Endliche Automaten (EA)

Übersicht

- Endliche Automaten als Berechnungsmodell
- Beispiel-Automat: Der Lachautomat
  - Verabeiten von Eingabeketten
  - Akzeptieren von Eingabeketten
- Mengentheoretische Formalisierung
- Endliche Automaten in Prolog
- Nicht-Deterministische Endliche Automaten
- Sprachen von Endlichen Automaten
- Reguläre Ausdrücke
- Literatur

Motivation

Endliche Automaten *(Finite-State Automata)* sind …

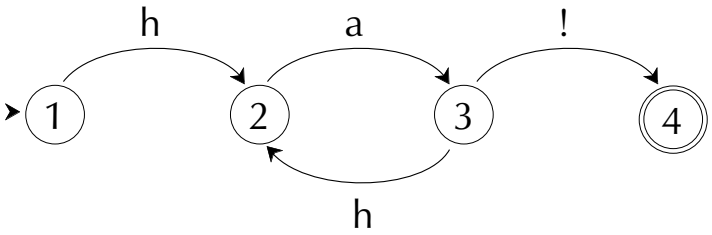
- mathematisch wohl-definiert und theoretisch aufgearbeitet
  - Informatik: Grundlage der Berechenbarkeitstheorie
  - Linguistik: Sind Teile der menschlichen Sprache mit Endlichen Automaten beschreibbar? Struktur von Wörtern, Sätzen, Dialogen…
- leicht implementierbar und effizient ausführbar auf Computer
- in unterschiedlichsten Gebieten anwendbar
  - Sprachverarbeitung: Tokenizer, Morphologie, Lexikon, Informationsextraktion, Phrasenerkennung, (Partielle) syntaktische Analyse …
    - Eigentliches Revival der sog. *finite state methods* in *NLP* feststellbar!
  - Informatik: Compilertechnik, Kommunikationsprotokolle, Prozessmodellierung,…

… abstrakte Maschinenmodelle!

Beispiel: Ein Lachautomat

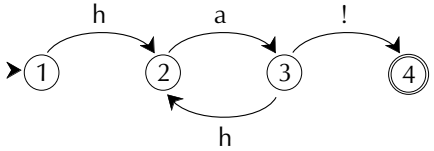
Graphische Repräsentation

- Zustandsübergangdiagramm (*transition diagram*)



Eingabe des Automaten

Der Lachautomat erhält eine Zeichenkette als Eingabe:



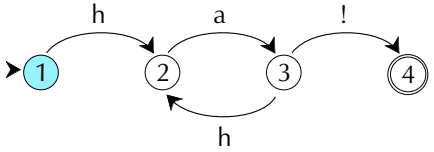
h a h a !

- ***Was macht der Automat damit?***

Beginn der Verarbeitung

Zu Beginn

- der Automat ist im Startzustand
- er schaut auf das erste Zeichen der Eingabe

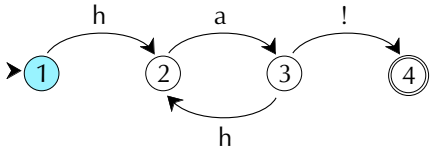


h a h a !

Ein einzelner Verarbeitungsschritt

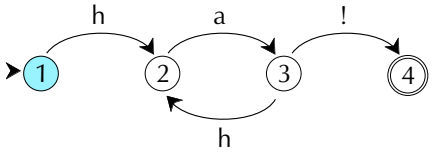
Der Automat nimmt jenen Übergang,

- der vom aktuellen Zustand ausgeht
- und mit jenem Zeichen in der Eingabekette beschriftet ist, auf das der Automat gerade schaut.



h a h a !

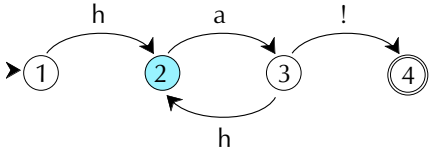
Ein einzelner Verarbeitungsschritt



h a h a !

Beim Nehmen eines Übergangs

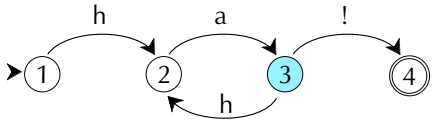
- springt der Automat in einen **neuen Zustand** und
- schaut auf das **nächste Zeichen** in der Eingabekette



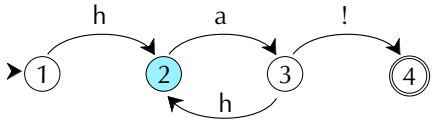
h **a** h a !

Abarbeiten der Eingabe

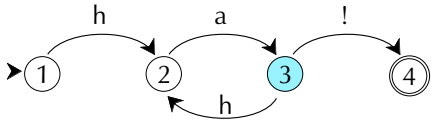
Der Automat konsumiert so Zeichen um Zeichen.



h a **h** a !



h a h **a** !

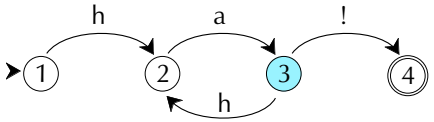


h a h a **!**

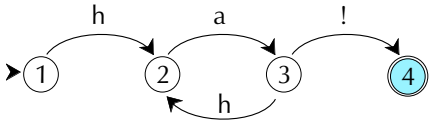
Abarbeiten der Eingabe

Der Automat konsumiert Zeichen um Zeichen,

- bis auch das letzte Zeichen der Eingabe konsumiert wurde.



h a h a !



h a h a !

Ende der Verarbeitung I

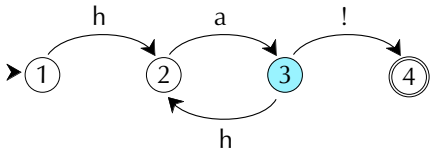
Wenn die Eingabe vollständig konsumiert ist, gibt es zwei Möglichkeiten

- der aktuelle Zustand ist **ein Endzustand** (4)
  - *Automat hat die Eingabe akzeptiert*
- der aktuelle Zustand ist **kein Endzustand** (3)
  - *Automat hat die Eingabe nicht akzeptiert*

- *Ein Automat kann mehrere Endzustände besitzen!*

Ende der Verarbeitung II

Kommt der Automat nicht weiter, weil kein Übergang zum aktuellen Eingabezeichen passt, ist die Eingabe ebenfalls *nicht* akzeptiert.



h a i !

Akzeptoren

Der Lachautomat ist ein Akzeptor.

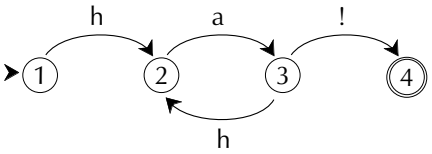
- Eingabe: Zeichenkette
- Ausgabe: »akzeptiert« oder »nicht akzeptiert«

ha! **hahaha!** **haha!** **hahahaha!**

Ausgabe: Ja

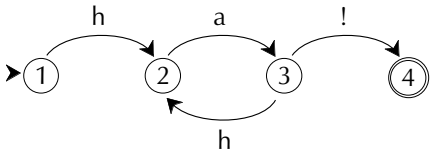
hi! **aha!** **ah!** **hahah!** **ha**

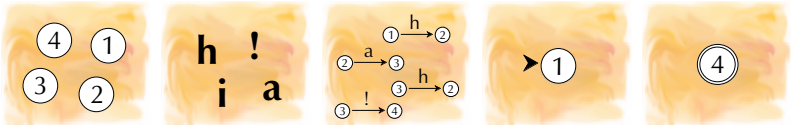
Ausgabe: Nein



Bestandteile

Bestandteile eines Endlichen Automaten





Zustände *States*

Alphabet *Alphabet*

Übergänge *Transitions*

Startzustand *Start State*

Endzustände *Final States*

Mengentheoretische Definition

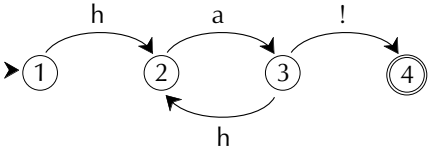
Ein Endlicher Automat ist ein Fünf-Tupel

$$\langle S,\ \Sigma,\ \delta,\ s,\ F \rangle$$

Zustände *Alphabet* *Übergänge* *Startzustand* *Endzustände*

- endliche, nicht leere Menge von Zuständen S
- Eingabe-Alphabet Σ
- partielle Übergangsfunktion δ: $(S \times \Sigma) \rightarrow S$
- Startzustand $s \in S$
- Menge von Endzuständen $F \subseteq S$

Mengentheoretischer Lachautomat



Dieser Automat sei ein 5-Tupel $\langle S, \Sigma, \delta, s, F \rangle$ mit

$S = \{1, 2, 3, 4\}$

$\Sigma = \{a, i, h, !\}$

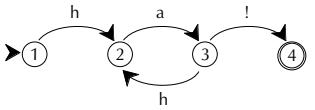
$\delta = \{\langle\langle 1, h\rangle, 2\rangle, \langle\langle 2, a\rangle, 3\rangle, \langle\langle 3, h\rangle, 2\rangle, \langle\langle 3, !\rangle, 4\rangle\}$

$s = 1$

$F = \{4\}$

EA-Akzeptor in Prolog I

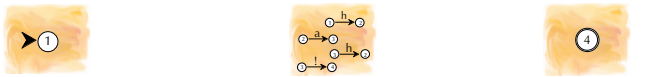
Die Struktur




```
start(1).
```

```
delta(1, h, 2).
delta(2, a, 3).
delta(3, h, 2).
delta(3, !, 4).
```

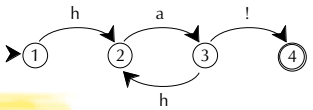
```
final(4).
```



Startzustand *Übergänge* *Endzustände*

EA-Akzeptor in Prolog II

Die Abarbeitung



- *Initialisierung*

```
init(String) :-
  start(StartState),
  accept(String, StartState).
```

- *Abarbeitung der Eingabekette*

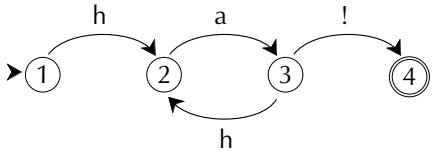
```
accept([], State) :-
  final(State).
accept([Char|Chars], State) :-
  delta(State, Char, NextState),
  accept(Chars, NextState).
```


(Deterministische) Endliche Automaten

Deterministische Endliche Automaten (DEA)
Deterministic Finite-State Automata (DFA)

- Von einem Zustand gehen nur Übergänge mit verschiedenen Beschriftungen aus.
- Jeder Übergang konsumiert ein Zeichen der Eingabekette.

▸ **Es kommt immer höchstens ein Übergang in Frage.**



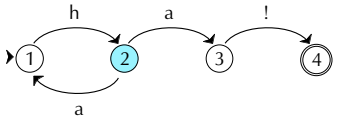
Nicht-deterministische Endliche Automaten

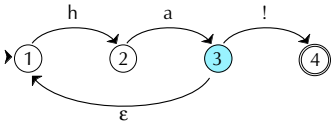
Nicht-deterministische Endliche Automaten (NEA)
Non-deterministic Finite-State Automata (NFA)

- **Mehrere gleich beschriftete** Übergänge von einem Zustand möglich
- ***ε*-Übergänge** (*epsilon*) möglich, bei denen kein Eingabesymbol konsumiert wird

▸ Mehrere Übergänge können gewählt werden.

▸ **Trotzdem**: Jeder NEA kann in einen DEA konvertiert werden!



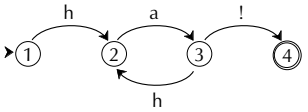


Sprache Endlicher Automaten

Definition: Sprache eines Endlichen Automaten

- Die Menge aller Eingabeketten, die von einem Endlichen Automaten *A* akzeptiert werden, heisst Sprache des Automaten *A*, meist geschrieben als *L*(*A*).

  L(Lachautomat) = {ha!, haha!, hahaha!, hahahaha!,…}

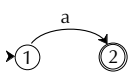


▸ *Die Sprachen Endlicher Automaten können unendlich viele Elemente enthalten!*

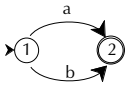
Endliche Automaten und Reguläre Ausdrücke

Die Sprachen, welche mit Endlichen Automaten erkannt werden können, heissen Regulären Sprachen.

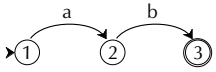
- Reguläre Sprachen können auch durch Reguläre Ausdrücke beschrieben werden:



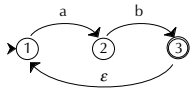
Einzelnes Symbol des Alphabets: **a**



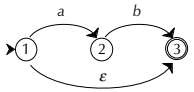
Alternative: **(a|b)**



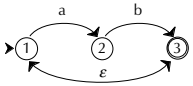
Verkettung: **ab**



Repetition: **(ab)+**



Optionalität: **(ab)?**

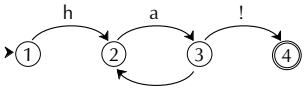


Optionale Repetition: **(ab)***

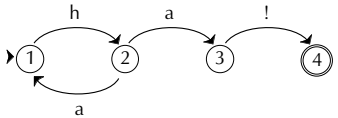
Lachen als Regulärer Ausdruck

Die Sprache, welche unser Lachautomat akzeptiert, kann als Regulärer Ausdruck spezifiziert werden.

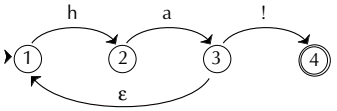
- *Achtung: Gewisse Diagramme lassen sich nicht 1:1 übertragen!*
- *Unterschiedliche Umformungen sind oft möglich!*



ha(ha)*!



h(ah)*a!



(ha)+!

Endliche Automaten Generatoren

Aus Regulären Ausdrücken lassen sich automatisch Endliche Automaten generieren, die die Sprache akzeptieren, welche die Regulären Ausdrücke beschreiben!

- In der Computerlinguistik oft verwendet, insbesondere für morphologische Verarbeitung
- Anwendungen in sogenannten `lex`-Werkzeugen, die für lexikalische Analyse beim Kompilieren von Programmiersprachen verwendet werden
- Anwendung beim Verabeiten von Suchmustern (*pattern matching*), die als Reguläre Ausdrücke angegeben werden. Z.B. in den Programmiersprachen Perl, JavaScript, Java, `grep`-Tools von UNIX, Suche in MS Word usw.

Literaturhinweise

Mathematische Grundlagen der Linguistik

- Barbara H. Partee/Alice ter Meulen/Robert E. Wall: *Mathematical Methods in Linguistics*. Dordrecht: Kluwer Academic Publishers, 1990.

  Ausführliche, gut verständliche Einführung in Mengenlehre, Logik, Algebra, Lambda-Kalkül, Automatentheorie. Empfehlenswert.

Verarbeitung Endlicher Automaten in Prolog

- Gerald Gazdar/Chris Mellis: *Natural Language Processing in PROLOG: An Introduction to Computational Linguistics*. Wokingham: Addison-Wesley, 1989. Seiten 21-59

  Programmierung einfacher computerlinguistischer Anwendungen mit EAs
- Wilhelm Weisweber: *Prolog: Logische Programmierung in der Praxis*: Thomson, 1997. Seiten 281-293

  Verarbeitung von EAs und Umwandlung von NEA zu minimalen DEA

Reguläre Ausdrücke, Endliche Automaten und Prolog

- `http://odur.let.rug.nl/~vannoord/prolog-rx/PrologAndRegex.html`