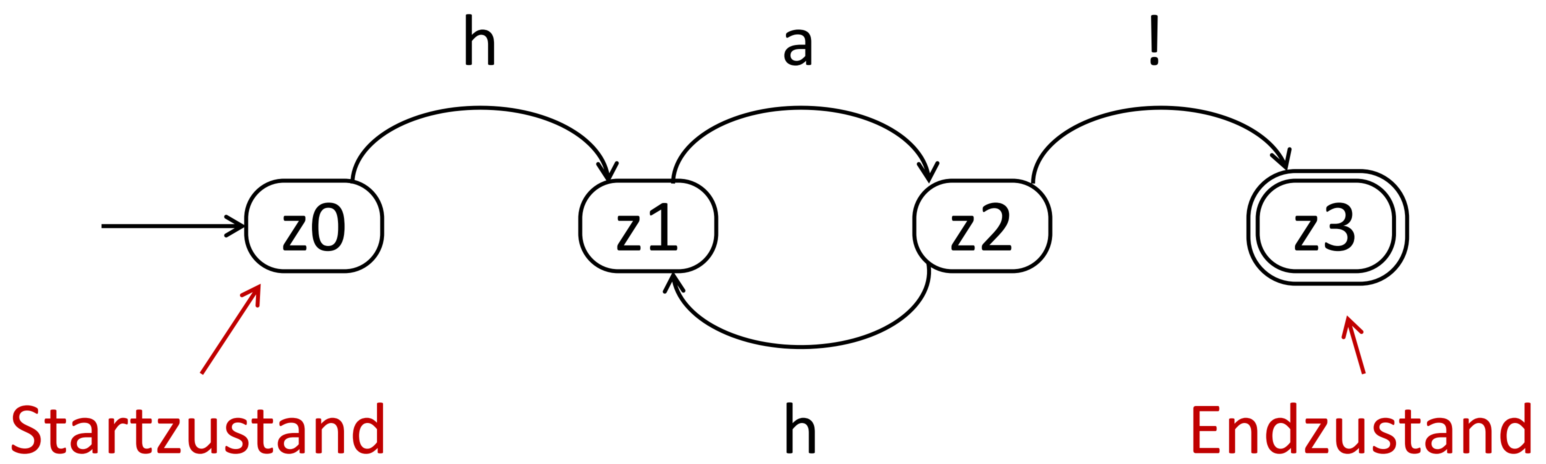
4 Erkennung formaler Sprachen

Lachautomat

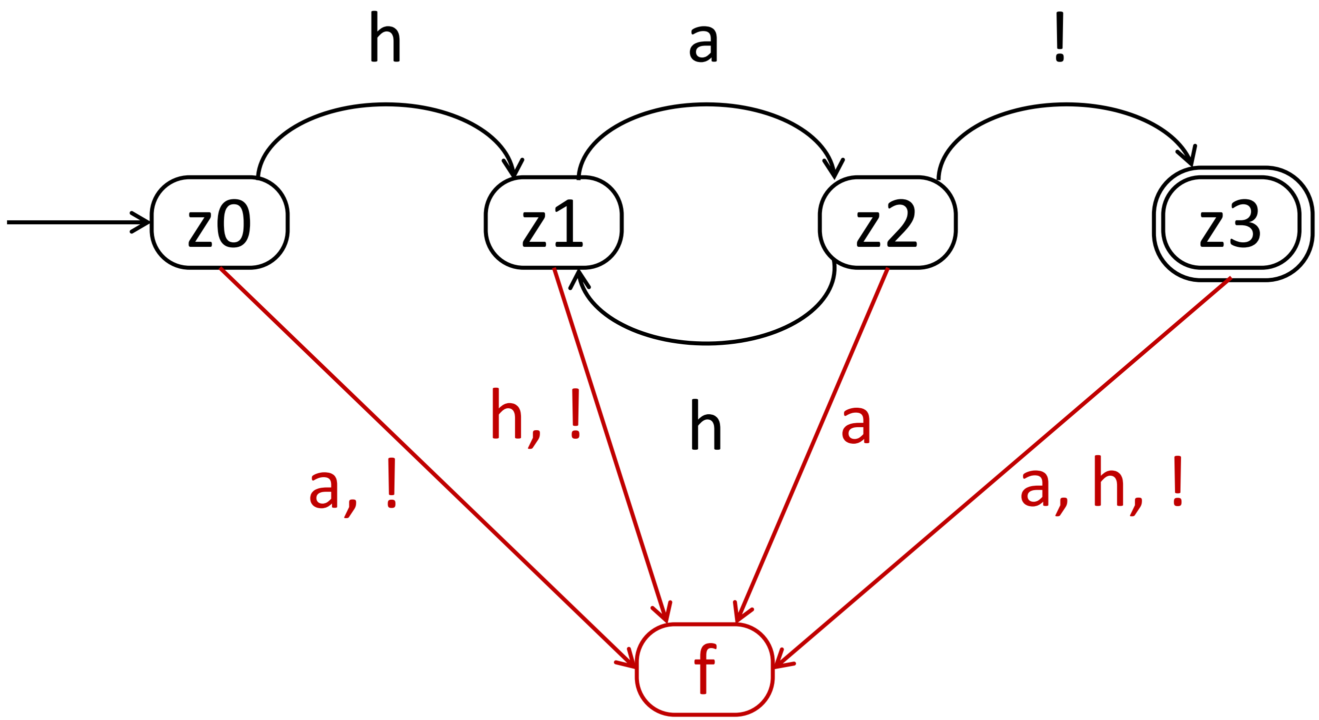


Akzeptiert:
ha!
haha!
hahahahaha!

Akzeptiert nicht:
aha!
haha!ha!
haha

Prüfe: ah!

Einführung eines Fehlerzustands f



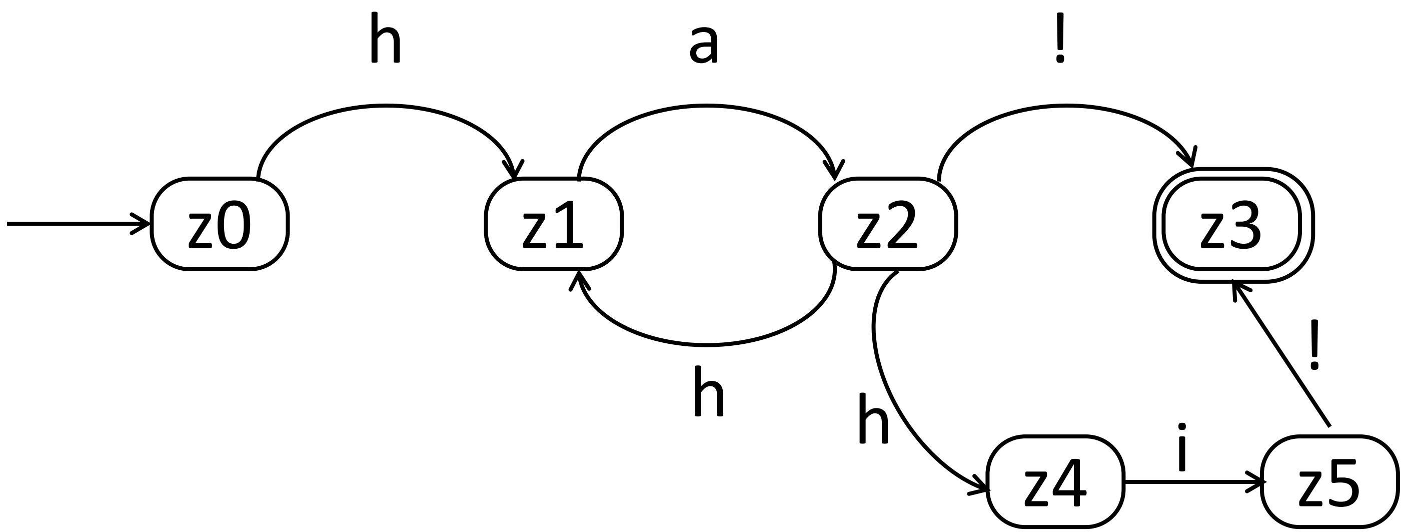
In der vereinfachten Darstellung notiert man nur Zustände und Transitionen, die für die Erkennung aller korrekten Wörter notwendig sind. Fehlerzustände werden weggelassen.

Ein Wort wird akzeptiert, wenn sich der Automat – beginnend im Startzustand – nach Abarbeitung des kompletten Wortes in einem Endzustand befindet.

Ein Automat, der die erfolgte Eingabe (z.B. haha!) auf Korrektheit überprüft, heißt **erkennender** Automat oder **Akzeptor**.

Endliche Automaten bestehen aus endlich vielen Zuständen und verarbeiten eine aus endlich vielen Zeichen bestehende Zeichenkette.

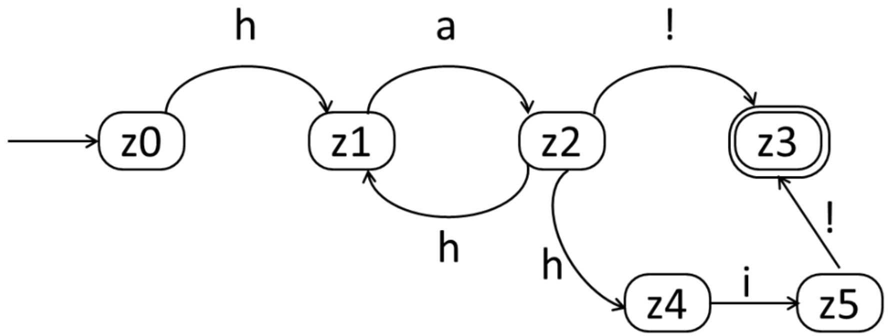
Lachautomat (erweitert)



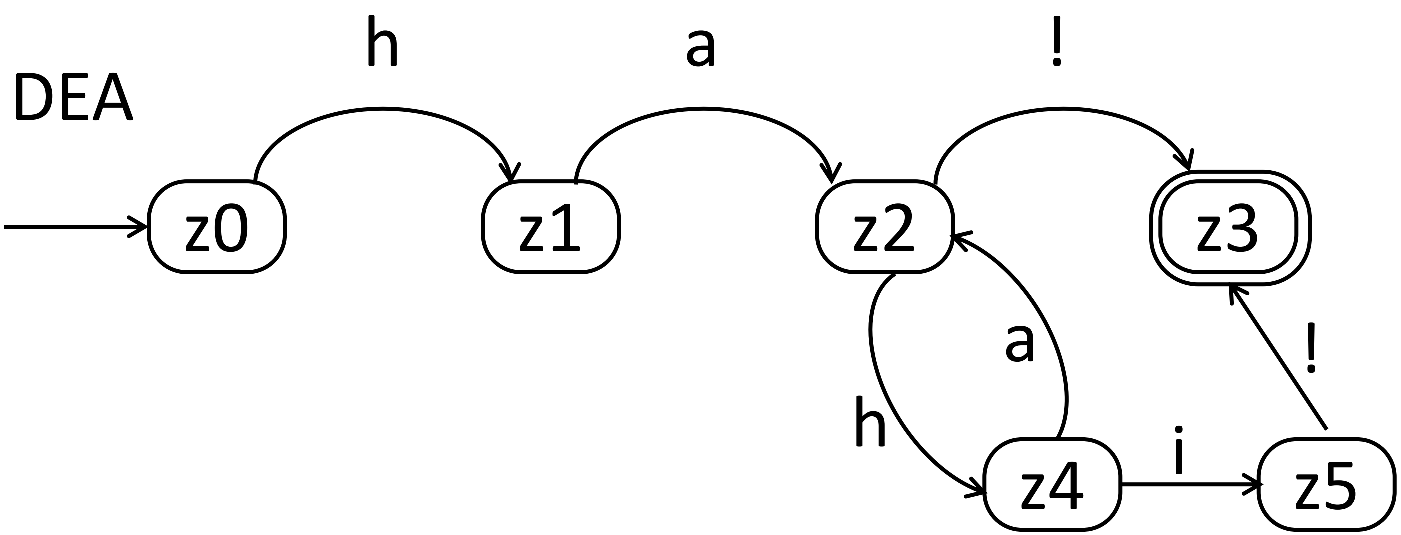
Wenn es mindestens einen Zustand (hier: z2) gibt, von dem aus ein Eingabezeichen (h) eine Transition zu verschiedenen Folgezuständen auslösen kann, ist der Automat nichtdeterministisch (NEA).
Wir implementieren nur DEA (deterministische, endliche Automaten).

Zu jedem NEA gibt es einen DEA, der die gleiche Sprache akzeptiert. Entwickle einen DEA für den erweiterten Lachautomaten.

NEA



DEA



Definition: Deterministischer endlicher Automat (DEA)

Jeder DEA besteht aus 5 Komponenten:

1. endliche Menge Z von Zuständen,
2. endliches Eingabealphabet Σ,
3. einem Startzustand,
4. Menge E von Endzuständen
5. zweistellige Übergangsfunktion δ, die jedem möglichen Paar aus Zustand und Eingabezeichen einen Folgezustand zuordnet.

Lachautomat:

Z = {z0, z1, z2, z3}

Σ = {a, h, !}

Startzustand: z0

E = {z3}

δ(z0, 'h') = z1;

δ(z1, 'a') = z2;

δ(z2, 'h') = z1;

δ(z2, '!') = z3;

Äquivalente Grammatik:

V = {<z0>, <z1>, <z2>, <z3>}

Σ = {a, h, !}

Startsymbol: <z0>

P:

<z0> → 'h' <z1>;

<z1> → 'a' <z2>;

<z2> → 'h' <z1>;

<z2> → '!' <z3>;

<z3> → ε;

(oder: <z2> → '!';
falls von <z3> keine weitere Produktion ausgeht.
Dann ist <z3> überflüssig.)

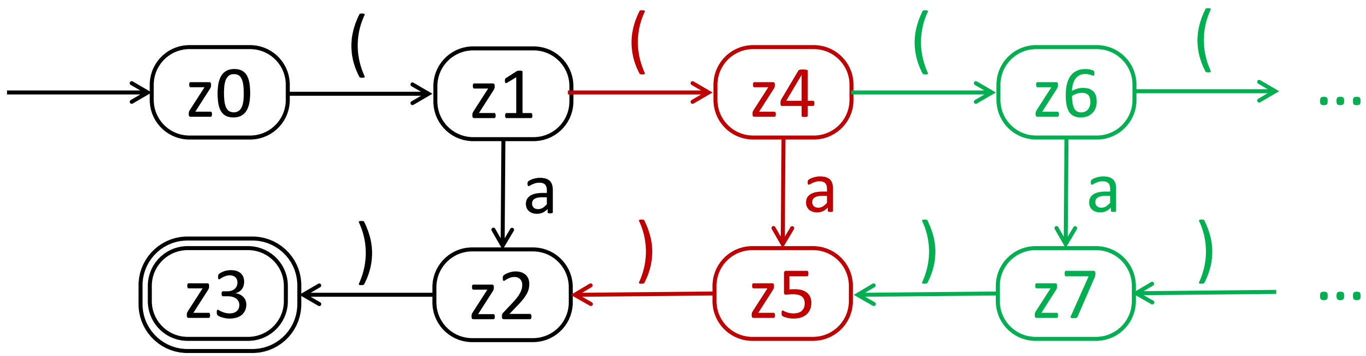
Sprachen, die von einem endlichen Automaten erkannt werden, heißen **reguläre Sprachen**.
Die Produktionsregeln entsprechender regulärer Grammatiken können stets in folgender Form angegeben werden:

<A> → 'a' <B>;
<A> → 'a'; Rechtslineare Grammatik
<A> → ε;

(<A>, <B> beliebige Nichtterminale, 'a' Terminal)

Nicht alle formalen Sprachen sind regulär: (S.26)

L1 = { (a) }   L2 = { (a), ((a)) }   Ln = { (a), ((a)), (((a))), ... }



Für Ln werden unendlich viele Zustände benötigt!

Reguläre Sprachen

Kontextfreie Sprachen

<S> → '(' <A> ')';
<A> → '(' <A> ')' | 'a';

Kontextsensitive Sprachen

<S> → a b c
<S> → a <A> b <B> c c
<B> → <A> b <B> c
<B> → b
b <A> → <A> b
a <A> → a a

<S> → a b c

<S> → a <A> b <B> c c

<B> → <A> b <B> c

<B> → b

b <A> → <A> b

a <A> → a a

Ableitung von aaabbbccc:

<S> → **a <A>** b <B> c c

→ a a **b <A>** b <B> c c c

→ a **a <A>** b b b c c c

→ a a a b b b c c c

<S> → a b c

<S> → a <A> b <B> c c

<B> → <A> b <B> c

<B> → b

b <A> → <A> b

a <A> → a a

Alle Wörter die von der Grammatik erzeugt werden, enthalten gleich oft die Buchstaben a, b, c (in dieser Reihenfolge, hintereinander).

Ableitung von aaaabbbbccc:

<S> → **a <A>** b <B> c c

→ a a **b <A>** b <B> c c c

→ a **a <A>** b **b <A>** b <B> c c c c

→ a a a **b <A>** b b b c c c c

→ a a **a <A>** b b b b c c c c

→ a a a a b b b b c c c c

Chomsky-Hierarchie