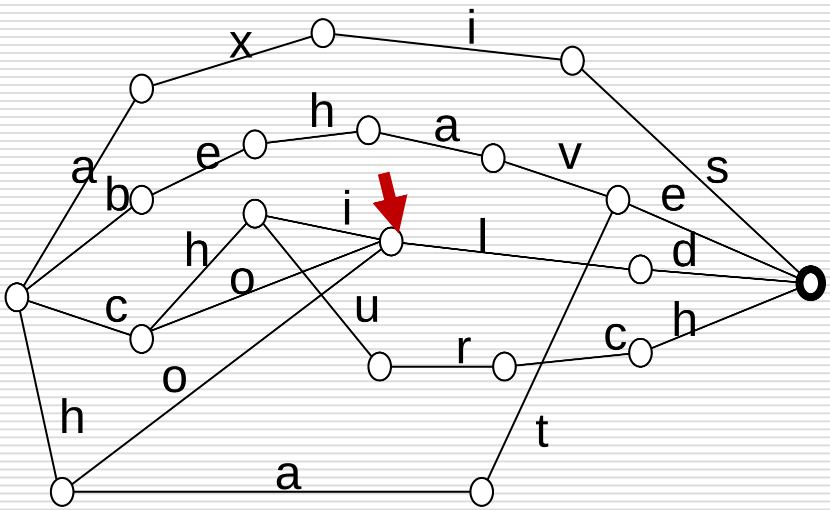


Figure 6
The universal deterministic Levenshtein automaton $A^{\forall}(1)$. See Example 4 for notation.

Example: input “chiid”, b=1



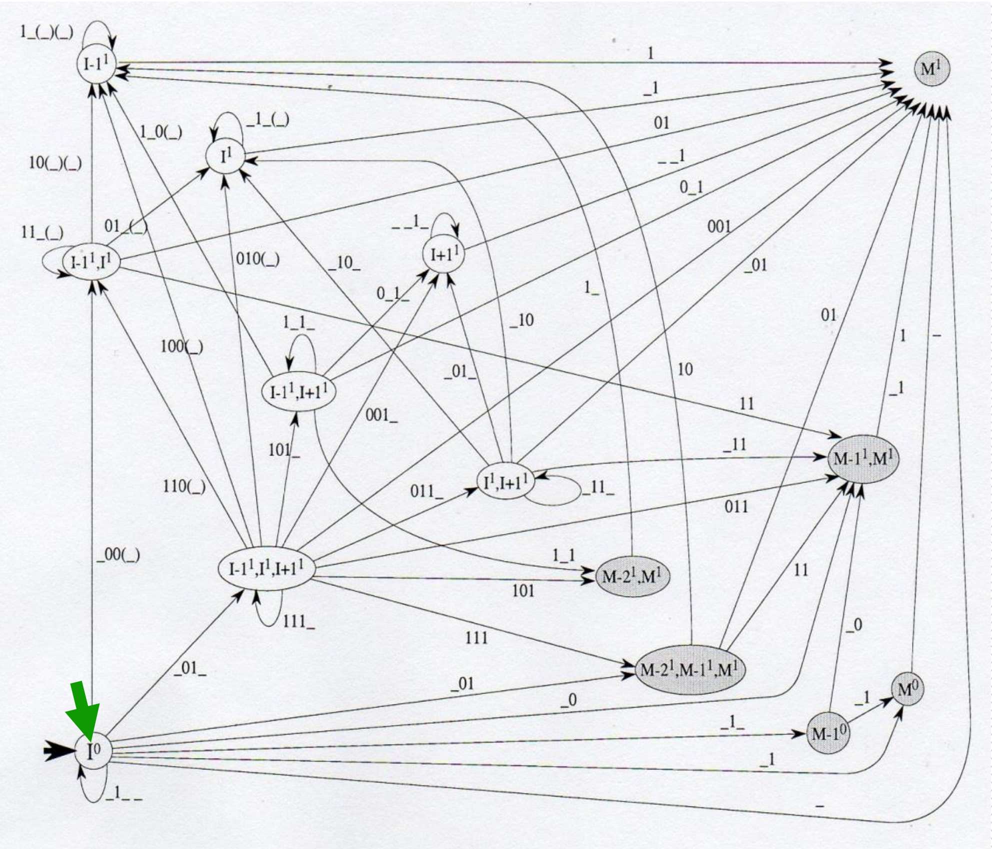
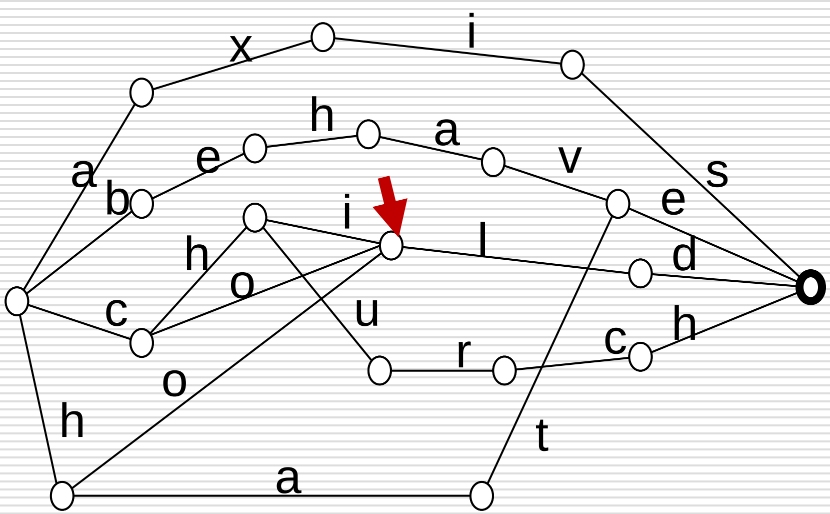


Figure 6
The universal deterministic Levenshtein automaton $A^{\forall}(1)$. See Example 4 for notation.

Example: input "chiid", b=1



χ(l, iid)=000

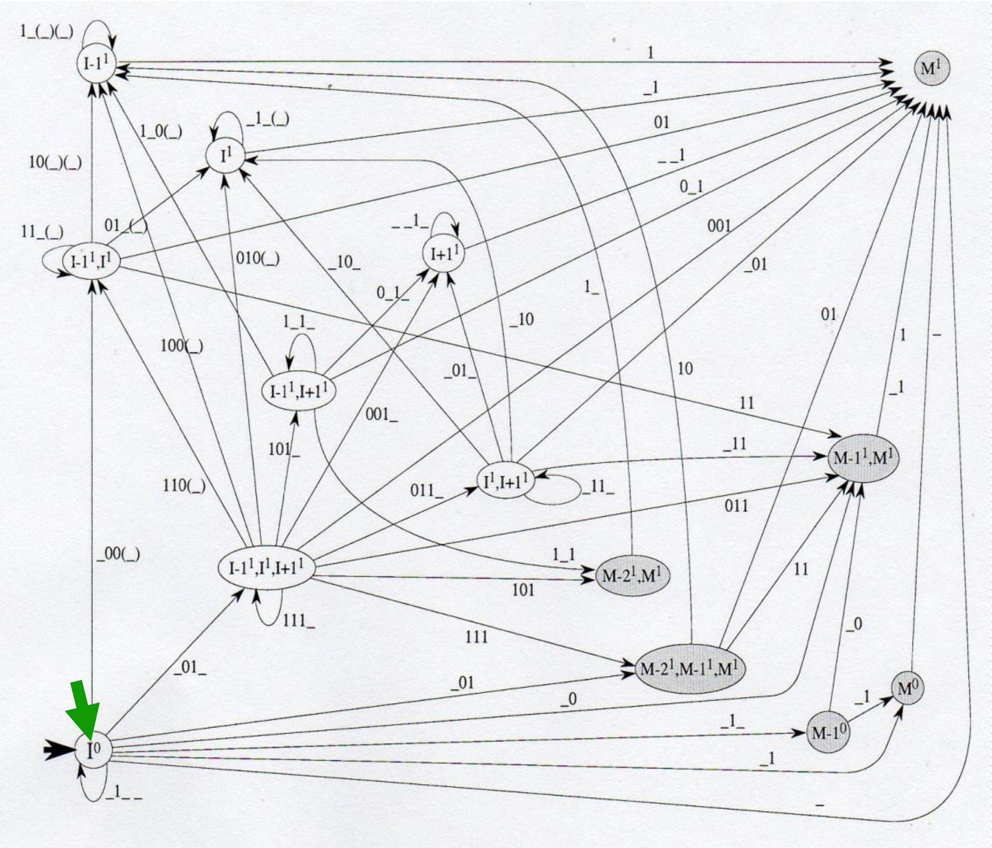
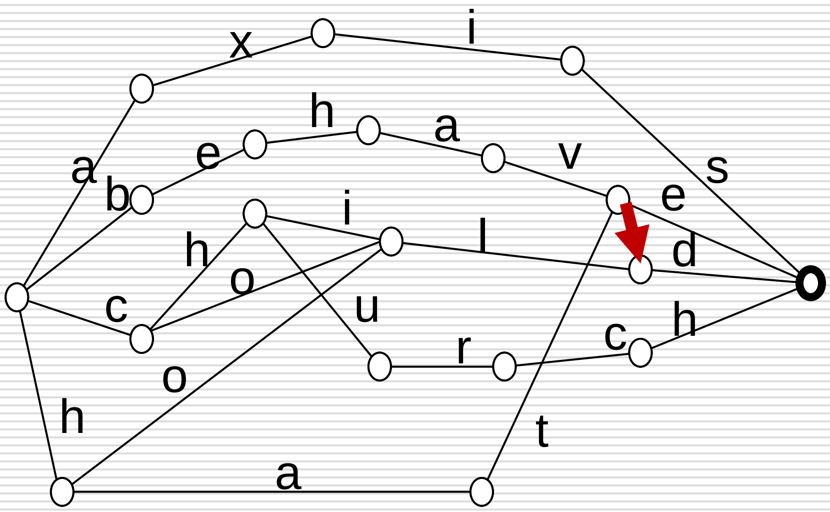


Figure 6
The universal deterministic Levenshtein automaton $A^{\forall}(1)$. See Example 4 for notation.

Example: input “chiid”, b=1



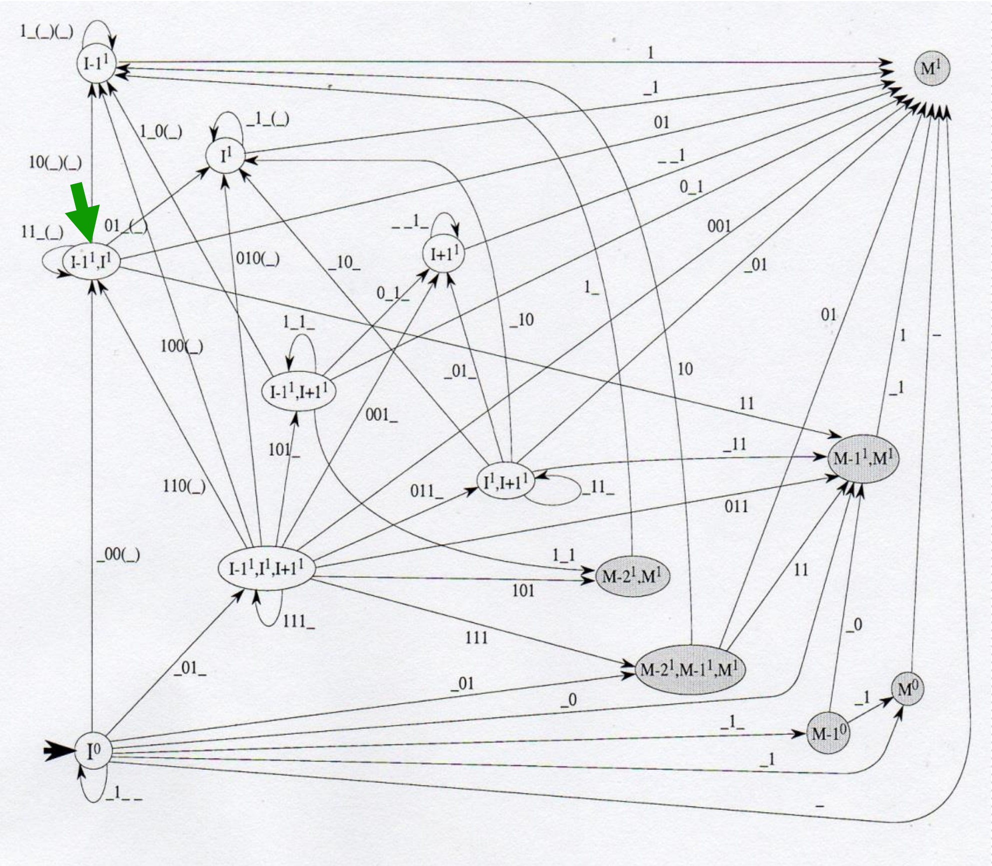
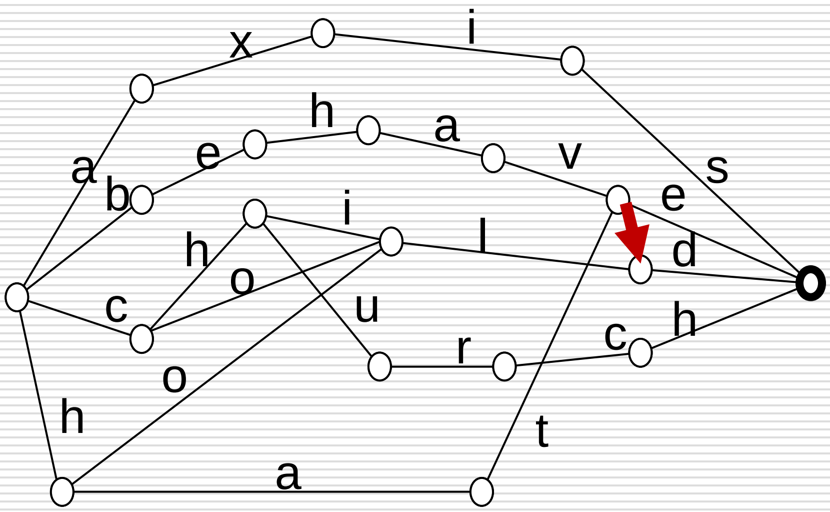


Figure 6
The universal deterministic Levenshtein automaton $A^{\forall}(1)$. See Example 4 for notation.

Example: input “chiid”, b=1



$\chi(d, id)=01$

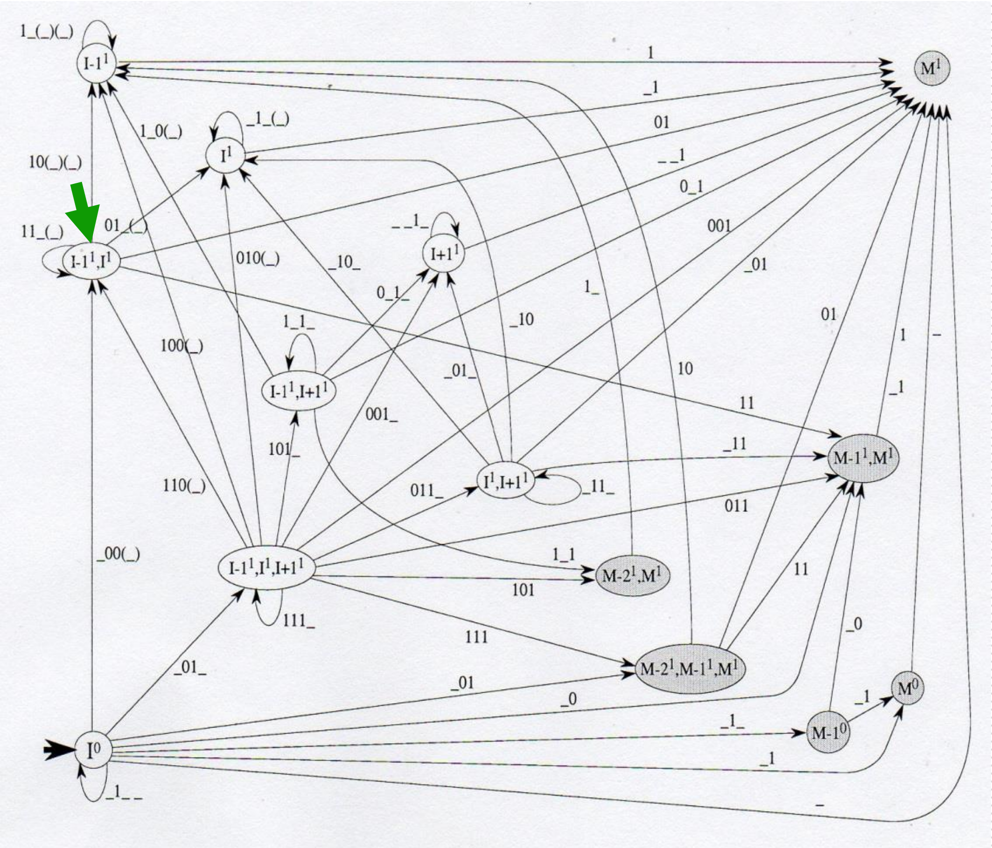
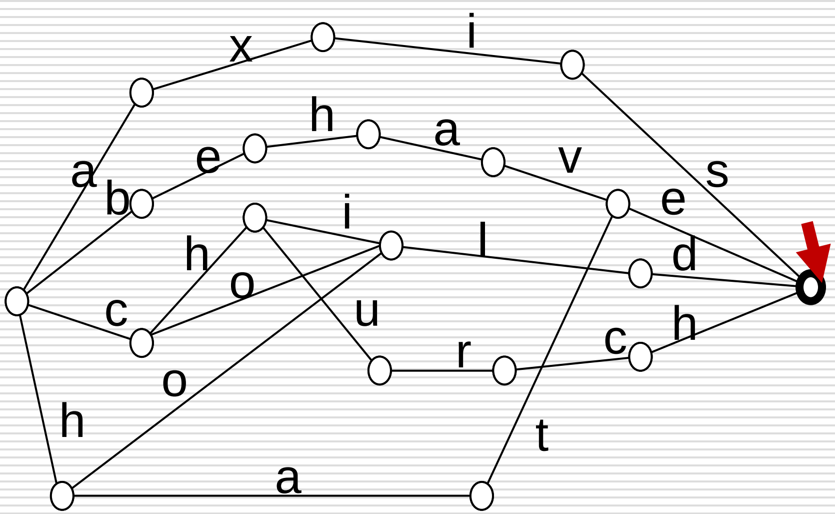


Figure 6
The universal deterministic Levenshtein automaton $A^{\forall}(1)$. See Example 4 for notation.

Example: input "chiid", b=1



Output „child“

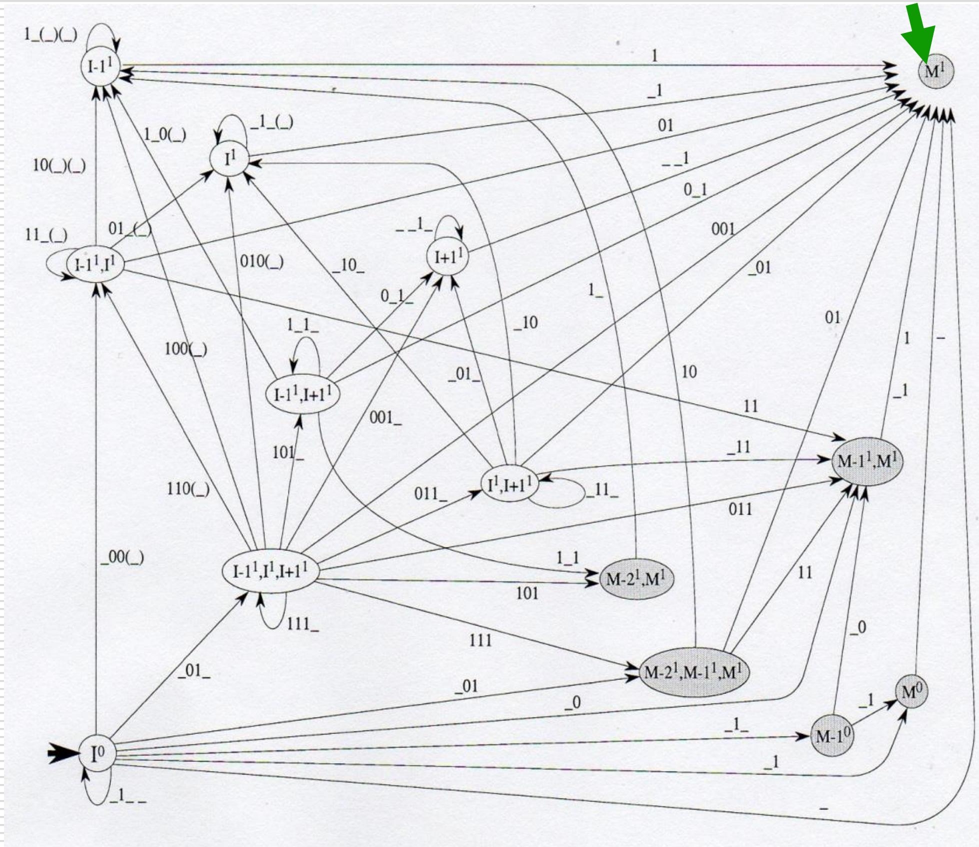


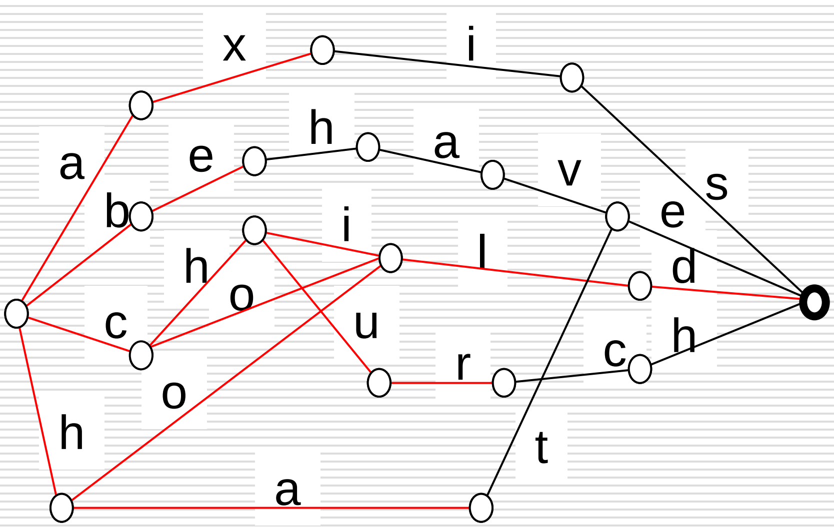
Figure 6
The universal deterministic Levenshtein automaton $A^{\forall}(1)$. See Example 4 for notation.

Advantages

- For each bound n=1,2,3,4 universal Levenshtein automaton offline precomputed once.
- Computing bitvectors and tracing states in the universal Levenshtein automaton cheap!!

Very fast control mechanism for traversal of lexicon automaton.
Useful solution for approximate search in a large lexicon (Lucene).

but not fully satisfactory…



Checked transitions

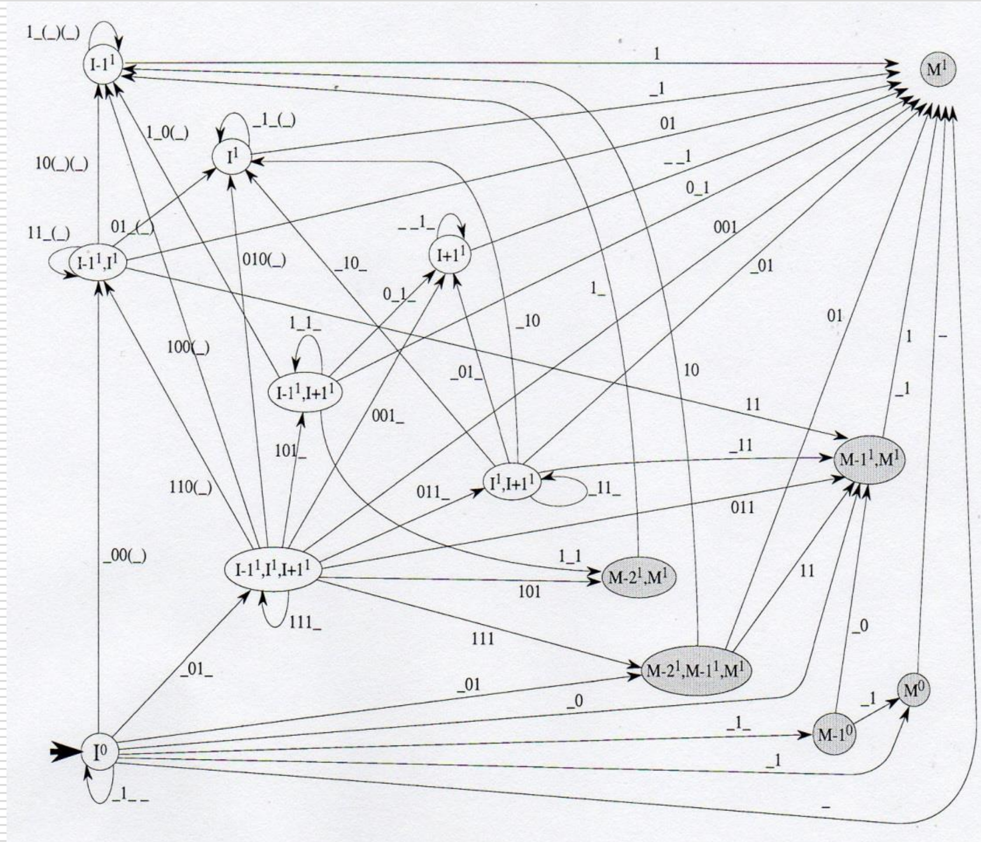
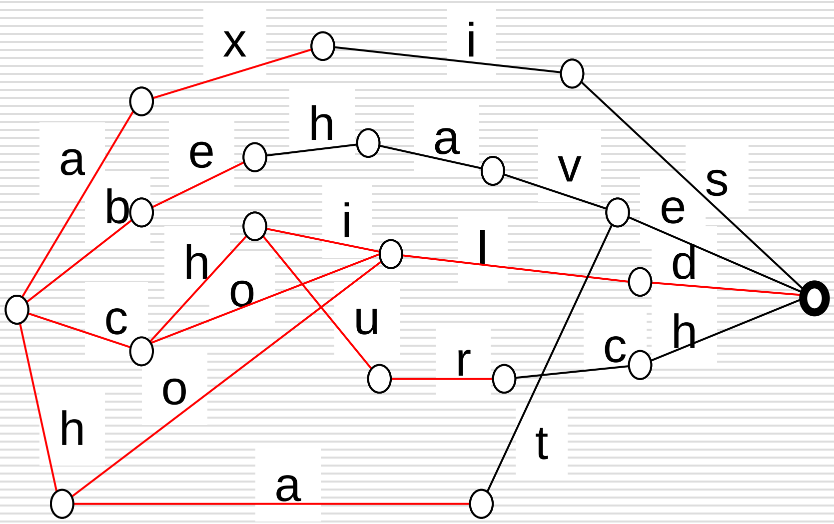


Figure 6
The universal deterministic Levenshtein automaton $A^{\forall}(1)$. See Example 4 for notation.

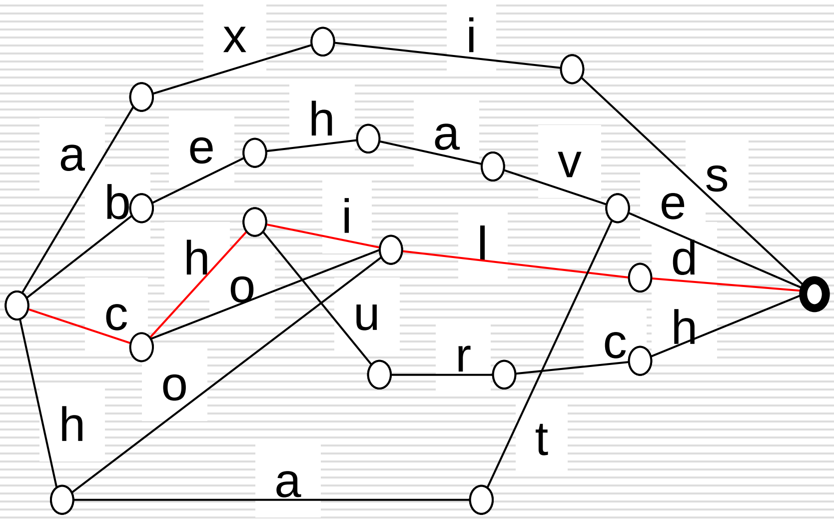
… better solution …



Checked transitions

Assume we would know that the there cannot be an error in the first part „chi“ of pattern „chiid“………

… better solution …



Assume we would know that the there cannot be an error in the first part „chi“ of pattern „chiid“………

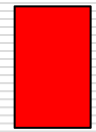
Transitions to be checked

… better solution …

Error somewhere in pattern

Effort

????????????????????



(a) Error in second half of pattern

??????????

… better solution …

Error somewhere in pattern

????????????????????

(a) Error in second half of pattern

??????????

Effort

(b) Error in first half of pattern

?

??????????

… better solution …

Error somewhere in pattern

Effort

????????????????????

(a) Error in second half of pattern

??????????

(b) Error in first half of pattern

??????????

Use an automaton for **inverted lexicon** words for backward search!

… better solution …

Error somewhere in pattern

?????????????????????

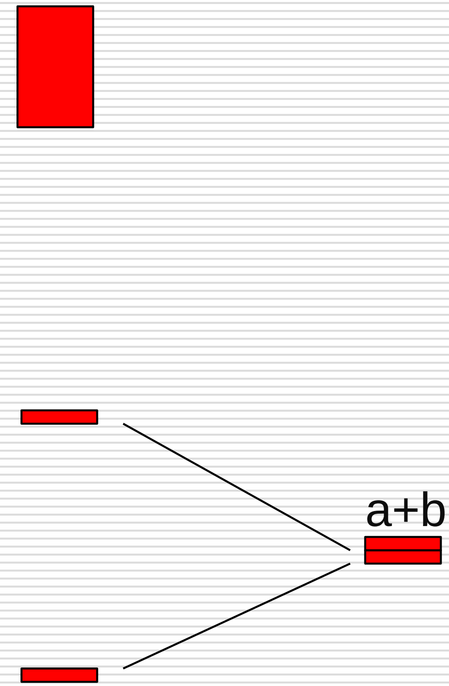
(a) Error in second half of pattern

✓ ??????????

(b) Error in first half of pattern

?????????? ✓

Effort



Use an automaton for **inverted lexicon** words for backward search!

Forward-backward method

Can be generalized to larger bounds n=2, 3,…

n=1	0 | ≤ 1 →	← ≤ 1 | 0
n=2	≤ 1 | ≤ 1 (2) →	← ≤ 1 (2) | ≤ 1
n=3	≤ 1 | ≤ 2 →	← ≤ 2 | ≤ 1

Enormous speed-up factor! Fully satisfactory solution for

- small bounds (1,2,3,4)
- normal lexica (entries not too long)

Postcorrection of OCRed historical Texts

Postcorrection of OCRed historical Texts

Given a historical text obtained via Optical Character Recognition (OCR)

- How can we found tokens that „probably“ represent errors?
- How can we find correction suggestions – also in original historical spelling?
- How can we find corresponding modern words?

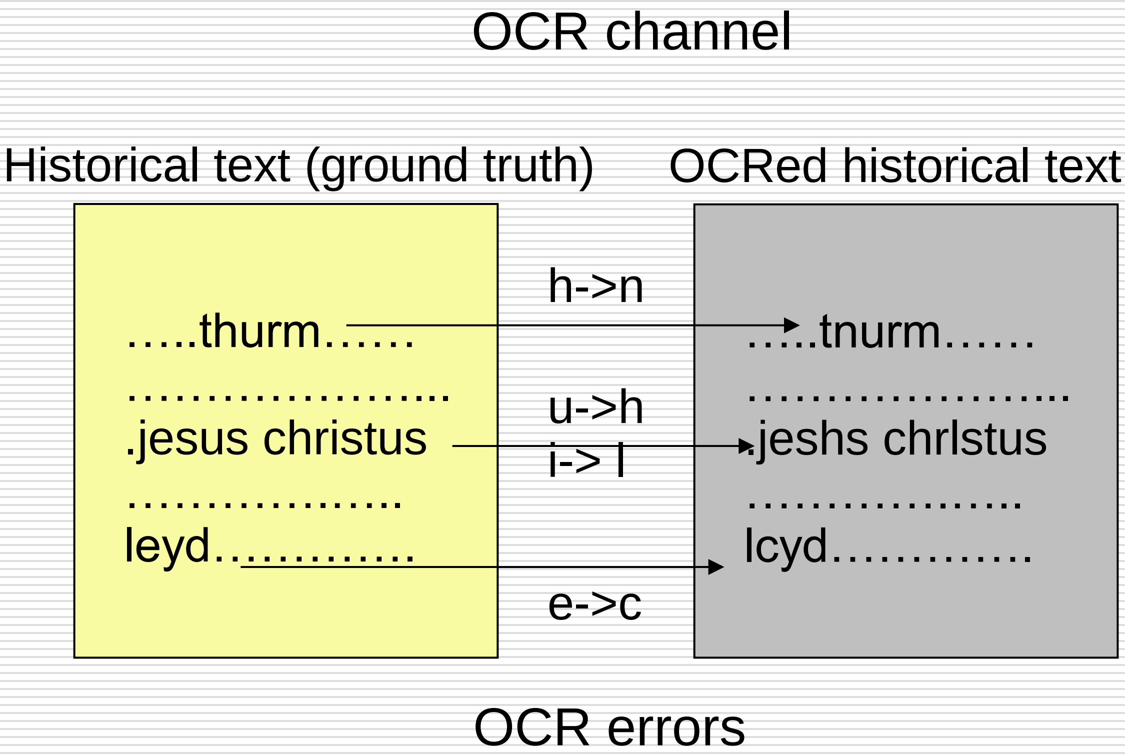
- How can we find typical OCR errors and errors series?
- How can we find characteristics of historical orthography?

Two-channel model for OCRed historical texts

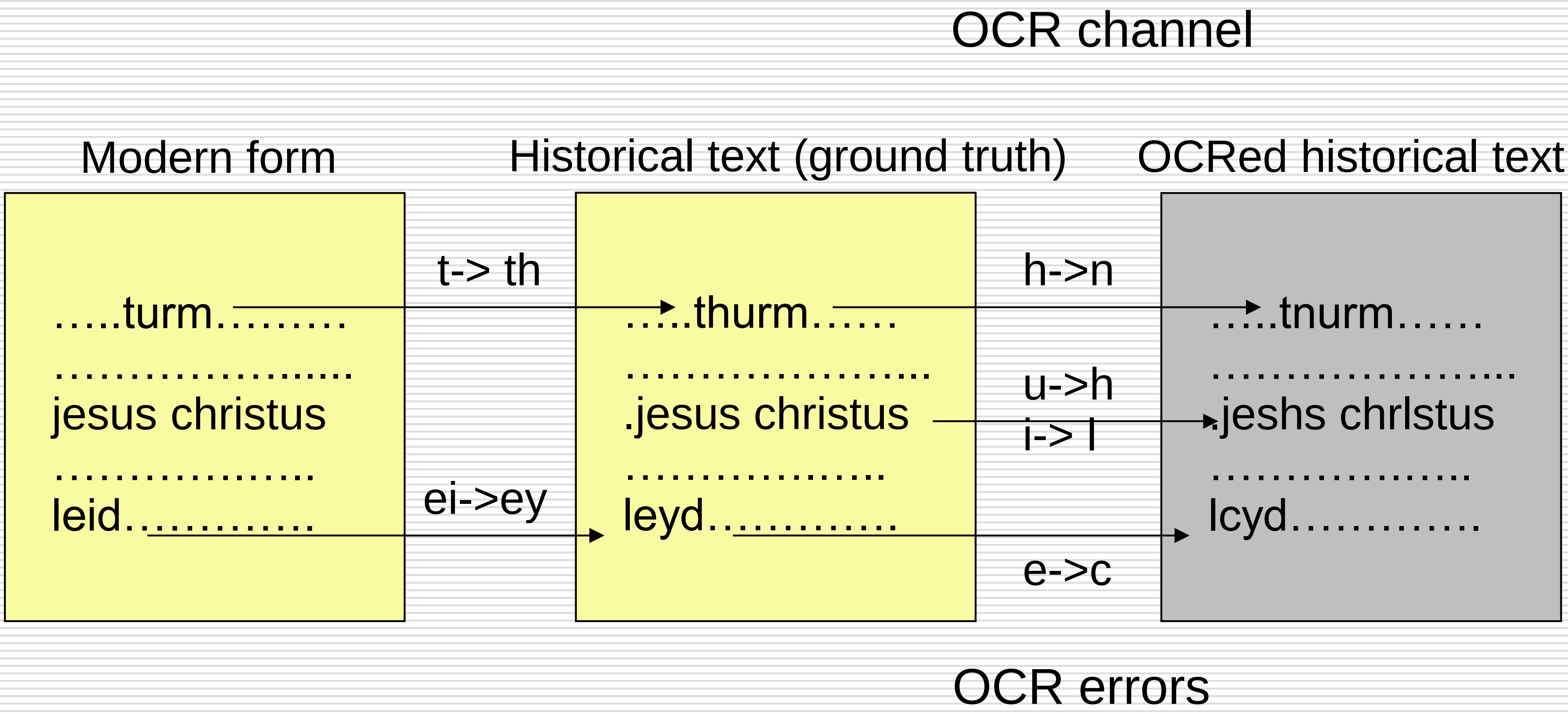
OCRed historical text

…..tnurm……
………………...
.jeshs chrlstus
……………..
lcyd………….

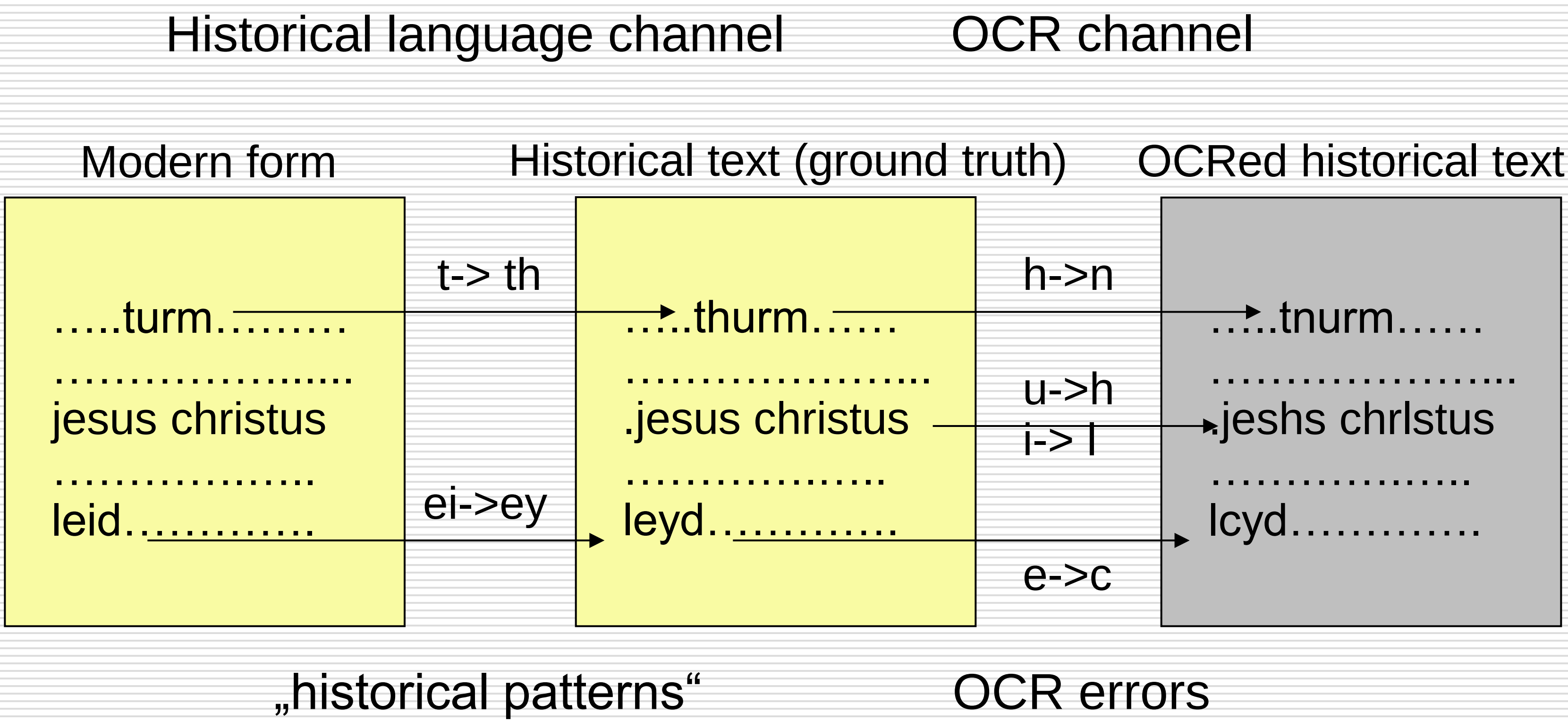
Two-channel model for OCRed historical texts



Two-channel model for OCRed historical texts

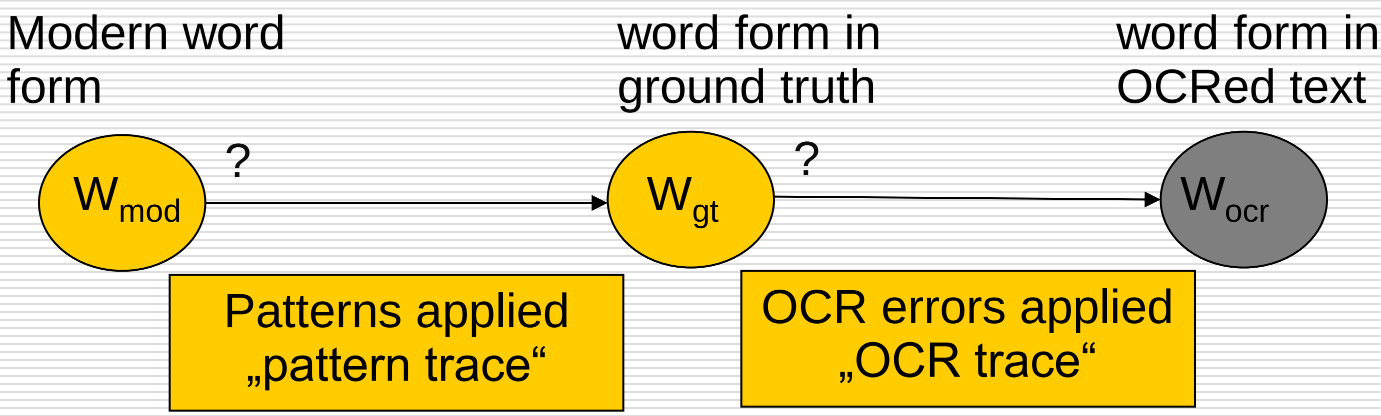


Two-channel model for OCRed historical texts



Computing all „interpretations“ of OCR token

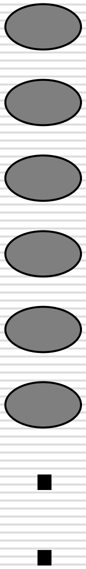
„Interpretation“: starting from OCR token Wocr „guess“ channel history



Variant of forward-backword procedure:
Given lexicon of modern words, set of patterns, set of possible OCR errors.
Efficiently generate all interpretations for an OCRed word
(number of pattern applications and OCR errors bounded)

Profiling historical OCRed corpora

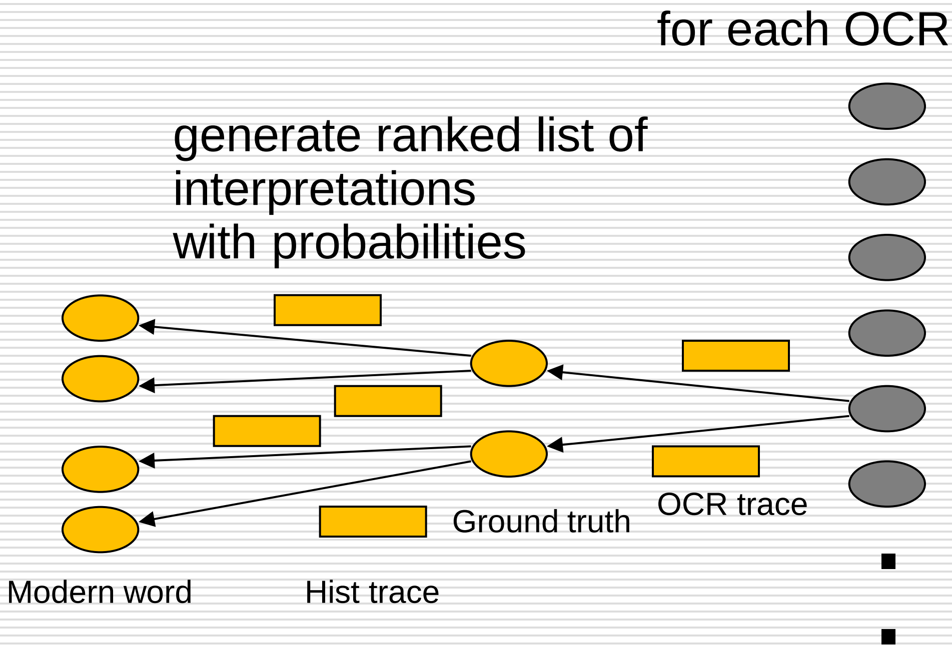
for each OCR token W_{ocr}



Initial „rough“ model for
- words
- patterns
- OCR errors

and their probabilities

Profiling historical OCRed corpora

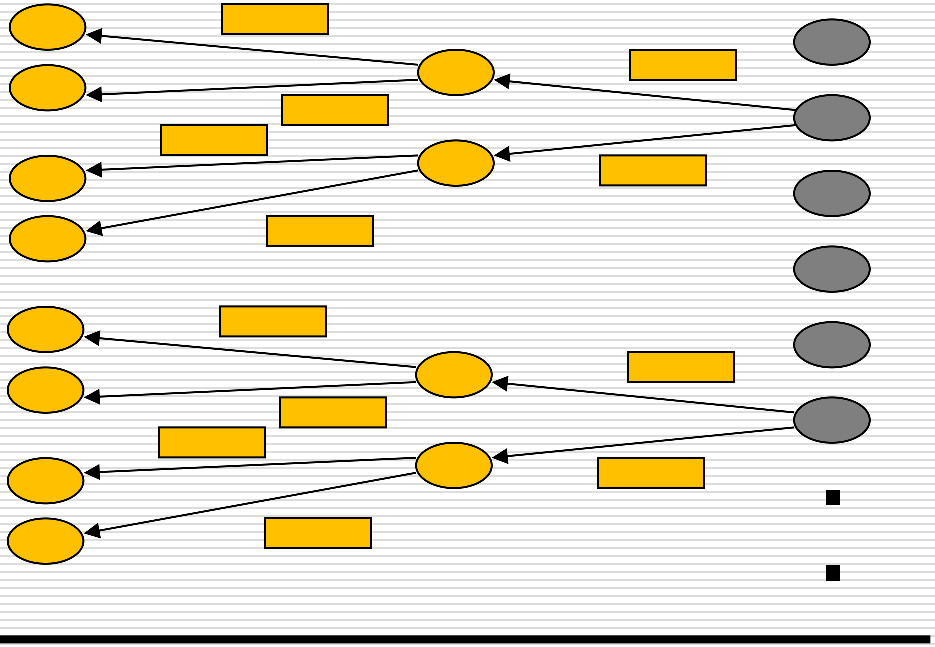


Initial „rough“ model for
- words
- patterns
- OCR errors

and their probabilities

Profiling historical OCRed corpora

for each OCR token W_{ocr}

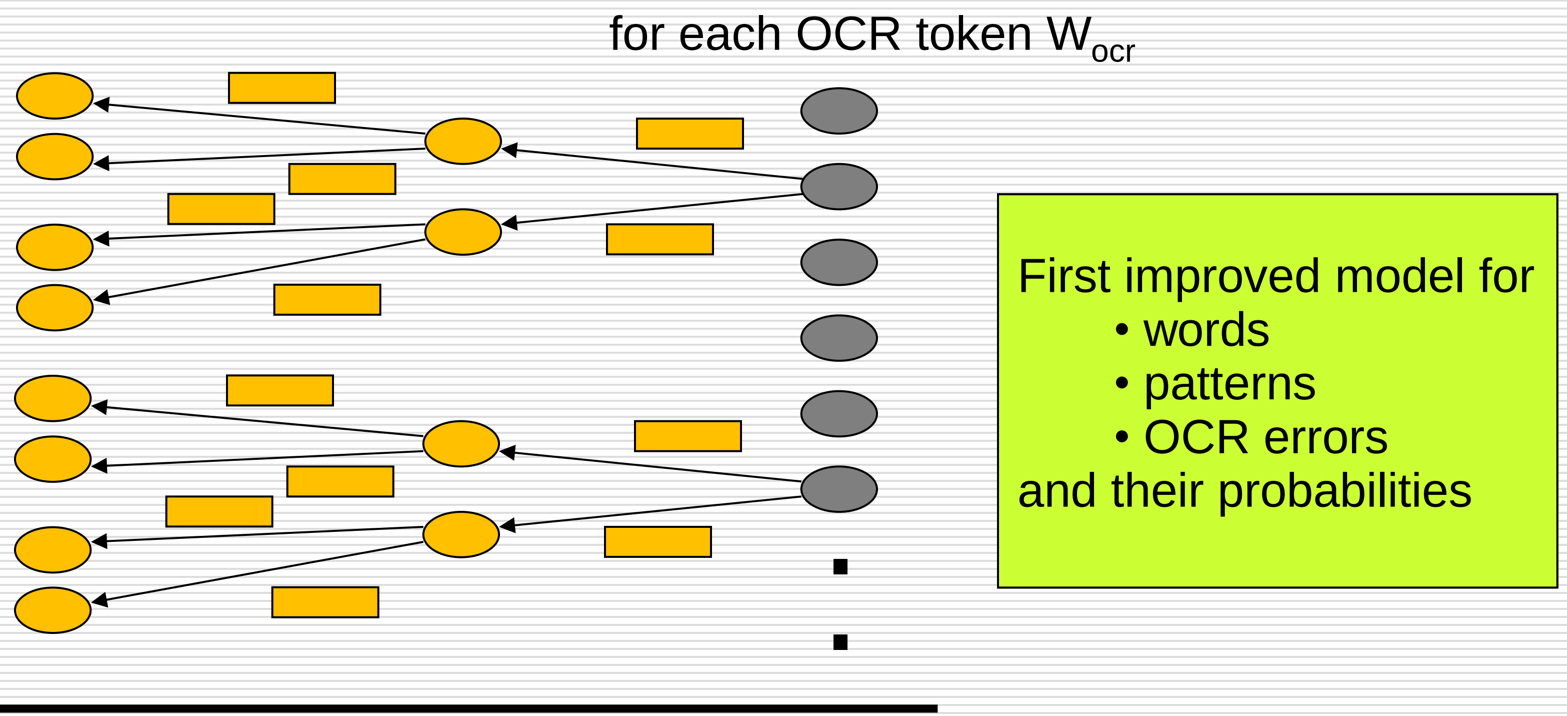


Initial „rough“ model for
- words
- patterns
- OCR errors

and their probabilities

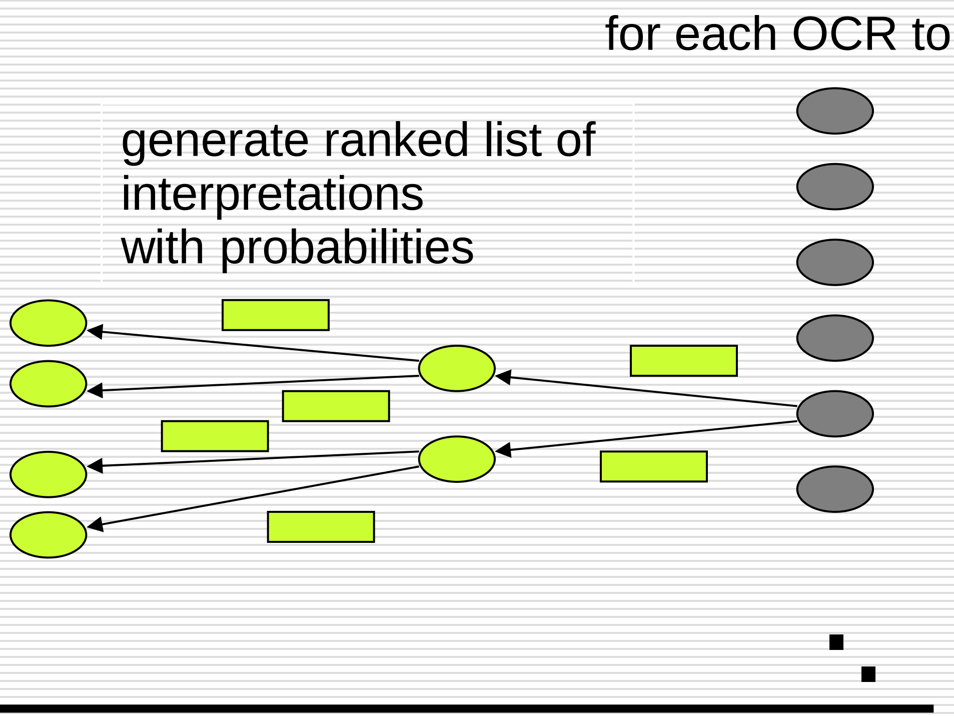
Summing up relative frequencies over all OCR tokens obtain

Profiling historical OCRed corpora



Summing up relative frequencies over all OCR tokens obtain

Profiling historical OCRed corpora

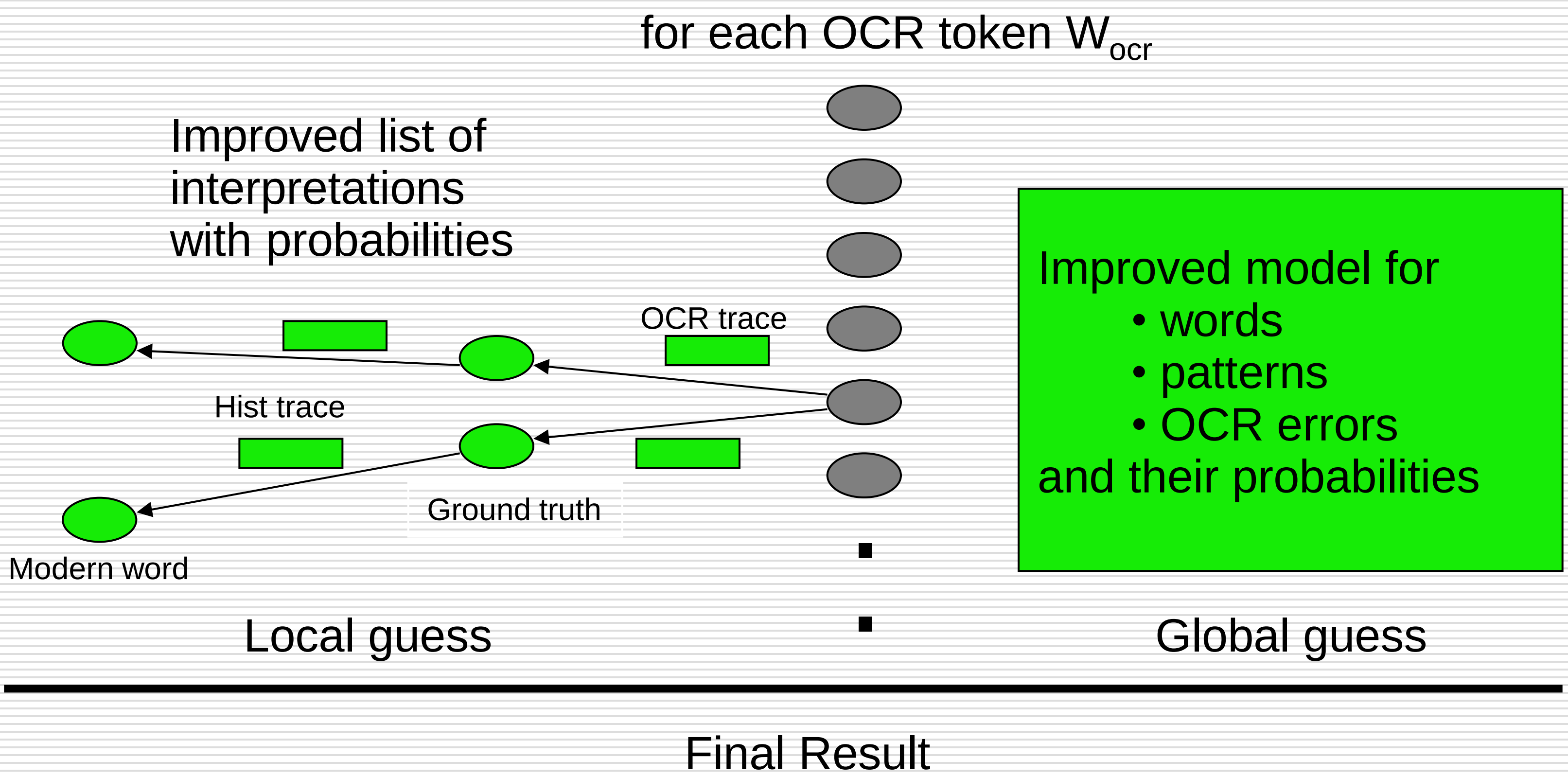


First improved model for
- words
- patterns
- OCR errors

and their probabilities

Iterate - after n rounds obtain

Profiling historical OCRed corpora

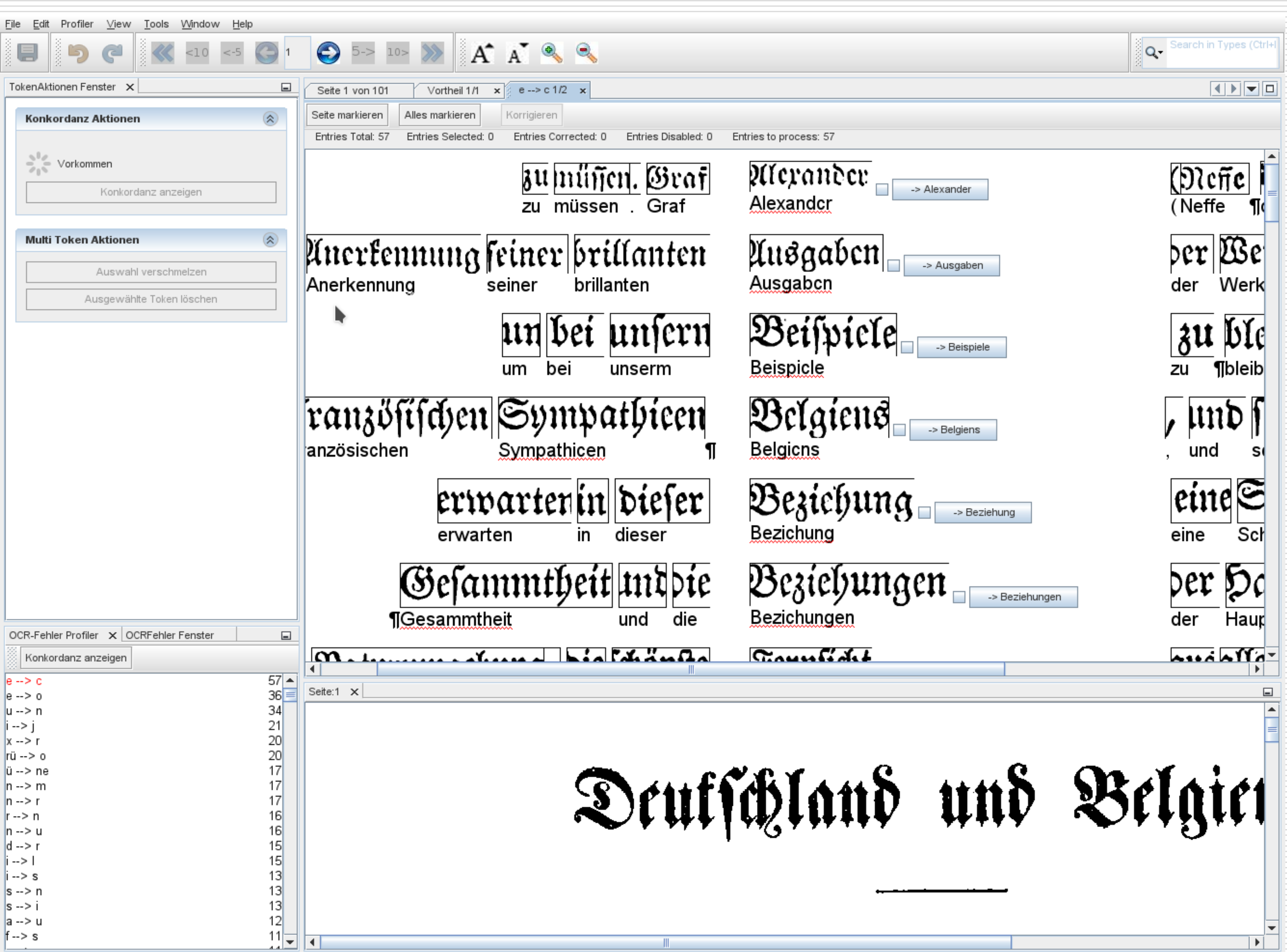


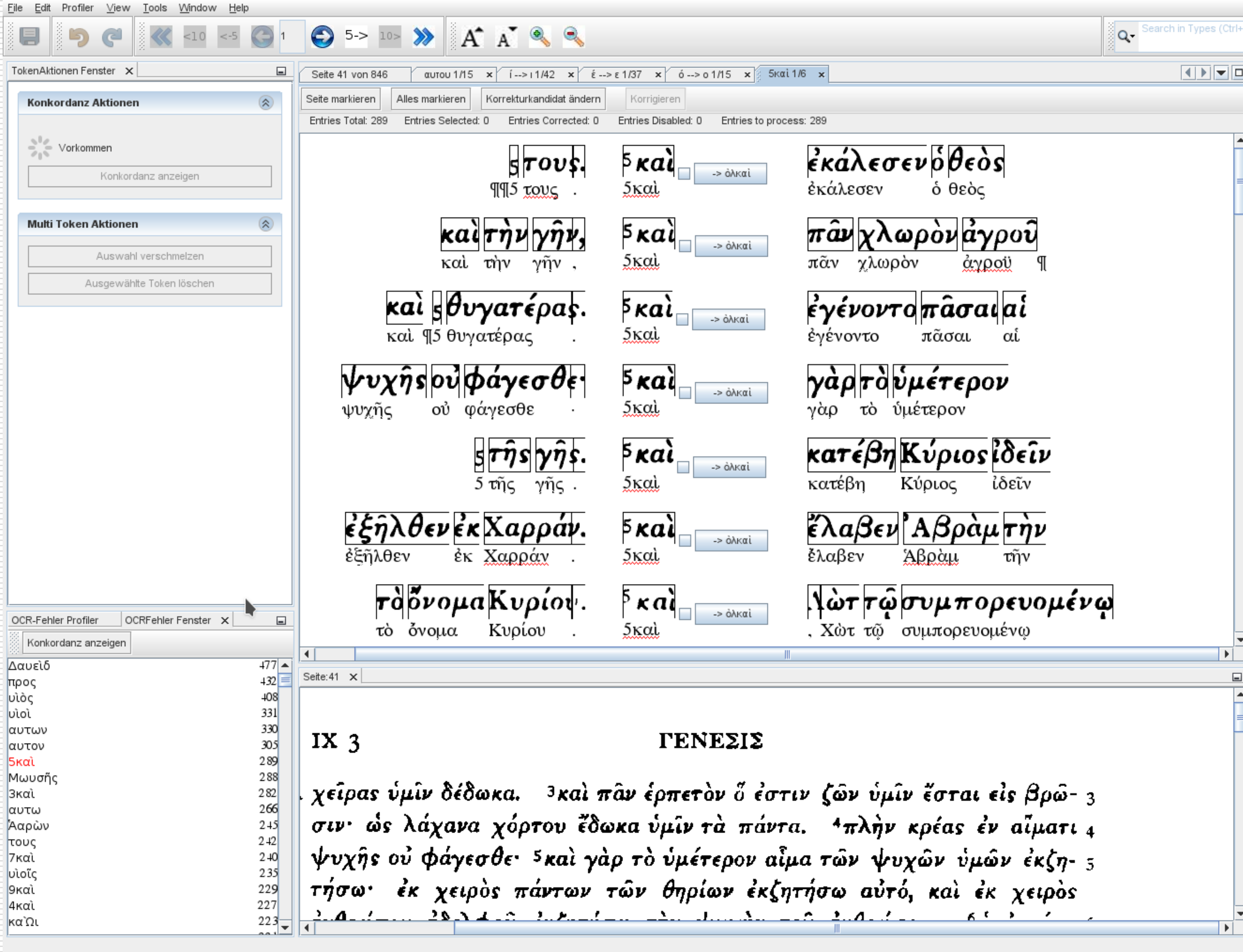
PoCoTo Postcorrection Tool

Profiles used for interactive postcorrection of OCRed historical texts

- finding tokens that „probably“ represent errors
- finding correction suggestions, also in original historical spelling
- finding corresponding modern words (not realized)

- finding typical OCR errors and errors series
- finding characteristics of historic orthography (not realized)


File Edit Profiler View Tools Window Help
Search in Types (Ctrl+I)
TokenAktionen Fenster
Konkordanz Aktionen
Vorkommen
Konkordanz anzeigen
Multi Token Aktionen
Auswahl verschmelzen
Ausgewählte Token löschen
Seite 1 von 101
Vortheil 1/1
e --> c 1/2
Seite markieren
Alles markieren
Korrigieren
Entries Total: 57
Entries Selected: 0
Entries Corrected: 0
Entries Disabled: 0
Entries to process: 57
zu müssen . Graf
Alexandcr
-> Alexander
(Neffe
Anerkennung seiner brillanten
Ausgabcn
-> Ausgaben
der Werk
um bei unserm
Beispicle
-> Beispiele
zu ¶bleib
anzösischen Sympathicen ¶
Belgicns
-> Belgiens
, und
erwarten in dieser
Bezichung
-> Beziehung
eine Sch
¶Gesammtheit und die
Bezichungen
-> Beziehungen
der Haup
OCR-Fehler Profiler
OCRFehler Fenster
Konkordanz anzeigen
e --> c 57
e --> o 36
u --> n 34
i --> j 21
x --> r 20
rü --> o 20
ü --> ne 17
n --> m 17
n --> r 17
r --> n 16
n --> u 16
d --> r 15
i --> l 15
i --> s 13
s --> n 13
s --> i 13
a --> u 12
f --> s 11
Seite:1
Deutschland und Belgien


File Edit Profiler View Tools Window Help
Search in Types (Ctrl+I)
TokenAktionen Fenster
Konkordanz Aktionen
Vorkommen
Konkordanz anzeigen
Multi Token Aktionen
Auswahl verschmelzen
Ausgewählte Token löschen
Seite 41 von 846
αυτου 1/15
ί --> ι 1/42
έ --> ε 1/37
ό --> ο 1/15
5καὶ 1/6
Seite markieren
Alles markieren
Korrekturkandidat ändern
Korrigieren
Entries Total: 289
Entries Selected: 0
Entries Corrected: 0
Entries Disabled: 0
Entries to process: 289
¶¶5 τους . 5καὶ -> ὁλκαὶ ἐκάλεσεν ὁ θεὸς
καὶ τὴν γῆν , 5καὶ -> ὁλκαὶ πᾶν χλωρὸν ἀγροῦ ¶
καὶ ¶5 θυγατέρας . 5καὶ -> ὁλκαὶ ἐγένοντο πᾶσαι αἱ
ψυχῆς οὐ φάγεσθε · 5καὶ -> ὁλκαὶ γὰρ τὸ ὑμέτερον
5 τῆς γῆς . 5καὶ -> ὁλκαὶ κατέβη Κύριος ἰδεῖν
ἐξῆλθεν ἐκ Χαρράν . 5καὶ -> ὁλκαὶ ἔλαβεν Ἀβρὰμ τὴν
τὸ ὄνομα Κυρίου . 5καὶ -> ὁλκαὶ , Χὼτ τῷ συμπορευομένῳ
OCR-Fehler Profiler
OCRFehler Fenster
Konkordanz anzeigen
Δαυεὶδ 477
προς 432
υἱὸς 408
υἱοὶ 331
αυτων 330
αυτον 305
5καὶ 289
Μωυσῆς 288
3καὶ 282
αυτω 266
Ἀαρὼν 245
τους 242
7καὶ 240
υἱοῖς 235
9καὶ 229
4καὶ 227
κα'Ωι 223
Seite:41
IX 3 ΓΕΝΕΣΙΣ
χεῖρας ὑμῖν δέδωκα. 3καὶ πᾶν ἑρπετὸν ὅ ἐστιν ζῶν ὑμῖν ἔσται εἰς βρῶ- 3
σιν· ὡς λάχανα χόρτου ἔδωκα ὑμῖν τὰ πάντα. 4πλὴν κρέας ἐν αἵματι 4
ψυχῆς οὐ φάγεσθε· 5καὶ γὰρ τὸ ὑμέτερον αἷμα τῶν ψυχῶν ὑμῶν ἐκζη- 5
τήσω· ἐκ χειρὸς πάντων τῶν θηρίων ἐκζητήσω αὐτό, καὶ ἐκ χειρὸς

Approximate search
…the story continued……

Approximate search
...the story continued......

Forward-backward

	Forward	Backward
n=1	0 | ≤ 1	≤ 1 | 0
n=2	≤ 1 | ≤ 2	≤ 2 | ≤ 1
n=3	≤ 1 | ≤ 3	≤ 3 | ≤ 1
n=4	≤ 2 | ≤ 4	≤ 4 | ≤ 2

....

Approximate search
…the story continued……

If a patterns containing ≤ n errors is split into n+1 parts, one part has 0 errors!!

```
???????????????????????????????????????????????????????????????????          ?????????????????????????????????????
-----------------------------------------------|--------|--------------------------
```

?Generalization of forward-backward where search can start with any infix of the pattern?

Need a data structure for the lexicon that

1. gives immediate access to any infix
2. infixes can be extended both to the left and to the right

Approximate search
...the story continued......

Using „symmetric compacted directed acyclic word graphs SCDAWGs“:

Improved algorithm for approximate search in lexica and collections of strings - much better for longer strings and larger distance bounds.

„Wallbreaker“ system – winner of international competition 2013 for string similarity search.

…...Insight

SCDAWGs and related text index structures very interesting for many natural language processing tasks!!