3.0 VU Formale Modellierung

Gernot Salzer

Forschungsbereich Theory and Logic
Institut für Logic and Computation

13.3.2018

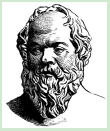
Was Sie letztes Mal hörten

Organisatorisches

1. tiss.tuwien.ac.at
2. tuwel.tuwien.ac.at
3. „Infos zu Ablauf und Organisation“ (in TUWEL)

Was Sie letztes Mal hörten

Die Logik untersucht allgemeine Prinzipien korrekten Schließens.



Alle Menschen sind sterblich.
Sokrates ist ein Mensch.

Sokrates ist sterblich.

} Prämissen, Annahmen

Konklusion, Folgerung

Inferenzregel

Alle x sind y.
z ist ein x.
---
z ist y.

x . . . Mensch
y . . . sterblich
z . . . Sokrates

Kriterium für die Gültigkeit von Inferenzregeln

Immer wenn alle Prämissen wahr sind, ist auch die Konklusion wahr.

Unzulässige Inferenzen

Alle Menschen sind sterblich.
Sokrates ist sterblich.

Sokrates ist ein Mensch.

Sokrates ist ein Mensch.
Sokrates ist sterblich.

Alle Menschen sind sterblich.

Alle Menschen sind sterblich.
Sokrates ist sterblich.

Sokrates ist ein Mensch.

Inferenzregel:

Alle x sind y.
z ist y.
---
z ist ein x.

- Diese Regel erfüllt nicht das Kriterium.
- Gegenbeispiel: $x =$ Fußball, $y =$ rund, $z =$ Sonne

Alle Fußbälle sind rund.	wahr
Die Sonne ist rund.	wahr
Die Sonne ist ein Fußball.	falsch

- Die Inferenzregel ist daher nicht gültig,
  obwohl sie gelegentlich zu wahren Konklusionen führt.
  Aber eben nicht immer!

Was Sie heute erwartet

Logische Junktoren (Operatoren, Konnektive, Funktionen)

. . . ermöglichen die Bildung zusammengesetzter Aussagen,
so wie Addition und Subtraktion bei arithmetischen Ausdrücken.

Wenn Feiertag ist oder der Professor krank ist,
findet die Vorlesung nicht statt.

- „es ist Feiertag“ (x), „Prof ist krank“ (y), „VO findet statt“ (z)
  . . . elementare Aussagen aus dem Uni-Milieu
- Logische Struktur: „Wenn x oder y, dann nicht z“
- Junktoren: wenn-dann, oder, nicht

Weitere Junktoren in . . .

- der Aussagenlogik: und, entweder-oder, genau dann-wenn, . . .
- Zeitlogiken: morgen, gestern, im nächsten Moment, bis, . . .
- Modallogiken: notwendigerweise, möglicherweise, . . .
- . . .

Quantoren

. . . ermöglichen Aussagen über die Anzahl betroffener Individuen, Zeitpunkte etc.

Jeder Mann liebt eine Frau.

- Wertebereiche: Männer (x) , Frauen (y)
- Logische Struktur:
  „Für alle x gibt es mindestens ein y, sodass x y liebt."
- Quantoren: für alle, mindestens ein

Weitere Quantoren: einige, viele, mindestens fünf, höchstens drei, immer, manchmal, irgendwann später, . . .

Komponenten einer Logik

- logische Symbole, Variablen:
  notwendig für kompakte und unmissverständliche Schreibung
- Syntax: Regeln für Wohlgeformtheit
  Wann ist eine Folge logischer Symbole eine Formel?
- Semantik: Bedeutung von Formeln
  Welche Wahrheitswerte gibt es?
  Wann ist eine Formel wahr, wann falsch?
  Was bedeuten die Symbole?
- Konsequenzrelation:
  Wann folgt eine Formel logisch aus anderen Formeln?
- Inferenzregeln (logischer Kalkül)
  Wie lassen sich Formeln beweisen?

Unterscheidung von Logiken

... nach Wahrheitswerten:

- Zweiwertige Logik: wahr/falsch
- Mehrwertige Logiken: wahr/falsch/unbekannt/widersprüchlich, ...
- Fuzzy logic: [0,1] (alle reellen Zahlen zwischen 0 und 1)

... nach Quantoren:

- Aussagenlogik: keine Quantoren
- Quantifizierte Aussagenlogik: Quantoren über Aussagenvariablen
- Prädikatenlogik: Quantoren über Individuenvariablen
- Logiken höherer Stufen: Quantoren über Funktionen und Prädikate

... nach den Ausdrucksmöglichkeiten:

- Elementare Logik: und, oder, nicht, für alle, ...
- Zeitlogiken: im nächsten Moment, für immer, irgendwann später, ...
- Modallogiken: „ich glaube/weiß, dass“, „es ist möglich/notwendig, dass“, ...

... nach wahren/beweisbaren Formeln, nach Kalkülen, ...

Klassische Aussagenlogik (Propositionallogik)

- zwei Wahrheitswerte: wahr/falsch, true/false, verum/falsum, 1/0, ein/aus, ...
- Aussagenvariablen, die wahr oder falsch sein können
- elementare Operatoren wie „und“, „oder“, „nicht“, ...
- keine Quantoren

Geht zurück auf die Antike

Grundlegend für

- Philosophie
- Mathematik
- Informatik

Clipart courtesy FCIT

Aristoteles
384–322 v.Chr.

Was Sie heute erwartet

Negation

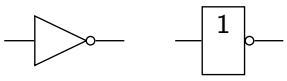
Ich gehe nicht ins Kino.

Ist falsch, wenn ich ins Kino gehe, und wahr andernfalls.

x	not x
1	0
0	1

Andere Bezeichnung: non

Symbole: $\neg x$, $-x$, $\sim x$, x', $!x$, $\overline{x}$, $\mathrm{N}x$, ...

Logikgatter: 

Konjunktion

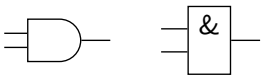
Der Himmel ist blau und die Sonne scheint.

Trifft nur zu, wenn jede der beiden Teilaussagen wahr ist.

x	y	x and y
1	1	1
1	0	0
0	1	0
0	0	0

Andere Bezeichnung: et

Symbole: $x \wedge y$, $x \cdot y$, xy, $x\&y$, $\mathrm{K}xy$, ...

Logikgatter: 

Disjunktion, Alternative

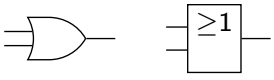
Ich trinke zum Essen Wein oder Bier (oder auch beides).

Nur falsch, wenn ich weder Wein noch Bier trinke.

x	y	x or y
1	1	1
1	0	1
0	1	1
0	0	0

Andere Bezeichnung: vel

Symbole: $x \vee y$, $x + y$, $x \mid y$, $\mathrm{A}xy$

Logikgatter: 

Ausschließende Disjunktion (Antivalenz)

Ich bin entweder gut drauf oder saugrantig,
etwas anderes gibt es bei mir nicht.

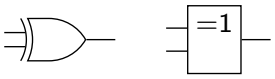
Trifft zu, wenn ich in genau einer der Stimmungslagen bin (die sich ausschließen).

x	y	x xor y
1	1	0
1	0	1
0	1	1
0	0	0

Andere Formulierungen: x oder y

Andere Bezeichnungen: exor, aut

Symbole: $x \not\equiv y$, $x \oplus y$, $x \not\Leftrightarrow y$, $x \not\leftrightarrow y$, Jxy, ...

Logikgatter: 

Äquivalenz

Ich springe dann (und nur dann), wenn du es auch tust.

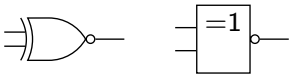
Trifft zu, wenn beide springen oder keiner.

x	y	x iff y
1	1	1
1	0	0
0	1	0
0	0	1

Andere Formulierungen: x genau dann wenn y, x if and only if y, x ist notwendig und hinreichend für y, x ist äquivalent zu y

Andere Bezeichnungen: eq, äq, xnor, nxor, ...

Symbole: $x \equiv y$, $x \Leftrightarrow y$, $x \leftrightarrow y$, Exy, ...

Logikgatter: 

Implikation

Wenn/Falls ich ins Kino gehe, (dann) esse ich dort Popcorn.
Ich gehe nur dann ins Kino, wenn ich dort Popcorn esse.

Falsch, wenn ich im Kino kein Popcorn esse, und wahr, wenn doch.
Keine Festlegung betreffend Popcorn außerhalb des Kinos, daher wahr.

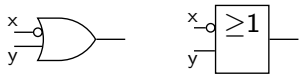
x	y	x implies y
1	1	1
1	0	0
0	1	1
0	0	1

„Verum ex quolibet“: x implies $1 = 1$
„Ex falso quodlibet“: 0 implies $y = 1$

Andere Formulierungen: aus x folgt y, x impliziert y, x hinreichend für y

Andere Bezeichnung: seq (sequi)

Symbole: $x \supset y$, $x \Rightarrow y$, $x \rightarrow y$, Cxy, ...

Logikgatter: 

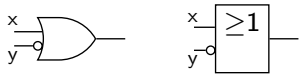
Implikation (Umkehrung)

Ich esse (dann) Popcorn, wenn/falls ich ins Kino gehe.

x	y	x if y
1	1	1
1	0	1
0	1	0
0	0	1

Andere Formulierungen: x folgt aus y, x wird von y impliziert,
x ist notwendig für y

Symbole: $x \subset y$, $x \Leftarrow y$, $x \leftarrow y$, ...

Logikgatter: 

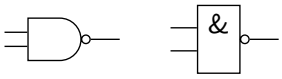
Negierte Konjunktion

x	y	x and y	x nand y
1	1	1	0
1	0	0	1
0	1	0	1
0	0	0	1

Andere Bezeichnungen:
Sheffer-Strich, nd (J.Nicod)

Symbole: $x \uparrow y$, $x \mid y$, x/y, Dxy, ...

Logikgatter:



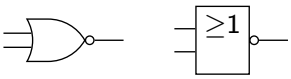
Negierte Disjunktion

x	y	x or y	x nor y
1	1	1	0
1	0	1	0
0	1	1	0
0	0	0	1

Andere Bezeichnungen:
Peirce-Pfeil, sh (H.M.Sheffer)

Symbole: $x \downarrow y$, Xxy, ...

Logikgatter:



Wenn Feiertag ist oder der Professor krank ist,
findet die Vorlesung nicht statt.

Logische Struktur: „Wenn x oder y, dann nicht z.“

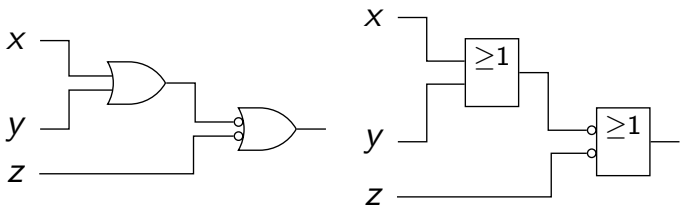
Logische Funktion: (x or y) implies not z implies (or (x, y), not(z))

Logische Formel: $(x \vee y) \supset \neg z$

Prefix-Notation: C A x y N z

Algebraische Notation: $(x + y) \rightarrow -z$ $\overline{x}\,\overline{y} + \overline{z}$

Logischer Schaltkreis:



Implikation oder Äquivalenz?

Natürlichsprachliche Implikationen sind oft logische Äquivalenzen.

Wenn du mein Auto wäscht, bekommst du 10 Euro.

Und was, wenn ich es nicht tue?
Nur ein Logiker hält in diesem Fall 10 Euro für möglich.
Alle anderen interpretieren den Satz als:

Du bekommst 10 Euro dann und nur dann, wenn du das Auto wäscht.

Der positive Erfolg bei allen Lehrveranstaltungen und Prüfungen der Studieneingangs- und Orientierungsphase berechtigt zur Absolvierung der weiteren Lehrveranstaltungen und Prüfungen sowie zum Verfassen der im Curriculum vorgesehenen Bachelor- oder Diplomarbeiten.
Universitätsgesetz 2002, Stand Bgbl I Nr. 13/2011, § 66(1a)

Logiker: Keine Einschränkung bei nicht bestandener STEOP.
Ministerium: Restliches Studium dann und nur dann, wenn STEOP.

Der Besitz eines Führerscheins berechtigt zum Lenken eines Autos.

Ohne Führerschein keine Berechtigung? (Äquivalenz)
Auch nicht auf Privatgelände? (Doch nur Implikation?)

In formalen Kontexten wird strikt zwischen Implikation und Äquivalenz unterschieden. „Implikation = halbe Äquivalenz“

Wenn eine Zahl durch 4 teilbar ist, ist sie gerade.

4-Teilbarkeit ist eine hinreichende Bedingung für Geradheit,
aber keine notwendige.

Äquivalenz führt zu einer falschen Aussage:

Eine Zahl ist durch 4 teilbar genau dann, wenn sie gerade ist.

2 ist eine gerade Zahl, aber nicht durch 4 teilbar.

Inklusive oder exklusive Disjunktion?

Natürlichsprachliche Disjunktionen sind meist ausschließend gemeint.

Falls du mich suchst: Ich bin zu Hause oder in der Arbeit.

Physikalisch kann ein Körper nicht an zwei Orten gleichzeitig sein.
Andererseits: Das Büro kann Teil der Wohnung sein.

„Tee oder Kaffee?“ – „Beides, bitte!“

Eher unüblich, aber der Gast ist König.

Ich besuche dich morgen oder übermorgen.

Ein Besuch an beiden Tagen wäre unerwartet.

Ich fahre entweder Auto oder höre Musik.
(Auf beides gleichzeitig kann ich mich nicht konzentrieren.)

Wirklich ein Beispiel für ausschließende Disjunktion?

Habe ich außerhalb des Autos tatsächlich keine ruhige Minute?

Die exklusive Disjunktion ist hier als Implikation gemeint (und wird auch so verstanden):

Wenn ich Auto fahre, höre ich nicht Musik.

Legt nicht fest, was ich außerhalb des Autos mache.

Rezept für Zweifelsfälle der aussagenlogischen Modellierung

1. Identifiziere die elementaren Aussagen.
2. Analysiere alle Wahrheitsbelegungen.
3. Wähle geeignete logische Funktionen
   (unbeirrt von Intuition und natürlicher Sprache).

z = Entweder „ich fahre Auto“ (x) oder „ich höre Musik“ (y).

x	y	z	x or y	x xor y	x implies not y	x nand y
1	1	0	1	0	0	0
1	0	1	1	1	1	1
0	1	1	1	1	1	1
0	0	1	0	0	1	1

x implies not y: Wenn ich Auto fahre, dann höre ich nicht Musik.

x nand y: Es kommt nicht vor, dass ich Auto fahre und Musik höre.

Aussagenlogische Funktionen – Überblick

true	false
1	0
$\top$	$\bot$

x	not
1	0
0	1
	$\neg$

x	y	and	nand	or	nor	iff	xor	implies		if	
1	1	1	0	1	0	1	0	1	0	1	0
1	0	0	1	1	0	0	1	0	1	1	0
0	1	0	1	1	0	0	1	1	0	0	1
0	0	0	1	0	1	1	0	1	0	1	0
		$\wedge$	$\uparrow$	$\vee$	$\downarrow$	$\equiv$	$\not\equiv$	$\supset$	$\not\supset$	$\subset$	$\not\subset$

2 nullstellige Funktionen (= Konstanten): true, false
4 einstellige Funktionen: not, ...
16 zweistellige Funktionen: and, nand, or, ...

Es gibt 2^{2^n} verschiedene n-stellige logische (Boolesche) Funktionen.

- 2^n verschiedene Argumentkombinationen („Zeilen“)
- 2 Ergebnismöglichkeiten für jede Argumentkombination

Funktionale Vollständigkeit

Eine Menge von Funktionen heißt vollständig (für eine Funktionsklasse), wenn damit alle Funktionen (der Klasse) ausgedrückt werden können.

$\{\text{not}, \text{and}, \text{or}\}$ ist funktional vollständig.

Begründung siehe später.

$\{\text{not}, \text{and}\}$ ist funktional vollständig.

- $\{\text{not}, \text{and}, \text{or}\}$ ist vollständig (siehe oben).
- or kann durch $\{\text{not}, \text{and}\}$ ausgedrückt werden kann:
  $x \text{ or } y = \text{not}(\text{not } x \text{ and not } y)$

x	y	not x	not y	not x and not y	not(not x and not y)
1	1	0	0	0	1
1	0	0	1	0	1
0	1	1	0	0	1
0	0	1	1	1	0

$\{\text{nand}\}$ ist funktional vollständig.

- $\{\text{not}, \text{and}\}$ ist funktional vollständig (siehe oben).
- not $x = x$ nand x
- x and $y = \text{not}(x \text{ nand } y) = (x \text{ nand } y) \text{ nand } (x \text{ nand } y)$

Praktisch relevant: nand ist einfach als Halbleiter-Schaltkreis realisierbar. Somit ist jede logische Funktion als Schaltkreis realisierbar.

Die Mengen $\{\text{nor}\}$, $\{\text{not}, \text{or}\}$, $\{\text{not}, \text{implies}\}$ und $\{\text{implies}, \text{false}\}$ sind ebenfalls funktional vollständig.

Was Sie heute erwartet

Syntax versus Semantik

Syntax:

- Zeichenfolge, mit der etwas notiert wird
- Regeln dafür, welche Zeichenfolgen zulässig sind

Semantik:

- Bedeutung einer Zeichenfolge
- Funktion, die jeder zulässigen Zeichenfolge eine Bedeutung zuordnet

Syntax und Semantik sind grundsätzlich voneinander unabhängig.
Die Bedeutung von Zeichen muss explizit vereinbart werden.

Syntax	Semantik
eins (deutsch) one (englisch) 1 (mathematisch)	das abstrakte Konzept der Zahl „1“
Bug	Schiffsvorderteil (deutsch) Käfer, Programmfehler (englisch)

and, nand ... mathematische Funktionen

x and y
$(x \text{ nand } y) \text{ nand } (x \text{ nand } y)$ } ... ununterscheidbar, identische Funktion:

x	y	x and y	$(x \text{ nand } y) \text{ nand } (x \text{ nand } y)$
1	1	1	1
1	0	0	0
0	1	0	0
0	0	0	0

Für Aussagen über die Form der Ausdrücke brauchen wir eine Formelsprache.

$x \wedge y$
$(x \uparrow y) \uparrow (x \uparrow y)$ } ... unterschiedliche Zeichenketten mit $\begin{matrix}3\\11\end{matrix}$ Symbolen

Induktive Definition unendlicher Mengen

Stufenweise Konstruktion der geraden Zahlen:

- 0 ist eine gerade Zahl: $G_0 = \{0\}$
- Addiert man zu geraden Zahlen 2, erhält man wieder gerade Zahlen:
  $G_1 = G_0 \cup \{ n+2 \mid n \in G_0 \} = \{0, 2\}$
  $G_2 = G_1 \cup \{ n+2 \mid n \in G_1 \} = \{0, 2, 4\}$
  $G_{i+1} = G_i \cup \{ n+2 \mid n \in G_i \} = \{0, 2, 4, \ldots, 2(i+1)\}$
- Die geraden Zahlen sind alle so konstruierten Zahlen:
  $\mathbb{G} = \lim_{i \to \infty} G_i = \bigcup_{i \geq 0} G_i$

Umständlich, aber konstruktiv: Beginnend mit G_0 lassen sich systematisch alle geraden Zahlen berechnen.

Beobachtung:

- $G_0 \subseteq \mathbb{G}$
- Wenn $n \in \mathbb{G}$, dann auch $n + 2 \in \mathbb{G}$.
- $\mathbb{G}$ ist die kleinste Menge mit diesen beiden Eigenschaften.

Induktive Definition der geraden Zahlen

$\mathbb{G}$ ist die kleinste Menge, für die gilt:

- $0 \in \mathbb{G}$
- Wenn $n \in \mathbb{G}$, dann auch $n + 2 \in \mathbb{G}$.

Kompakte Definition, aber nicht konstruktiv:
In den Bedingungen kommt die zu definierende Menge $\mathbb{G}$ selbst vor.

Beiden Methoden definieren dieselbe Menge.
(Nicht offensichtlich, Beweis erforderlich!).

Daher: „Use the best of both worlds."

- Definiere die Menge induktiv.
- Konstruiere benötigte Elemente stufenweise.

Anmerkung: Der Zusatz „ist kleinste Menge" ist wesentlich.
Die natürlichen Zahlen erfüllen ebenfalls beide Bedingungen, sind aber nicht die kleinste derartige Menge.

Induktive Definition – allgemeine Situation

$\mathcal{U}$ ... Universum, Menge aller relevanten Elemente
$\mathcal{M}_0 \subseteq \mathcal{U}$ ... Menge von Grundelementen
$f_1 : \mathcal{U}^{n_1} \mapsto \mathcal{U}$, $f_2 : \mathcal{U}^{n_2} \mapsto \mathcal{U}$, ... Konstruktionsfunktionen

Stufenweise Konstruktion der Menge $\mathcal{M}$

- $\mathcal{M}_{i+1} = \mathcal{M}_i \cup \{ f_1(x_1, \ldots, x_{n_1}) \mid x_1, \ldots, x_{n_1} \in \mathcal{M}_i \}$
  $\cup \{ f_2(x_1, \ldots, x_{n_2}) \mid x_1, \ldots, x_{n_2} \in \mathcal{M}_i \}$
  $\cup \ldots$
- $\mathcal{M} = \lim_{i \to \infty} \mathcal{M}_i = \bigcup_{i \geq 0} \mathcal{M}_i$

Induktive Definition der Menge $\mathcal{M}$

$\mathcal{M}$ ist die kleinste Menge, für die gilt:

- $\mathcal{M}_0 \subseteq \mathcal{M}$
- Wenn $x_1, \ldots, x_{n_1} \in \mathcal{M}$, dann $f_1(x_1, \ldots, x_{n_1}) \in \mathcal{M}$.
- Wenn $x_1, \ldots, x_{n_2} \in \mathcal{M}$, dann $f_2(x_1, \ldots, x_{n_2}) \in \mathcal{M}$.
- ...

Aussagenlogik – Syntax

Ausdrücke wie x and y und $(x$ nand $y)$ nand $(x$ nand $y)$ sind ununterscheidbar (gleiche Funktion!). Um Aussagen über ihre Form treffen zu können, benötigen wir eine Formelsprache.

$\mathcal{V} = \{A, B, C, \ldots, A_0, A_1, \ldots\}$ aussagenlogische Variablen

Syntax aussagenlogischer Formeln

Die Menge $\mathcal{A}$ der aussagenlogischen Formeln ist die kleinste Menge, für die gilt:

(a1) $\mathcal{V} \subseteq \mathcal{A}$ Variablen sind Formeln.

(a2) $\{\top, \bot\} \subseteq \mathcal{A}$ $\top$ und $\bot$ sind Formeln.

(a3) $\neg F \in \mathcal{A}$, wenn $F \in \mathcal{A}$. $\neg F$ ist eine Formel, falls F eine ist.

(a4) $(F * G) \in \mathcal{A}$, wenn $F, G \in \mathcal{A}$ und $* \in \{\wedge, \uparrow, \vee, \downarrow, \equiv, \not\equiv, \supset, \subset\}$.
$(F * G)$ ist eine Formel, falls F und G welche sind und $*$ ein binäres Op.symbol ist.

(a1) $\mathcal{V} \subseteq \mathcal{A}$

(a2) $\{\top, \bot\} \subseteq \mathcal{A}$

(a3) $\neg F \in \mathcal{A}$, wenn $F \in \mathcal{A}$.

(a4) $(F * G) \in \mathcal{A}$, wenn $F, G \in \mathcal{A}$ und $* \in \{\wedge, \uparrow, \vee, \downarrow, \equiv, \not\equiv, \supset, \subset\}$.

$((A \uparrow B) \uparrow (A \uparrow B))$ ist eine aussagenlogische Formel, weil:

1. A und B sind Formeln. (a1)
2. $(A \uparrow B)$ ist eine Formel, (a4)
   - da A und B Formeln sind (Punkt 1)
   - und $\uparrow$ ein binäres Operatorsymbol ist.
3. $((A \uparrow B) \uparrow (A \uparrow B))$ ist eine Formel, (a4)
   - da $(A \uparrow B)$ und $(A \uparrow B)$ Formeln sind (Punkt 2)
   - und $\uparrow$ ein binäres Operatorsymbol ist.

(a1) $\mathcal{V} \subseteq \mathcal{A}$

(a2) $\{\top, \bot\} \subseteq \mathcal{A}$

(a3) $\neg F \in \mathcal{A}$, wenn $F \in \mathcal{A}$.

(a4) $(F * G) \in \mathcal{A}$, wenn $F, G \in \mathcal{A}$ und $* \in \{\wedge, \uparrow, \vee, \downarrow, \equiv, \not\equiv, \supset, \subset\}$.

$A \wedge B$ ist keine aussagenlogische Formel.

- $\mathcal{A}$ ist die kleinste Menge mit den Eigenschaften (a1)–(a4), daher kann $\wedge$ nur aufgrund von (a4) in einer Formel vorkommen.
- Dann muss es aber auch ein Klammernpaar geben.
- $A \wedge B$ enthält $\wedge$, aber keine Klammern – Widerspruch.

Formelsyntax: Beispiel einer induktiven Definition

$\mathcal{A}$ ist die kleinste Menge, für die gilt:

(a1) $\mathcal{V} \subseteq \mathcal{A}$

(a2) $\{\top, \bot\} \subseteq \mathcal{A}$

(a3) $\neg F \in \mathcal{A}$, wenn $F \in \mathcal{A}$.

(a4) $(F * G) \in \mathcal{A}$, wenn $F, G \in \mathcal{A}$ und $* \in \{\wedge, \uparrow, \vee, \downarrow, \equiv, \not\equiv, \supset, \subset\}$.

$\mathcal{U}$ ... Menge aller Zeichenketten bestehend aus Variablen, Operatorsymbolen und Klammern

$\mathcal{V} \cup \{\text{„}\top\text{“}, \text{„}\bot\text{“}\}$ ... Grundelemente

$$\left.\begin{array}{l} f_\neg(F) = \text{„}\neg\text{“}\ F \\ f_\wedge(F,G) = \text{„(“}\ F\ \text{„}\wedge\text{“}\ G\ \text{„)“} \\ f_\uparrow(F,G) = \text{„(“}\ F\ \text{„}\uparrow\text{“}\ G\ \text{„)“} \\ \quad\vdots \\ f_\subset(F,G) = \text{„(“}\ F\ \text{„}\subset\text{“}\ G\ \text{„)“} \end{array}\right\} \ldots \text{ Konstruktionsfunktionen}$$

Aussagenlogik – Semantik

$((A \wedge \neg B) \supset \bot)$ – wahr oder falsch?

Hängt ab

- vom Wert der Variablen A und B und
- von der Bedeutung der Symbole $\wedge$, $\neg$, $\supset$ und $\bot$.

Interpretationen

$\mathbb{B} = \{1, 0\}$ ... Wahrheitswerte
$I \colon \mathcal{V} \mapsto \mathbb{B}$ ... Wahrheitsbelegung, Interpretation
$\mathcal{I} = \{\, I \mid I \colon \mathcal{V} \mapsto \mathbb{B} \,\}$ ... Menge aller Interpretationen

$I(A) = I(C) = 1$ und $I(v) = 0$ sonst
„Die elementaren Aussagen A und C sind wahr, die übrigen sind falsch."

Semantik aussagenlogischer Formeln

Der Wert einer Formel in einer Interpretation I wird festgelegt durch die Funktion $\mathrm{val} \colon \mathcal{I} \times \mathcal{A} \mapsto \mathbb{B}$:

(v1) $\mathrm{val}_I(A) = I(A)$ für $A \in \mathcal{V}$;

(v2) $\mathrm{val}_I(\top) = 1$ und $\mathrm{val}_I(\bot) = 0$;

(v3) $\mathrm{val}_I(\neg F) = \mathrm{not}\, \mathrm{val}_I(F)$;

(v4) $\mathrm{val}_I(\,(F * G)\,) = \mathrm{val}_I(F) \circledast \mathrm{val}_I(G)$,
wobei $\circledast$ die logische Funktion zum Operator $*$ ist.

(v4) ist eine Abkürzung für:

$$\begin{aligned}
\mathrm{val}_I(\,(F \wedge G)\,) &= \mathrm{val}_I(F) \text{ and } \mathrm{val}_I(G) \\
\mathrm{val}_I(\,(F \vee G)\,) &= \mathrm{val}_I(F) \text{ or } \mathrm{val}_I(G) \\
\mathrm{val}_I(\,(F \equiv G)\,) &= \mathrm{val}_I(F) \text{ iff } \mathrm{val}_I(G) \\
\mathrm{val}_I(\,(F \supset G)\,) &= \mathrm{val}_I(F) \text{ implies } \mathrm{val}_I(G) \\
&\vdots
\end{aligned}$$

(v1) $\mathrm{val}_I(A) = I(A)$ für $A \in \mathcal{V}$;

(v2) $\mathrm{val}_I(\top) = 1$ und $\mathrm{val}_I(\bot) = 0$;

(v3) $\mathrm{val}_I(\neg F) = \mathrm{not}\,\mathrm{val}_I(F)$;

(v4) $\mathrm{val}_I(\,(F * G)\,) = \mathrm{val}_I(F) \circledast \mathrm{val}_I(G)$,
wobei $\circledast$ die logische Funktion zum Operator $*$ ist.

Wert von $((A \wedge \neg B) \supset \bot)$ für $I(A) = 1$ und $I(B) = 0$

$$
\begin{aligned}
\mathrm{val}_I(((A \wedge \neg B) \supset \bot)) &= \mathrm{val}_I((A \wedge \neg B)) \text{ implies } \mathrm{val}_I(\bot) \\
&= (\mathrm{val}_I(A) \text{ and } \mathrm{val}_I(\neg B)) \text{ implies } 0 \\
&= (1 \text{ and not } \mathrm{val}_I(B)) \text{ implies } 0 \\
&= (1 \text{ and not } 0) \text{ implies } 0 \\
&= (1 \text{ and } 1) \text{ implies } 0 \\
&= 1 \text{ implies } 0 = 0
\end{aligned}
$$

Wahrheitstafel

- Kompakte Berechnung der Formelwerte für alle Interpretationen
- Unter jedem Operator steht der Wert der entsprechenden Teilformel.

A	B	$((A \wedge \neg B) \supset \bot)$
1	1	1 0 0 1 1 0
1	0	1 1 1 0 0 0
0	1	0 0 0 1 1 0
0	0	0 0 1 0 1 0

bedeutet:

$I(A) = 1$, $I(B) = 1$: $\mathrm{val}_I(\cdots) = \cdots = 1$

$I(A) = 1$, $I(B) = 0$: $\mathrm{val}_I(\cdots) = \cdots = 0$

$I(A) = 0$, $I(B) = 1$: $\mathrm{val}_I(\cdots) = \cdots = 1$

$I(A) = 0$, $I(B) = 0$: $\mathrm{val}_I(\cdots) = \cdots = 1$

false
0
$\bot$

x	not
1	0
0	1
	$\neg$

x	y	and	implies
1	1	1	1
1	0	0	0
0	1	0	1
0	0	0	1
		$\wedge$	$\supset$

Eine Formel F heißt

- gültig, wenn $\text{val}_I(F) = 1$ für alle $I \in \mathcal{I}$; „Tautologie“
- erfüllbar, wenn $\text{val}_I(F) = 1$ für mindestens ein $I \in \mathcal{I}$;
- widerlegbar, wenn $\text{val}_I(F) = 0$ für mindestens ein $I \in \mathcal{I}$;
- unerfüllbar, wenn $\text{val}_I(F) = 0$ für alle $I \in \mathcal{I}$. „Kontradiktion“

Folgerungen:

- Eine gültige Formel ist erfüllbar, aber weder widerlegbar noch unerfüllbar.
- Eine erfüllbare Formel kann gültig oder widerlegbar sein, aber nicht unerfüllbar.
- Eine widerlegbare Formel kann erfüllbar oder unerfüllbar sein, aber nicht gültig.
- Eine unerfüllbare Formel ist widerlegbar, aber weder gültig noch erfüllbar.
- F ist gültig/erfüllbar/widerlegbar/unerfüllbar genau dann, wenn $\neg F$ unerfüllbar/widerlegbar/erfüllbar/gültig ist.

$((A \wedge \neg B) \supset \bot)$ ist erfüllbar und widerlegbar.

A	B	$((A \wedge \neg B) \supset \bot)$
1	1	1 0 0 1 1 0
1	0	1 1 1 0 0 0
0	1	0 0 0 1 1 0
0	0	0 0 1 0 1 0

Die Formel ist

- erfüllbar (daher nicht unerfüllbar),
- widerlegbar (daher nicht gültig).

$(A \vee \neg A)$ ist gültig und erfüllbar.

A	$(A \vee \neg A)$
1	1 1 0 1
0	0 1 1 0

Die Formel ist

- gültig (daher nicht widerlegbar),
- erfüllbar (daher nicht unerfüllbar).

$(A \wedge \neg A)$ ist unerfüllbar und widerlegbar.

A	$(A \wedge \neg A)$
1	1 0 0 1
0	0 0 1 0

Die Formel ist

- unerfüllbar (daher nicht erfüllbar),
- widerlegbar (daher nicht gültig).

Semantische Äquivalenz

Zwei Formeln F und G heißen äquivalent, geschrieben $F = G$,
wenn $\mathrm{val}_I(F) = \mathrm{val}_I(G)$ für alle Interpretationen I gilt.

$\neg(A \wedge B)$ und $(\neg A \vee \neg B)$ sind äquivalent

A	B	$\neg(A \wedge B) = (\neg A \vee \neg B)$
1	1	0 1 1 1 ✓ 0 1 0 0 1
1	0	1 1 0 0 ✓ 0 1 1 1 0
0	1	1 0 0 1 ✓ 1 0 1 0 1
0	0	1 0 0 0 ✓ 1 0 1 1 0

Äquivalenz bleibt bei der Ersetzung von Variablen durch Formeln erhalten.

$\neg(\quad A \quad \wedge \; B\;) = (\neg \quad A \quad \vee \neg\, B\;)$ $\qquad [A \mapsto (C \vee D),\, B \mapsto \neg D]$

Ersetzen einer Teilformel durch eine äquivalente liefert eine äquiv. Formel.

$(A \supset \neg(A \wedge B))$ $\qquad \neg(A \wedge B) = (\neg A \vee \neg B)$

Semantische Äquivalenz

Zwei Formeln F und G heißen äquivalent, geschrieben $F = G$,
wenn $\mathrm{val}_I(F) = \mathrm{val}_I(G)$ für alle Interpretationen I gilt.

$\neg(A \wedge B)$ und $(\neg A \vee \neg B)$ sind äquivalent

A	B	$\neg(A \wedge B) = (\neg A \vee \neg B)$
1	1	0 1 1 1 ✓ 0 1 0 0 1
1	0	1 1 0 0 ✓ 0 1 1 1 0
0	1	1 0 0 1 ✓ 1 0 1 0 1
0	0	1 0 0 0 ✓ 1 0 1 1 0

Äquivalenz bleibt bei der Ersetzung von Variablen durch Formeln erhalten.

$\neg((C \vee D) \wedge \neg D) = (\neg(C \vee D) \vee \neg\neg D)$ $\qquad [A \mapsto (C \vee D),\, B \mapsto \neg D]$

Ersetzen einer Teilformel durch eine äquivalente liefert eine äquiv. Formel.

$(A \supset \neg(A \wedge B)) = (A \supset (\neg A \vee \neg B))$ $\qquad \neg(A \wedge B) = (\neg A \vee \neg B)$

$\langle \mathbb{B}, \text{and}, \text{or}, \text{not}, 0, 1 \rangle$ ist eine Boolesche Algebra

Das heißt, es gelten folgende Gleichungen.

$(A \wedge B) \wedge C = A \wedge (B \wedge C)$	$(A \vee B) \vee C = A \vee (B \vee C)$	Assoziativität
$A \wedge B = B \wedge A$	$A \vee B = B \vee A$	Kommutativität
$A \wedge A = A$	$A \vee A = A$	Idempotenz
$A \wedge \top = A$	$A \vee \bot = A$	Neutralität
$A \wedge \neg A = \bot$	$A \vee \neg A = \top$	Komplement
$A \wedge (A \vee B) = A$	$A \vee (A \wedge B) = A$	Absorption
$A \wedge (B \vee C) = (A \wedge B) \vee (A \wedge C)$		Distributivität
$A \vee (B \wedge C) = (A \vee B) \wedge (A \vee C)$		

Schreibvereinfachung: keine Außenklammern, keine Klammern bei geschachteltem $\wedge$ oder $\vee$ (Assoziativität!)

$$A \wedge B \wedge C = ((A \wedge B) \wedge C) = (A \wedge (B \wedge C))$$

$\langle 2^M, \cap, \cup, \overline{\ }, \emptyset, M \rangle$ ... Beispiel einer anderen Booleschen Algebra
2^M ... Menge aller Teilmengen der Menge M, Potenzmenge
$\overline{X}$ ... Komplement der Menge X bzgl. M

3.0 VU Formale Modellierung

Gernot Salzer

Forschungsbereich Theory and Logic
Institut für Logic and Computation

20.3.2018

Was Sie letztes Mal hörten

Unterscheidung von Logiken

... nach Wahrheitswerten:

- Zweiwertige Logik: wahr/falsch
- Mehrwertige Logiken: wahr/falsch/unbekannt/widersprüchlich, ...
- Fuzzy logic: [0,1] (alle reellen Zahlen zwischen 0 und 1)

... nach Quantoren:

- Aussagenlogik: keine Quantoren
- Quantifizierte Aussagenlogik: Quantoren über Aussagenvariablen
- Prädikatenlogik: Quantoren über Individuenvariablen
- Logiken höherer Stufen: Quantoren über Funktionen und Prädikate

... nach den Ausdrucksmöglichkeiten:

- Elementare Logik: und, oder, nicht, für alle, ...
- Zeitlogiken: im nächsten Moment, für immer, irgendwann später, ...
- Modallogiken: „ich glaube/weiß, dass“, „es ist möglich/notwendig, dass“, ...

... nach wahren/beweisbaren Formeln, nach Kalkülen, ...

Was Sie letztes Mal hörten

Aussagenlogische Funktionen – Überblick

true	false
1	0
$\top$	$\bot$

x	not
1	0
0	1
	$\neg$

x	y	and	nand	or	nor	iff	xor	implies		if	
1	1	1	0	1	0	1	0	1	0	1	0
1	0	0	1	1	0	0	1	0	1	1	0
0	1	0	1	1	0	0	1	1	0	0	1
0	0	0	1	0	1	1	0	1	0	1	0
		$\wedge$	$\uparrow$	$\vee$	$\downarrow$	$\equiv$	$\not\equiv$	$\supset$	$\not\supset$	$\subset$	$\not\subset$

Eine Menge von Funktionen heißt vollständig (für eine Funktionsklasse), wenn damit alle Funktionen (der Klasse) ausgedrückt werden können.

$\{\text{not}, \text{and}, \text{or}\}$ ist vollständig für die aussagenlogischen Funktionen.

Begründung siehe später.

Die Mengen $\{\text{not}, \text{and}\}$, $\{\text{nand}\}$, $\{\text{not}, \text{or}\}$, $\{\text{nor}\}$, $\{\text{not}, \text{implies}\}$ und $\{\text{implies}, \text{false}\}$ sind ebenfalls funktional vollständig.

5

Was Sie letztes Mal hörten

Induktive Definitionen

$\mathcal{U}$ ... Universum, Menge aller relevanten Elemente
$\mathcal{M}_0 \subseteq \mathcal{U}$ ... Menge von Grundelementen
$f_1 : \mathcal{U}^{n_1} \mapsto \mathcal{U}$, $f_2 : \mathcal{U}^{n_2} \mapsto \mathcal{U}$, ... Konstruktionsfunktionen

Stufenweise Konstruktion der Menge $\mathcal{M}$

- $\mathcal{M}_{i+1} = \mathcal{M}_i \cup \{ f_1(x_1, \ldots, x_{n_1}) \mid x_1, \ldots, x_{n_1} \in \mathcal{M}_i \}$
  $\cup \{ f_2(x_1, \ldots, x_{n_2}) \mid x_1, \ldots, x_{n_2} \in \mathcal{M}_i \}$
  $\cup \ \ldots$
- $\mathcal{M} = \lim_{i \to \infty} \mathcal{M}_i = \bigcup_{i \geq 0} \mathcal{M}_i$

Induktive Definition der Menge $\mathcal{M}$

$\mathcal{M}$ ist die kleinste Menge, für die gilt:

- $\mathcal{M}_0 \subseteq \mathcal{M}$
- Wenn $x_1, \ldots, x_{n_1} \in \mathcal{M}$, dann $f_1(x_1, \ldots, x_{n_1}) \in \mathcal{M}$.
- Wenn $x_1, \ldots, x_{n_2} \in \mathcal{M}$, dann $f_2(x_1, \ldots, x_{n_2}) \in \mathcal{M}$.
- ...

Beispiel: Geschachtelte Klammern

Gesucht: Spezifikation aller richtig geschachtelten Folgen von runden, eckigen und geschwungenen Klammern, wie etwa `([[{()}]])`

Induktive Definition:
Die Menge $\mathcal{K}$ der Klammernfolgen ist die kleinste Menge, für die gilt:

(k1) `()`, `[]`, `{}` $\in \mathcal{K}$ alternativ: $\{$`()`, `[]`, `{}`$\} \subseteq \mathcal{K}$

(k2) Wenn $x \in \mathcal{K}$, dann auch `(`x`)` $\in \mathcal{K}$.

(k3) Wenn $x \in \mathcal{K}$, dann auch `[`x`]` $\in \mathcal{K}$.

(k4) Wenn $x \in \mathcal{K}$, dann auch `{`x`}` $\in \mathcal{K}$.

Wir zeigen, dass `([[{()}]])` in der Menge $\mathcal{K}$ liegt.

1. `()` $\in \mathcal{K}$ wegen (k1)
2. Da `()` $\in \mathcal{K}$, gilt auch `{()}` $\in \mathcal{K}$. wegen (k4)
3. Da `{()}` $\in \mathcal{K}$, gilt auch `[{()}]` $\in \mathcal{K}$. wegen (k3)
4. Da `[{()}]` $\in \mathcal{K}$, gilt auch `[[{()}]]` $\in \mathcal{K}$. wegen (k3)
5. Da `[[{()}]]` $\in \mathcal{K}$, gilt auch `([[{()}]])` $\in \mathcal{K}$. wegen (k2)

Aussagenlogik – Syntax

$\mathcal{V} = \{A, B, C, \ldots, A_0, A_1, \ldots\}$ aussagenlogische Variablen

Syntax aussagenlogischer Formeln

Die Menge $\mathcal{A}$ der aussagenlogischen Formeln ist die kleinste Menge, für die gilt:

(a1) $\mathcal{V} \subseteq \mathcal{A}$ Variablen sind Formeln.

(a2) $\{\top, \bot\} \subseteq \mathcal{A}$ $\top$ und $\bot$ sind Formeln.

(a3) $\neg F \in \mathcal{A}$, wenn $F \in \mathcal{A}$. $\neg F$ ist eine Formel, falls F eine ist.

(a4) $(F * G) \in \mathcal{A}$, wenn $F, G \in \mathcal{A}$ und $* \in \{\wedge, \uparrow, \vee, \downarrow, \equiv, \not\equiv, \supset, \subset\}$.
$(F * G)$ ist eine Formel, falls F und G welche sind und $*$ ein binäres Op.symbol ist.

$\neg(((A \vee B) \equiv \neg(B \downarrow C)) \subset ((A \vee \bot) \wedge B)) \in \mathcal{A}$

Aussagenlogik – Semantik

$\mathcal{I} = \{ I \mid I \colon \mathcal{V} \mapsto \mathbb{B} \}$ ... Menge aller Interpretationen

Semantik aussagenlogischer Formeln

Der Wert einer Formel in einer Interpretation I wird festgelegt durch die Funktion val: $\mathcal{I} \times \mathcal{A} \mapsto \mathbb{B}$:

(v1) $\mathrm{val}_I(A) = I(A)$ für $A \in \mathcal{V}$;

(v2) $\mathrm{val}_I(\top) = 1$ und $\mathrm{val}_I(\bot) = 0$;

(v3) $\mathrm{val}_I(\neg F) = \mathrm{not}\,\mathrm{val}_I(F)$;

(v4) $\mathrm{val}_I(\,(F * G)\,) = \mathrm{val}_I(F) \circledast \mathrm{val}_I(G)$,
wobei $\circledast$ die logische Funktion zum Operator $*$ ist.

A	B	$((A \wedge \neg B) \supset \bot)$
1	1	1 0 0 1 1 0
1	0	1 1 1 0 0 0
0	1	0 0 0 1 1 0
0	0	0 0 1 0 1 0

bedeutet:

$I(A) = 1, I(B) = 1$: $\mathrm{val}_I(\cdots) = \cdots = 1$

$I(A) = 1, I(B) = 0$: $\mathrm{val}_I(\cdots) = \cdots = 0$

$I(A) = 0, I(B) = 1$: $\mathrm{val}_I(\cdots) = \cdots = 1$

$I(A) = 0, I(B) = 0$: $\mathrm{val}_I(\cdots) = \cdots = 1$

Semantische Äquivalenz

Zwei Formeln F und G heißen äquivalent, geschrieben $F = G$, wenn $\mathrm{val}_I(F) = \mathrm{val}_I(G)$ für alle Interpretationen I gilt.

$(A \supset B)$ und $(\neg A \vee B)$ sind äquivalent

A	B	$(A \supset B)$	$=$	$(\neg A \vee B)$
1	1	1	✓	0 1
1	0	0	✓	0 0
0	1	1	✓	1 1
0	0	1	✓	1 1

Was Sie heute erwartet

$\langle \mathbb{B}, \text{and}, \text{or}, \text{not}, 0, 1 \rangle$ ist eine Boolesche Algebra

Das heißt, es gelten folgende Gleichungen.

$(A \wedge B) \wedge C = A \wedge (B \wedge C)$	$(A \vee B) \vee C = A \vee (B \vee C)$	Assoziativität
$A \wedge B = B \wedge A$	$A \vee B = B \vee A$	Kommutativität
$A \wedge A = A$	$A \vee A = A$	Idempotenz
$A \wedge \top = A$	$A \vee \bot = A$	Neutralität
$A \wedge \neg A = \bot$	$A \vee \neg A = \top$	Komplement
$A \wedge (A \vee B) = A$	$A \vee (A \wedge B) = A$	Absorption
$A \wedge (B \vee C) = (A \wedge B) \vee (A \wedge C)$		Distributivität
$A \vee (B \wedge C) = (A \vee B) \wedge (A \vee C)$		

Schreibvereinfachung: keine Außenklammern, keine Klammern bei geschachteltem $\wedge$ oder $\vee$ (Assoziativität!)

$$A \wedge B \wedge C = ((A \wedge B) \wedge C) = (A \wedge (B \wedge C))$$

$\langle 2^M, \cap, \cup, \overline{\ }, \emptyset, M \rangle$ ... Beispiel einer anderen Booleschen Algebra
2^M ... Menge aller Teilmengen der Menge M, Potenzmenge
$\overline{X}$ ... Komplement der Menge X bzgl. M

Weitere Äquivalenzen

Ersetzen von Junktoren durch $\wedge$, $\vee$ und $\neg$

$$A \uparrow B = \neg A \vee \neg B \qquad A \equiv B = (\neg A \vee B) \wedge (A \vee \neg B)$$
$$A \downarrow B = \neg A \wedge \neg B \qquad = (A \wedge B) \vee (\neg A \wedge \neg B)$$
$$A \supset B = \neg A \vee B \qquad A \not\equiv B = (\neg A \vee \neg B) \wedge (A \vee B)$$
$$A \subset B = A \vee \neg B \qquad = (A \wedge \neg B) \vee (\neg A \wedge B)$$

Verschieben der Negation

$$\left.\begin{array}{l} \neg(A \wedge B) = \neg A \vee \neg B \\ \neg(A \vee B) = \neg A \wedge \neg B \end{array}\right\} \text{ De Morgan Regeln} \qquad \neg\neg A = A$$

Äquivalenzen für $\top$ und $\bot$

$$A \wedge \top = A \qquad A \wedge \bot = \bot \qquad A \wedge \neg A = \bot \qquad \neg\top = \bot$$
$$A \vee \bot = A \qquad A \vee \top = \top \qquad A \vee \neg A = \top \qquad \neg\bot = \top$$

$$
\begin{aligned}
& (X \uparrow Y) \uparrow (\top \uparrow Y) && A \uparrow B = \neg A \vee \neg B \\
= {} & (\neg X \vee \neg Y) \uparrow (\neg \top \vee \neg Y) && A \uparrow B = \neg A \vee \neg B \\
= {} & \neg(\neg X \vee \neg Y) \vee \neg(\neg \top \vee \neg Y) && \neg(A \vee B) = \neg A \wedge \neg B \\
= {} & (\neg\neg X \wedge \neg\neg Y) \vee (\neg\neg \top \wedge \neg\neg Y) && \neg\neg A = A \\
= {} & (X \wedge Y) \vee (\top \wedge Y) && A \wedge \top = A, \text{ Komm. } \wedge \\
= {} & (X \wedge Y) \vee Y && A \vee (A \wedge B) = A, \text{ Komm. } \wedge, \vee \\
= {} & Y
\end{aligned}
$$

Logische Konsequenz

$F_1, \ldots, F_n \models_I G$: „Aus $\mathrm{val}_I(F_1) = \cdots = \mathrm{val}_I(F_n) = 1$ folgt $\mathrm{val}_I(G) = 1$.“

„Falls in der Wahrheitsbelegung I alle Prämissen wahr sind,
dann ist auch die Konklusion wahr in I.“

$I(A)$	$I(B)$	A	$A \vee B$	$\models_I$	B
1	1	1	1	✓	1
1	0	1	1	↯	0
0	1	0	1	✓	1
0	0	0	0	✓	0

Logische Konsequenz

$F_1, \ldots, F_n \models G$: $F_1, \ldots, F_n \models_I G$ gilt für alle Interpretationen I.

„Die Formel G ist eine logische Konsequenz der Formeln $F_1, \ldots, F_n$.“

„Die Formel G folgt aus den Formeln $F_1, \ldots, F_n$.“

Konvention: „$\models G$“ ($n = 0$) bedeutet „G ist gültig.“

$A, A \vee B \models B$? Nein!

$I(A)$	$I(B)$	A	$A \vee B$	$\models_I$	B
1	1	1	1	✓	1
1	0	1	1	↯	0
0	1	0	1	✓	1
0	0	0	0	✓	0

$I(A) = 1$, $I(B) = 0$:
Es gilt $\mathrm{val}_I(A) = \mathrm{val}_I(A \vee B) = 1$,
aber $\mathrm{val}_I(B) \neq 1$!

I heißt Gegenbeispiel.

$A, A \supset B \models B$? Ja!

$I(A)$	$I(B)$	A	$A \supset B$	$\models_I$	B
1	1	1	1	✓	1
1	0	1	0	✓	0
0	1	0	1	✓	1
0	0	0	1	✓	0

$$\frac{A \qquad A \supset B}{B} \qquad \frac{x \qquad \text{Wenn } x\text{, dann } y.}{y}$$

Ist eine gültige Inferenzregel!

Kriterium für die Gültigkeit von Inferenzregeln

Immer wenn alle Prämissen wahr sind, ist auch die Konklusion wahr.

Äquivalenz, Konsequenz und Gültigkeit

Die Formeln F und G sind äquivalent ($F = G$) genau dann,
wenn $F \equiv G$ eine gültige Formel ist.

Deduktionstheorem

G folgt aus $F_1, \ldots, F_n$ genau dann, wenn $F_n \supset G$ aus $F_1, \ldots, F_{n-1}$ folgt.

$F_1, \ldots, F_n \models G$ genau dann, wenn $F_1, \ldots, F_{n-1} \models F_n \supset G$.

Mehrfache Anwendung liefert:

$F_1, \ldots, F_n \models G$ genau dann, wenn $F_1 \supset (F_2 \supset \cdots (F_n \supset G) \cdots)$ gültig.

Wegen $A \supset (B \supset C) = (A \wedge B) \supset C$ erhalten wir weiters:

$F_1, \ldots, F_n \models G$ genau dann, wenn $(F_1 \wedge \cdots \wedge F_n) \supset G$ gültig.

Das heißt: Semantik ($=$ und $\models$) ausdrückbar in der Syntax ($\equiv$ und $\supset$).

Ist nicht in jeder Logik möglich!

Was Sie heute erwartet

Rezept für Zweifelsfälle der aussagenlogischen Modellierung

1. Identifiziere die elementaren Aussagen.
2. Analysiere alle Wahrheitsbelegungen.
3. Wähle geeignete logische Funktionen
   (unbeirrt von Intuition und natürlicher Sprache)

Klingt ja nicht schlecht, aber:

Wie kann man eine beliebige Funktion auf eine Kombination der bekannten logischen Grundfunktionen zurückführen?

Beziehungsweise:

Wie kann man eine beliebige Funktion mit den bekannten Operatoren als Formel darstellen?

Gesucht: Ein allgemeines Verfahren (ein Algorithmus), das zu einer gegebenen Funktion eine passende Formel liefert.

Von der Funktion zur Formel

Gegeben: Funktion $f : \mathbb{B}^n \mapsto \mathbb{B}$ (z.B. als Wahrheitstafel)
Gesucht: Formel, die f darstellt

A	B	C	$F[A,B,C]?$
x	y	z	$f(x,y,z)$
1	1	1	1
1	1	0	0
1	0	1	0
1	0	0	1
0	1	1	1
0	1	0	0
0	0	1	0
0	0	0	0

Von der Funktion zur Formel

Gegeben: Funktion $f : \mathbb{B}^n \mapsto \mathbb{B}$ (z.B. als Wahrheitstafel)
Gesucht: Formel, die f darstellt

A	B	C	$F[A,B,C] := (A \wedge B \wedge C) \vee (A \wedge \neg B \wedge \neg C) \vee (\neg A \wedge B \wedge C)$			
x	y	z	$f(x,y,z)$			
1	1	1	1	1	0	0
1	1	0	0	0	0	0
1	0	1	0	0	0	0
1	0	0	1	0	1	0
0	1	1	1	0	0	1
0	1	0	0	0	0	0
0	0	1	0	0	0	0
0	0	0	0	0	0	0

DNF_f ... „Disjunktive Normalform zur Funktion f“

Von der Funktion zur Formel

Gegeben: Funktion $f : \mathbb{B}^n \mapsto \mathbb{B}$ (z.B. als Wahrheitstafel)
Gesucht: Formel, die f darstellt

A	B	C	$F[A,B,C]$?
x	y	z	$f(x,y,z)$
1	1	1	1
1	1	0	0
1	0	1	0
1	0	0	1
0	1	1	1
0	1	0	0
0	0	1	0
0	0	0	0

Von der Funktion zur Formel

Gegeben: Funktion $f : \mathbb{B}^n \mapsto \mathbb{B}$ (z.B. als Wahrheitstafel)
Gesucht: Formel, die f darstellt

A	B	C	$F[A,B,C] := (\neg A \vee \neg B \vee C) \wedge (\neg A \vee B \vee \neg C) \wedge (A \vee \neg B \vee C) \wedge \cdots$				
x	y	z	$f(x,y,z)$				
1	1	1	1	1	1	1	1 1
1	1	0	0	0	1	1	1 1
1	0	1	0	1	0	1	1 1
1	0	0	1	1	1	1	1 1
0	1	1	1	1	1	1	1 1
0	1	0	0	1	1	0	1 1
0	0	1	0	1	1	1	0 1
0	0	0	0	1	1	1	1 0

KNF_f ... „Konjunktive Normalform zur Funktion f“

Von der Funktion $f : \mathbb{B}^n \mapsto \mathbb{B}$ zur Formel DNF_f

Notation: $\bigwedge\{F, G, H, \dots\} = F \wedge G \wedge H \wedge \cdots \qquad \bigwedge\{\} = \top$
$\bigvee\{F, G, H, \dots\} = F \vee G \vee H \vee \cdots \qquad \bigvee\{\} = \bot$

Charakteristisches Konjunkt für $\vec{b} = (b_1, \dots, b_n) \in \mathbb{B}^n$:
$K_{\vec{b}} = \bigwedge\{ A_i \mid b_i = 1,\ i = 1..n \} \wedge \bigwedge\{ \neg A_i \mid b_i = 0,\ i = 1..n \}$

$\vec{b} = (1, 0, 1, 1) \quad \Longrightarrow \quad K_{\vec{b}} = A_1 \wedge \neg A_2 \wedge A_3 \wedge A_4$

$I_{\vec{b}}$ ... Interpretation definiert durch $I_{\vec{b}}(A_i) = b_i$

$K_{\vec{b}}$ hat den Wert 1 für $I_{\vec{b}}$, und 0 für alle anderen Interpretationen.

$I_{\vec{b}}$: $A_1 \mapsto 1,\ A_2 \mapsto 0,\ A_3 \mapsto 1,\ A_4 \mapsto 1 \quad \Longrightarrow \quad \mathrm{val}_{I_{\vec{b}}}(K_{\vec{b}}) = 1$

Disjunktive Normalform für $f : \mathbb{B}^n \mapsto \mathbb{B}$

$\mathrm{DNF}_f = \bigvee\{ K_{\vec{b}} \mid f(\vec{b}) = 1,\ \vec{b} \in \mathbb{B}^n \}$ repräsentiert die Funktion f, d.h.:
$\mathrm{val}_{I_{\vec{b}}}(\mathrm{DNF}_f) = f(\vec{b})$ für alle $\vec{b} \in \mathbb{B}^n$.

Von der Funktion $f : \mathbb{B}^n \mapsto \mathbb{B}$ zur Formel KNF_f

Notation: $\bigwedge\{F, G, H, \dots\} = F \wedge G \wedge H \wedge \cdots \qquad \bigwedge\{\} = \top$
$\bigvee\{F, G, H, \dots\} = F \vee G \vee H \vee \cdots \qquad \bigvee\{\} = \bot$

Charakteristisches Disjunkt für $\vec{b} = (b_1, \dots, b_n) \in \mathbb{B}^n$:
$D_{\vec{b}} = \bigvee\{ A_i \mid b_i = 0,\ i = 1..n \} \vee \bigvee\{ \neg A_i \mid b_i = 1,\ i = 1..n \}$

$\vec{b} = (1, 0, 1, 1) \quad \Longrightarrow \quad D_{\vec{b}} = \neg A_1 \vee A_2 \vee \neg A_3 \vee \neg A_4$

$I_{\vec{b}}$ ... Interpretation definiert durch $I_{\vec{b}}(A_i) = b_i$

$D_{\vec{b}}$ hat den Wert 0 für $I_{\vec{b}}$, und 1 für alle anderen Interpretationen.

$I_{\vec{b}}$: $A_1 \mapsto 1,\ A_2 \mapsto 0,\ A_3 \mapsto 1,\ A_4 \mapsto 1 \quad \Longrightarrow \quad \mathrm{val}_{I_{\vec{b}}}(D_{\vec{b}}) = 0$

Konjunktive Normalform für $f : \mathbb{B}^n \mapsto \mathbb{B}$

$\mathrm{KNF}_f = \bigwedge\{ D_{\vec{b}} \mid f(\vec{b}) = 0,\ \vec{b} \in \mathbb{B}^n \}$ repräsentiert die Funktion f, d.h.:
$\mathrm{val}_{I_{\vec{b}}}(\mathrm{KNF}_f) = f(\vec{b})$ für alle $\vec{b} \in \mathbb{B}^n$.

A_1	A_2	A_3						
	$\vec{b}$		$f(\vec{b})$	$K_{\vec{b}}$			$D_{\vec{b}}$	
1	1	1	1	$A_1 \wedge$	$A_2 \wedge$	$A_3 =: K_{111}$		
1	1	0	0				$\neg A_1 \vee \neg A_2 \vee$	$A_3 =: D_{110}$
1	0	1	0				$\neg A_1\vee$	$A_2 \vee \neg A_3 =: D_{101}$
1	0	0	1	$A_1 \wedge \neg A_2 \wedge \neg A_3 =: K_{100}$				
0	1	1	1	$\neg A_1 \wedge$	$A_2 \wedge$	$A_3 =: K_{011}$		
0	1	0	0				$A_1 \vee \neg A_2 \vee$	$A_3 =: D_{010}$
0	0	1	0				$A_1 \vee$	$A_2 \vee \neg A_3 =: D_{001}$
0	0	0	0				$A_1 \vee$	$A_2 \vee$ $A_3 =: D_{000}$

$\text{DNF}_f = K_{111} \vee K_{100} \vee K_{011}$

$\text{KNF}_f = D_{110} \wedge D_{101} \wedge D_{010} \wedge D_{001} \wedge D_{000}$

Folgerung:

$\{\text{not}, \text{and}, \text{or}\}$ ist funktional vollständig.

Was Sie heute erwartet

Normalformen

Literal: Variable oder negierte Variable, also A, $\neg A$, B, $\neg B$, ...

Negationsnormalform (NNF)

- Literale sowie $\top$ und $\bot$ sind in NNF.
- $(F \wedge G)$ und $(F \vee G)$ sind in NNF, wenn F und G in NNF sind.
- Keine Formel sonst ist in NNF.

NNF: $(\neg A \vee ((B \vee \neg C) \wedge \top))$ Keine NNFs: $\neg\neg A$, $\neg(A \wedge B)$, $\neg\bot$
DNF_f und KNF_f sind Formeln in NNF.

Disjunktive Normalform (DNF)

$\top$, $\bot$ sowie Disjunktionen von Konjunktion von Literalen:
$(^{(\neg)}A_{1,1} \wedge {}^{(\neg)}A_{1,2} \wedge {}^{(\neg)}A_{1,3} \wedge \cdots) \vee (^{(\neg)}A_{2,1} \wedge {}^{(\neg)}A_{2,2} \wedge {}^{(\neg)}A_{2,3} \wedge \cdots) \vee \cdots$

Konjunktive Normalform (KNF)

$\top$, $\bot$ sowie Konjunktionen von Disjunktion von Literalen:
$(^{(\neg)}A_{1,1} \vee {}^{(\neg)}A_{1,2} \vee {}^{(\neg)}A_{1,3} \vee \cdots) \wedge (^{(\neg)}A_{2,1} \vee {}^{(\neg)}A_{2,2} \vee {}^{(\neg)}A_{2,3} \vee \cdots) \wedge \cdots$

Normalformen

Formeln, die gleichzeitig in DNF und KNF sind:

- $\top$
- $\bot$
- $^{(\neg)}A_1 \wedge {}^{(\neg)}A_2 \wedge \cdots \wedge {}^{(\neg)}A_n$
- $^{(\neg)}A_1 \vee {}^{(\neg)}A_2 \vee \cdots \vee {}^{(\neg)}A_n$

Normalformen für die Funktion f von vorhin

$\mathrm{DNF}_f = K_{111} \vee K_{100} \vee K_{011}$	kanonische (maximale) DNF, NNF
$(A_2 \wedge A_3) \vee (A_1 \wedge \neg A_2 \wedge \neg A_3)$	minimale DNF, NNF
$\mathrm{KNF}_f = D_{110} \wedge D_{101} \wedge D_{010} \wedge D_{001} \wedge D_{000}$	kanonische KNF, NNF
$(A_1 \vee A_3) \wedge (\neg A_2 \vee A_3) \wedge (A_2 \vee \neg A_3)$	minimale KNF, NNF
$(A_1 \vee A_2) \wedge (\neg A_2 \vee A_3) \wedge (A_2 \vee \neg A_3)$	andere minimale KNF, NNF

Normalformen sind in der Regel nicht eindeutig.
Typische Problemstellung: Finde kleine oder kleinste Normalform.

Normalformen

Weitere Normalformen:

- Beschränkung auf andere Operatoren, etwa $\uparrow$
- Andere Einschränkungen der Struktur, etwa
  Konjunktion von Disjunktionen von Konjunktionen von Literalen
  (ermöglicht kleinere Formeln als DNF oder KNF)

Noch mehr Normalformen für die Funktion f von vorhin

$(A_2 \uparrow A_3) \uparrow (A_1 \uparrow ((A_2 \uparrow A_2) \uparrow (A_3 \uparrow A_3) \uparrow (A_2 \uparrow A_2) \uparrow (A_3 \uparrow A_3)))$

$((A_1 \wedge \neg A_3) \vee A_2) \wedge (\neg A_2 \vee A_3)$ NNF

Konstruktion von DNFs/KNFs – Semantische Methode

Gegeben: Aussagenlogische Formel F
Gesucht: Äquivalente Formel in DNF/KNF

1. Stelle die zu F gehörige Funktion f als Wahrheitstafel dar.
2. Konstruiere DNF_f bzw. KNF_f.

A_1	A_2	A_3	$F := (A_1 \supset (A_2 \equiv A_3)) \wedge (\neg A_1 \supset (A_2 \wedge A_3))$	
1	1	1	1	K_{111}
1	1	0	0	D_{110}
1	0	1	0	D_{101}
1	0	0	1	K_{100}
0	1	1	1	K_{011}
0	1	0	0	D_{010}
0	0	1	0	D_{001}
0	0	0	0	D_{000}

DNF: $F = K_{111} \vee K_{100} \vee K_{011}$
KNF: $F = D_{110} \wedge D_{101} \wedge D_{010} \wedge D_{001} \wedge D_{000}$

Konstruktion von DNFs/KNFs – Algebraische Methode

Gegeben: Aussagenlogische Formel F
Gesucht: Äquivalente Formel in DNF/KNF

1. Ersetze alle Junktoren durch $\wedge$, $\vee$ und $\neg$.
   $A \uparrow B = \neg A \vee \neg B \qquad A \downarrow B = \neg A \wedge \neg B \qquad A \supset B = \neg A \vee B \qquad A \subset B = A \vee \neg B$
   $A \equiv B = (\neg A \vee B) \wedge (A \vee \neg B) = (A \wedge B) \vee (\neg A \wedge \neg B)$
   $A \not\equiv B = (\neg A \vee \neg B) \wedge (A \vee B) = (A \wedge \neg B) \vee (\neg A \wedge B)$
2. Verschiebe Negationen nach innen, eliminiere Doppelnegationen.
   $\neg(A \wedge B) = \neg A \vee \neg B \qquad \neg(A \vee B) = \neg A \wedge \neg B \qquad \neg\neg A = A$
3. Wende das Distributivgesetz an.
   DNF: Schiebe Disjunktionen nach außen mittels
   $A \wedge (B \vee C) = (A \wedge B) \vee (A \wedge C)$
   KNF: Schiebe Konjunktionen nach außen mittels
   $A \vee (B \wedge C) = (A \vee B) \wedge (A \vee C)$
4. Eliminiere $\top$ und $\bot$.
   $A \wedge \top = A \qquad A \wedge \bot = \bot \qquad A \wedge \neg A = \bot \qquad \neg\top = \bot$
   $A \vee \bot = A \qquad A \vee \top = \top \qquad A \vee \neg A = \top \qquad \neg\bot = \top$

(Äquivalenzen werden hier von links nach rechts angewendet.)

$((A_1 \uparrow A_2) \supset \neg A_2) \wedge (\neg A_1 \supset (A_2 \wedge \bot))$

1. Ersetze alle Junktoren durch $\wedge$, $\vee$ und $\neg$:
   $(\ \ (\neg A_1 \vee \neg A_2) \supset \neg A_2) \wedge (\ \neg A_1 \supset (A_2 \wedge \bot))$
   $(\neg(\neg A_1 \vee \ \neg A_2) \vee \neg A_2) \wedge (\ \neg A_1 \supset (A_2 \wedge \bot))$
   $(\neg(\neg A_1 \vee \ \neg A_2) \vee \neg A_2) \wedge (\neg\neg A_1 \vee (A_2 \wedge \bot))$
2. Verschiebe Negationen nach innen, eliminiere Doppelnegationen:
   $((\neg\neg A_1 \wedge \neg\neg A_2) \vee \neg A_2) \wedge (\neg\neg A_1 \vee (A_2 \wedge \bot))$
   $(\quad (A_1 \wedge \quad A_2) \vee \neg A_2) \wedge (\quad A_1 \vee (A_2 \wedge \bot))$
3. Wende das Distributivgesetz an:
   DNF: $(((A_1 \wedge A_2) \vee \neg A_2) \wedge A_1) \vee (((A_1 \wedge A_2) \vee \neg A_2) \wedge (A_2 \wedge \bot))$
   $(((A_1 \wedge A_2) \vee \neg A_2) \wedge A_1) \vee (A_1 \wedge A_2 \wedge A_2 \wedge \bot) \vee (\neg A_2 \wedge A_2 \wedge \bot)$
   $(A_1 \wedge A_2 \wedge A_1) \vee (\neg A_2 \wedge A_1) \vee (A_1 \wedge A_2 \wedge A_2 \wedge \bot) \vee (\neg A_2 \wedge A_2 \wedge \bot)$
   $(A_1 \wedge A_2) \vee (\neg A_2 \wedge A_1) \vee (A_1 \wedge A_2 \wedge \bot) \vee (\neg A_2 \wedge A_2 \wedge \bot)$ (Idemp.)
   KNF: $(A_1 \vee \neg A_2) \wedge (A_2 \vee \neg A_2) \wedge (A_1 \vee (A_2 \wedge \bot))$
   $(A_1 \vee \neg A_2) \wedge (A_2 \vee \neg A_2) \wedge (A_1 \vee A_2) \wedge (A_1 \vee \bot)$

4. Vereinfache mit den Regeln für $\top$ und $\bot$:

DNF: $(A_1 \wedge A_2) \vee (\neg A_2 \wedge A_1) \vee (A_1 \wedge A_2 \wedge \bot) \vee (\neg A_2 \wedge A_2 \wedge \bot)$
$(A_1 \wedge A_2) \vee (\neg A_2 \wedge A_1) \vee (A_1 \wedge A_2 \wedge \bot) \vee \bot$
$(A_1 \wedge A_2) \vee (\neg A_2 \wedge A_1) \vee (A_1 \wedge A_2 \wedge \bot)$
$(A_1 \wedge A_2) \vee (\neg A_2 \wedge A_1) \vee \bot$
$(A_1 \wedge A_2) \vee (\neg A_2 \wedge A_1)$ DNF erreicht!
$A_1 \wedge (A_2 \vee \neg A_2)$ (Distributivgesetz, keine DNF mehr)
$A_1 \wedge \top$
A_1 (wieder DNF)

KNF: $(A_1 \vee \neg A_2) \wedge (A_2 \vee \neg A_2) \wedge (A_1 \vee A_2) \wedge (A_1 \vee \bot)$
$(A_1 \vee \neg A_2) \wedge (A_2 \vee \neg A_2) \wedge (A_1 \vee A_2) \wedge A_1$ KNF erreicht!
$(A_1 \vee \neg A_2) \wedge (A_2 \vee \neg A_2) \wedge A_1$ (Absorption)
$(A_1 \vee \neg A_2) \wedge \top \wedge A_1$ (keine KNF mehr!)
$(A_1 \vee \neg A_2) \wedge A_1$ (wieder KNF)
A_1 (Absorption)

Welche Methode ist besser?

Gefühlsmäßig: Die semantische Methode ist übersichtlicher.

Theoretisch: Beide Methoden sind schlecht, denn beide sind im schlechtesten Fall exponentiell.

- Semantische Methode: Aufwand immer exponentiell in Variablenzahl! Wahrheitstafel besitzt $2^{\text{Variablenzahl}}$ Zeilen.
- Algebraische Methode: Schritt 3 (Distributivgesetz) ist aufwändig, kann zu einer exponentiellen Verlängerung der Formel führen.

Praktisch:

- Semantische Methode nur brauchbar bei Formeln mit sehr wenigen Variablen. Immer exponentiell in Variablenzahl, liefert immer die maximale DNF/KNF.
- Algebraische Methode teilweise auch für große Formeln brauchbar, insbesondere mit Computerunterstützung. Kann auch kleine DNFs/KNFs liefern.

Was Sie heute erwartet

Das Erfüllbarkeitsproblem der Aussagenlogik

Erfüllbarkeitsproblem (Satisfiability, SAT)

Gegeben: aussagenlogische Formel F
Frage: Ist F erfüllbar, d.h., gibt es ein $I \in \mathcal{I}$, sodass $\text{val}_I(F) = 1$?

Effiziente Verfahren zur Lösung von SAT sind wichtig in der Praxis:

- Viele praktische Aufgaben lassen sich als Probleme der Aussagenlogik formulieren, wie z.B.
  - Verifikation von Hard- und Software
  - Planungsaufgaben, Logistik-Probleme
- Die meisten aussagenlogischen Fragen lassen sich zu einem (Un)Erfüllbarkeitsproblem umformulieren:

$$
\begin{aligned}
G \text{ gültig} &\iff \neg G \text{ unerfüllbar} \\
G \text{ widerlegbar} &\iff \neg G \text{ erfüllbar} \\
G = H &\iff G \not\equiv H \text{ unerfüllbar} \\
F_1, \ldots, F_n \models G &\iff F_1 \wedge \cdots \wedge F_n \wedge \neg G \text{ unerfüllbar}
\end{aligned}
$$

Methoden zur Lösung von SAT

Wahrheitstafel:

- Berechne den Formelwert der Reihe nach für jede Interpretation.
  Antwort „ja“, sobald man den Wert 1 erhält; „nein“, wenn immer 0.
- Unbrauchbar, da exponentiell: $2^{\text{Variablenzahl}}$ Interpretationen!

Umwandlung in DNF:

- Wandle F in eine disjunktive Normalform um.
  Antwort „nein“, wenn man $\bot$ erhält; „ja“ sonst.
- Unbrauchbar: F meistens in Fast-KNF. Distributivgesetz verlängert F exponentiell.

SAT-Solver: Programme, die SAT lösen.

- Verwenden fortgeschrittene algebraische/graphenorientierte/logische Methoden mit besonderen Datenstrukturen.
- Können SAT für Formeln mit Millionen von Variablen lösen.
- Stand der Technik bei der Verifikation von Prozessoren etc.
- Aber: Exponentielle Laufzeit für manche Formelarten!

$ 1.000.000,– Prämie für einen effizienten SAT-Solver

. . . oder für den Beweis, dass es diesen nicht geben kann.

Abzuholen beim Clay Mathematics Institute (www.claymath.org)
für das offene Milleniumsproblem „P versus NP“.

Weiters warten ewiger Ruhm, eine Universitätsstelle, . . .

P: Klasse der Probleme, die sich effizient (polynomiell) lösen lassen.

NP: Klasse jener Probleme, deren Lösungen sich effizient (polynomiell) verifizieren lassen; die Suche nach der Lösung kann aber aufwändig sein.

P versus NP (Stephen Cook, 1971)

Gilt $\mathrm{P} = \mathrm{NP}$ oder $\mathrm{P} \neq \mathrm{NP}$ (gleichbedeutend mit $\mathrm{P} \subsetneq \mathrm{NP}$)?

NP-Vollständigkeit

Die schwierigsten Probleme in NP heißen NP-vollständig.
Ihr Kennzeichen:

Kann man ein NP-vollständiges Problem effizient lösen,
dann kann man alle Probleme in NP effizient lösen.

HAMILTON-KREIS ist NP-vollständig

Gegeben: Party-Gäste, von denen sich einige nicht mögen.
Frage: Kann man die Gruppe so um einen runden Tisch setzen, dass sich je zwei Sitznachbarn vertragen?

- Wenn alle sitzen, ist leicht zu prüfen, ob sich alle Nachbarn verstehen.
- Das Finden einer geeigneten Sitzordnung ist aber im Allgemeinen schwierig. Exponentiell?

SAT ist NP-vollständig

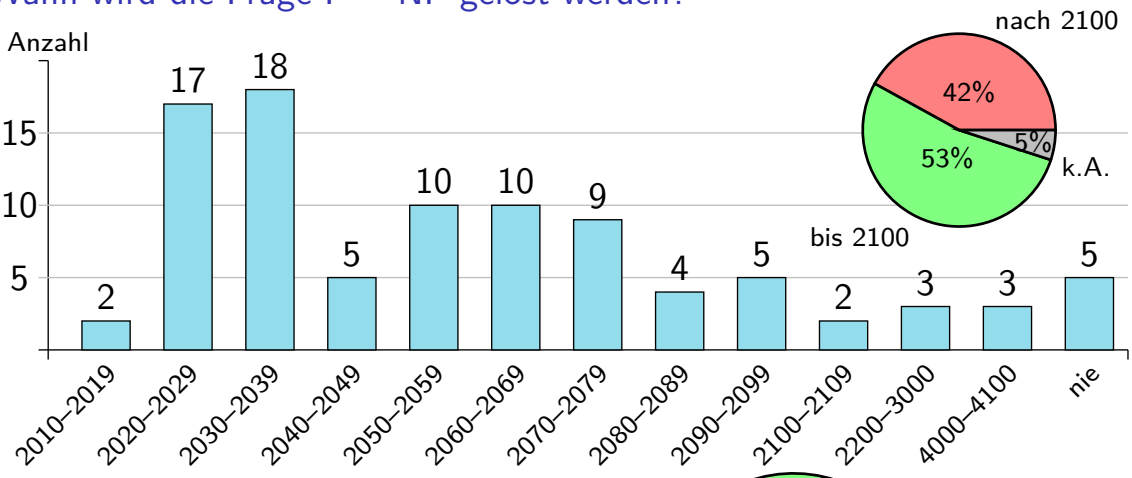
Gegeben: eine aussagenlogische Formel.
Frage: Ist die Formel erfüllbar?

- Ist die Interpretation I gegeben, lässt sich $\mathrm{val}_I(F) = 1$ leicht überprüfen.
- Das Finden der Interpretation ist aber schwierig. Exponentiell?

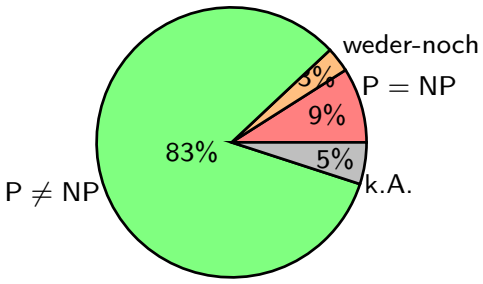
SAT polynomiell lösbar $\Longrightarrow$ P = NP

SAT nicht polynomiell lösbar $\Longrightarrow$ P $\neq$ NP

Wann wird die Frage $P \stackrel{?}{=} NP$ gelöst werden?



Wie wird die Antwort lauten?



[W.I.Gasarch, 2012, Meinungsumfrage unter 152 Experten]

Falls Sie SAT nicht ausreichend inspiriert . . .

MINESWEEPER ist NP-vollständig

Gegeben: eine Minesweeper-Stellung
Frage: Ist die Stellung möglich?

Beispiel einer unmöglichen Stellung:

- „2“, aber 5 Bomben in der Umgebung
- „6“, aber nur drei Bomben möglich
- „1“, aber keine Bombe in der Umgebung

MINESWEEPER polynomiell lösbar $\Longrightarrow$ P = NP

MINESWEEPER nicht polynomiell lösbar $\Longrightarrow$ P $\neq$ NP

Es sind mittlerweile hunderte von NP-vollständigen Problemen aus allen Bereichen der Informatik bekannt.

Was Sie heute erwartet

House

Max wird mit hohem Fieber und ausgeprägten Gliederschmerzen in das Spital eingeliefert. Dr. House diskutiert die Diagnose mit einer Kollegin.

House: „Wenn der Patient Fieber hat, handelt es sich um Grippe oder Erkältung.“

Cameron: „Wenn er keine starken Gliederschmerzen hat, dann hat er auch keine Grippe.“

House: „Jedenfalls weisen hohes Fieber und starke Gliederschmerzen immer auf Grippe hin.“

Cameron: “Er hat sicher nicht beide Krankheiten gleichzeitig.“

Wie lautet die Diagnose?
Wie lässt sie sich mit Hilfe der Aussagenlogik finden und begründen?

House – Wahl der Aussagenvariablen

Aussagenvariablen können nur Aussagen repräsentieren, die einen Wahrheitswert besitzen.
Einzelne Haupt-, Zeit- oder Eigenschaftswörter sind keine Aussagen!

Falsch: $A =$ „krank“ oder $A =$ „Fieber“.

Möglich: $A =$ „Max ist krank“ oder $A =$ „Der Patient hat Fieber“.

Max wird mit hohem Fieber und ausgeprägten Gliederschmerzen in das Spital eingeliefert.

„Max wird mit . . . eingeliefert“ $= A$?

„Max hat hohes Fieber“ $= A$,
„Max hat ausgeprägte Gliederschmerzen“ $= B$ und
„Max wird in das Spital eingeliefert“ $= C$?

„Max hat hohes Fieber“ = „Max hat Fieber“ $= A$ und
„Max hat ausgeprägte Gl.schmerzen“ = „Max hat Gl.schmerzen“ $= B$? 47

House – Wahl der Aussagenvariablen

Dr. House diskutiert die Diagnose mit einer Kollegin.

„Dr. House diskutiert . . . mit einer Kollegin“ $= D$?

Wenn der Patient Fieber hat, handelt es sich um Grippe oder Erkältung.

„Der Patient hat Fieber“ $= E$?

„Der Patient hat Fieber“ = „Max hat Fieber“ $= A$?

Cameron: “Er hat sicher nicht beide Krankheiten gleichzeitig.“

„Cameron sagt, dass er nicht beide Krankheiten gleichzeitig hat.“ $= F$?

„Max kann nicht beide Krankheiten gleichzeitig haben.“ $= F$?

- Elimination von Abkürzungen und Referenzen
  „Er hat beide Krankheiten“ = „P. hat Grippe“ + „P. hat Erkältung“
- Generalisierung: Zusammenfassen von gleichartigen Aussagen
- Abstraktion: Weglassen von Details
- Konzentration auf das Wesentliche: Identifikation der relevanten Teilaussagen

Aber:

- Was zusammengefasst wurde, kann nicht mehr getrennt analysiert werden.
- Was weggelassen wurde, kann nicht für die Argumentation verwendet werden.
- Was nicht zusammenfasst wurde, aber zusammengehört, muss durch zusätzliche Formeln in Beziehung gesetzt werden.

Was kann man zusammenfassen? Was weglassen? Was ist wesentlich?

House – Wahl der Aussagenvariablen

Max wird mit hohem Fieber und ausgeprägten Gliederschmerzen in das Spital eingeliefert. Dr. House diskutiert die Diagnose mit einer Kollegin.

House: „Wenn der Patient Fieber hat, handelt es sich um Grippe oder Erkältung.“

Cameron: „Wenn er keine starken Gliederschmerzen hat, dann hat er auch keine Grippe.“

House: „Jedenfalls weisen hohes Fieber und starke Gliederschmerzen immer auf Grippe hin.“

Cameron: “Er hat sicher nicht beide Krankheiten gleichzeitig.“

F . . . „Max/Patient hat (hohes) Fieber.“
S . . . „Max/Patient hat starke/ausgeprägte Gliederschmerzen.“
G . . . „Max/Patient hat eine Grippe.“
E . . . „Max/Patient hat eine Erkältung.“

House – aussagenlogische Modellierung

F ... „Max/Patient hat (hohes) Fieber.“
S ... „Max/Patient hat starke/ausgeprägte Gliederschmerzen.“
G ... „Max/Patient hat eine Grippe.“
E ... „Max/Patient hat eine Erkältung.“

Max wird mit hohem Fieber und ausgeprägten Gliederschmerzen in das Spital eingeliefert. Dr. House diskutiert die Diagnose mit einer Kollegin.	$F_1 := F \wedge S$
House: „Wenn der Patient Fieber hat, handelt es sich um Grippe oder Erkältung.“	$F_2 := F \supset (G \vee E)$
Cameron: „Wenn er keine starken Gliederschmerzen hat, dann hat er auch keine Grippe.“	$F_3 := \neg S \supset \neg G$
House: „Jedenfalls weisen hohes Fieber und starke Gliederschmerzen immer auf Grippe hin.“	$F_4 := (F \wedge S) \supset G$
Cameron: “Er hat sicher nicht beide Krankheiten gleichzeitig.“	$F_5 := \neg(G \wedge E)$

House – Diagnose

Finde alle Interpretationen I, in denen alle Formeln wahr sind.

Methode 1: Wahrheitstafel

Vereinfachung: Prüfe nur Interpretationen, in denen $F_1 = F \wedge S$ wahr ist.

F	S	G	E	F_1	$F \supset (G \vee E)$	$\neg S \supset \neg G$	$(F \wedge S) \supset G$	$\neg(G \wedge E)$
1	1	0	0	1	0	1	0	1
1	1	0	1	1	1	1	0	1
1	1	1	0	1	1	1	1	1
1	1	1	1	1	1	1	1	0

$I(G) = 1$, $I(E) = 0 \Longrightarrow$ Die Diagnose lautet auf „Grippe“.

Methode 2: Umwandlung in KNF, SAT-Solver aufrufen

$$\underbrace{F \wedge S}_{F_1} \wedge \underbrace{(\neg F \vee G \vee E)}_{F_2} \wedge \underbrace{(S \vee \neg G)}_{F_3} \wedge \underbrace{(\neg F \vee \neg S \vee G)}_{F_4} \wedge \underbrace{(\neg G \vee \neg E)}_{F_5}$$

SAT-Solver liefert „erfüllbar“ sowie die Interpretation I mit $I(F) = I(S) = I(G) = 1$ und $I(E) = 0$.

Weitere Lösungen durch Ausschluss der bereits gefundenen mit der zusätzlichen Formel $F_6 = \neg(F \wedge S \wedge G \wedge \neg E) = \neg F \vee \neg S \vee \neg G \vee E$

SAT-Solver liefert für $F_1 \wedge \cdots \wedge F_5 \wedge F_6$ das Ergebnis „unerfüllbar“, es gibt also keine weiteren Lösungen.

Methode 3: Umwandlung in DNF und Vereinfachung

$$\overbrace{F \wedge S}^{F_1} \wedge \overbrace{(\neg F \vee G \vee E)}^{F_2} \wedge \overbrace{(S \vee \neg G)}^{F_3} \wedge \overbrace{(\neg F \vee \neg S \vee G)}^{F_4} \wedge \overbrace{(\neg G \vee \neg E)}^{F_5}$$

$$(\underbrace{(F \wedge S \wedge \neg F)}_{=\bot} \vee (F \wedge S \wedge G) \vee (F \wedge S \wedge E)) \wedge \cdots$$

$$((F \wedge S \wedge G) \vee (F \wedge S \wedge E)) \wedge (S \vee \neg G) \wedge \cdots$$

$$((F \wedge S \wedge G) \vee (F \wedge S \wedge E) \vee \underbrace{(F \wedge S \wedge G \wedge \neg G)}_{=\bot} \vee \underbrace{(F \wedge S \wedge E \wedge \neg G)}_{\text{Absorption } F \wedge S \wedge E}) \wedge \cdots$$

$$((F \wedge S \wedge G) \vee (F \wedge S \wedge E)) \wedge (\neg F \vee \neg S \vee G) \wedge \cdots$$

$$((F \wedge S \wedge G \wedge G) \vee (F \wedge S \wedge E \wedge G)) \wedge \cdots$$

$$((F \wedge S \wedge G) \vee \underbrace{(F \wedge S \wedge E \wedge G)}_{\text{Absorption } F \wedge S \wedge G}) \wedge \cdots$$

$$(F \wedge S \wedge G) \wedge (\neg G \vee \neg E)$$

$$\underbrace{(F \wedge S \wedge G \wedge \neg G)}_{=\bot} \vee (F \wedge S \wedge G \wedge \neg E)$$

$F \wedge S \wedge G \wedge \neg E$ DNF mit Lösung $I(F) = I(S) = I(G) = 1$ und $I(E) = 0$

3.0 VU Formale Modellierung

Gernot Salzer

Forschungsbereich Theory and Logic
Institut für Logic and Computation

10.4.2018

Was Sie letztes Mal hörten

Semantische Äquivalenz

Zwei Formeln F und G heißen äquivalent, geschrieben $F = G$, wenn $\mathrm{val}_I(F) = \mathrm{val}_I(G)$ für alle Interpretationen I gilt.

$(A \supset B)$ und $(\neg A \vee B)$ sind äquivalent

A	B	$(A \supset B)$	$=$	$(\neg A \vee B)$
1	1	1	✓	1
1	0	0	✓	0
0	1	1	✓	1
0	0	1	✓	1

$F = G$ gilt genau dann, wenn $F \equiv G$ eine gültige Formel ist.

$\langle \mathbb{B}, \mathrm{and}, \mathrm{or}, \mathrm{not}, 0, 1 \rangle$ ist eine Boolesche Algebra

$(A \wedge B) \wedge C = A \wedge (B \wedge C)$	$(A \vee B) \vee C = A \vee (B \vee C)$	Assoziativität
$A \wedge B = B \wedge A$	$A \vee B = B \vee A$	Kommutativität
$A \wedge A = A$	$A \vee A = A$	Idempotenz
$A \wedge \top = A$	$A \vee \bot = A$	Neutralität
$A \wedge \neg A = \bot$	$A \vee \neg A = \top$	Komplement
$A \wedge (A \vee B) = A$	$A \vee (A \wedge B) = A$	Absorption
$A \wedge (B \vee C) = (A \wedge B) \vee (A \wedge C)$		Distributivität
$A \vee (B \wedge C) = (A \vee B) \wedge (A \vee C)$		

Weitere Äquivalenzen

$A \uparrow B = \neg A \vee \neg B$	$A \equiv B = (\neg A \vee B) \wedge (A \vee \neg B)$	$\neg(A \wedge B) = \neg A \vee \neg B$
$A \downarrow B = \neg A \wedge \neg B$	$= (A \wedge B) \vee (\neg A \wedge \neg B)$	$\neg(A \vee B) = \neg A \wedge \neg B$
$A \supset B = \neg A \vee B$	$A \not\equiv B = (\neg A \vee \neg B) \wedge (A \vee B)$	$\neg\neg A = A$
$A \subset B = A \vee \neg B$	$= (A \wedge \neg B) \vee (\neg A \wedge B)$	

$A \wedge \top = A$	$A \wedge \bot = \bot$	$A \wedge \neg A = \bot$	$\neg\top = \bot$
$A \vee \bot = A$	$A \vee \top = \top$	$A \vee \neg A = \top$	$\neg\bot = \top$

Logische Konsequenz

$F_1, \ldots, F_n \models_I G$: „Aus $\mathrm{val}_I(F_1) = \cdots = \mathrm{val}_I(F_n) = 1$ folgt $\mathrm{val}_I(G) = 1$.“

Logische Konsequenz

Die Formel G folgt aus den Formeln $F_1, \ldots, F_n$, geschrieben $F_1, \ldots, F_n \models G$, wenn $F_1, \ldots, F_n \models_I G$ für alle Interpretationen I gilt.

„Die Formel G folgt aus den Formeln $F_1, \ldots, F_n$.“

$A,\ A \supset B \models B$

$I(A)$	$I(B)$	A	$A \supset B$		B
1	1	1	1	✓	1
1	0	1	0	✓	0
0	1	0	1	✓	1
0	0	0	1	✓	0

$A,\ A \vee B \not\models B$

Gegenbeispiel: $I(A) = 1$, $I(B) = 0$

$F_1, \ldots, F_n \models G$ gilt genau dann, wenn $(F_1 \wedge \cdots \wedge F_n) \supset G$ gültig ist.

Was Sie letztes Mal hörten

Von der Funktion zur Formel

Gegeben: Funktion $f : \mathbb{B}^n \mapsto \mathbb{B}$ (z.B. als Wahrheitstafel)
Gesucht: Formel, die f darstellt

A_1	A_2	A_3							
	$\vec{b}$		$f(\vec{b})$	$K_{\vec{b}}$			$D_{\vec{b}}$		
1	1	1	1	$A_1 \wedge$	$A_2 \wedge$	$A_3 =: K_{111}$			
1	1	0	0				$\neg A_1 \vee \neg A_2 \vee$	$A_3 =: D_{110}$	
1	0	1	0				$\neg A_1 \vee$	$A_2 \vee \neg A_3 =: D_{101}$	
1	0	0	1	$A_1 \wedge \neg A_2 \wedge \neg A_3 =: K_{100}$					
0	1	1	1	$\neg A_1 \wedge$	$A_2 \wedge$	$A_3 =: K_{011}$			
0	1	0	0				$A_1 \vee \neg A_2 \vee$	$A_3 =: D_{010}$	
0	0	1	0				$A_1 \vee$	$A_2 \vee \neg A_3 =: D_{001}$	
0	0	0	0				$A_1 \vee$	$A_2 \vee$	$A_3 =: D_{000}$

$\text{DNF}_f = K_{111} \vee K_{100} \vee K_{011}$

$\text{KNF}_f = D_{110} \wedge D_{101} \wedge D_{010} \wedge D_{001} \wedge D_{000}$

7

Was Sie letztes Mal hörten

Normalformen

Literal: Variable oder negierte Variable, also A, $\neg A$, B, $\neg B$, ...

Negationsnormalform (NNF)

- Literale sowie $\top$ und $\bot$ sind in NNF.
- $(F \wedge G)$ und $(F \vee G)$ sind in NNF, wenn F und G in NNF sind.
- Keine Formel sonst ist in NNF.

Disjunktive Normalform (DNF)

$\top$, $\bot$ sowie Disjunktionen von Konjunktion von Literalen:
$(^{(\neg)}A_{1,1} \wedge {}^{(\neg)}A_{1,2} \wedge {}^{(\neg)}A_{1,3} \wedge \cdots) \vee (^{(\neg)}A_{2,1} \wedge {}^{(\neg)}A_{2,2} \wedge {}^{(\neg)}A_{2,3} \wedge \cdots) \vee \cdots$

Konjunktive Normalform (KNF)

$\top$, $\bot$ sowie Konjunktionen von Disjunktion von Literalen:
$(^{(\neg)}A_{1,1} \vee {}^{(\neg)}A_{1,2} \vee {}^{(\neg)}A_{1,3} \vee \cdots) \wedge (^{(\neg)}A_{2,1} \vee {}^{(\neg)}A_{2,2} \vee {}^{(\neg)}A_{2,3} \vee \cdots) \wedge \cdots$

Konstruktion von DNFs/KNFs – Semantische Methode

Gegeben: Aussagenlogische Formel F
Gesucht: Äquivalente Formel in DNF/KNF

1. Stelle die zu F gehörige Funktion f als Wahrheitstafel dar.
2. Konstruiere DNF_f bzw. KNF_f.

A_1	A_2	A_3	$F := (A_1 \supset (A_2 \equiv A_3)) \wedge (\neg A_1 \supset (A_2 \wedge A_3))$		
1	1	1	1	K_{111}	
1	1	0	0		D_{110}
1	0	1	0		D_{101}
1	0	0	1	K_{100}	
0	1	1	1	K_{011}	
0	1	0	0		D_{010}
0	0	1	0		D_{001}
0	0	0	0		D_{000}

DNF: $F = K_{111} \vee K_{100} \vee K_{011}$
KNF: $F = D_{110} \wedge D_{101} \wedge D_{010} \wedge D_{001} \wedge D_{000}$

Konstruktion von DNFs/KNFs – Algebraische Methode

Gegeben: Aussagenlogische Formel F
Gesucht: Äquivalente Formel in DNF/KNF

1. Ersetze alle Junktoren durch $\wedge$, $\vee$ und $\neg$.
2. Verschiebe Negationen nach innen, eliminiere Doppelnegationen.
3. Wende das Distributivgesetz an.
4. Eliminiere $\top$ und $\bot$.

$(A_1 \supset (A_2 \equiv A_3)) \wedge (\neg A_1 \supset (A_2 \wedge A_3))$

1. $(\neg A_1 \vee (A_2 \wedge A_3) \vee (\neg A_2 \wedge \neg A_3)) \wedge (\neg\neg A_1 \vee (A_2 \wedge A_3))$
2. $(\neg A_1 \vee (A_2 \wedge A_3) \vee (\neg A_2 \wedge \neg A_3)) \wedge (A_1 \vee (A_2 \wedge A_3))$
3. DNF: $(\neg A_1 \wedge A_1) \vee (\neg A_1 \wedge A_2 \wedge A_3) \vee (A_2 \wedge A_3 \wedge A_1) \vee$
   $(A_2 \wedge A_3 \wedge A_2 \wedge A_3) \vee (\neg A_2 \wedge \neg A_3 \wedge A_1) \vee (\neg A_2 \wedge \neg A_3 \wedge A_2 \wedge A_3)$
   KNF: $(\neg A_1 \vee A_2 \vee \neg A_2) \wedge (\neg A_1 \vee A_2 \vee \neg A_3) \wedge (\neg A_1 \vee A_3 \vee \neg A_2) \wedge$
   $(\neg A_1 \vee A_3 \vee \neg A_3) \wedge (A_1 \vee A_2) \wedge (A_1 \vee A_3)$
4. DNF: $(\neg A_1 \wedge A_2 \wedge A_3) \vee (A_2 \wedge A_3) \vee (A_1 \wedge \neg A_2 \wedge \neg A_3)$
   KNF: $(\neg A_1 \vee A_2 \vee \neg A_3) \wedge (\neg A_1 \vee \neg A_2 \vee A_3) \wedge (A_1 \vee A_2) \wedge (A_1 \vee A_3)$

Was Sie letztes Mal hörten

Das Erfüllbarkeitsproblem der Aussagenlogik

Erfüllbarkeitsproblem (Satisfiability, SAT)

Gegeben: aussagenlogische Formel F
Frage: Ist F erfüllbar, d.h., gibt es ein $I \in \mathcal{I}$, sodass $\text{val}_I(F) = 1$?

Effiziente Verfahren zur Lösung von SAT sind wichtig in der Praxis:

- Viele praktische Aufgaben sind Probleme der Aussagenlogik.
- Aussagenlogische Fragen lassen sich auf SAT zurückführen.

Unbrauchbare Verfahren:

- Wahrheitstafel: Prüfe, ob es Interpretation mit Ergebnis 1 gibt.
- Umwandlung in DNF: Antwort „nein“, wenn man $\bot$ erhält, sonst „ja“.

SAT-Solver: Programme, die SAT lösen.

- Lösen SAT mit fortgeschrittenen Methoden und Datenstrukturen.
- Können SAT für Formeln mit Millionen von Variablen lösen.
- Aber: immer noch ineffizient für bestimmte Formeltypen.

P = NP?

P: Klasse der Probleme, deren Lösungen sich polynomiell finden lassen.

NP: Klasse jener Probleme, deren Lösungen sich polynomiell verifizieren lassen; die Suche nach der Lösung kann aber aufwändig sein.

P versus NP

Gilt P = NP oder P $\neq$ NP (gleichbedeutend mit P $\subsetneq$ NP)?

SAT ist in NP: Gegeben I, lässt sich $\text{val}_I(F) = 1$ leicht überprüfen.
Die Suche nach so einem I scheint aber aufwändig. Exponentiell?

SAT ist NP-vollständig: SAT gehört zu den schwierigsten Problemen in NP. Kann man ein NP-vollständiges Problem effizient lösen, dann kann man alle Probleme in NP effizient lösen.

- SAT polynomiell lösbar $\Longrightarrow$ P = NP
- SAT nicht polynomiell lösbar $\Longrightarrow$ P $\neq$ NP

Was Sie heute erwartet

Dualität von Funktionen, Operatoren und Formeln

Beobachtung 1: „and“ und „or“ verhalten sich spiegelbildlich bzgl. 0 und 1.

x	y	x and y	x or y
1	1	1	1
1	0	0	1
0	1	0	1
0	0	0	0

x	y	x or y	x and y
0	0	0	0
0	1	1	0
1	0	1	0
1	1	1	1

„and“ ist eine Konjunktion für 1 und eine Disjunktion für 0.
„or“ ist eine Konjunktion für 0 und eine Disjunktion für 1.

Beobachtung 2: Boolesche Algebra ist symmetrisch bzgl. $\wedge/\vee$ und $\top/\bot$.

$$(A \wedge B) \wedge C = A \wedge (B \wedge C) \qquad (A \vee B) \vee C = A \vee (B \vee C)$$
$$A \wedge B = B \wedge A \qquad A \vee B = B \vee A$$
$$A \wedge A = A \qquad A \vee A = A$$
$$A \wedge \top = A \qquad A \vee \bot = A$$
$$A \wedge \neg A = \bot \qquad A \vee \neg A = \top$$
$$A \wedge (A \vee B) = A \qquad A \vee (A \wedge B) = A$$
$$A \wedge (B \vee C) = (A \wedge B) \vee (A \wedge C) \qquad A \vee (B \wedge C) = (A \vee B) \wedge (A \vee C)$$

Duale Funktionen

Zwei n-stellige Funktionen f und g heißen dual zueinander, wenn gilt:
$\text{not}\, f(x_1, \ldots, x_n) = g(\text{not}\, x_1, \ldots, \text{not}\, x_n)$.

Diese Bedingung ist gleichbedeutend mit jeder der folgenden:

$$\begin{aligned} \text{not}\, f(\text{not}\, x_1, \ldots, \text{not}\, x_n) &= g(x_1, \ldots, x_n) \\ f(x_1, \ldots, x_n) &= \text{not}\, g(\text{not}\, x_1, \ldots, \text{not}\, x_n) \\ f(\text{not}\, x_1, \ldots, \text{not}\, x_n) &= \text{not}\, g(x_1, \ldots, x_n) \end{aligned}$$

„and“ und „or“ sind dual, da $\text{not}(x \text{ and } y) = (\text{not}\, x) \text{ or } (\text{not}\, y)$ gilt.

Duale Operatoren

Zwei Operatoren heißen dual, wenn die zugehörigen Funktionen dual sind.

true / $\top$	not / $\neg$	and / $\wedge$	nand / $\uparrow$	iff / $\equiv$	implies / $\supset$	if / $\subset$
			ist dual zu			
false / $\bot$	not / $\neg$	or / $\vee$	nor / $\downarrow$	xor / $\not\equiv$	— / $\not\subset$	— / $\not\supset$

(und umgekehrt).

$G[A_1, \ldots, A_n]$ ... „Formel G enthält die Variablen A_1, ..., A_n“
$G[H_1, \ldots, H_n]$ ... Formel, die aus $G[A_1, \ldots, A_n]$ entsteht, wenn A_i überall durch H_i ersetzt wird.

Duale Formeln

Zwei Formeln $F[A_1, \ldots, A_n]$ und $G[A_1, \ldots, A_n]$ heißen dual zueinander, wenn gilt: $\neg F[A_1, \ldots, A_n] = G[\neg A_1, \ldots, \neg A_n]$

$\neg F[\neg A_1, \ldots, \neg A_n]$ ist dual zu $F[A_1, \ldots, A_n]$.

$\neg((\neg A \vee \neg\neg B) \supset \neg A)$ ist dual zu $(A \vee \neg B) \supset A$.

Sei G die Formel, die aus F durch Ersetzen aller Operatoren durch ihre dualen hervorgeht. Dann ist G dual zu F.

$\neg(((A \wedge B) \not\equiv \neg(B \uparrow C)) \not\supset ((A \wedge \top) \vee B))$ ist dual zu
$\neg(((A \vee B) \equiv \neg(B \downarrow C)) \subset ((A \vee \bot) \wedge B))$

F^*, G^* ... irgendwelche dualen Formeln zu F bzw. G

$F = G$ gilt genau dann, wenn $F^* = G^*$ gilt.

$F = G$	$F^* = G^*$
$(A \wedge B) \wedge C = A \wedge (B \wedge C)$	$(A \vee B) \vee C = A \vee (B \vee C)$
$A \wedge B = B \wedge A$	$A \vee B = B \vee A$
$A \wedge A = A$	$A \vee A = A$
$A \wedge \top = A$	$A \vee \bot = A$
$A \wedge \neg A = \bot$	$A \vee \neg A = \top$
$A \wedge (A \vee B) = A$	$A \vee (A \wedge B) = A$
$A \wedge (B \vee C) = (A \wedge B) \vee (A \wedge C)$	$A \vee (B \wedge C) = (A \vee B) \wedge (A \vee C)$

- $(F^*)^* = F$
- F ist gültig genau dann, wenn F^* unerfüllbar ist.
- $F \supset G$ ist gültig genau dann, wenn $G^* \supset F^*$ gültig ist.
- ...

Was Sie heute erwartet

Gone Maggie gone

„The Simpsons“, Staffel 20, Folge 13

The Simpsons – aussagenlogische Modellierung

If all you have is a hammer, everything looks like a nail.

M_i ... Maggie
K_i ... Knecht Ruprecht
G_i ... Gift
H_i ... Homer

} ... befindet sich zum Zeitpunkt i auf der anderen Seite des Flusses.

- Zum Zeitpunkt i befinden sich alle auf dieser Flussseite.

$\text{AlleHier}(i) := \neg M_i \wedge \neg K_i \wedge \neg G_i \wedge \neg H_i$

- Zum Zeitpunkt i befinden sind alle auf der anderen Flussseite.

$\text{AlleDort}(i) := M_i \wedge K_i \wedge G_i \wedge H_i$

- Wenn sich Maggie und KR oder Maggie und das Gift am selben Flussufer befinden, muss Homer bei Maggie sein.

$\text{Sicher}(i) := ((M_i \equiv K_i) \vee (M_i \equiv G_i)) \supset (M_i \equiv H_i)$

MH_i ... Maggie
KH_i ... Knecht Ruprecht
GH_i ... Gift
} ... fährt mit Homer über den Fluss (zw. den Zeitpunkten $i-1$ und i).
HH_i ... Homer fährt alleine über den Fluss (zwischen $i-1$ und i).

- Genau eine Überfahrt zwischen den Zeitpunkten $i-1$ und i.

$$\begin{aligned}&\text{Überfahrt}(i) :=\\&(MH_i \wedge \neg KH_i \wedge \neg GH_i \wedge \neg HH_i) \vee (\neg MH_i \wedge KH_i \wedge \neg GH_i \wedge \neg HH_i) \vee\\&(\neg MH_i \wedge \neg KH_i \wedge GH_i \wedge \neg HH_i) \vee (\neg MH_i \wedge \neg KH_i \wedge \neg GH_i \wedge HH_i)\end{aligned}$$

- Definition der Überfahrten:

$$\begin{aligned}&\text{DefÜberfahrt}(i) :=\\&\quad (MH_i \supset ((M_{i-1} \not\equiv M_i) \wedge (K_{i-1} \equiv K_i) \wedge (G_{i-1} \equiv G_i) \wedge (H_{i-1} \not\equiv H_i) \wedge (H_i \equiv M_i)))\\&\wedge (KH_i \supset ((M_{i-1} \equiv M_i) \wedge (K_{i-1} \not\equiv K_i) \wedge (G_{i-1} \equiv G_i) \wedge (H_{i-1} \not\equiv H_i) \wedge (H_i \equiv K_i)))\\&\wedge (GH_i \supset ((M_{i-1} \equiv M_i) \wedge (K_{i-1} \equiv K_i) \wedge (G_{i-1} \not\equiv G_i) \wedge (H_{i-1} \not\equiv H_i) \wedge (H_i \equiv G_i)))\\&\wedge (HH_i \supset ((M_{i-1} \equiv M_i) \wedge (K_{i-1} \equiv K_i) \wedge (G_{i-1} \equiv G_i) \wedge (H_{i-1} \not\equiv H_i))\end{aligned}$$

Gesamtformel: Nach n Überfahrten sollen alle auf der anderen Seite sein.

$$\begin{aligned}\text{Simpsons}(n) :=\ &\text{AlleHier}(0) \wedge \text{AlleDort}(n) \wedge \bigwedge_{i=0}^{n} \text{Sicher}(i)\\&\wedge \bigwedge_{i=1}^{n} \text{Überfahrt}(i) \wedge \bigwedge_{i=1}^{n} \text{DefÜberfahrt}(i)\end{aligned}$$

Methode zum Lösen des Rätsels:

1. Errate die benötigte Zahl n der Überfahrten.
2. Finde eine erfüllende Interpretation für die Formel Simpsons(n) (z.B. mit Hilfe eines SAT-Solvers).

Eine mögliche Lösung:
$n = 7$,
$I(MH_1) = I(HH_2) = I(GH_3) = I(MH_4) = I(KH_5) = I(HH_6) = I(MH_7) = 1$

Gone Maggie gone (Fortsetzung)

„The Simpsons“, Staffel 20, Folge 13

Vor- und Nachteile dieser aussagenlogischen Modellierung

Vorteile:

- deklarativ-statisch, nicht prozedural-dynamisch
  Welche Eigenschaften sollen gelten?
  Nicht: Welche Schritte sind für Lösung erforderlich?
- modular
  Neue Bedingungen werden durch zusätzliche Formeln berücksichtigt.

Nachteile:

- Erraten von Parametern
  n muss durch Probieren gefunden werden
- Große Zahl an Variablen und Formeln
  Dynamik muss durch indizierte Variablen simuliert werden.
- Frame Problem
  Bei jeder Aktion muss auch definiert werden, was sich nicht ändert.
- unintuitiv
  Bei der Modellierung von Abläufen denkt man an Zustände und Übergänge, nicht an statische Bedingungen.

Was Sie heute erwartet

The Simpsons – Modellierung als Automat

Systemkomponenten: Maggie (M), Knecht Ruprecht (K), Gift (G), Homer+Boot (H), linkes/rechtes Flussufer

Situationsbeschreibung (Systemzustand): $\frac{\text{Lebewesen/Dinge links}}{\text{Lebewesen/Dinge rechts}}$

(Wer/Was befindet sich momentan auf welcher Seite des Flusses?)

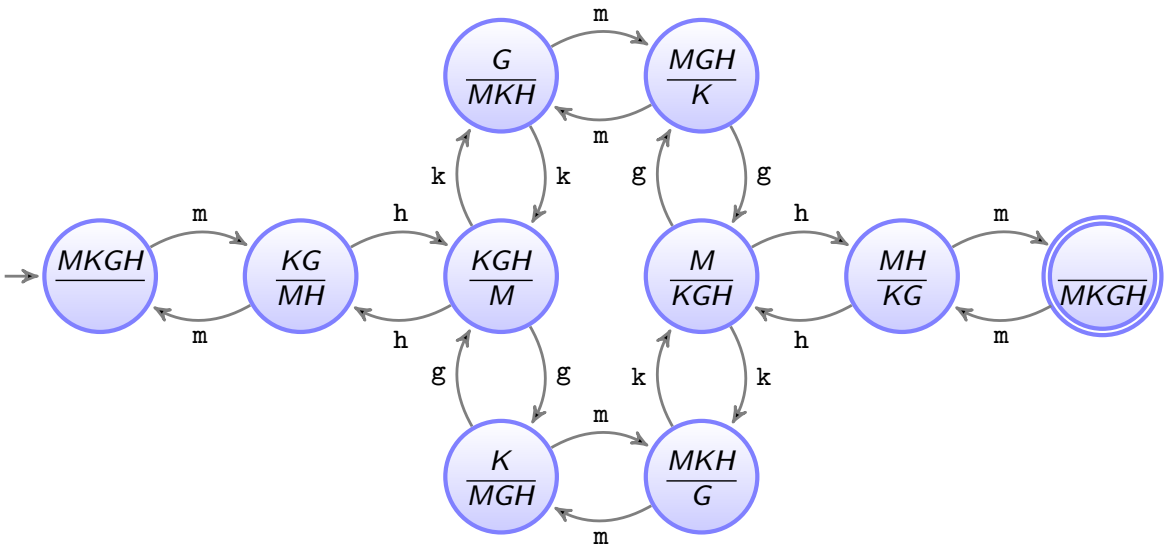
Anfangszustand: $\underline{MKGH}$

Endzustand (Ziel): $\overline{MKGH}$

Verbotene Zustände: $\frac{GH}{MK}$, $\frac{MK}{GH}$, $\frac{KH}{MG}$, $\frac{MG}{KH}$, $\frac{H}{MKG}$, $\frac{MKG}{H}$

Zustandsübergänge:
`h`, `m`, `k`, `g` ... Homer fährt alleine/mit Maggie/KR/Gift über den Fluss.

Automat (ohne verbotene Zustände und Übergänge):



Mögliche Lösungen: $\{\texttt{mhkmghm}, \texttt{mmmhhhgmkhm}, \ldots, \texttt{mhkmgkmgkmghm}, \ldots\}$

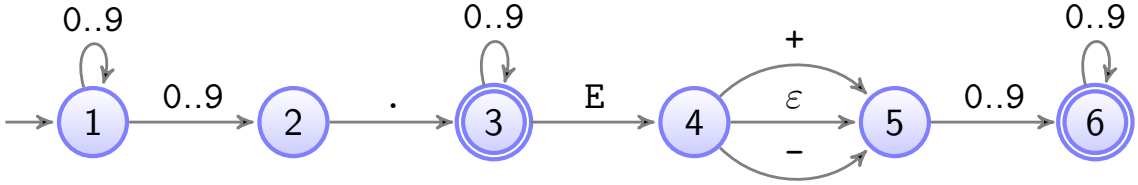
„Sprache des Automaten“

Beispiel: Reelle Numerale mit Exponentialteil

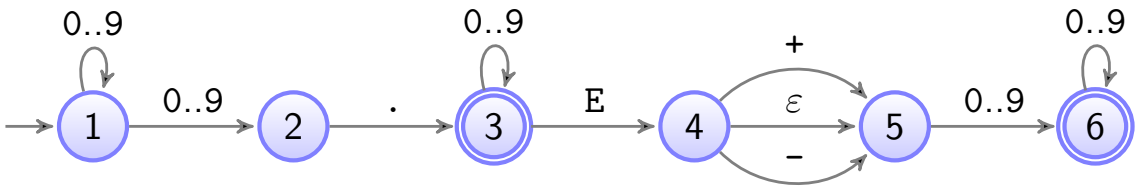
Z.B. `3.14`, `0.314E1` ($= 0.314 \cdot 10^1$), `314.E-2` ($= 314 \cdot 10^{-2}$)

- Mindestens eine Ziffer vor Dezimalpunkt
- Dezimalpunkt
- Nachkommastellen optional
- Exponentialteil optional:
  - eingeleitet durch `E`
  - Vorzeichen optional
  - mindestens eine Ziffer

Endlicher Automat für die reellen Numerale



`0..9` ... Abkürzung für 10 parallele Übergänge beschriftet mit 0 bis 9.
ε ... Leerwort, „Nichts“



- Zustandsbeschriftungen 1–6 dienen nur der Bezugnahme, irrelevant für das Verhalten des Automaten
- Kanten sind mit Symbolen beschriftet, die gelesen/geschrieben werden.
- Anfangszustand (1) ist durch einen Pfeil aus dem Nichts markiert.
- Endzustände (3, 6) sind durch einen Doppelkreis markiert.

Zwei Sichtweisen:

- Akzeptor: Der Automat liest Symbole und akzeptiert alle Zeichenketten, die vom Anfangs- zu einem der Endzustände führen.
- Generator: Der Automat schreibt Symbole und generiert jene Zeichenketten, die vom Anfangs- zu einem der Endzustände führen.

Was Sie heute erwartet

Endliche Automaten modellieren Systeme bzw. Abläufe, die nur eine begrenzte, feste Zahl an unterscheidbaren Zuständen besitzen.

Kennzeichen:

- endliche Menge von Zuständen
- Übergänge zwischen Zuständen
- Eingaben, die die Übergänge steuern.
- Ausgaben oder Aktionen, die in den Zuständen oder während der Übergänge getätigt werden.
- Anfangszustand
- Endzustände (optional)
- deterministisch: Der momentane Zustand und die nächste Eingabe bestimmen eindeutig den Folgezustand.
  nichtdeterministisch: Es gibt Zustände, die bei manchen Eingaben mehrere mögliche Folgezustände besitzen.

Arten endlicher Automaten

(Klassischer) Endlicher Automat:

- Anfangs- und Endzustände
- nur Eingaben (bzw. nur Ausgaben)
- Ein-/Ausgaben verknüpft mit Zustandsübergängen
- verarbeitet endliche Symbolfolgen
- Unterarten: deterministisch, nichtdeterministisch mit/ohne ε-Übergängen

Transducer: wie endlicher Automat, aber mit Ein- und Ausgaben.

Mealy-Automat: deterministischer Transducer; in der Regel keine Endzustände, verarbeitet daher unendliche Symbolfolgen.

Moore-Automat: wie Mealy-Automat, die Ausgaben sind aber mit den Zuständen verknüpft.

Büchi-Automat: wie endlicher Automat, verarbeitet aber unendliche Symbolfolgen

Weitere Typen: Verallgemeinerter endlicher Automat, Muller-Automat, Rabin-Automat, Baumautomaten, ...

Englische Begriffe: automaton/automata, finite state machine, DFA (Deterministic Finite Automaton), NFA (Non-Determinstic FA)

Weitere (nicht-endliche) Automatenarten: Kellerautomaten (Push-down automata), Turing-Maschinen, Registermaschinen etc. können Ausgaben wieder lesen $\Rightarrow$ zusätzlicher Speicher, mächtiger als endliche Automaten.

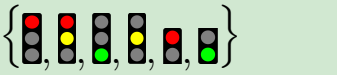
Spezifikation von Automaten:

- Graphisch: Zustände sind Knoten, Übergänge sind Kanten, Ein- und Ausgaben sind Beschriftungen von Knoten und Kanten.
- Tabellarisch: Zu jedem Zustand und Eingabesymbol gibt es einen Eintrag mit zugehöriger Ausgabe und den Folgezuständen.

Was Sie heute erwartet

Formale Sprachen

Alphabet (Σ): endliche, nicht-leere Menge atomarer Symbole

- Menge aller lateinischen Buchstaben, Ziffern und Sonderzeichen
- Menge aller ägyptischen Hieroglyphen
- $\{$  $\}$
- $\{0, \ldots, 9, ., \mathtt{E}, +, -\}$
- $\{0, 1\}$
- $\{00, 01, 10, 11\}$

Wort über Σ: (endliche) Folge von Zeichen aus dem Alphabet Σ

ε ... Leerwort
$\Sigma^+ = \{ s_1 \cdots s_n \mid s_i \in \Sigma, 1 \leq i \leq n \}$ ... Menge aller nicht-leeren Wörter über Σ
$\Sigma^* = \Sigma^+ \cup \{\varepsilon\}$ ... Menge aller Wörter über Σ (inklusive Leerwort)

$w_1 \cdot w_2 = w_1 w_2$ ... Verkettung der Wörter $w_1, w_2 \in \Sigma^*$

$\langle \Sigma^*, \cdot, \varepsilon \rangle$ bildet ein Monoid

D.h.: Für alle Wörter $u, v, w \in \Sigma^*$ gelten folgende Gleichungen:
$(u \cdot v) \cdot w = u \cdot (v \cdot w)$ Assoziativität
$w \cdot \varepsilon = \varepsilon \cdot w = w$ Neutralität

$\Sigma = \{0, 1\}$
$\Sigma^* = \{\varepsilon, 0, 1, 00, 01, 10, 11, 000, 001, \ldots\}$
$10 \cdot \varepsilon \cdot 11101 \cdot 000 = 1011101000$ (Klammerung irrelevant, Assoziativität!)
$\varepsilon \cdot \varepsilon \cdot \varepsilon = \varepsilon$

Formale Sprache über Σ: beliebige Teilmenge von Σ^*

- die Menge aller deutschen Sätze (Alphabet: Buchstaben+Satzzeichen)
- die Menge aller Java-Programme (Alphabet: ASCII-Zeichen)
- $\{\}$, $\{\varepsilon\}$, Σ^*

2^{Σ^*} ... Menge aller Sprachen über Σ = Menge aller Teilmengen von Σ^*

Was Sie heute erwartet

Deterministische endliche Automaten

Deterministischer endlicher Automat (DEA)

... wird beschrieben durch ein 5-Tupel $\mathcal{A} = \langle Q, \Sigma, \delta, q_0, F \rangle$, wobei

- Q ... endliche Menge der Zustände
- Σ ... Eingabealphabet (*input alphabet*)
- $\delta\colon Q \times \Sigma \mapsto Q$ ... Übergangsfunktion (total) (*transition function*)
- $q_0 \in Q$ ... Anfangszustand (*initial state*)
- $F \subseteq Q$ ... Menge der Endzustände (*final states*)

δ ist eine totale Funktion: Folgezustand $\delta(q, s)$ ist für jeden Zustand $q \in Q$ und jede Eingabe $s \in \Sigma$ eindeutig definiert. $\Longrightarrow$ „deterministisch“

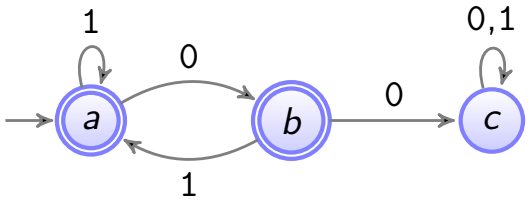
Erweiterte Übergangsfunktion $\delta^*\colon Q \times \Sigma^* \mapsto Q$

$\delta^*(q, \varepsilon) = q, \quad \delta^*(q, sw) = \delta^*(\delta(q, s), w)$ für alle $q \in Q$, $s \in \Sigma$, $w \in \Sigma^*$.

Akzeptierte/Generierte Sprache

$\mathcal{L}(\mathcal{A}) = \{ w \in \Sigma^* \mid \delta^*(q_0, w) \in F \}$

Beispiel: 00-freie Binärstrings



c ... „Falle“, Fehlerzustand
wird oft auch weggelassen

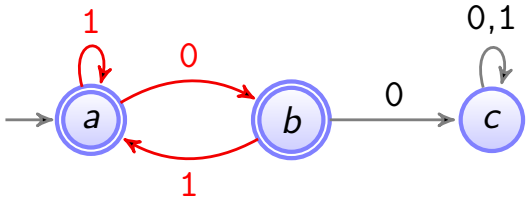
$\mathcal{A} = \langle Q, \Sigma, \delta, q_0, F \rangle$, wobei

- $Q = \{a, b, c\}$ ... Zustandsmenge
- $\Sigma = \{0, 1\}$ ... Eingabealphabet
- $\delta\colon Q \times \Sigma \mapsto Q$ ... Übergangsfunktion definiert durch:

δ	0	1
a	b	a
b	c	a
c	c	c

- $q_0 = a$ ... Anfangszustand
- $F = \{a, b\}$ ... Endzustände

Beispiel: 00-freie Binärstrings



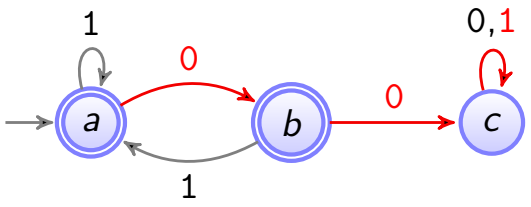
$$\begin{aligned}\delta^*(a, 101) &= \delta^*(\delta(a, 1), 01) \\ &= \delta^*(a, 01) \\ &= \delta^*(\delta(a, 0), 1) \\ &= \delta^*(b, 1) \\ &= \delta^*(\delta(b, 1), \varepsilon) \\ &= \delta^*(a, \varepsilon) \\ &= a\end{aligned}$$

$\delta^*(q, sw) = \delta^*(\delta(q, s), w)$

$\delta^*(q, \varepsilon) = q$

Das Wort 101 wird von $\mathcal{A}$ akzeptiert/generiert, d.h., $101 \in \mathcal{L}(\mathcal{A})$, weil $\delta^*(a, 101) = a$ ein Endzustand ist.

Beispiel: 00-freie Binärstrings

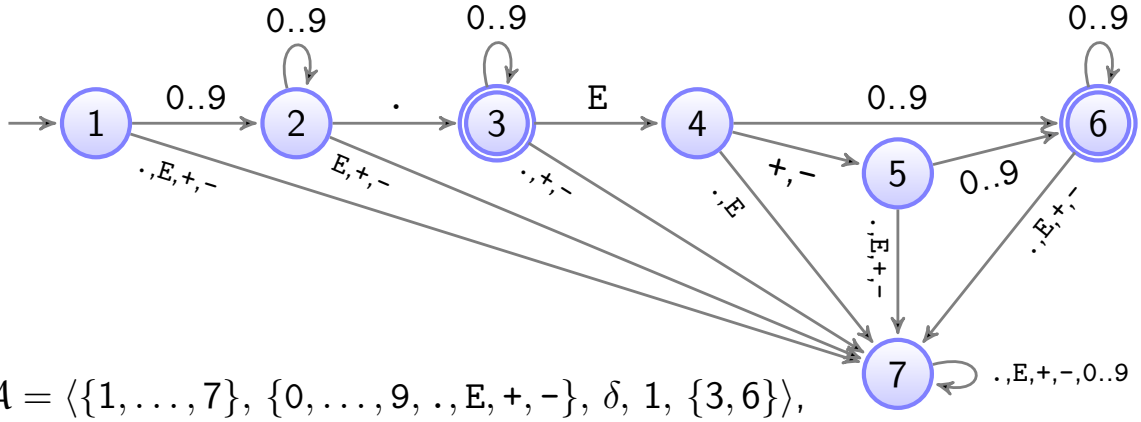


$$\begin{aligned}\delta^*(a,001) &= \delta^*(\delta(a,0),01) \\ &= \delta^*(b,01) \\ &= \delta^*(\delta(b,0),1) \\ &= \delta^*(c,1) \\ &= \delta^*(\delta(c,1),\varepsilon) \\ &= \delta^*(c,\varepsilon) \\ &= c\end{aligned} \qquad \begin{aligned}&\delta^*(q,sw) = \delta^*(\delta(q,s),w) \\ &\\ &\\ &\\ &\\ &\delta^*(q,\varepsilon) = q\end{aligned}$$

Das Wort 001 wird von $\mathcal{A}$ nicht akzeptiert/generiert, $001 \notin \mathcal{L}(\mathcal{A})$, weil $\delta^*(a,001) = c$ kein Endzustand ist.

$\mathcal{L}(\mathcal{A}) = \{\, w \in \{0,1\}^* \mid 00 \text{ kommt nicht in } w \text{ vor} \,\}$

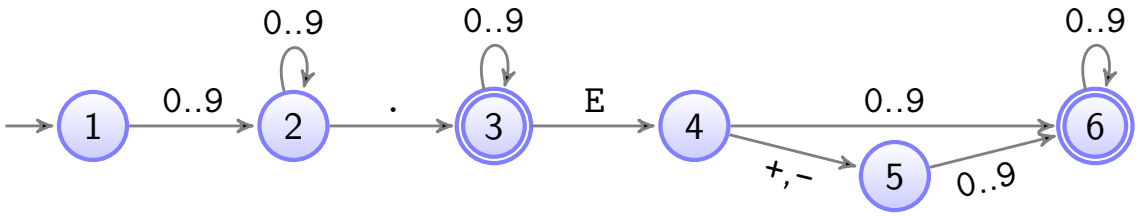
Beispiel: reelle Numerale



$\mathcal{A} = \langle \{1,\ldots,7\}, \{0,\ldots,9,\,.,\mathtt{E},+,-\}, \delta, 1, \{3,6\} \rangle$,
wobei

δ	$0 \cdots 9$	.	E	+	−
1	$2 \cdots 2$	7	7	7	7
2	$2 \cdots 2$	3	7	7	7
3	$3 \cdots 3$	7	4	7	7
4	$6 \cdots 6$	7	7	5	5
5	$6 \cdots 6$	7	7	7	7
6	$6 \cdots 6$	7	7	7	7
7	$7 \cdots 7$	7	7	7	7

Beispiel: reelle Numerale



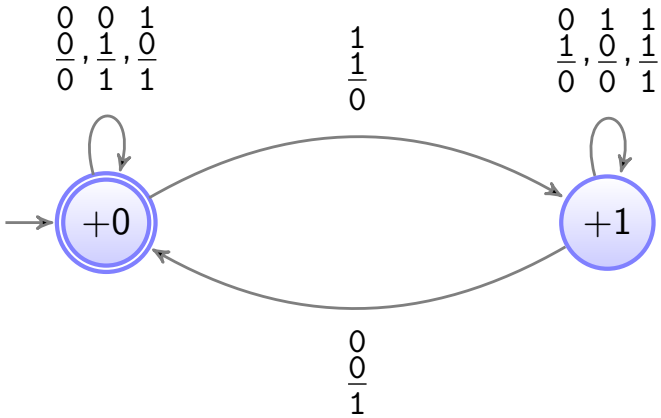
	δ	0 ··· 9	.	E	+	−
AZ	1	2 ··· 2	7	7	7	7
	2	2 ··· 2	3	7	7	7
EZ	3	3 ··· 3	7	4	7	7
	4	6 ··· 6	7	7	5	5
	5	6 ··· 6	7	7	7	7
EZ	6	6 ··· 6	7	7	7	7
	7	7 ··· 7	7	7	7	7

Beispiel: Binäraddition von rechts nach links (Kontrolle)

$$\begin{array}{cccccc|l} 0 & 0 & 1 & 0 & 1 & 1 & = 11_{10} \\ 0 & 0 & 0 & 1 & 0 & 1 & = 5_{10} \\ \hline 0 & 1 & 0 & 0 & 0 & 0 & = 16_{10} \end{array}$$

$\longleftarrow$

$Q = \{+0, +1\}$ $\quad \Sigma = \{\binom{0}{0}, \binom{0}{1}, \binom{1}{0}, \binom{1}{1}, \binom{0}{0}, \binom{0}{1}, \binom{1}{0}, \binom{1}{1}\}$



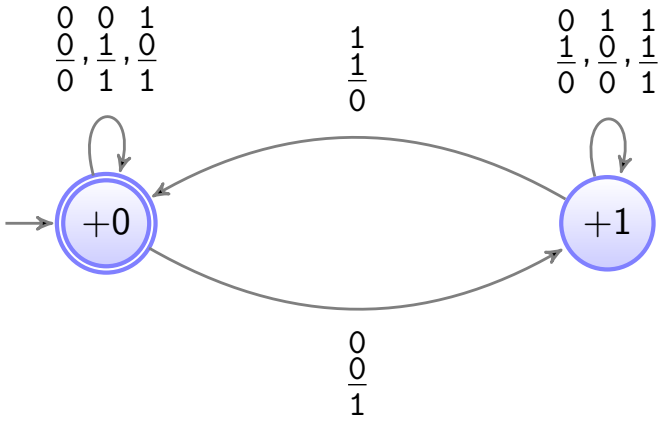
+0 ... kein Übertrag
+1 ... Übertrag

Beispiel: Binäraddition von links nach rechts (Kontrolle)

$$\begin{array}{cccccc|l} 0 & 0 & 1 & 0 & 1 & 1 & = 11_{10} \\ 0 & 0 & 0 & 1 & 0 & 1 & = 5_{10} \\ \hline 0 & 1 & 0 & 0 & 0 & 0 & = 16_{10} \end{array}$$

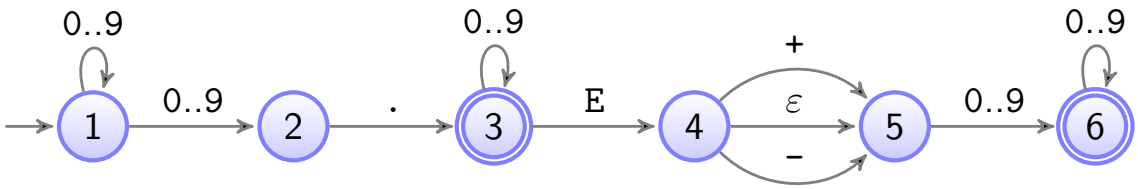
$\longrightarrow$

$Q = \{+0, +1\}$ $\quad \Sigma = \{\frac{0}{0}, \frac{0}{1}, \frac{1}{0}, \frac{1}{1}, \frac{0}{0}, \frac{0}{1}, \frac{1}{0}, \frac{1}{1}\}$



+0 . . . kein Übertrag
+1 . . . Übertrag

Was Sie heute erwartet



Kein deterministischer Automat!

- $\delta(1, 0) = 1$?
  $\delta(1, 0) = 2$?
  Die Übergangsfunktion muss ein eindeutiges Ergebnis besitzen.
- $\delta(4, \varepsilon) = 5$?
  Die Übergangsfunktion ist vom Typ $Q \times \Sigma \mapsto Q$, aber $\varepsilon \notin \Sigma$!

Indeterminismus: Der momentane Zustand und das Eingabesymbol legen den nächsten Zustand bzw. die nächste Aktion nicht eindeutig fest.

- In Zustand 1 sind bei Eingabe 0 die Folgezustände 1 und 2 möglich.
- In Zustand 3 sind bei Eingabe E die Folgezustände 4 und 5 möglich.
- In Zustand 4 ist Zustand 5 mit und ohne Eingabe erreichbar.

Ob die richtige Entscheidung getroffen wurde, wird erst später klar.

Nichtdeterministische endliche Automaten

Nichtdeterministischer endlicher Automat (NEA)

... wird beschrieben durch ein 5-Tupel $\mathcal{A} = \langle Q, \Sigma, \delta, q_0, F \rangle$, wobei

- Q, Σ, q_0, F ... siehe DEAs
- $\delta \subseteq Q \times (\Sigma \cup \{\varepsilon\}) \times Q$ ... Übergangsrelation

DEA: $\delta\colon Q \times \Sigma \mapsto Q$ ... totale Übergangsfunktion

Erweiterte Übergangsrelation $\delta^* \subseteq Q \times \Sigma^* \times Q$

δ^* ist die kleinste Menge mit folgenden Eigenschaften:

- $(q, \varepsilon, q) \in \delta^*$ für alle $q \in Q$
- Wenn $(q_1, w, q_2) \in \delta^*$ und $(q_2, s, q_3) \in \delta$, dann $(q_1, ws, q_3) \in \delta^*$.

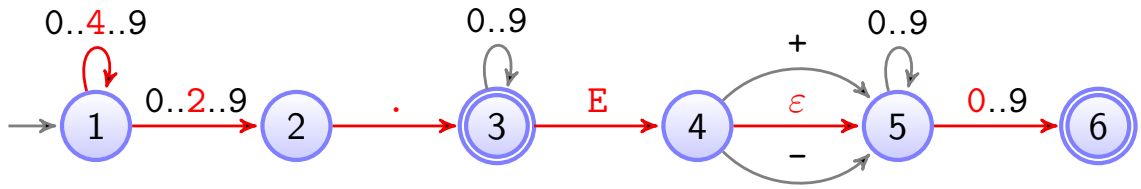
DEA: $\delta^*(q, \varepsilon) = q, \quad \delta^*(q, sw) = \delta^*(\delta(q, s), w)$

Akzeptierte/Generierte Sprache

$\mathcal{L}(\mathcal{A}) = \{ w \in \Sigma^* \mid (q_0, w, q_f) \in \delta^* \text{ für ein } q_f \in F \}$

DEA: $\mathcal{L}(\mathcal{A}) = \{ w \in \Sigma^* \mid \delta^*(q_0, w) \in F \}$

Beispiel: 42.E0 ist ein reelles Numeral



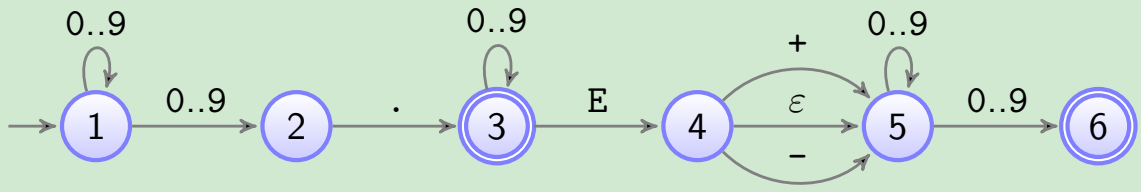
$\mathcal{L}(\mathcal{A}) = \{\, w \in \Sigma^* \mid (1, w, 3) \in \delta^* \text{ oder } (1, w, 6) \in \delta^* \,\}$

Zu zeigen: $\texttt{42.E0} \in \mathcal{L}(\mathcal{A})$

Wenn $(q_1, w, q_2) \in \delta^*$ und $(q_2, s, q_3) \in \delta$, dann $(q_1, ws, q_3) \in \delta^*$.		
$(1, \varepsilon, 1)$	$(1, 4, 1)$	$(1, 4, 1)$
$(1, 4, 1)$	$(1, 2, 2)$	$(1, 42, 2)$
$(1, 42, 2)$	$(2, \texttt{.}, 3)$	$(1, \texttt{42.}, 3)$
$(1, \texttt{42.}, 3)$	$(3, \texttt{E}, 4)$	$(1, \texttt{42.E}, 4)$
$(1, \texttt{42.E}, 4)$	$(4, \varepsilon, 5)$	$(1, \texttt{42.E}, 5)$
$(1, \texttt{42.E}, 5)$	$(5, 0, 6)$	$(1, \texttt{42.E0}, 6)$

$(1, \texttt{42.E0}, 6) \in \delta^*$, 1 ist Anfangs- und 6 Endzustand $\Longrightarrow \texttt{42.E0} \in \mathcal{L}(\mathcal{A})$

Tabellarische Darstellung der Übergangsrelation $\delta \subseteq Q \times (\Sigma \cup \{\varepsilon\}) \times Q$
Für jeden Zustand $q \in Q$ und jede Eingabe $s \in \Sigma \cup \{\varepsilon\}$:
Tabelleneintrag mit der Menge $\{\, q' \in Q \mid (q, s, q') \in \delta \,\}$ der Folgezustände



	δ	0	$\cdots$	9	.	E	+	−	ε
AZ	1	{1,2}	$\cdots$	{1,2}	{}	{}	{}	{}	{}
	2	{}	$\cdots$	{}	{3}	{}	{}	{}	{}
EZ	3	{3}	$\cdots$	{3}	{}	{4}	{}	{}	{}
	4	{}	$\cdots$	{}	{}	{}	{5}	{5}	{5}
	5	{5,6}	$\cdots$	{5,6}	{}	{}	{}	{}	{}
EZ	6	{}	$\cdots$	{}	{}	{}	{}	{}	{}

Alternative Definition von NEAs:
Übergangsfunktion $\delta\colon Q \times (\Sigma \cup \{\varepsilon\}) \mapsto 2^Q$ (ist total!) an Stelle von
Übergangsrelation $\delta \subseteq Q \times (\Sigma \cup \{\varepsilon\}) \times Q$

Vergleich DEA – NEA

NEAs sind flexibler:

- Mehrere Folgezustände pro Zustand und Eingabe möglich;
- Kein Folgezustand zu einem Zustand und einer Eingabe erlaubt;
- Zustandswechsel ohne Eingabe möglich (ε-Übergang).

DEAs und NEAs besitzen dieselbe Ausdrucksstärke.

- Jeder DEA ist per Definition auch ein NEA.
- Zu jedem NEA gibt es einen DEA, der dieselbe Sprache akzeptiert. (Lässt sich automatisch finden, siehe später.)

Vorteile von NEAs:

- Benötigen teilweise erheblich weniger Zustände und Übergänge als äquivalente DEAs.
  Die Zustandszahl im DEA kann exponentiell größer sein als im NEA.
- Bei Modellierungsaufgaben leichter zu konstruieren.

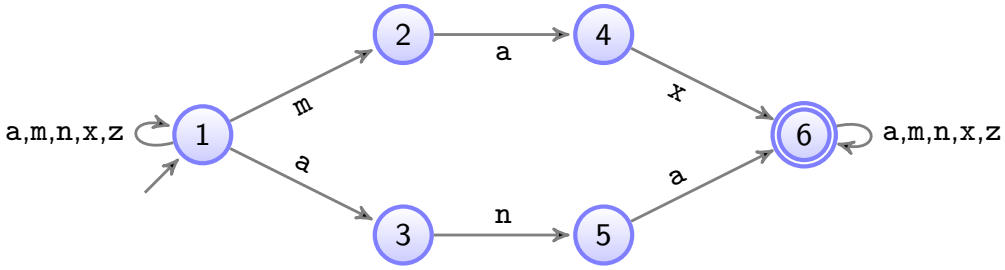
Vorteile von DEAs:

- Effiziente Abarbeitung, kein Backtracking.

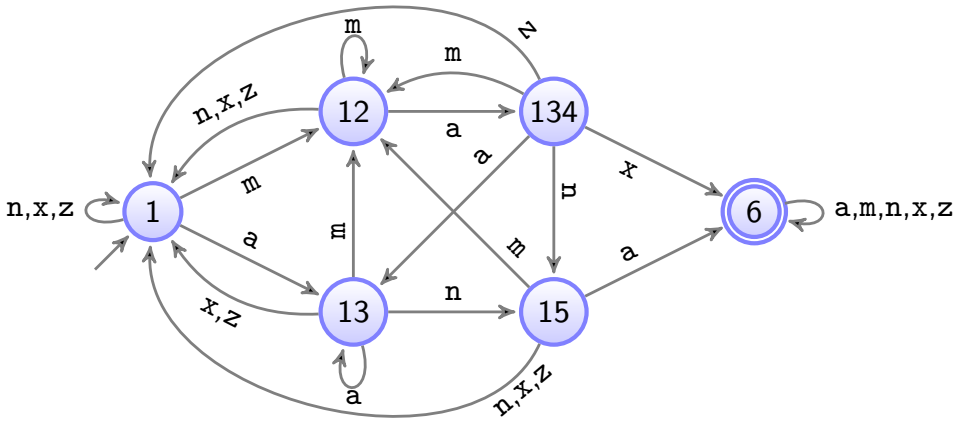
Beispiel: Suche nach Max und Ana

Gesucht: Automat zur Suche nach „max“ und „ana“ in einem Text
$\Sigma = \{\mathtt{a}, \mathtt{m}, \mathtt{n}, \mathtt{x}, \mathtt{z}\}$ (z ... Stellvertreter für b–l, o–w, y, z, ...)

NEA:



DEA:



3.0 VU Formale Modellierung

Gernot Salzer

Forschungsbereich Theory and Logic
Institut für Logic and Computation

17.4.2018

Was Sie letztes Mal hörten

Dualität von Funktionen, Operatoren und Formeln

Duale Funktionen

Zwei n-stellige Funktionen f und g heißen dual zueinander, wenn gilt: $\text{not}\, f(x_1, \ldots, x_n) = g(\text{not}\, x_1, \ldots, \text{not}\, x_n)$.

Duale Operatoren

Zwei Operatoren heißen dual, wenn die zugehörigen Funktionen dual sind.

Duale Formeln

Zwei Formeln $F[A_1, \ldots, A_n]$ und $G[A_1, \ldots, A_n]$ heißen dual zueinander, wenn gilt: $\neg F[A_1, \ldots, A_n] = G[\neg A_1, \ldots, \neg A_n]$

$\neg F[\neg A_1, \ldots, \neg A_n]$ ist dual zu $F[A_1, \ldots, A_n]$.

Sei G die Formel, die aus F durch Ersetzen aller Operatoren durch ihre dualen hervorgeht. Dann ist G dual zu F.

$F = G$ gilt genau dann, wenn $F^* = G^*$ gilt.

Was Sie letztes Mal hörten

The Simpsons – aussagenlogische Modellierung

If all you have is a hammer, everything looks like a nail.

M_i ... Maggie
K_i ... Knecht Ruprecht
G_i ... Gift
H_i ... Homer

} ... befindet sich zum Zeitpunkt i auf der anderen Seite des Flusses.

- Zum Zeitpunkt i befinden sich alle auf dieser Flussseite.

$\text{AlleHier}(i) := \neg M_i \wedge \neg K_i \wedge \neg G_i \wedge \neg H_i$

- Zum Zeitpunkt i befinden sind alle auf der anderen Flussseite.

$\text{AlleDort}(i) := M_i \wedge K_i \wedge G_i \wedge H_i$

- Wenn sich Maggie und KR oder Maggie und das Gift am selben Flussufer befinden, muss Homer bei Maggie sein.

$\text{Sicher}(i) := ((M_i \equiv K_i) \vee (M_i \equiv G_i)) \supset (M_i \equiv H_i)$

MH_i ... Maggie
KH_i ... Knecht Ruprecht
GH_i ... Gift

} ... fährt mit Homer über den Fluss (zw. den Zeitpunkten $i-1$ und i).

HH_i ... Homer fährt alleine über den Fluss (zwischen $i-1$ und i).

- Genau eine Überfahrt zwischen den Zeitpunkten $i-1$ und i.

$\text{Überfahrt}(i) :=$

$(MH_i \wedge \neg KH_i \wedge \neg GH_i \wedge \neg HH_i) \vee (\neg MH_i \wedge KH_i \wedge \neg GH_i \wedge \neg HH_i) \vee$
$(\neg MH_i \wedge \neg KH_i \wedge GH_i \wedge \neg HH_i) \vee (\neg MH_i \wedge \neg KH_i \wedge \neg GH_i \wedge HH_i)$

- Definition der Überfahrten:

$\text{DefÜberfahrt}(i) :=$

$$\begin{aligned}
&(MH_i \supset ((M_{i-1} \not\equiv M_i) \wedge (K_{i-1} \equiv K_i) \wedge (G_{i-1} \equiv G_i) \wedge (H_{i-1} \not\equiv H_i) \wedge (H_i \equiv M_i))) \\
\wedge\ &(KH_i \supset ((M_{i-1} \equiv M_i) \wedge (K_{i-1} \not\equiv K_i) \wedge (G_{i-1} \equiv G_i) \wedge (H_{i-1} \not\equiv H_i) \wedge (H_i \equiv K_i))) \\
\wedge\ &(GH_i \supset ((M_{i-1} \equiv M_i) \wedge (K_{i-1} \equiv K_i) \wedge (G_{i-1} \not\equiv G_i) \wedge (H_{i-1} \not\equiv H_i) \wedge (H_i \equiv G_i))) \\
\wedge\ &(HH_i \supset ((M_{i-1} \equiv M_i) \wedge (K_{i-1} \equiv K_i) \wedge (G_{i-1} \equiv G_i) \wedge (H_{i-1} \not\equiv H_i))
\end{aligned}$$

Gesamtformel: Nach n Überfahrten sollen alle auf der anderen Seite sein.

$$\text{Simpsons}(n) := \text{AlleHier}(0) \wedge \text{AlleDort}(n) \wedge \bigwedge_{i=0}^{n} \text{Sicher}(i) \wedge \bigwedge_{i=1}^{n} \text{Überfahrt}(i) \wedge \bigwedge_{i=1}^{n} \text{DefÜberfahrt}(i)$$

Methode zum Lösen des Rätsels:

1. Errate die benötigte Zahl n der Überfahrten.
2. Finde eine erfüllende Interpretation für die Formel Simpsons(n) (z.B. mit Hilfe eines SAT-Solvers).

Eine mögliche Lösung:
$n = 7$,
$I(MH_1) = I(HH_2) = I(GH_3) = I(MH_4) = I(KH_5) = I(HH_6) = I(MH_7) = 1$

Vor- und Nachteile dieser aussagenlogischen Modellierung

Vorteile:

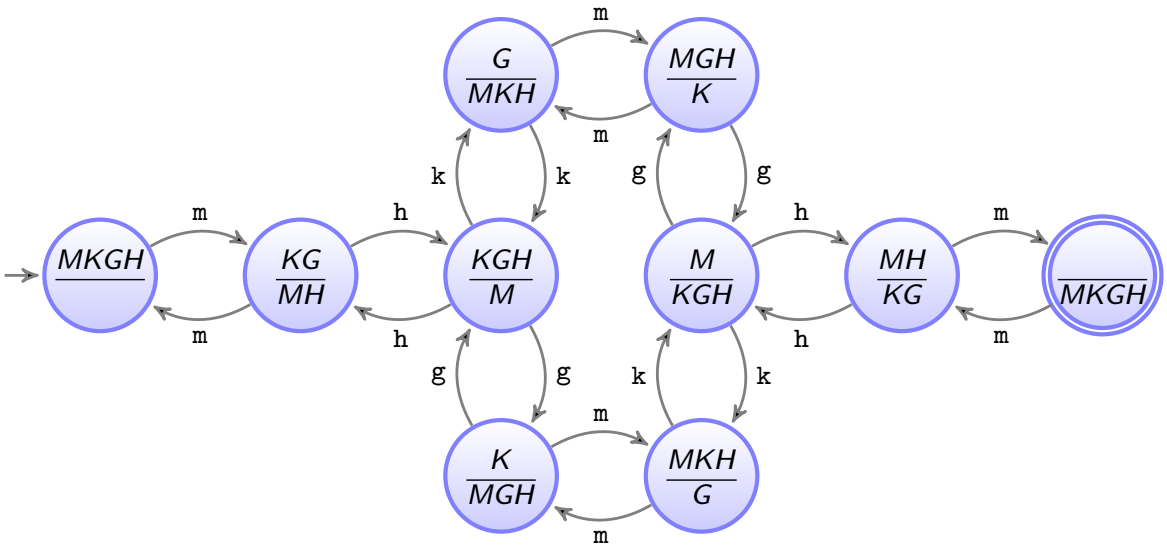
- deklarativ-statisch, nicht prozedural-dynamisch
  Welche Eigenschaften sollen gelten?
  Nicht: Welche Schritte sind für Lösung erforderlich?
- modular
  Neue Bedingungen werden durch zusätzliche Formeln berücksichtigt.

Nachteile:

- Erraten von Parametern
  n muss durch Probieren gefunden werden
- Große Zahl an Variablen und Formeln
  Dynamik muss durch indizierte Variablen simuliert werden.
- Frame Problem
  Bei jeder Aktion muss auch definiert werden, was sich nicht ändert.
- unintuitiv
  Bei der Modellierung von Abläufen denkt man an Zustände und Übergänge, nicht an statische Bedingungen.

Was Sie letztes Mal hörten

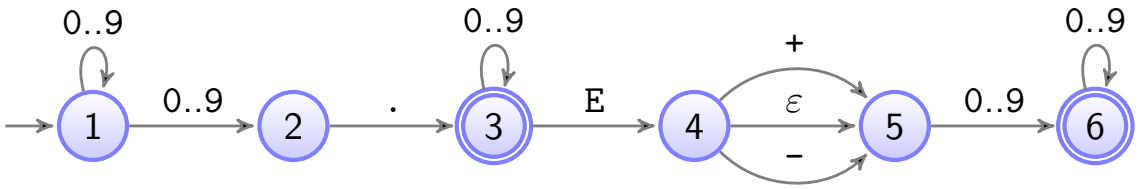
Beispiel: Flussüberquerung als Automat



Mögliche Lösungen: $\{\texttt{mhkmghm}, \texttt{mmmhhhgmkhm}, \ldots, \texttt{mhkmgkmgkmghm}, \ldots\}$

„Sprache des Automaten“

Beispiel: Reelle Numerale



- Zustandsbeschriftungen 1–6 dienen nur der Bezugnahme, irrelevant für das Verhalten des Automaten
- Kanten sind mit Symbolen beschriftet, die gelesen/geschrieben werden.
- Anfangszustand (1) ist durch einen Pfeil aus dem Nichts markiert.
- Endzustände (3, 6) sind durch einen Doppelkreis markiert.

Zwei Sichtweisen:

- Akzeptor: Der Automat liest Symbole und akzeptiert alle Zeichenketten, die vom Anfangs- zu einem der Endzustände führen.
- Generator: Der Automat schreibt Symbole und generiert jene Zeichenketten, die vom Anfangs- zu einem der Endzustände führen.

Was Sie letztes Mal hörten

Endliche Automaten modellieren Systeme bzw. Abläufe, die nur eine begrenzte, feste Zahl an unterscheidbaren Zuständen besitzen.

(Klassischer) Endlicher Automat:

- Anfangs- und Endzustände
- nur Eingaben (bzw. nur Ausgaben)
- verarbeitet endliche Symbolfolgen
- Unterarten: deterministisch, nichtdeterministisch mit/ohne ε-Übergängen

Transducer: wie endlicher Automat, aber mit Ein- und Ausgaben.
Unterarten: Mealy-Automaten, Moore-Automaten

Büchi-Automat: wie endlicher Automat, verarbeitet aber unendliche Symbolfolgen

Spezifikation von Automaten:

- Graphisch: Zustände sind Knoten, Übergänge sind Kanten, Ein- und Ausgaben sind Beschriftungen von Knoten und Kanten.
- Tabellarisch: Zu jedem Zustand und Eingabesymbol gibt es einen Eintrag mit zugehöriger Ausgabe und den Folgezuständen.

Was Sie letztes Mal hörten

Formale Sprachen

Σ	Alphabet, d.h., endliche, nicht-leere Menge atomarer Symbole
w	Wort über Σ, (endliche) Folge von Zeichen aus dem Alphabet Σ
ε	Leerwort
Σ^+	Menge aller nicht-leeren endlichen Wörter über Σ
Σ^*	Menge aller endlichen Wörter über Σ (inklusive Leerwort)

$w_1 \cdot w_2 = w_1 w_2$ Verkettung der Wörter $w_1, w_2 \in \Sigma^*$
$L \subseteq \Sigma^*$ formale Sprache über Σ
2^{Σ^*} Menge aller Sprachen über Σ

$\langle \Sigma^*, \cdot, \varepsilon \rangle$ bildet ein Monoid.

$\Sigma = \{0, 1\}$
$\Sigma^* = \{\varepsilon, 0, 1, 00, 01, 10, 11, 000, 001, \dots\}$
$10 \cdot \varepsilon \cdot 11101 \cdot 000 = 1011101000$ (Klammerung irrelevant, Assoziativität!)
$\varepsilon \cdot \varepsilon \cdot \varepsilon = \varepsilon$

Was Sie letztes Mal hörten

Endliche Automaten

Bezeichnungen

Q	endliche Menge von Zuständen
q_0	Anfangszustand
F	Menge von Endzuständen
Σ	Eingabealphabet
δ	Übergangsfunktion/-relation
δ^*	erweiterte Übergangsfunktion/-relation
$\mathcal{A}$	Automat
$\mathcal{L}(\mathcal{A})$	die von $\mathcal{A}$ akzeptierte Sprache

Übergangsfunktionen und -relationen

$\delta\colon Q \times \Sigma \mapsto Q$	det. endlicher Automat (DEA)
$\delta \subseteq Q \times \Sigma \times Q$ $\delta\colon Q \times \Sigma \mapsto 2^Q$	nichtdet. endl. Aut. (NEA) ohne ε-Ü.
$\delta \subseteq Q \times (\Sigma \cup \{\varepsilon\}) \times Q$ $\delta\colon Q \times (\Sigma \cup \{\varepsilon\}) \mapsto 2^Q$	nichtdet. endl. Aut. (NEA) mit ε-Ü.

ε-Übergänge, Relation oder Potenzmenge $\Longrightarrow$ nichtdeterm. Automat

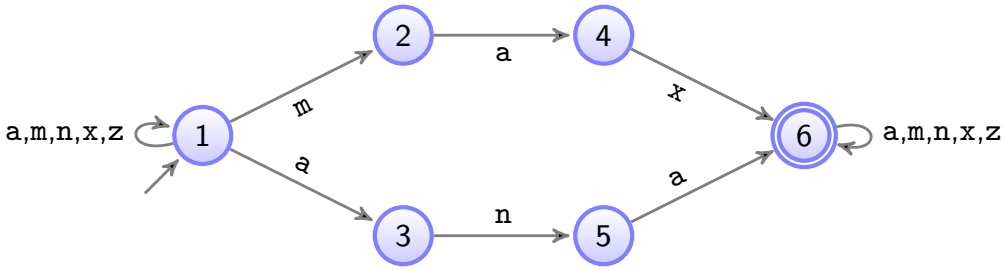
Akzeptierte Wörter

Alle Wörter, mit denen man vom Anfangszustand aus einen der Endzustände erreicht.

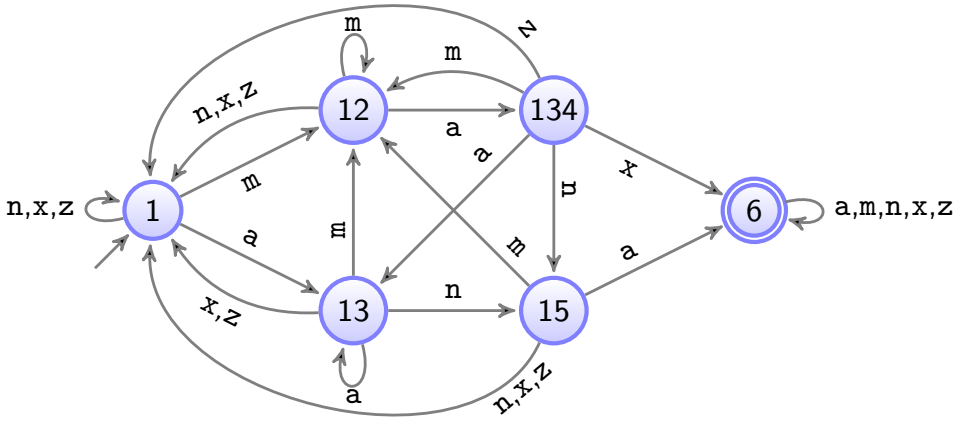
Beispiel: Suche nach Max und Ana

Gesucht: Automat zur Suche nach „max“ und „ana“ in einem Text
$\Sigma = \{\mathtt{a}, \mathtt{m}, \mathtt{n}, \mathtt{x}, \mathtt{z}\}$ (z ... Stellvertreter für b–l, o–w, y, z, ...)

NEA:

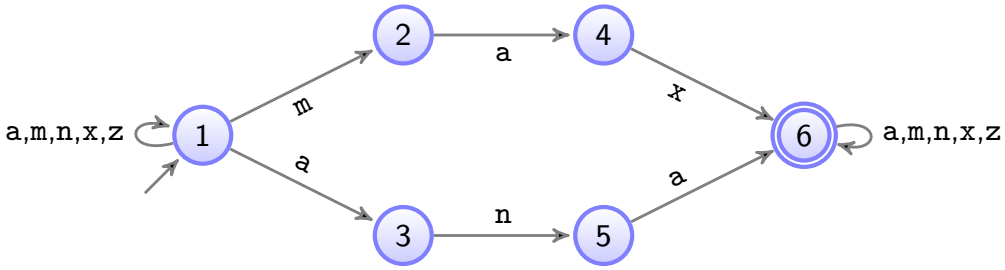


DEA:



Was Sie heute erwartet

Beispiel: Suche nach Max und Ana

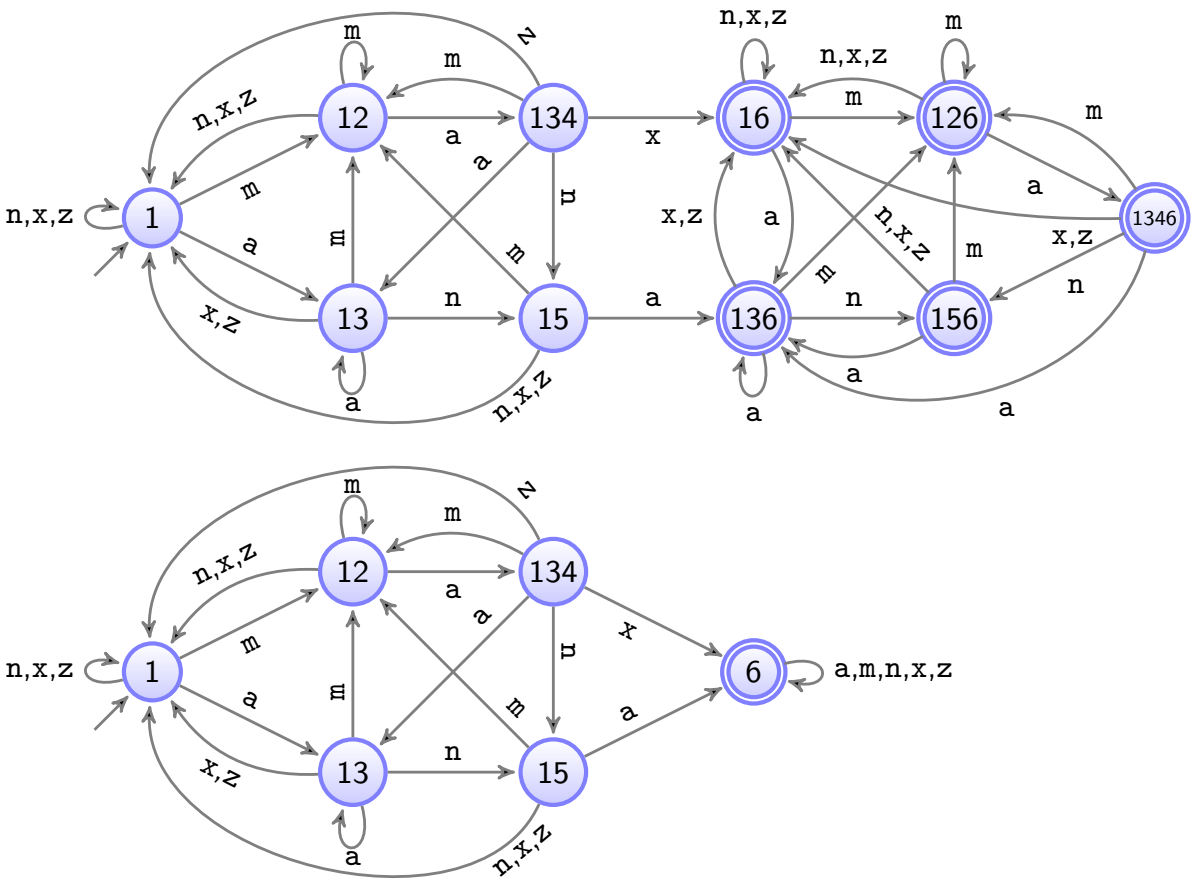


	δ^*	a	m	n	x	z
SZ	1	$\{1,3\}$	$\{1,2\}$	$\{1\}$	$\{1\}$	$\{1\}$
	2	$\{4\}$	$\{\}$	$\{\}$	$\{\}$	$\{\}$
	3	$\{\}$	$\{\}$	$\{5\}$	$\{\}$	$\{\}$
	4	$\{\}$	$\{\}$	$\{\}$	$\{6\}$	$\{\}$
	5	$\{6\}$	$\{\}$	$\{\}$	$\{\}$	$\{\}$
EZ	6	$\{6\}$	$\{6\}$	$\{6\}$	$\{6\}$	$\{6\}$

Identisch mit der Tabelle für δ, wenn es keine ε-Kanten gibt.

	δ^*	a	m	n	x	z
SZ	1	$\{1,3\}$	$\{1,2\}$	$\{1\}$	$\{1\}$	$\{1\}$
	2	$\{4\}$	$\{\}$	$\{\}$	$\{\}$	$\{\}$
	3	$\{\}$	$\{\}$	$\{5\}$	$\{\}$	$\{\}$
	4	$\{\}$	$\{\}$	$\{\}$	$\{6\}$	$\{\}$
	5	$\{6\}$	$\{\}$	$\{\}$	$\{\}$	$\{\}$
EZ	6	$\{6\}$	$\{6\}$	$\{6\}$	$\{6\}$	$\{6\}$

	$\widehat{\delta}$	a	m	n	x	z
SZ	$\{1\}$	$\{1,3\}$	$\{1,2\}$	$\{1\}$	$\{1\}$	$\{1\}$
	$\{1,2\}$	$\{1,3,4\}$	$\{1,2\}$	$\{1\}$	$\{1\}$	$\{1\}$
	$\{1,3\}$	$\{1,3\}$	$\{1,2\}$	$\{1,5\}$	$\{1\}$	$\{1\}$
	$\{1,3,4\}$	$\{1,3\}$	$\{1,2\}$	$\{1,5\}$	$\{1,6\}$	$\{1\}$
	$\{1,5\}$	$\{1,3,6\}$	$\{1,2\}$	$\{1\}$	$\{1\}$	$\{1\}$
EZ	$\{1,6\}$	$\{1,3,6\}$	$\{1,2,6\}$	$\{1,6\}$	$\{1,6\}$	$\{1,6\}$
EZ	$\{1,3,6\}$	$\{1,3,6\}$	$\{1,2,6\}$	$\{1,5,6\}$	$\{1,6\}$	$\{1,6\}$
EZ	$\{1,2,6\}$	$\{1,3,4,6\}$	$\{1,2,6\}$	$\{1,6\}$	$\{1,6\}$	$\{1,6\}$
EZ	$\{1,5,6\}$	$\{1,3,6\}$	$\{1,2,6\}$	$\{1,6\}$	$\{1,6\}$	$\{1,6\}$
EZ	$\{1,3,4,6\}$	$\{1,3,6\}$	$\{1,2,6\}$	$\{1,5,6\}$	$\{1,6\}$	$\{1,6\}$



Determinisierung

Gegeben: NEA $\mathcal{A} = \langle Q, \Sigma, \delta, q_0, F \rangle$ mit $\delta \subseteq Q \times (\Sigma \cup \{\varepsilon\}) \times Q$

Gesucht: DEA $\widehat{\mathcal{A}} = \langle \widehat{Q}, \Sigma, \widehat{\delta}, \widehat{q_0}, \widehat{F} \rangle$ mit $\widehat{\delta}\colon \widehat{Q} \times \Sigma \mapsto \widehat{Q}$,
sodass $\mathcal{A}$ und $\widehat{\mathcal{A}}$ dieselbe Sprache akzeptieren.

Wir definieren den deterministischen Automaten $\widehat{\mathcal{A}}$ folgendermaßen:

- $\widehat{Q} = 2^Q$
- $\widehat{q_0} = \{q_0\}$
- $\widehat{F} = \begin{cases} \{\, \widehat{q} \in \widehat{Q} \mid \widehat{q} \cap F \neq \emptyset \,\} \cup \{\widehat{q_0}\} & \text{falls } \varepsilon \in \mathcal{L}(\mathcal{A}) \\ \{\, \widehat{q} \in \widehat{Q} \mid \widehat{q} \cap F \neq \emptyset \,\} & \text{sonst} \end{cases}$
- Für alle Zustände $\widehat{q} \in \widehat{Q}$ und alle Symbole $s \in \Sigma$:

  $\widehat{\delta}(\widehat{q}, s) = \{\, q' \in Q \mid \text{es gibt } q \in \widehat{q}, \text{ sodass } (q, s, q') \in \delta^* \,\}$

$\mathcal{A}$ und $\widehat{\mathcal{A}}$ sind äquivalent, d.h., sie akzeptieren dieselbe Sprache:
$\mathcal{L}(\mathcal{A}) = \mathcal{L}(\widehat{\mathcal{A}})$.

Anmerkungen

Neue Endzustände: Ein neuer Zustand $\widehat{q}$ ist Endzustand, ...

- wenn seine Bezeichnung einen der alten Endzustände enthält, d.h., wenn $\widehat{q} \cap F \neq \emptyset$, oder
- wenn es sich um den neuen Startzustand handelt und der alte Automat das Leerwort akzeptiert,
  d.h., wenn $\widehat{q} = \widehat{q_0}$ und $\varepsilon \in \mathcal{L}(\mathcal{A})$,
  d.h., wenn $\widehat{q} = \widehat{q_0}$ und $(q_0, \varepsilon, f) \in \delta^*$ für einen Endzustand $f \in F$.

Neue Übergangsfunktion: Wegen

$$\widehat{\delta}(\widehat{q}, s) = \bigcup_{q \in \widehat{q}} \{\, q' \in Q \mid (q, s, q') \in \delta^* \,\} = \bigcup_{q \in \widehat{q}} \delta^*(q, s)$$

spart es Arbeit, wenn man zuerst $\delta^*(q, s)$ für alle $q \in Q$ und alle $s \in \Sigma$ berechnet. Danach müssen nur mehr die Zeilen, die $\widehat{q}$ entsprechen, vereinigt werden.

Neue Zustände: Betrachte nur jene Zustände aus 2^Q, die von $\widehat{q_0}$ aus erreichbar sind.

Was Sie heute erwartet

Transducer

Endlicher Transducer

... wird beschrieben durch ein 6-Tupel $\mathcal{A} = \langle Q, \Sigma, \Gamma, \delta, I, F \rangle$, wobei

- Q, Σ, F ... siehe DEAs
- Γ ... Ausgabealphabet (*output alphabet*)
- $\delta \subseteq Q \times (\Sigma \cup \{\varepsilon\}) \times (\Gamma \cup \{\varepsilon\}) \times Q$ ... Übergangsrelation
- $I \subseteq Q$ ... Anfangszustände

Erweiterte Übergangsrelation $\delta^* \subseteq Q \times \Sigma^* \times \Gamma^* \times Q$

δ^* ist die kleinste Menge mit folgenden Eigenschaften:

- $(q, \varepsilon, \varepsilon, q) \in \delta^*$ für alle $q \in Q$
- $(q_1, w, w', q_2) \in \delta^*,\ (q_2, s, s', q_3) \in \delta \quad \Longrightarrow \quad (q_1, ws, w's', q_3) \in \delta^*$

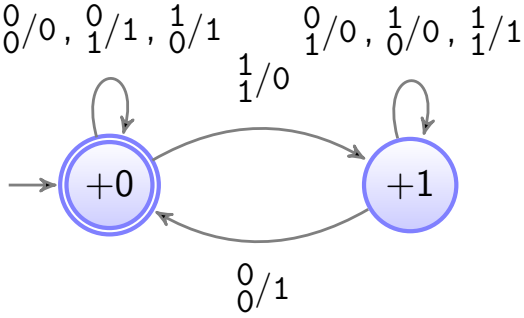
Übersetzungsrelation $[\mathcal{A}] \subseteq \Sigma^* \times \Gamma^*$

$[\mathcal{A}] = \{\, (w, w') \in \Sigma^* \times \Gamma^* \mid (i, w, w', f) \in \delta^* \text{ für ein } i \in I \text{ und ein } f \in F \,\}$

Beispiel: Binäraddition von rechts nach links

$$\begin{array}{cccccc} 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 \\ \hline 0 & 1 & 0 & 0 & 0 & 0 \end{array} \quad \begin{array}{l} = 11_{10} \\ = 5_{10} \\ \\ = 16_{10} \end{array}$$

$\leftarrow$



$\mathcal{A} = \langle \{+0, +1\}, \{\frac{0}{0}, \frac{0}{1}, \frac{1}{0}, \frac{1}{1}\}, \{0, 1\}, \delta, \{+0\}, \{+0\} \rangle$, wobei

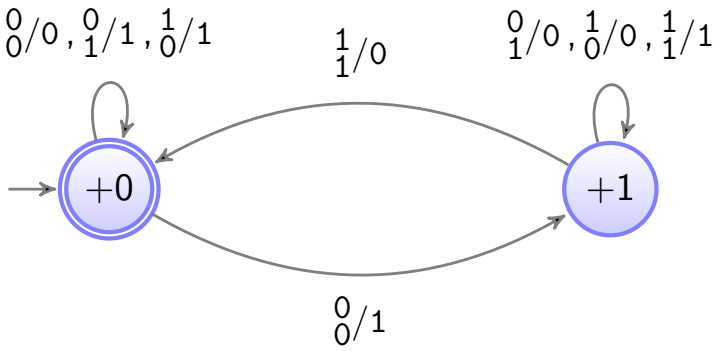
δ	$\frac{0}{0}$	$\frac{0}{1}$	$\frac{1}{0}$	$\frac{1}{1}$
$+0$	$\{(0, +0)\}$	$\{(1, +0)\}$	$\{(1, +0)\}$	$\{(0, +1)\}$
$+1$	$\{(1, +0)\}$	$\{(0, +1)\}$	$\{(0, +1)\}$	$\{(1, +1)\}$

$$[\mathcal{A}] = \{\, (\varepsilon, \varepsilon),\ (\tfrac{0}{0}, 0),\ (\tfrac{0}{1}, 1),\ (\tfrac{1}{0}, 1),$$
$$(\tfrac{0\,0}{0\,0}, 00),\ (\tfrac{0\,0}{0\,1}, 01), \ldots, (\tfrac{1\,0}{1\,0}, 01), \ldots, (\tfrac{1\,0\,1\,0}{1\,0\,1\,0}, 0101), \ldots \}$$

Binäraddition von links nach rechts

$$\begin{array}{cccccc|l} 0 & 0 & 1 & 0 & 1 & 1 & = 11_{10} \\ 0 & 0 & 0 & 1 & 0 & 1 & = 5_{10} \\ \hline 0 & 1 & 0 & 0 & 0 & 0 & = 16_{10} \end{array}$$

$\longrightarrow$



+0 . . . kein Übertrag
+1 . . . Übertrag

Achtung, Indeterminismus! Zustand „+0“ besitzt bei Eingabe $\binom{0}{0}$ zwei Folgezustände, ebenso Zustand „+1“ bei Eingabe $\binom{1}{1}$.

Was Sie heute erwartet

Mealy-Automat

... wird beschrieben durch ein 6-Tupel $\mathcal{A} = \langle Q, \Sigma, \Gamma, \delta, \gamma, q_0 \rangle$, wobei

- Q, Σ, δ, q_0 ... siehe DEAs
- Γ ... Ausgabealphabet (*output alphabet*)
- $\gamma\colon Q \times \Sigma \mapsto \Gamma$ ... Ausgabefunktion (*output function*)

Erweiterte Übergangsfunktion $\delta^*\colon Q \times \Sigma^* \mapsto Q$ siehe DEA.

Erweiterte Ausgabefunktion $\gamma^*\colon Q \times \Sigma^* \mapsto \Gamma^*$

$\gamma^*(q, \varepsilon) = \varepsilon$ für alle $q \in Q$, $s \in \Sigma$, $w \in \Sigma^*$
$\gamma^*(q, sw) = \gamma(q, s) \cdot \gamma^*(\delta(q, s), w)$

Übersetzungsfunktion $[\mathcal{A}]\colon \Sigma^* \mapsto \Gamma^*$

$[\mathcal{A}](w) = \gamma^*(q_0, w)$

Mealy-Automaten sind ein Spezialfall von Transducern:

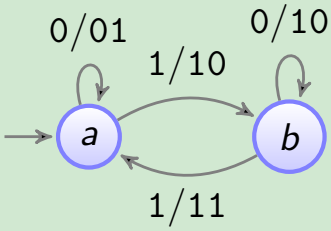
- Nur ein Anfangszustand: $I = \{q_0\}$
- Die Übergangsrelation ist deterministisch:
  - Der Folgezustand $\delta(q, s)$ ist eindeutig durch q und s bestimmt.
  - Keine ε-Übergänge
- Relationstupel: $(q, s, \gamma(q, s), \delta(q, s))$
- Alle Zustände sind Endzustände: $F = Q$

$(1:2)$-$(0,1)$-RLL-Encoder als Mealy-Automat

$\mathcal{A} = \langle \{a, b\}, \{0, 1\}, \{01, 10, 11\}, \delta, \gamma, a \rangle$

δ	0	1
a	a	b
b	b	a

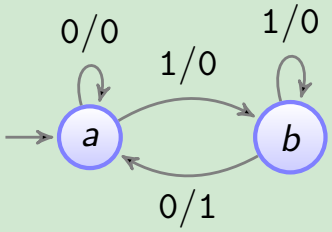
γ	0	1
a	01	10
b	10	11



w:	ε	0	1	00	10	01	11	000	100	...
$[\mathcal{A}](w)$:	ε	01	10	0101	1010	0110	1011	010101	101010	...

Detektor für fallende Flanken

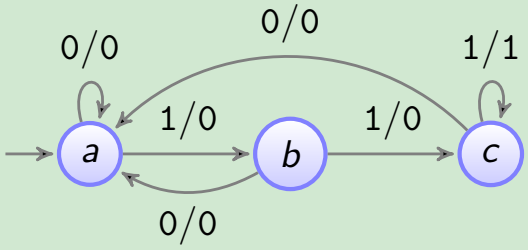
Ausgabe 1, wenn in der Eingabe ein Wechsel von 1 auf 0 stattfindet.



w:	$00110001010\cdots$
$[\mathcal{A}](w)$:	$00001000101 0\cdots$

Detektor für 111-Blöcke

Ausgabe 1, wenn in der Eingabe drei 1er aufeinander folgen.



w:	$0011011111010111 0\cdots$
$[\mathcal{A}](w)$:	$0000000111000001 0\cdots$

Was Sie heute erwartet

Moore-Automat

... wird beschrieben durch ein 6-Tupel $\mathcal{A} = \langle Q, \Sigma, \Gamma, \delta, \gamma, q_0 \rangle$, wobei

- Q, Σ, δ, q_0 ... siehe DEAs
- Γ ... Ausgabealphabet (*output alphabet*)
- $\gamma\colon Q \mapsto \Gamma$ ... Ausgabefunktion (*output function*)

(Mealy: $\gamma\colon Q \times \Sigma \mapsto \Gamma$)

Erweiterte Übergangsfunktion $\delta^*\colon Q \times \Sigma^* \mapsto Q$ siehe DEA.

Erweiterte Ausgabefunktion $\gamma^*\colon Q \times \Sigma^* \mapsto \Gamma^*$

$\gamma^*(q, \varepsilon) = \gamma(q)$ für alle $q \in Q$, $s \in \Sigma$, $w \in \Sigma^*$
$\gamma^*(q, sw) = \gamma(q) \cdot \gamma^*(\delta(q, s), w)$

(Mealy: $\gamma^*(q, sw) = \gamma(q, s) \cdot \gamma^*(\delta(q, s), w)$)

Übersetzungsfunktion $[\mathcal{A}]\colon \Sigma^* \mapsto \Gamma^*$

$[\mathcal{A}](w) = \gamma^*(q_0, w)$

Moore-Automaten sind (fast) ein Spezialfall von Transducern:

- Nur ein Anfangszustand: $I = \{q_0\}$
- Alle Übergange in einen Zustand geben dasselbe Symbol aus.
- Die Übergangsrelation ist deterministisch:
  - Der Folgezustand $\delta(q, s)$ ist eindeutig durch q und s bestimmt.
  - Keine ε-Übergänge
- Relationstupel: $(q, s, \gamma(q), \delta(q, s))$
- Alle Zustände sind Endzustände: $F = Q$

Vergleich von Moore- und Mealy-Automaten

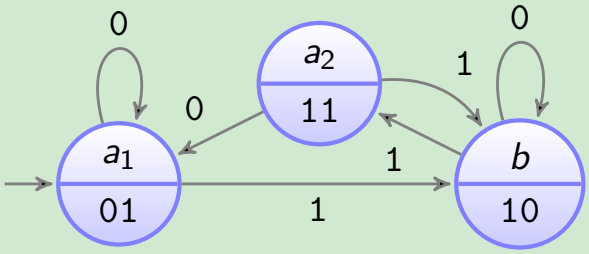
- Die Ausgabe erfolgt
  - ... bei Moore-Automaten durch den momentanen Zustand.
  - ... bei Mealy-Automaten beim Zustandswechsel, der durch Ursprungszustand und Eingabe festgelegt ist.
- Moore- und Mealy-Automaten besitzen dieselbe Ausdrucksstärke, sind aber schwächer als Transducer.
- Moore-Automaten haben in der Regel mehr Zustände als Mealy-Automaten.

(1 : 2)-(0, 1)-RLL-Encoder als Moore-Automat

$\mathcal{A} = \langle \{a_1, a_2, b\}, \{0, 1\}, \{01, 10, 11\}, \delta, \gamma, a \rangle$

δ	0	1
a_1	a_1	b
a_2	a_1	b
b	b	a_2

γ	
a_1	01
a_2	11
b	10

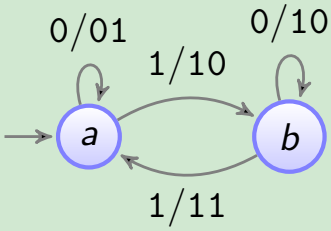


w:	ε	0	1	00	10	01	$\cdots$	100	$\cdots$
$[\mathcal{A}](w)$:	01	0101	0110	010101	011010	0110	$\cdots$	01101010	$\cdots$

Zum Vergleich: Mealy-Automat

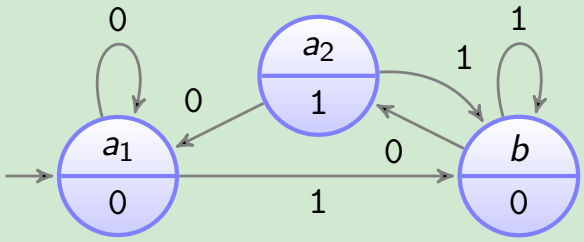
δ	0	1
a	a	b
b	b	a

γ	0	1
a	01	10
b	10	11



Detektor für fallende Flanken

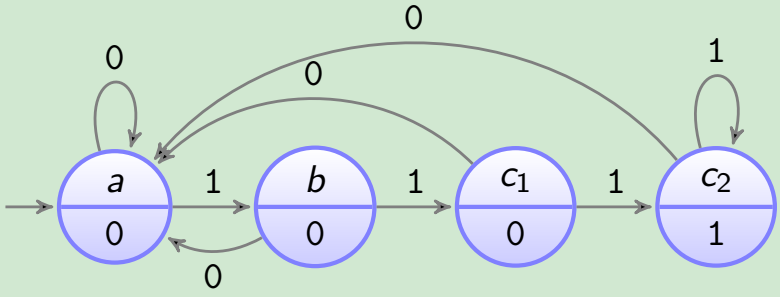
Ausgabe 1, wenn in der Eingabe ein Wechsel von 1 auf 0 stattfindet.



w:	00110001010 $\cdots$
$[\mathcal{A}](w)$:	000001000101 $\cdots$

Detektor für 111-Blöcke

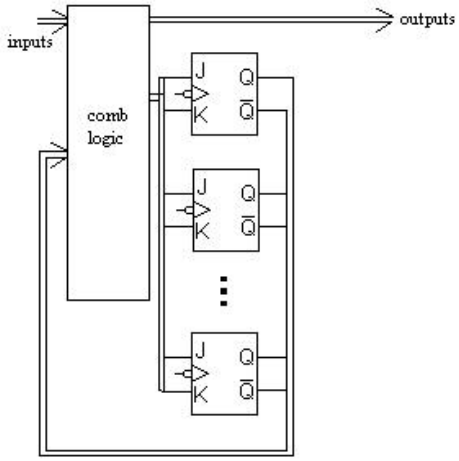
Ausgabe 1, wenn in der Eingabe drei 1er aufeinander folgen.



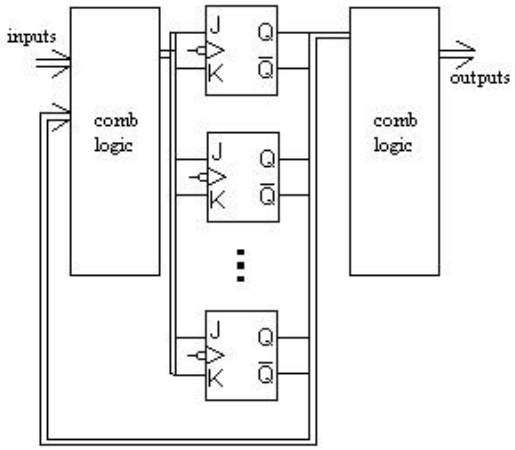
w:	00110111110101110 $\cdots$
$[\mathcal{A}](w)$:	000000001110000010 $\cdots$

Relevanz von Mealy- und Moore-Automaten

Mealy-Schaltwerk



Moore-Schaltwerk



- „inputs“ stammen aus dem Eingabealphabet Σ
- „outputs“ stammen aus dem Ausgabealphabet Γ
- Flip-Flops speichern den Zustand aus Q; Reset: Anfangszustand q_0
- „combination logic“: realisiert die Übergangsfunktionen δ und γ.

Graphiken: Ken Reid, Indiana University – Purdue University Indianapolis

Was Sie heute erwartet

Büchi-Automaten

Deterministischer Büchi-Automat

... wird beschrieben durch ein 5-Tupel $\mathcal{A} = \langle Q, \Sigma, \delta, q_0, F \rangle$, wobei

- Q, Σ, δ, q_0, F wie bei DEAs definiert sind.

Σ^ω ... Menge aller unendlichen Wörter ($= \omega$-Wörter) über Σ

Akzeptanz von Wörtern

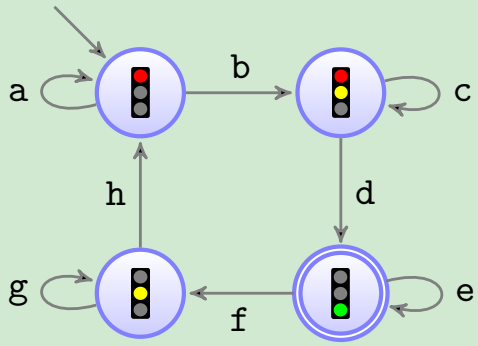
Ein deterministischer Büchi-Automat $\mathcal{A}$ akzeptiert ein Wort $s_1 s_2 s_3 \cdots \in \Sigma^\omega$, wenn es Zustände $q_0, q_1, q_2, q_3, \cdots \in Q$ gibt, sodass

- $q_0 \in Q$ der Startzustand ist,
- $\delta(q_{i-1}, s_i) = q_i$ für alle $i \in \mathbb{N}$ gilt und
- es unendlich viele i gibt, sodass q_i ein Endzustand ist ($q_i \in F$).

Akzeptierte/Generierte Sprache

$\mathcal{L}(\mathcal{A}) = \{ w \in \Sigma^\omega \mid w \text{ wird von } \mathcal{A} \text{ akzeptiert} \}$

Faire Verkehrsampel



Der Automat akzeptiert genau jene Wörter aus $\{\mathtt{a}, \ldots, \mathtt{h}\}^\omega$, bei denen es immer wieder grün wird.

Nichtdeterministischer Büchi-Automat

... wird beschrieben durch ein 5-Tupel $\mathcal{A} = \langle Q, \Sigma, \delta, I, F \rangle$, wobei

- Q, Σ, F ... siehe DEA
- $I \subseteq Q$ ... Anfangszustände
- $\delta \subseteq Q \times \Sigma \times Q$ ... Übergangsrelation

Akzeptanz von Wörtern

Ein nichtdeterministischer Büchi-Automat $\mathcal{A}$ akzeptiert ein Wort $s_1 s_2 s_3 \cdots \in \Sigma^\omega$, wenn es Zustände $q_0, q_1, q_2, q_3, \cdots \in Q$ gibt, sodass

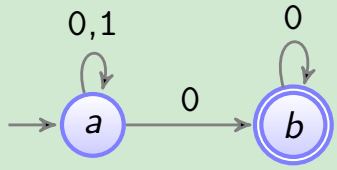
- $q_0 \in Q$ ein Startzustand ist ($q_0 \in I$),
- $(q_{i-1}, s_i, q_i) \in \delta$ für alle $i \in \mathbb{N}$ gilt und
- es unendlich viele i gibt, sodass q_i ein Endzustand ist ($q_i \in F$).

Akzeptierte/Generierte Sprache

$\mathcal{L}(\mathcal{A}) = \{ w \in \Sigma^\omega \mid w \text{ wird von } \mathcal{A} \text{ akzeptiert} \}$

Nur endlich viele 1er

Gesucht: Ein Büchi-Automat, der genau jene ω-Wörter über $\{0, 1\}$ akzeptiert, die nur eine endliche Anzahl an 1ern enthalten.



Kein deterministischer Büchi-Automaten akzeptiert diese Sprache.
$\Longrightarrow$ Nichtdeterministische Büchi-Automaten sind ausdrucksstärker als deterministische!

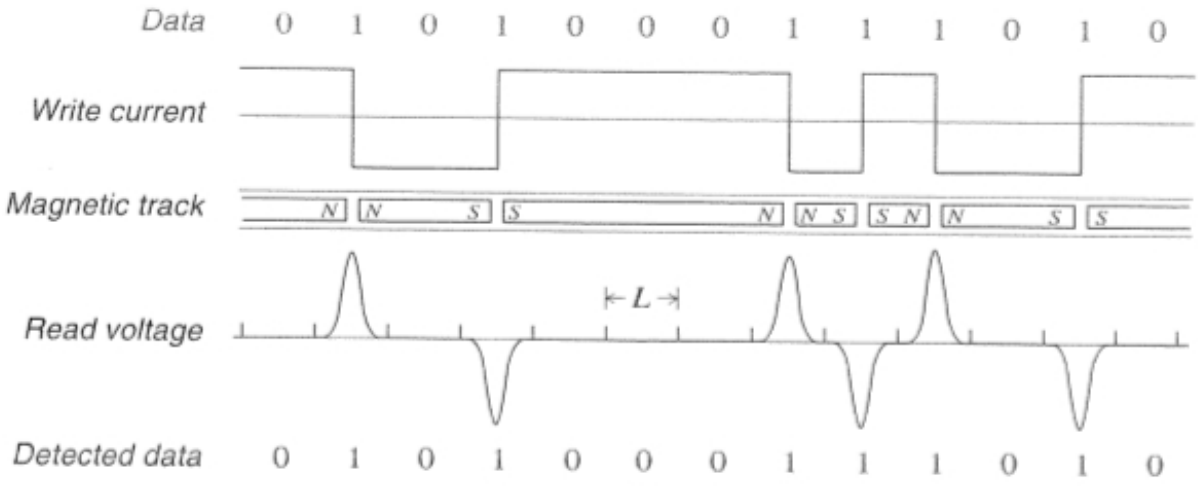
Modellierung mit endlichen Automaten

Vorgangsweise:

1. Was sind die Zustände des Systems? Wieviele sind notwendig? Zustandsbezeichnungen?
2. Startzustand? Endzustände?
3. Was sind die Aktionen/Eingaben, die zu Zustandsübergängen führen? Bezeichnung?
4. Was sind die Aktionen/Ausgaben, die bei Zustandsübergängen stattfinden? Bezeichnung?
5. Lege für jeden Zustand und jede Eingabe die Folgezustände und die Ausgaben fest.

Anwendung: RLL (run-length limited) Codes

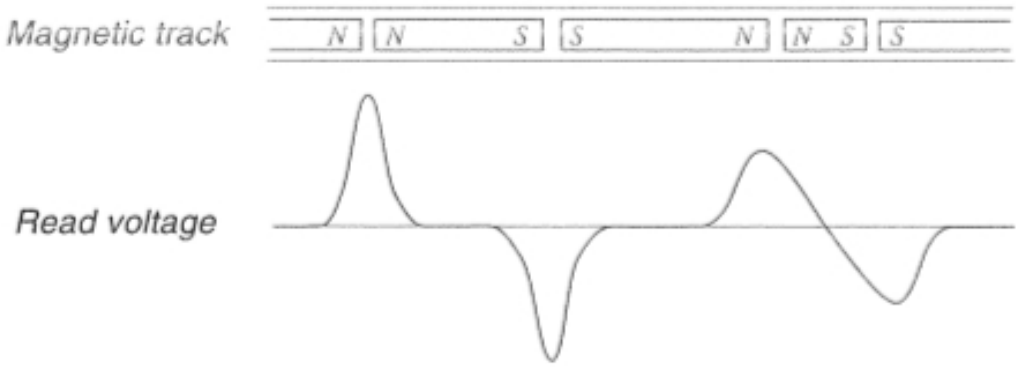
- Daten werden auf Festplatten, CDs und DVDs binär kodiert.
- Binär-1: Magnetisierungswechsel bzw. Vertiefung („pit“)



Graphik aus: D.Lind, B.Marcus: An Introduction to Symbolic Dynamics and Coding, Cambridge Unversity Press 1995.

Probleme:

- 1er zu weit auseinander: Synchronisation geht verloren.
- 1er zu dicht: werden ununterscheidbar bzw. schwächen sich ab.



Graphik aus: D.Lind, B.Marcus: An Introduction to Symbolic Dynamics and Coding, Cambridge Unversity Press 1995.

Einfache Lösung:

- Füge nach jedem Datenbit ein zusätzliches Synchronisationsbit ein.
- Wähle Bitabstand so groß, dass 1er-Folgen richtig erkannt werden.

$(m : n)$-(d, k)-RLL-Codes

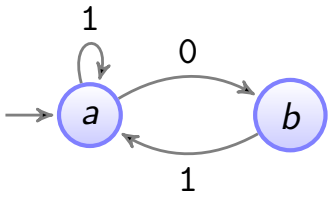
Nachteil der einfachen Lösung: Benötigt mehr Platz als notwendig.

Idee: Nütze die „natürlichen“ 0er-Strecken und 1er im Datenstrom.
m Datenbits werden so in n Codebits übersetzt, dass zwischen zwei 1ern mindestens d und höchstens k 0er auftreten.

- Existiert nur für bestimmte Werte von m, n, d, k
- Ziel: minimiere n/m, maximiere d

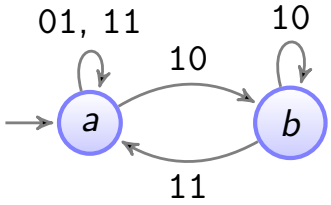
DVD: $(8 : 16)$-$(2, 10)$-RLL-Code

(1 : 2)-(0, 1)-RLL-Code: 2 Codebits/Datenbit, max. ein 0er zwischen 1ern.



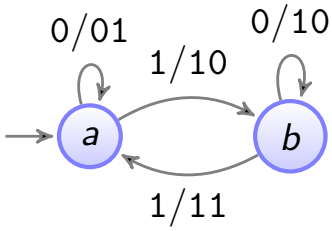
Zustand *a*: zwei Codeworte für zwei Datenbits – reicht.
Zustand *b*: ein Codewort für zwei Datenbits – reicht nicht.

„Potenzieren“ des Automaten:

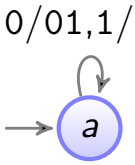


Zustand *a*: drei Codeworte für zwei Datenbits – reicht.
Zustand *b*: zwei Codeworte für zwei Datenbits – reicht.

Codierungsautomat (Mealy-Automat):



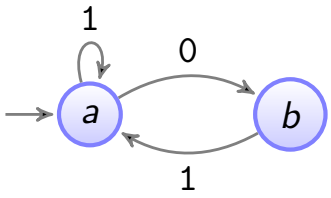
oder



oder . . .

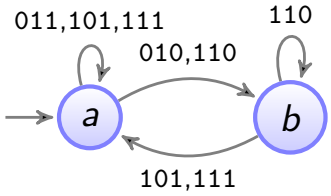
10010 ⇒ 10 10 10 11 01 oder 11 01 01 11 01 oder . . .

(2 : 3)-(0, 1)-RLL-Code: 3 Code-/2 Datenbits, max. ein 0er zwischen 1ern.



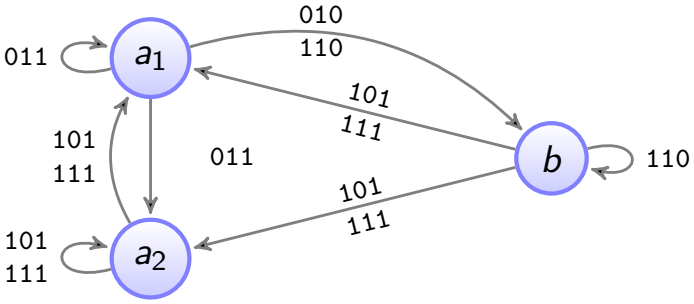
Zustand *a*: zwei Code- für vier Datenworte – reicht nicht.
Zustand *b*: ein Codewort für vier Datenworte – reicht nicht.

Dritte Potenz des Automaten:

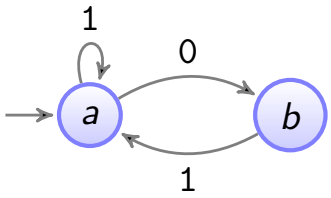


Zustand *a*: fünf Code- für vier Datenworte – reicht.
Zustand *b*: drei Code- für vier Datenworte – reicht nicht.

Teilung von Zustand *a*:

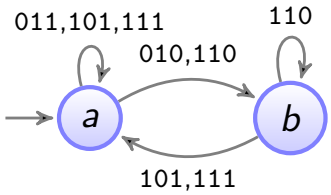


(2 : 3)-(0, 1)-RLL-Code: 3 Code-/2 Datenbits, max. ein 0er zwischen 1ern.



Zustand *a*: zwei Code- für vier Datenworte – reicht nicht.
Zustand *b*: ein Codewort für vier Datenworte – reicht nicht.

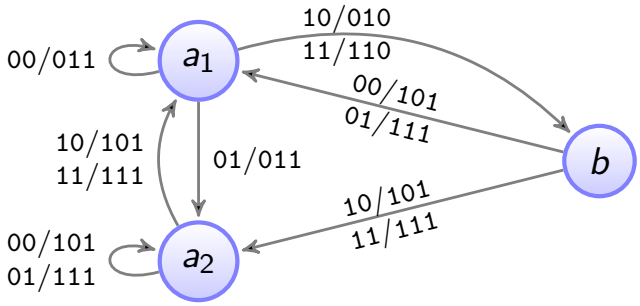
Dritte Potenz des Automaten:



Zustand *a*: fünf Code- für vier Datenworte – reicht.
Zustand *b*: drei Code- für vier Datenworte – reicht nicht.

Teilung von Zustand *a*:

Mealy-Automat



Was Sie heute erwartet

Operationen auf formalen Sprachen

Σ Alphabet, d.h., endliche, nicht-leere Menge atomarer Symbole
w Wort über Σ, (endliche) Folge von Zeichen aus dem Alphabet Σ
ε Leerwort
Σ^* Menge aller endlichen Wörter über Σ (inklusive Leerwort)
$w \cdot w' = ww'$ Verkettung der Wörter $w, w' \in \Sigma^*$

Seien $L, L' \subseteq \Sigma^*$ zwei Sprachen.

$$L \cup L' = \{ w \mid w \in L \text{ oder } w \in L' \} \qquad \text{Vereinigung}$$

$$L \cdot L' = \{ w \cdot w' \mid w \in L, w' \in L' \} \qquad \text{Verkettung}$$

$$L^0 = \{\varepsilon\} \qquad \text{Potenzen}$$
$$L^{n+1} = L \cdot L^n \quad (n \geq 0)$$

$$L^+ = \bigcup_{n \geq 1} L^n$$
$$L^* = \bigcup_{n \geq 0} L^n = L^0 \cup L^+ = \{\varepsilon\} \cup L^+ \qquad \text{Kleene-Stern}$$

Verkettung

$\{\mathtt{a}, \mathtt{b}\} \cdot \{\mathtt{b}, \mathtt{c}, \mathtt{d}\} = \{\mathtt{ab}, \mathtt{ac}, \mathtt{ad}, \mathtt{bb}, \mathtt{bc}, \mathtt{bd}\}$
$\{\mathtt{b}, \mathtt{c}, \mathtt{d}\} \cdot \{\mathtt{a}, \mathtt{b}\} = \{\mathtt{ba}, \mathtt{bb}, \mathtt{ca}, \mathtt{cb}, \mathtt{da}, \mathtt{db}\}$

$(\{\mathtt{a}, \mathtt{b}\} \cdot \{1, 2\}) \cdot \{\#, \$\} = \{\mathtt{a1\#}, \mathtt{a1\$}, \mathtt{a2\#}, \mathtt{a2\$}, \mathtt{b1\#}, \mathtt{b1\$}, \mathtt{b2\#}, \mathtt{b2\$}\}$
$= \{\mathtt{a}, \mathtt{b}\} \cdot (\{1, 2\} \cdot \{\#, \$\})$

$\{\mathtt{a}, \mathtt{b}\} \cdot \{\varepsilon\} = \{\mathtt{a} \cdot \varepsilon, \mathtt{b} \cdot \varepsilon\} = \{\mathtt{a}, \mathtt{b}\} = \{\varepsilon \cdot \mathtt{a}, \varepsilon \cdot \mathtt{b}\} = \{\varepsilon\} \cdot \{\mathtt{a}, \mathtt{b}\}$

$\{\mathtt{a}, \mathtt{b}\} \cdot \{\} = \{\} \cdot \{\mathtt{a}, \mathtt{b}\} = \{\}$

$\{\varepsilon\} \cdot \{\varepsilon\} = \{\varepsilon\}$
$\{\} \cdot \{\} = \{\varepsilon\} \cdot \{\} = \{\} \cdot \{\varepsilon\} = \{\}$

Beobachtungen:

- Sprachverkettung ist nicht kommutativ.
- Sprachverkettung ist assoziativ.
- $\{\varepsilon\}$ ist neutrales Element bzgl. Sprachverkettung.
- $\{\}$ ist Nullelement bzgl. Sprachverkettung.

Potenzen von $\{a, 42\}$

$$
\begin{aligned}
L &= \{a, 42\} \\
L^0 &= \{\varepsilon\} \\
L^1 &= L \cdot L^0 = L \cdot \{\varepsilon\} = L = \{a, 42\} \\
L^2 &= L \cdot L^1 = L \cdot L = \{aa, a42, 42a, 4242\} \\
L^3 &= L \cdot L^2 = \{aaa, aa42, a42a, a4242, 42aa, 42a42, 4242a, 424242\} \\
&\vdots \\
L^+ &= \bigcup_{n\geq 1} L^n = L^1 \cup L^2 \cup L^3 \cup \cdots \\
&= \{a, 42, aa, a42, 42a, 4242, aaa, aa42, a42a, a4242, 42aa, \dots\} \\
L^* &= \bigcup_{n\geq 0} L^n = L^0 \cup L^1 \cup L^2 \cup L^3 \cup \cdots \\
&= \{\varepsilon, a, 42, aa, a42, 42a, 4242, aaa, aa42, a42a, a4242, 42aa, \dots\}
\end{aligned}
$$

Potenzen eines Alphabets Σ

$\Sigma^0 = \{\varepsilon\}$	$\Sigma^1 = \Sigma$
Σ^n	alle Σ-Wörter der Länge n (d.h., mit n Symbolen)
$\Sigma^+ = \bigcup_{n\geq 1} \Sigma^n$	alle Σ-Wörter ohne Leerwort
$\Sigma^* = \bigcup_{n\geq 0} \Sigma^n$	alle Σ-Wörter mit Leerwort

$\langle 2^{\Sigma^*}, \cup, \cdot, \{\}, \{\varepsilon\}\rangle$ bildet einen idempotenten Halbring

Das heißt, es gelten folgende Gleichungen.

$\langle 2^{\Sigma^*}, \cup, \{\}\rangle$ ... idemp.komm.Monoid

$(A \cup B) \cup C = A \cup (B \cup C)$
$\{\} \cup A = A \cup \{\} = A$
$A \cup B = B \cup A$
$A \cup A = A$

$\langle 2^{\Sigma^*}, \cdot, \{\varepsilon\}\rangle$ ... Monoid

$(A \cdot B) \cdot C = A \cdot (B \cdot C)$
$\{\varepsilon\} \cdot A = A \cdot \{\varepsilon\} = A$

Verkettung distribuiert über Vereinigung.

$A \cdot (B \cup C) = (A \cdot B) \cup (A \cdot C) \qquad (B \cup C) \cdot A = (B \cdot A) \cup (C \cdot A)$

$\{\}$ ist Nullelement bzgl. Verkettung.

$\{\} \cdot A = A \cdot \{\} = \{\}$

Weitere Identitäten für $^+$ und *:

$(A^*)^* = A^* \qquad (A \cup \{\varepsilon\})^* = A^* \qquad A^* \cdot A = A^+ \qquad A^+ \cup \{\varepsilon\} = A^*$

3.0 VU Formale Modellierung

Gernot Salzer

Forschungsbereich Theory and Logic
Institut für Logic and Computation

24.4.2018

Was Sie letztes Mal hörten

Endliche Automaten

Bezeichnungen

Q	endliche Menge von Zuständen
q_0	Anfangszustand
I	Menge von Anfangszuständen
F	Menge von Endzuständen
Σ	Eingabealphabet
Γ	Ausgabealphabet
δ	Übergangsfunktion/-relation
δ^*	erweiterte Übergangsfunktion/-relation
γ	Ausgabefunktion
γ^*	erweiterte Ausgabefunktion
$\mathcal{A}$	Automat
$\mathcal{L}(\mathcal{A})$	die von $\mathcal{A}$ akzeptierte Sprache
$[\mathcal{A}]$	die von $\mathcal{A}$ berechnete Übersetzungsfunktion/-relation

Übergangsfunktionen und -relationen

$\delta\colon Q \times \Sigma \mapsto Q$	det. endlicher Automat (DEA) det. Büchi-Automat
$\delta \subseteq Q \times \Sigma \times Q$ $\delta\colon Q \times \Sigma \mapsto 2^Q$	nichtdet. endl. Aut. (NEA) ohne ε-Ü. nichtdet. Büchi-Automat
$\delta \subseteq Q \times (\Sigma \cup \{\varepsilon\}) \times Q$ $\delta\colon Q \times (\Sigma \cup \{\varepsilon\}) \mapsto 2^Q$	nichtdet. endl. Aut. (NEA) mit ε-Ü.
$\delta \subseteq Q \times (\Sigma \cup \{\varepsilon\}) \times (\Gamma \cup \{\varepsilon\}) \times Q$ $\delta\colon Q \times (\Sigma \cup \{\varepsilon\}) \mapsto 2^{(\Gamma \cup \{\varepsilon\}) \times Q}$	Transducer (indet. mit ε-Ü.)
$\delta\colon Q \times \Sigma \mapsto \Gamma \times Q$ $\delta\colon Q \times \Sigma \mapsto Q \quad \gamma\colon Q \times \Sigma \mapsto \Gamma$	Mealy-Automat
$\delta\colon Q \times \Sigma \mapsto Q \quad \gamma\colon Q \mapsto \Gamma$	Moore-Automat

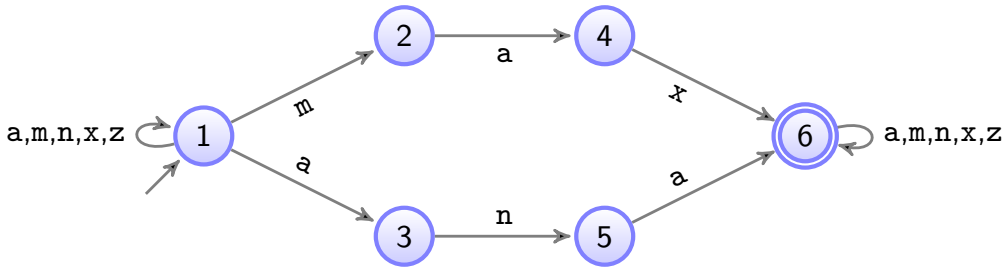
ε-Übergänge, Relation oder Potenzmenge $\Longrightarrow$ nichtdeterm. Automat

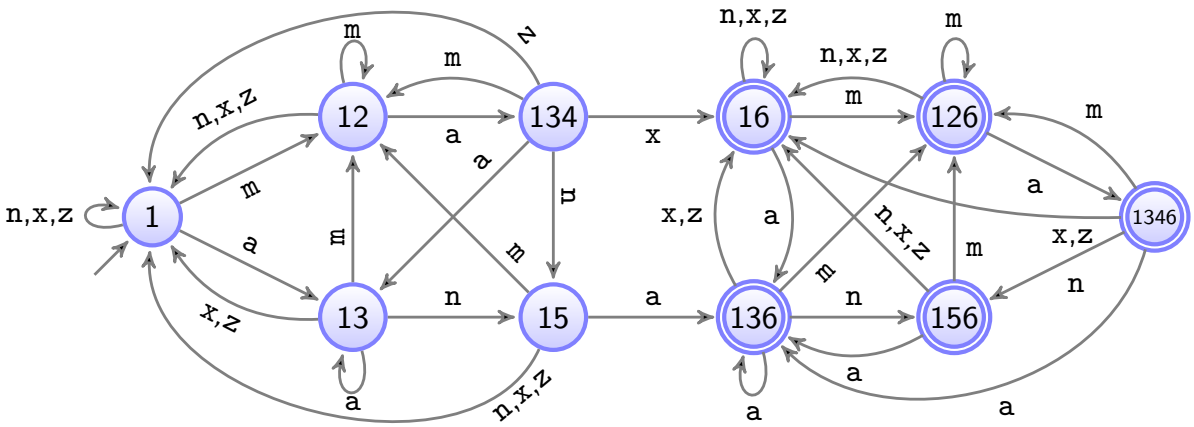
Akzeptierte Wörter bzw. Wortpaare

- (Nicht-)Deterministischer Automat:
  Alle Wörter, mit denen man vom Anfangszustand aus einen der Endzustände erreicht.
- Büchi-Automat:
  Alle unendlichen Wörter, mit denen man vom Anfangszustand aus unendlich oft an einem Endzustand vorbei kommt.
- Transducer:
  Alle Paare (u, v) von Ein-/Ausgabewörtern, bei denen man mit u vom Anfangszustand (von einem der Anfangszustände) aus in einen Endzustand gelangt und dabei v ausgibt.
  - Mealy: Pro Übergang wird genau ein Symbol gelesen und eines geschrieben. Die Ausgabe ist an den Übergang gebunden.
  - Moore: Pro Übergang wird genau ein Symbol gelesen und eines geschrieben. Die Ausgabe ist an den Zielzustand gebunden.

Was Sie letztes Mal hörten

Beispiel: Suche nach Max und Ana





Modellierung mit endlichen Automaten

Vorgangsweise:

1. Was sind die Zustände des Systems? Wieviele sind notwendig? Zustandsbezeichnungen?
2. Startzustand? Endzustände?
3. Was sind die Aktionen/Eingaben, die zu Zustandsübergängen führen? Bezeichnung?
4. Was sind die Aktionen/Ausgaben, die bei Zustandsübergängen stattfinden? Bezeichnung?
5. Lege für jeden Zustand und jede Eingabe die Folgezustände und die Ausgaben fest.

Was Sie letztes Mal hörten

Operationen auf formalen Sprachen

Σ	Alphabet, d.h., endliche, nicht-leere Menge atomarer Symbole
w	Wort über Σ, (endliche) Folge von Zeichen aus dem Alphabet Σ
ε	Leerwort
Σ^*	Menge aller endlichen Wörter über Σ (inklusive Leerwort)
$w \cdot w' = ww'$	Verkettung der Wörter $w, w' \in \Sigma^*$

Seien $L, L' \subseteq \Sigma^*$ zwei Sprachen.

$$L \cup L' = \{ w \mid w \in L \text{ oder } w \in L' \} \qquad \text{Vereinigung}$$

$$L \cdot L' = \{ w \cdot w' \mid w \in L, w' \in L' \} \qquad \text{Verkettung}$$

$$L^0 = \{\varepsilon\} \qquad \text{Potenzen}$$
$$L^{n+1} = L \cdot L^n \quad (n \geq 0)$$

$$L^+ = \bigcup_{n \geq 1} L^n$$

$$L^* = \bigcup_{n \geq 0} L^n = L^0 \cup L^+ = \{\varepsilon\} \cup L^+ \qquad \text{Kleene-Stern}$$

$\langle 2^{\Sigma^*}, \cup, \cdot, \{\}, \{\varepsilon\}\rangle$ bildet einen idempotenten Halbring

Das heißt, es gelten folgende Gleichungen.

$\langle 2^{\Sigma^*}, \cup, \{\}\rangle$ ... idemp.komm.Monoid

$(A \cup B) \cup C = A \cup (B \cup C)$
$\{\} \cup A = A \cup \{\} = A$
$A \cup B = B \cup A$
$A \cup A = A$

$\langle 2^{\Sigma^*}, \cdot, \{\varepsilon\}\rangle$ ... Monoid

$(A \cdot B) \cdot C = A \cdot (B \cdot C)$
$\{\varepsilon\} \cdot A = A \cdot \{\varepsilon\} = A$

Verkettung distribuiert über Vereinigung.

$A \cdot (B \cup C) = (A \cdot B) \cup (A \cdot C) \qquad (B \cup C) \cdot A = (B \cdot A) \cup (C \cdot A)$

$\{\}$ ist Nullelement bzgl. Verkettung.

$\{\} \cdot A = A \cdot \{\} = \{\}$

Weitere Identitäten für $^+$ und *:

$(A^*)^* = A^* \qquad (A \cup \{\varepsilon\})^* = A^* \qquad A^* \cdot A = A^+ \qquad A^+ \cup \{\varepsilon\} = A^*$

Was Sie heute erwartet

Reguläre Sprachen

Alle Sprachen, die aus einem Alphabet mit Hilfe von Vereinigung, Verkettung und Stern gebildet werden können.

Anwendungen:

- Betriebssytem-Shells: DOS („Wildcards“), UNIX-Shells (wie `sh`, `csh`, `ash`, `bash`, `zsh`), ...
- UNIX Kommandozeilenprogramme: `grep`, `awk`, `ed`, `sed`, ...
- Editoren: `vi`, `emacs`, ...
- Compilerbau: *Tokens* bilden reguläre Sprache, die durch sog. Scanner (Lexer) wie `lex` oder `flex` verarbeitet werden.
- Programmiersprachen: PERL, TCL, PHP, PYTHON, RUBY, R, JAVA, JAVASCRIPT, .NET-Sprachen, ...
- Websprachen: XML Schema, XQuery, XPath, DTDs, ...
- Datenbanken: MySQL, Oracle, PostgreSQL, ...
- ...

Regulären Sprachen über einem Alphabet

Die Menge der regulären Sprachen über Σ, $\mathcal{L}_{\text{reg}}(\Sigma)$, ist die kleinste Menge, sodass gilt:

- $\{\}$, $\{\varepsilon\}$ und $\{s\}$ sind reguläre Sprachen (für alle $s \in \Sigma$).
- Wenn L und L' reguläre Sprachen sind, dann auch $L \cup L'$, $L \cdot L'$ und L^*.

Reellen Numerale: reguläre Sprache über $\Sigma = \{0, \ldots, 9, ., \mathtt{E}, +, -\}$

$$\begin{aligned} \mathit{real} &= \mathit{digit} \cdot \mathit{digit}^* \cdot \{.\} \cdot \mathit{digit}^* \cdot (\{\varepsilon\} \cup \mathit{scale}) \\ \mathit{scale} &= \{\mathtt{E}\} \cdot \{+, -, \varepsilon\} \cdot \mathit{digit} \cdot \mathit{digit}^* \\ \mathit{digit} &= \{0, \ldots, 9\} = \{0\} \cup \cdots \cup \{9\} \end{aligned}$$

Wichtig: Unterscheide Symbole des Alphabets von Meta-Symbolen!

$0, \ldots, 9, ., \mathtt{E}, +, -$ ... Symbole des Alphabets
ε, *real*, *scale*, *digit* ... Meta-Symbole, Abkürzungen

Was Sie heute erwartet

Reguläre Ausdrücke

Ausdrücke wie $\mathit{digit} \cdot \mathit{digit}^*$ und digit^+ sind ununterscheidbar: Beides sind semantische Beschreibungen der Menge aller Ziffernfolgen. Um Aussagen über ihre Form treffen zu können, benötigen wir eine formale Sprache.

Reguläre Ausdrücke (algebraische Notation)

Die regulären Ausdrücke über Σ sind die kleinste Menge, für die gilt:

- $\emptyset$, ε und s sind reguläre Ausdrücke (für alle Symbole $s \in \Sigma$).
- Sind r und r' reguläre Ausdrücke, dann auch $(r + r')$, (rr') und r^*.

Vereinfachte Klammerung: $+$ bindet am schwächsten, $*$ am stärksten. Keine Klammern bei gleichartigen Operatoren (wegen Assoziativität).

Die Sprache $\mathcal{L}(r)$ zu einem regulären Ausdruck r ist definiert durch:

$$\begin{aligned} \mathcal{L}(\emptyset) &= \{\} & \mathcal{L}(r + r') &= \mathcal{L}(r) \cup \mathcal{L}(r') \\ \mathcal{L}(\varepsilon) &= \{\varepsilon\} & \mathcal{L}(rr') &= \mathcal{L}(r) \cdot \mathcal{L}(r') \\ \mathcal{L}(s) &= \{s\} \quad \text{für } s \in \Sigma & \mathcal{L}(r^*) &= (\mathcal{L}(r))^* \end{aligned}$$

Regulärer Ausdruck für die reellen Numerale

$R = DD^* \,.\, D^*(\varepsilon + S)$
$S = \mathtt{E}(\mathtt{+} + \mathtt{-} + \varepsilon)DD^*$
$D = \mathtt{0} + \mathtt{1} + \cdots + \mathtt{9}$

(R, S und D sind Abkürzungen für die jeweiligen regulären Ausdrücke.)

Die zugehörigen Sprachen:

$$\begin{aligned}\mathcal{L}(D) &= \mathcal{L}(\mathtt{0} + \mathtt{1} + \cdots + \mathtt{9}) \\ &= \mathcal{L}(\mathtt{0}) \cup \mathcal{L}(\mathtt{1}) \cup \cdots \cup \mathcal{L}(\mathtt{9}) \\ &= \{\mathtt{0}\} \cup \{\mathtt{1}\} \cup \cdots \cup \{\mathtt{9}\} \\ &= \mathit{digits}\end{aligned}$$

$$\begin{aligned}\mathcal{L}(S) &= \mathcal{L}(\mathtt{E}(\mathtt{+} + \mathtt{-} + \varepsilon)DD^*) \\ &= \mathcal{L}(\mathtt{E}) \cdot \mathcal{L}(\mathtt{+} + \mathtt{-} + \varepsilon) \cdot \mathcal{L}(D) \cdot \mathcal{L}(D^*) \\ &= \{\mathtt{E}\} \cdot (\mathcal{L}(\mathtt{+}) \cup \mathcal{L}(\mathtt{-}) \cup \mathcal{L}(\varepsilon)) \cdot \mathit{digits} \cdot \mathcal{L}(D)^* \\ &= \{\mathtt{E}\} \cdot (\{\mathtt{+}\} \cup \{\mathtt{-}\} \cup \{\varepsilon\}) \cdot \mathit{digits} \cdot \mathit{digits}^* \\ &= \mathit{scale}\end{aligned}$$

$$\mathcal{L}(R) = \cdots = \mathit{real}$$

Zwei reguläre Ausdrücke r und r' heißen äquivalent, geschrieben $r = r'$, wenn $\mathcal{L}(r) = \mathcal{L}(r')$ gilt.

$((\mathtt{a} + \mathtt{b})^* + \varepsilon)^* = (\mathtt{a} + \mathtt{b})^*$

$$\begin{aligned}\mathcal{L}(((\mathtt{a} + \mathtt{b})^* + \varepsilon)^*) &= \cdots \\ &= (\{\mathtt{a}, \mathtt{b}\}^* \cup \{\varepsilon\})^* \\ &= (\{\mathtt{a}, \mathtt{b}\}^*)^* \qquad \text{da } \varepsilon \in L^* \text{ für alle } L \\ &= (\{\mathtt{a}, \mathtt{b}\}^*)^0 \cup (\{\mathtt{a}, \mathtt{b}\}^*)^1 \cup (\{\mathtt{a}, \mathtt{b}\}^*)^2 \cup \cdots \\ &= \{\mathtt{a}, \mathtt{b}\}^* \cup (\{\mathtt{a}, \mathtt{b}\}^*)^0 \cup (\{\mathtt{a}, \mathtt{b}\}^*)^2 \cup \cdots \\ &= \{\mathtt{a}, \mathtt{b}\}^* \qquad \text{da } L^* \text{ alle Wörter über } L \text{ enthält} \\ &= \cdots \\ &= \mathcal{L}((\mathtt{a} + \mathtt{b})^*)\end{aligned}$$

Reguläre Ausdrücke in EBNF-Notation

EBNF ... Erweiterte Backus-Naur-Form (Formalismus zur Beschreibung der Syntax von Programmiersprachen, die reguläre Ausdrücke zulässt)

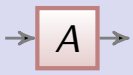
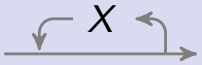
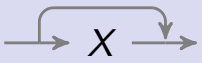
AB	$A \cdot B$	Aufeinanderfolge
$A \mid B$	$A \cup B$	Alternativen
$[A]$	$\{\varepsilon\} \cup A$	Option
$\{A\}$	A^*	Wiederholung
(A)	(A)	Gruppierung
`"`s`"`	$\{s\}$	Symbol

Reelle Numerale in EBNF-Notation

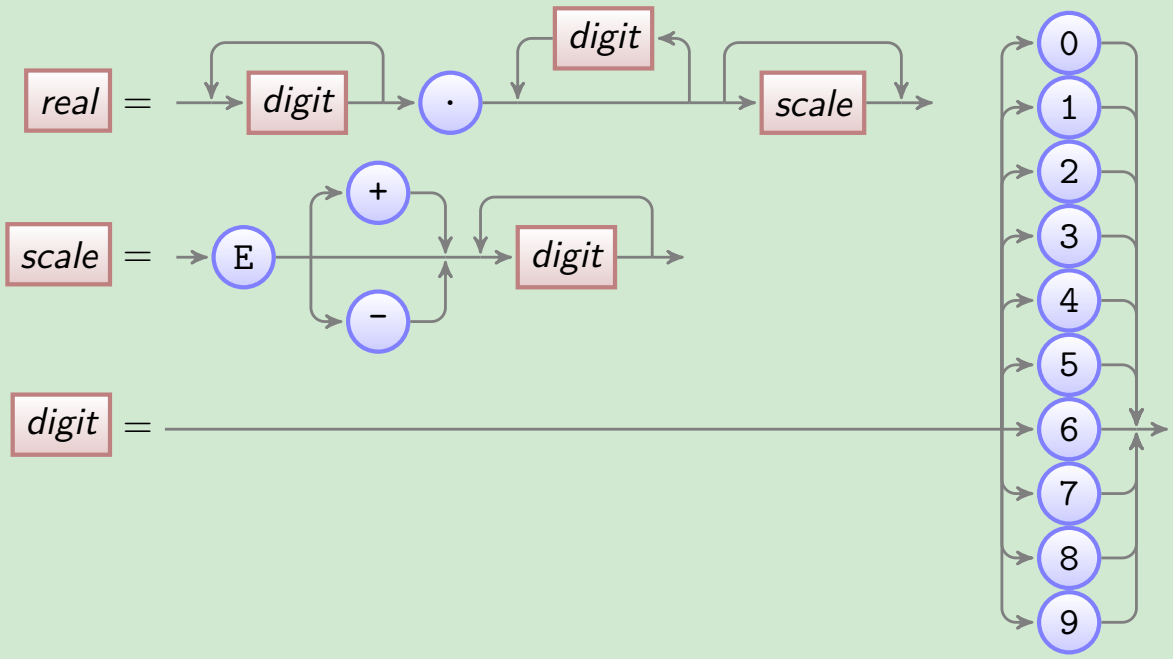
```
real  = digit {digit} "." {digit} [scale]
scale = "E" ["+" | "-"] digit {digit}
digit = "0" | "1" | "2" | ... | "9"
```


Reguläre Ausdrücke als Syntaxdiagramme

Syntaxdiagramme ... graphische Form der EBNF

	A	Abkürzung
	$\{s\}$	Symbol
	X^+	Wiederholung ≥ 1
	X^*	Wiederholung ≥ 0
	$X \cdot Y$	Aufeinanderfolge
	$X \cup Y$	Alternativen
	$\{\varepsilon\} \cup X$	Option

Reelle Numerale als Syntaxdiagramm



Reguläre Ausdrücke im informatischen Alltag

UNIX := *„30 definitions of regular expressions living under one roof“*

(Donald E. Knuth)

`grep -E -e "`*regexp*`"` *file*

liefert alle Zeilen der Datei *file*, die eine Zeichenkette enthalten, die dem regulären Ausdruck *regexp* (in POSIX ERE Syntax) entspricht.

Verschiedene „Standards“:

- POSIX Basic Regular Expressions
- POSIX Extended Regular Expressions
- PERL Regular Expressions
- ...

Posix Extended Regular Expressions (Ere)

regexp	trifft zu auf	*regexp*	trifft zu auf
$\backslash s$	Zeichen s	rr'	r gefolgt von r'
s	s, falls kein Sonderzeichen	$r \mid r'$	r oder r'
.	alle Zeichen	$r*$	≥ 0 Mal r
^	Zeilenanfang	$r+$	≥ 1 Mal r
$	Zeilenende	$r?$	≤ 1 Mal r
$[s_1 \cdots s_n]$	ein Zeichen aus $\{s_1, \ldots, s_n\}$	$r\{i\}$	i Mal r
$[\hat{}\, s_1 \cdots s_n]$	alle Zeichen außer $s_1, \ldots, s_n$	$r\{i,\}$	$\geq i$ Mal r
(r)	r	$r\{i,j\}$	i bis j Mal r

Reelle Numerale als Ere

$digit = \{0, \ldots, 9\}$ `[0-9]`

$scale = \{\mathtt{E}\} \cdot \{+, -, \varepsilon\} \cdot digit \cdot digit^*$ `E[+-]?[0-9]+`

$real = digit \cdot digit^* \cdot \{.\} \cdot digit^* \cdot (\{\varepsilon\} \cup scale)$

```
^[0-9]+\.[0-9]*(E[+-]?[0-9]+)?$
```

`[0-9]` ... Kurzform von `[0123456789]`; analog `[a-zA-Z]` für Buchstaben

Was Sie heute erwartet

Eigenschaften regulärer Sprachen

Reguläre Sprachen sind abgeschlossen gegenüber

- Vereinigung, Verkettung und Stern (warum?)
- Durchschnitt, Komplement, Differenz, Homomorphismen, Quotientenbildung, ...

Entscheidbarkeit eines Problems: Es gibt ein Verfahren (einen Algorithmus), der für jede Eingabe die richtige Antwort ja/nein liefert.

Nicht entscheidbar: Halteproblem Ihrer Lieblingsprogrammiersprache

- Gegeben ein Programm mit einer Eingabe, wird es anhalten?

Folgende Probleme regulärer Sprachen sind entscheidbar:

- Gegeben ein Wort w und einen regulären Ausdruck r, gilt $w \in \mathcal{L}(r)$? (Wortproblem)
- Gegeben zwei reguläre Ausdrücke r und r', gilt $\mathcal{L}(r) = \mathcal{L}(r')$? (Äquivalenzproblem)
- Gegeben einen regulären Ausdruck r, ist $\mathcal{L}(r)$ leer/endlich/unendlich?

Ausdruckskraft regulärer Sprachen

Die regulären Sprachen sind genau jene, die von endlichen Automaten akeptiert werden, d.h.:

- Zu jedem regulären Ausdruck r gibt es einen endlichen Automaten $\mathcal{A}$, sodass $\mathcal{L}(\mathcal{A}) = \mathcal{L}(r)$ gilt.
- Zu jedem endlichen Automaten $\mathcal{A}$ gibt es einen regulären Ausdruck r, sodass $\mathcal{L}(r) = \mathcal{L}(\mathcal{A})$ gilt.

Nicht regulär sind Sprachen, deren Analyse ein unbegrenztes Gedächtnis erfordert:

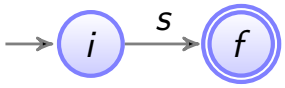
- Klammerausdrücke: $\{\texttt{()}, \texttt{(())}, \texttt{()()}, \texttt{((()))}, \texttt{(())()}, \texttt{()()()}, \ldots\}$
- $\{\, \mathtt{a}^n\mathtt{b}^n \mid n \geq 0 \,\} = \{\varepsilon, \mathtt{ab}, \mathtt{aabb}, \mathtt{aaabbb}, \ldots\}$
- $\{\, \mathtt{a}^n\mathtt{b}^n\mathtt{c}^n \mid n \geq 0 \,\} = \{\varepsilon, \mathtt{abc}, \mathtt{aabbcc}, \mathtt{aaabbbccc}, \ldots\}$
- Palindrome: Wörter, die identisch mit ihrem Spiegelbild sind. $\{\mathtt{otto}, \mathtt{anna}, \mathtt{reliefpfeiler}, \mathtt{o\,genie\,der\,herr\,ehre\,dein\,ego}, \ldots\}$
- Doppelwörter: $\{\, ww \mid w \in \Sigma^* \,\}$ (falls $|\Sigma| > 1$)

Was Sie heute erwartet

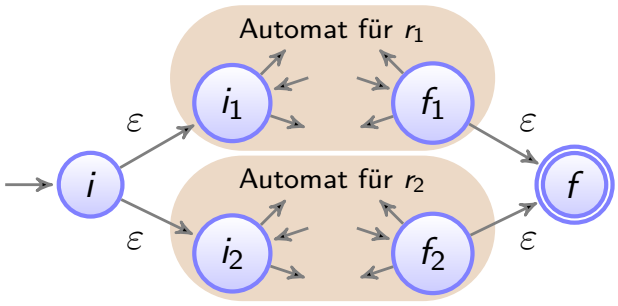
Vom regulären Ausdruck zum Automaten

Automat für $\emptyset$:

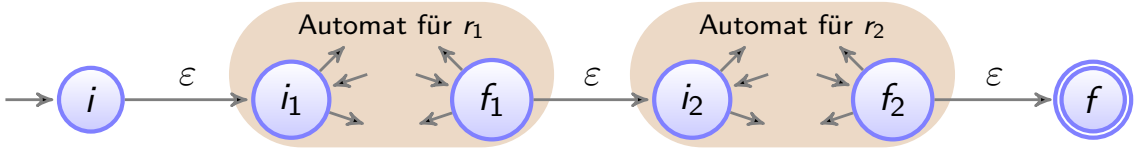
Automat für $s \in \Sigma \cup \{\varepsilon\}$:



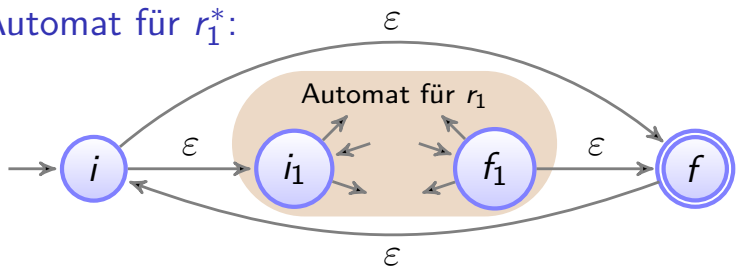
Automat für $r_1 + r_2$:



Automat für $r_1 r_2$:

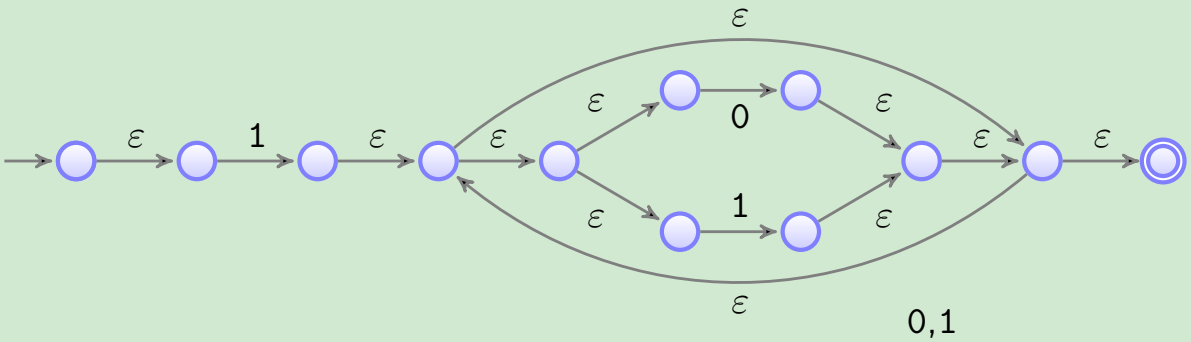


Automat für r_1^*:

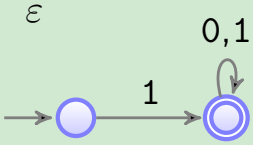


Automat für Binärnumerale ohne führende Null

Regulärer Ausdruck: $1(0+1)^*$



Minimaler deterministischer Automat:



Manuelle Konstruktion von Automaten:

1. Konstruiere die Automaten zu einfachen Teilsprachen „durch Hinschauen“.
2. Verwende die allgemeine Konstruktion mit ε-Übergängen für undurchsichtige Situationen.

Was Sie heute erwartet

Vom Automaten zum regulären Ausdruck

R ... Menge der regulären Ausdrücke über Σ

Verallgemeinerter endlicher Automat

... wird beschrieben durch ein 5-Tupel $\mathcal{A} = \langle Q, \Sigma, \delta, i, f \rangle$, wobei

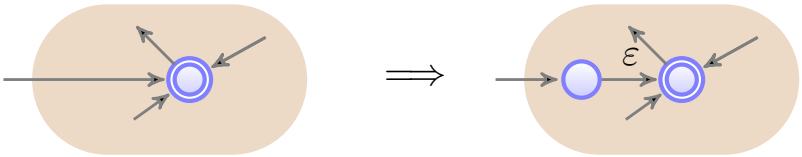
- Q ... endliche Zustandsmenge
- Σ ... Eingabealphabet
- $\delta\colon (Q - \{f\}) \times (Q - \{i\}) \mapsto R$ ... Übergangsfunktion
- $i \in Q$ ... Anfangszustand
- $f \in Q$, $f \neq i$ ... Endzustand

Unterschiede zu „normalen“ Automaten:

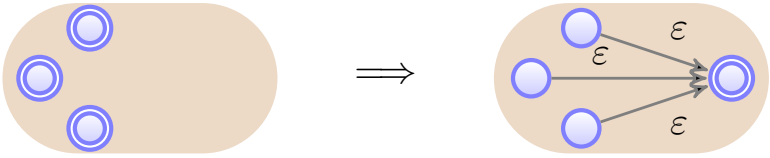
- keine Übergänge in den Anfangszustand;
- nur ein Endzustand, der nicht Anfangszustand ist;
- keine Übergänge weg vom Endzustand;
- nur ein Übergang zwischen je zwei Zuständen;
- Übergänge beschriftet mit regulären Ausdrücken.

Umwandlung eines endlichen Automaten in einen verallgemeinerten

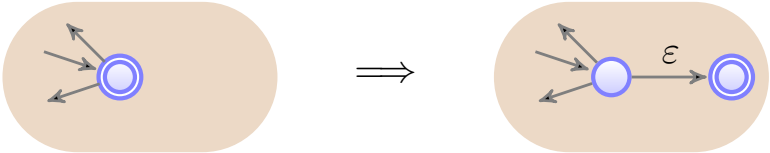
- Übergänge in den Anfangszustand oder Anfangszustand ist Endzustand:



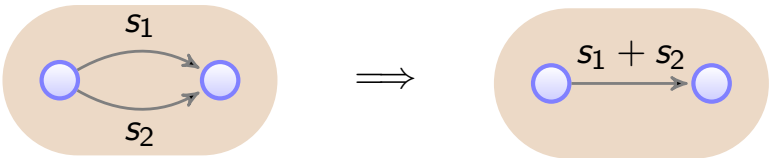
- Mehrere Endzustände:



- Übergänge weg vom Endzustand:



- Mehrere Übergänge zwischen zwei Zuständen:

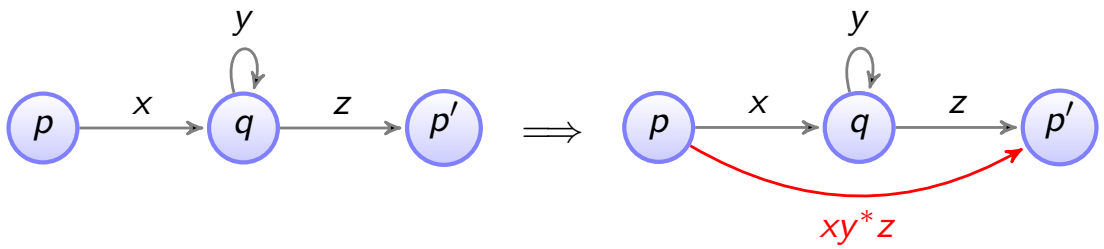


Vom verallgemeinerten Automaten zum regulären Ausdruck

Gegeben: Verallgemeinerter Automat $\mathcal{A} = \langle Q, \Sigma, \delta, i, f \rangle$

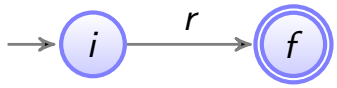
Für jeden Zustand $q \in Q - \{i, f\}$ führe folgende Schritte durch:

1. Füge zwischen allen Nachbarn p, p' von q neue Übergänge hinzu:



   (x, y und z bezeichnen reguläre Ausdrücke.)

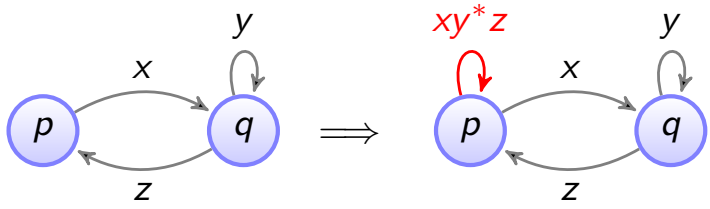
2. Entferne q und alle Kanten von und nach q aus dem Automaten.

Restautomat: 

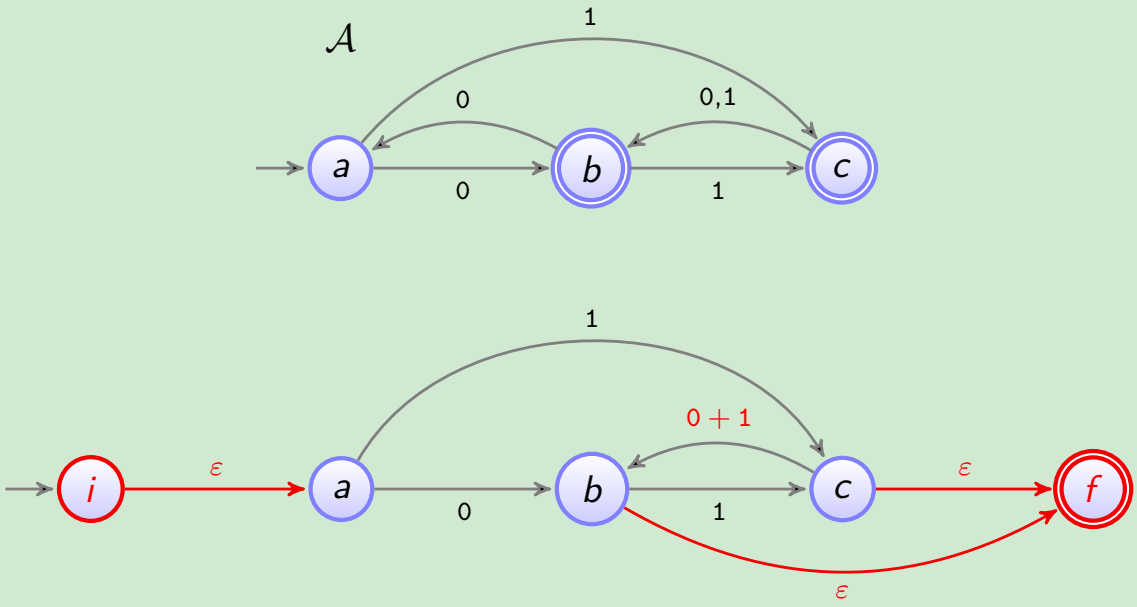
Ergebnis: r ist ein regulärer Ausdruck mit $\mathcal{L}(r) = \mathcal{L}(\mathcal{A})$.

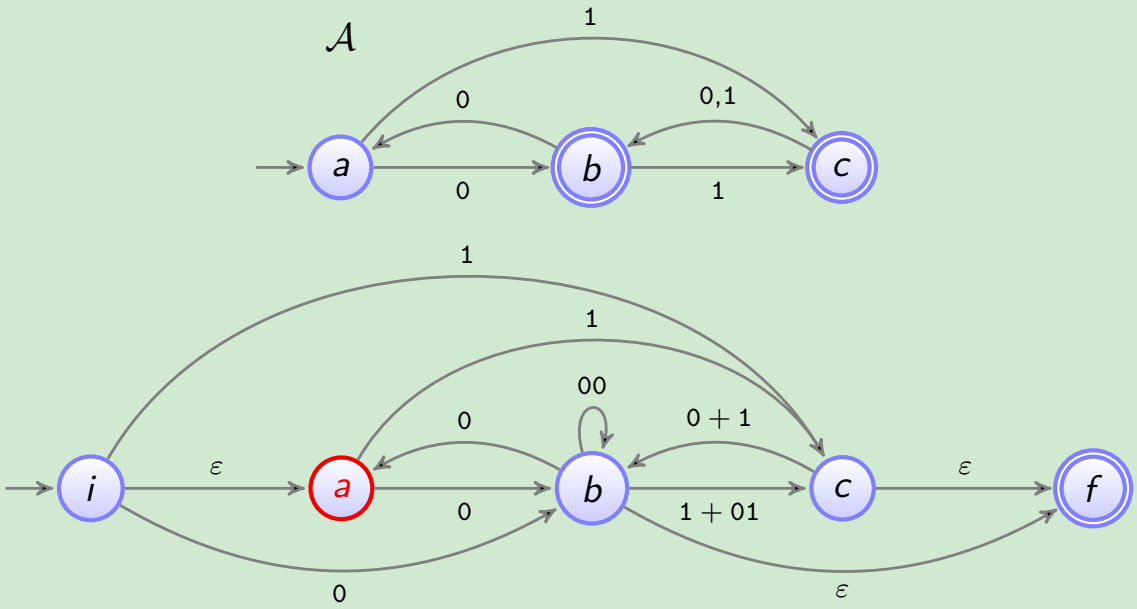
Anmerkungen:

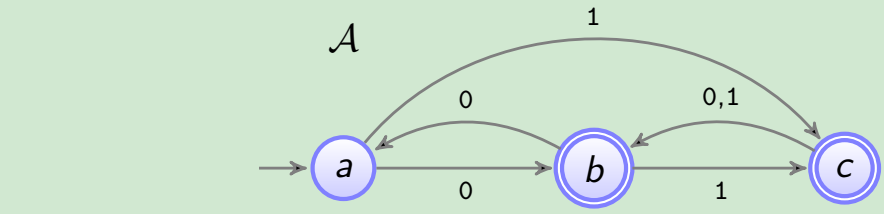
- Falls p und p' derselbe Knoten sind, erhält man:

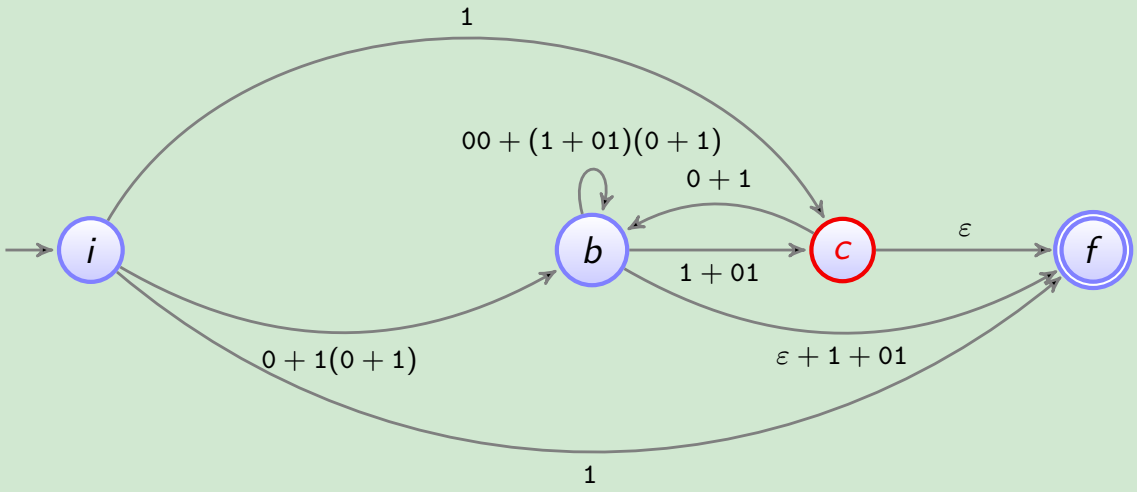


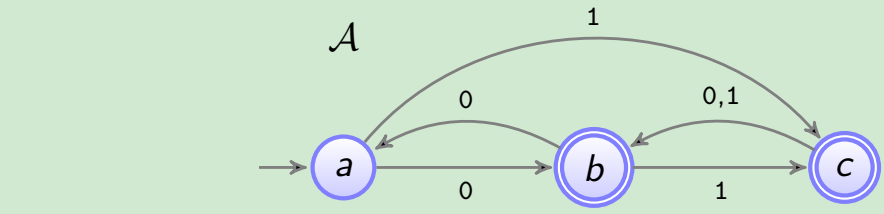
- Falls die Schleife mit dem Ausdruck y nicht existiert, entfällt y^*.
- Falls der Übergang p–p' bereits existiert, wird xy^*z addiert.

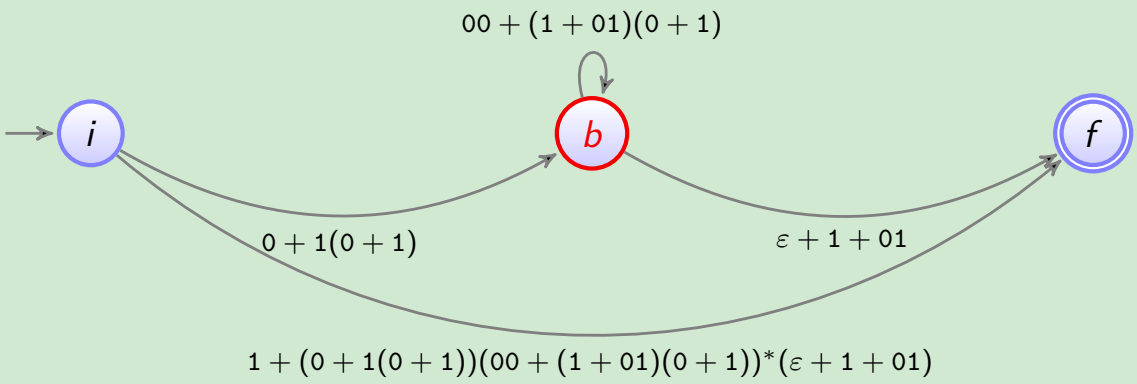

$\mathcal{A}$
1
0
0,1
a
b
c
0
1
1
0 + 1
i
ε
a
b
c
ε
f
0
1
ε

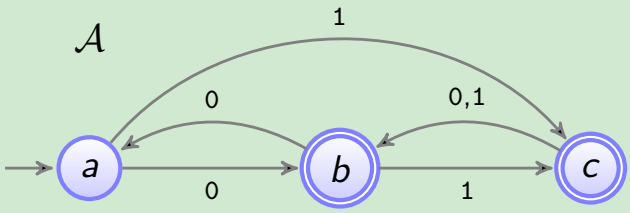

$\mathcal{A}$
1
0
0,1
a
b
c
0
1
1
1
00
0
0 + 1
i
ε
a
b
c
ε
f
0
1 + 01
0
ε


$\mathcal{A}$
1
0
0,1
a
b
c
0
1


1
$00 + (1 + 01)(0 + 1)$
$0 + 1$
i
b
c
ε
f
$1 + 01$
$0 + 1(0 + 1)$
$\varepsilon + 1 + 01$
1


$\mathcal{A}$
1
0
0,1
a
b
c
0
1


$00 + (1 + 01)(0 + 1)$
i
b
f
$0 + 1(0 + 1)$
$\varepsilon + 1 + 01$
$1 + (0 + 1(0 + 1))(00 + (1 + 01)(0 + 1))^*(\varepsilon + 1 + 01)$



$1 + (0 + 1(0 + 1))(00 + (1 + 01)(0 + 1))^*(\varepsilon + 1 + 01)$

$r = 1 + (0 + 10 + 11)(00 + 10 + 11 + 010 + 011)^*(\varepsilon + 1 + 01)$

Es gilt: $\mathcal{L}(r) = \mathcal{L}(\mathcal{A})$.

3.0 VU Formale Modellierung

Gernot Salzer

Forschungsbereich Theory and Logic
Institut für Logic and Computation

8.5.2018

Was Sie letztes Mal hörten

Reguläre Sprachen

Alle Sprachen, die aus einem Alphabet mit Hilfe von Vereinigung, Verkettung und Stern gebildet werden können.

Regulären Sprachen über einem Alphabet

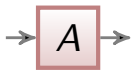
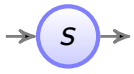
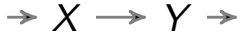
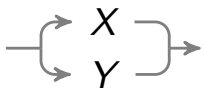
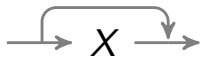
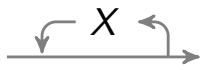
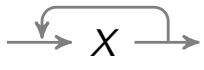
Die Menge der regulären Sprachen über Σ, $\mathcal{L}_{\text{reg}}(\Sigma)$, ist die kleinste Menge, sodass gilt:

- $\{\}$, $\{\varepsilon\}$ und $\{s\}$ sind reguläre Sprachen (für alle $s \in \Sigma$).
- Wenn L und L' reguläre Sprachen sind, dann auch $L \cup L'$, $L \cdot L'$ und L^*.

Reellen Numerale: reguläre Sprache über $\Sigma = \{0, \ldots, 9, ., \mathtt{E}, +, -\}$

$$
\begin{aligned}
\mathit{real} &= \mathit{digit} \cdot \mathit{digit}^* \cdot \{.\} \cdot \mathit{digit}^* \cdot (\{\varepsilon\} \cup \mathit{scale}) \\
\mathit{scale} &= \{\mathtt{E}\} \cdot \{+, -, \varepsilon\} \cdot \mathit{digit} \cdot \mathit{digit}^* \\
\mathit{digit} &= \{0, \ldots, 9\} = \{0\} \cup \cdots \cup \{9\}
\end{aligned}
$$

Was Sie letztes Mal hörten

Reg. Sprache	Algebra	EBNF	Syntaxdiagramm	
A	A	A		Abkürzung
$\{\}$	$\emptyset$			Leersprache
$\{\varepsilon\}$	ε			Leerwortsprache
$\{s\}$	s	"s"		Terminalsymbol
$X \cdot Y$	XY	XY		Aufeinanderfolge
$X \cup Y$	$X + Y$	$X \mid Y$		Alternativen
$\{\varepsilon\} \cup X$	$\varepsilon + X$	$[X]$		Option
X^*	X^*	$\{X\}$		Wiederholung ≥ 0
X^+	XX^*, X^+	$X\{X\}$		Wiederholung ≥ 1
(X)	(X)	(X)		Gruppierung

POSIX Extended Regular Expressions (ERE)

regexp	trifft zu auf	Reg. Sprache
\s	Zeichen s	$\{s\}$
s	s, falls kein Sonderzeichen	$\{s\}$
.	alle Zeichen	Σ
^	Zeilenanfang	
\$	Zeilenende	
$[s_1 \cdots s_n]$	eines der Zeichen s_i	$\{s_1, \ldots, s_n\}$
$[\hat{}s_1 \cdots s_n]$	alle Zeichen außer $s_1, \ldots, s_n$	$\Sigma - \{s_1, \ldots, s_n\}$
(X)	X	X
XY	X gefolgt von Y	$X \cdot Y$
$X \mid Y$	X oder Y	$X \cup Y$
$X*$	≥ 0 Mal X	X^*
$X+$	≥ 1 Mal X	X^+
$X?$	≤ 1 Mal X	$X \cup \{\varepsilon\}$
$X\{i\}$	i Mal X	X^i
$X\{i,\}$	$\geq i$ Mal X	$X^i \cdot X^*$
$X\{i,j\}$	i bis j Mal X	$X^i \cup X^{i+1} \cup \cdots \cup X^j$

Reelle Numerale

. . . sind eine reguläre Sprache über $\{0, \ldots, 9, ., \mathtt{E}, +, -\}$

$$real = digit \cdot digit^* \cdot \{.\} \cdot digit^* \cdot (\{\varepsilon\} \cup scale)$$
$$scale = \{\mathtt{E}\} \cdot \{+, -, \varepsilon\} \cdot digit \cdot digit^*$$
$$digit = \{0, \ldots, 9\} = \{0\} \cup \cdots \cup \{9\}$$

. . . dargestellt als regulärer Ausdruck in algebraischer Notation

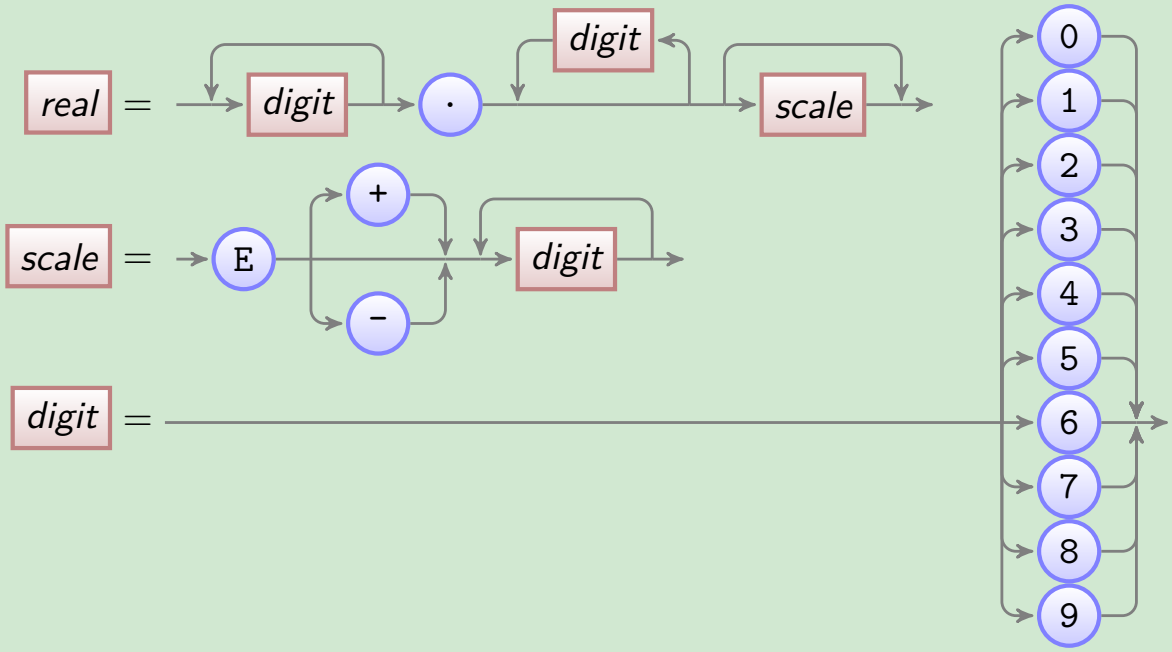
$$R = DD^*.D^*(\varepsilon + S)$$
$$S = \mathtt{E}(+ + - + \varepsilon)DD^*$$
$$D = 0 + 1 + \cdots + 9$$

. . . dargestellt als regulärer Ausdruck in EBNF-Notation

$$R = D\ \{\ D\ \}\ \texttt{"."}\ \{\ D\ \}\ [\ S\ ]$$
$$S = \texttt{"E"}\ [\ \texttt{"+"} \mid \texttt{"-"}\ ]\ D\ \{\ D\ \}$$
$$D = \texttt{"0"} \mid \texttt{"1"} \mid \texttt{"2"} \mid \cdots \mid \texttt{"9"}$$

Reelle Numerale

. . . dargestellt als Syntaxdiagramm



. . . dargestellt als Posix Extended Regular Expression

```
^[0-9]+\.[0-9]*(E[+-]?[0-9]+)?$
```

Was Sie letztes Mal hörten

Eigenschaften regulärer Sprachen

Reguläre Sprachen sind abgeschlossen gegenüber allen relevanten Operationen.

Beispiele für Operationen: Vereinigung, Verkettung, Stern, Durchschnitt, Komplement, Differenz, Homomorphismen, Quotientenbildung

Alle relevanten Probleme regulärer Sprachen sind entscheidbar.

Beispiele entscheidbarer Probleme:

- Gegeben ein Wort w und einen regulären Ausdruck r, gilt $w \in \mathcal{L}(r)$? (Wortproblem)
- Gegeben zwei reguläre Ausdrücke r und r', gilt $\mathcal{L}(r) = \mathcal{L}(r')$? (Äquivalenzproblem)
- Gegeben einen regulären Ausdruck r, ist $\mathcal{L}(r)$ leer/endlich/unendlich?

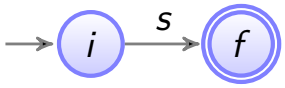
Eine Sprache ist regulär genau dann, wenn sie von einem endlichen Automaten akzeptiert wird.

Was Sie letztes Mal hörten

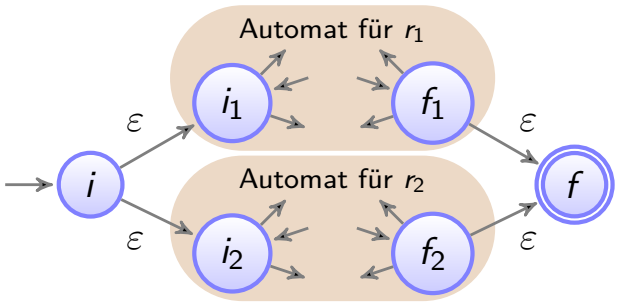
Vom regulären Ausdruck zum Automaten

Automat für $\emptyset$:

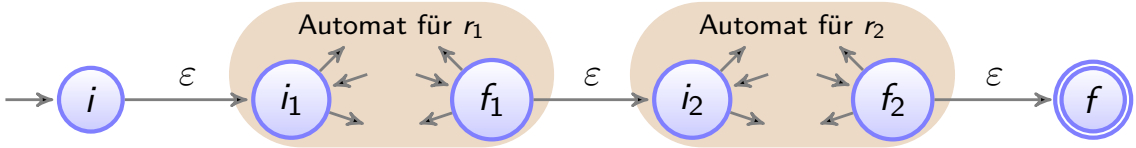
Automat für $s \in \Sigma \cup \{\varepsilon\}$:



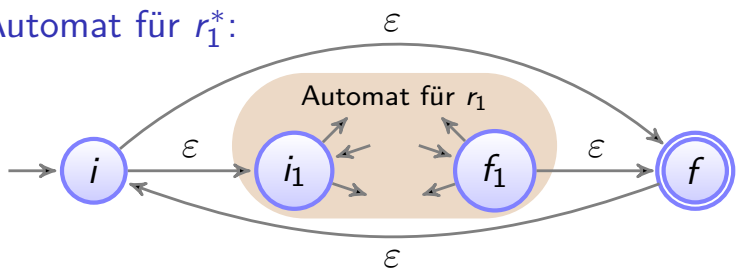
Automat für $r_1 + r_2$:



Automat für $r_1 r_2$:



Automat für r_1^*:



Was Sie letztes Mal hörten

Vom Automaten zum regulären Ausdruck

1. Umwandlung des endlichen in einen verallgemeinerten Automaten
2. Schrittweise Elimination der Zustände,
   Aufbau von regulären Ausdrücken
3. Ablesen des regulären Ausdrucks vom einzigen verbliebenen Übergang

Unterschiede zwischen verallgemeinertem und „normalem“ Automaten:

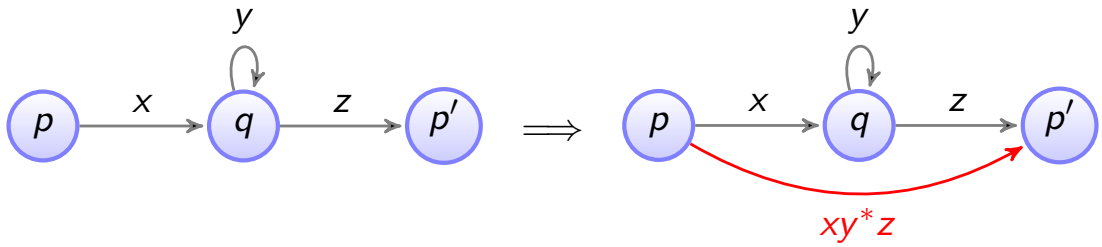
- keine Übergänge in den Anfangszustand
- nur ein Endzustand, der nicht Anfangszustand ist
- keine Übergänge weg vom Endzustand
- nur ein Übergang zwischen je zwei Zuständen
- Übergänge beschriftet mit regulären Ausdrücken

Vom verallgemeinerten Automaten zum regulären Ausdruck

Gegeben: Verallgemeinerter Automat $\mathcal{A} = \langle Q, \Sigma, \delta, i, f \rangle$

Für jeden Zustand $q \in Q - \{i, f\}$ führe folgende Schritte durch:

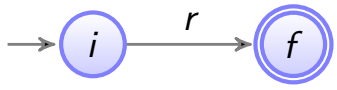
1. Füge zwischen allen Nachbarn p, p' von q neue Übergänge hinzu:



   (x, y und z bezeichnen reguläre Ausdrücke.)

2. Entferne q und alle Kanten von und nach q aus dem Automaten.

Restautomat:



Ergebnis: r ist ein regulärer Ausdruck mit $\mathcal{L}(r) = \mathcal{L}(\mathcal{A})$.

Was Sie heute erwartet

Grenzen regulärer Sprachen

Was reguläre Ausdrücke und endliche Automaten beschreiben können:

- lexikale Elemente in Programmiersprachen: Identifier, Numerale, ...
- Morphologie von Worten in natürlichen Sprachen: korrekte Fall-/Zahl-/Zeitendungen, ...
- Systeme, die nur ein endliches Gedächtnis besitzen

Was sie nicht beschreiben können:

- geklammerte Ausdrücke in Programmiersprachen
- geschachtelte Programmstrukturen wie `if ... while ... if ... endif ... endwhile ... endif`
- Schachtelung von Haupt- und Nebensätzen in natürlichen Sprachen
- Systeme mit einem (potentiell) unendlichen Speicher

Wohlgeklammerte Ausdrücke sind nicht regulär

$\{(), (()), ()(), ((())), (())(), ()()(), \ldots\}$

$\{ a^n b^n \mid n \geq 0 \}$ ist nicht regulär

$\{\varepsilon, ab, aabb, aaabbb, aaaabbbb, aaaaabbbbb, \ldots\}$

Kontextfreie Grammatiken

- Menge von Ersetzungsregeln
- 2 Arten von Symbolen: Terminale, Nonterminale
- Ausgehend vom Start-Nonterminal werden Nonterminale solange ersetzt, bis nur noch Terminale bleiben.

$$
\begin{array}{ll@{\qquad}ll}
\mathit{Satz} & \to \mathit{HwP}\ \mathit{ZwP} & \mathit{Art} & \to \texttt{der} \mid \texttt{den} \\
\mathit{HwP} & \to \mathit{Art}\ \mathit{Hw} & \mathit{Hw} & \to \texttt{Hund} \mid \texttt{Mond} \\
\mathit{ZwP} & \to \mathit{Hzw}\ \mathit{HwP}\ \mathit{Zw} & \mathit{Hzw} & \to \texttt{wird} \\
 & & \mathit{Zw} & \to \texttt{anbellen}
\end{array}
$$

$$
\begin{aligned}
\mathit{Satz} &\Rightarrow_p \mathit{HwP}\ \ \mathit{ZwP} \\
&\Rightarrow_p \mathit{Art}\ \ \mathit{Hw}\ \ \mathit{Hzw}\ \ \mathit{HwP}\ \ \mathit{Zw} \\
&\Rightarrow_p \texttt{der}\ \ \texttt{Hund}\ \ \texttt{wird}\ \ \mathit{Art}\ \ \mathit{Hw}\ \ \texttt{anbellen} \\
&\Rightarrow_p \texttt{der}\ \ \texttt{Hund}\ \ \texttt{wird}\ \ \texttt{den}\ \ \texttt{Mond}\ \ \texttt{anbellen}
\end{aligned}
$$

Generative Grammatiken gehen zurück auf Noam Chomsky (Linguist, Philosoph, Aktivist; *1928).

Kontextfreie Grammatik

... wird beschrieben durch ein 4-Tupel $G = \langle V, T, P, S \rangle$, wobei

- V ... Menge der Nonterminalsymbole (Variablen)
- T ... Menge der Terminalsymbole ($V \cap T = \{\}$)
- $P \subseteq V \times (V \cup T)^*$ ... Produktionen
- $S \in V$ ... Startsymbol

Schreibweise: $A \to w$ statt $(A, w) \in P$

$A \to w_1 \mid \cdots \mid w_n$ statt $A \to w_1, \ldots, A \to w_n$

Ableitbarkeit

... in einem Schritt:

$x\,A\,y \Rightarrow x\,w\,y$, falls $A \to w \in P$ und $x, y \in (V \cup T)^*$.

... in mehreren Schritten:

$u \overset{*}{\Rightarrow} v$, falls

- $u = v$, oder
- $u \Rightarrow u'$ und $u' \overset{*}{\Rightarrow} v$ für ein Wort $u' \in (V \cup T)^*$.

Von Grammatik G generierte Sprache

$\mathcal{L}(G) = \{\, w \in T^* \mid S \overset{*}{\Rightarrow} w \,\}$

Grammatiken G_1 und G_2 heißen äquivalent, wenn $\mathcal{L}(G_1) = \mathcal{L}(G_2)$ gilt.

$G = \langle \{S\}, \{\mathtt{a}, \mathtt{b}\}, \{S \to \varepsilon \mid \mathtt{a}\,S\,\mathtt{b}\}, S \rangle$

$S \overset{*}{\Rightarrow} \mathtt{aabb}$, weil $S \Rightarrow \mathtt{a}\,S\,\mathtt{b} \Rightarrow \mathtt{aa}\,S\,\mathtt{bb} \Rightarrow \mathtt{aa}\,\varepsilon\,\mathtt{bb} = \mathtt{aabb}$

$\mathcal{L}(G) = \{\, \mathtt{a}^n\mathtt{b}^n \mid n \geq 0 \,\} = \{\varepsilon, \mathtt{ab}, \mathtt{aabb}, \mathtt{aaabbb}, \ldots\}$

Kontextfreie Sprachen

Eine Sprache heißt kontextfrei, wenn es eine kontextfreie Grammatik gibt, die sie generiert.

Verschiedene Ableitbarkeitsbegriffe

Linksableitbarkeit: $x\,A\,y \Rightarrow_L x\,w\,y$ falls $A \to w \in P$ und $x \in T^*$
(In jedem Schritt wird die linkeste Variable ersetzt.)

Rechtsableitbarkeit: $x\,A\,y \Rightarrow_R x\,w\,y$ falls $A \to w \in P$ und $y \in T^*$
(In jedem Schritt wird die rechteste Variable ersetzt.)

Parallelableitbarkeit: $x_0\,A_1\,x_1 \cdots A_n\,x_n \Rightarrow_P x_0\,w_1\,x_1 \cdots w_n\,x_n$
falls $A_1 \to w_1, \ldots, A_n \to w_n \in P$ und $x_0, \ldots, x_n \in T^*$
(In jedem Schritt werden alle Variablen ersetzt.)

- $\overset{*}{\Rightarrow}_L$, $\overset{*}{\Rightarrow}_R$ und $\overset{*}{\Rightarrow}_P$ sind eingeschränkte Ableitungsrelationen: Manche Wörter $w \in (V \cup T)^*$ sind mit $\overset{*}{\Rightarrow}$ herleitbar ($S \overset{*}{\Rightarrow} w$), aber nicht mit diesen Relationen.
- Sie können aber jedes Wort der Sprache ableiten. Für alle $w \in T^*$ gilt: $S \overset{*}{\Rightarrow} w$ gdw. $S \overset{*}{\Rightarrow}_L w$ gdw. $S \overset{*}{\Rightarrow}_R w$ gdw. $S \overset{*}{\Rightarrow}_P w$

$$\mathcal{L}(G) = \{\, w \in T^* \mid S \overset{*}{\Rightarrow} w \,\} = \{\, w \in T^* \mid S \overset{*}{\Rightarrow}_L w \,\}$$
$$= \{\, w \in T^* \mid S \overset{*}{\Rightarrow}_R w \,\} = \{\, w \in T^* \mid S \overset{*}{\Rightarrow}_P w \,\}$$

21

Was Sie heute erwartet

Syntax aussagenlogischer Formeln

$G = \langle \{Fm, Op, Var\}, T, P, Fm \rangle$, wobei

$T = \{\top, \bot, \neg, (,), \wedge, \uparrow, \vee, \downarrow, \equiv, \not\equiv, \supset, \subset\} \cup \mathcal{V}$

$P = \{\ Fm \to Var \mid \top \mid \bot \mid \neg Fm \mid (\ Fm\ Op\ Fm\ )\ ,$
$\quad Op \to \wedge \mid \uparrow \mid \vee \mid \downarrow \mid \equiv \mid \not\equiv \mid \supset \mid \subset\ ,$
$\quad Var \to \mathtt{A} \mid \mathtt{B} \mid \mathtt{C} \mid \cdots\ \}$

$((\mathtt{A} \uparrow \mathtt{B}) \uparrow (\mathtt{A} \uparrow \mathtt{B}))$ ist eine aussagenlogische Formel, weil:

$$
\begin{array}{ll}
Fm \Rightarrow_L (\ Fm\ Op\ Fm\ ) & \Rightarrow_L (\ (\ Fm\ Op\ Fm\ )\ Op\ Fm\ ) \\
\Rightarrow_L (\ (\ Var\ Op\ Fm\ )\ Op\ Fm\ ) & \Rightarrow_L (\ (\mathtt{A}\ Op\ Fm\ )\ Op\ Fm\ ) \\
\Rightarrow_L (\ (\mathtt{A} \uparrow Fm\ )\ Op\ Fm\ ) & \Rightarrow_L (\ (\mathtt{A} \uparrow Var\ )\ Op\ Fm\ ) \\
\Rightarrow_L (\ (\mathtt{A} \uparrow \mathtt{B}\ )\ Op\ Fm\ ) & \Rightarrow_L (\ (\mathtt{A} \uparrow \mathtt{B}\ ) \uparrow Fm\ ) \\
\Rightarrow_L (\ (\mathtt{A} \uparrow \mathtt{B}\ ) \uparrow (\ Fm\ Op\ Fm\ )\ ) & \Rightarrow_L (\ (\mathtt{A} \uparrow \mathtt{B}\ ) \uparrow (\ Var\ Op\ Fm\ )\ ) \\
\Rightarrow_L (\ (\mathtt{A} \uparrow \mathtt{B}\ ) \uparrow (\mathtt{A}\ Op\ Fm\ )\ ) & \Rightarrow_L (\ (\mathtt{A} \uparrow \mathtt{B}\ ) \uparrow (\mathtt{A} \uparrow Fm\ )\ ) \\
\Rightarrow_L (\ (\mathtt{A} \uparrow \mathtt{B}\ ) \uparrow (\mathtt{A} \uparrow Var\ )\ ) & \Rightarrow_L (\ (\mathtt{A} \uparrow \mathtt{B}\ ) \uparrow (\mathtt{A} \uparrow \mathtt{B}\ )\ )
\end{array}
$$

Syntax aussagenlogischer Formeln

$G = \langle \{Fm, Op, Var\}, T, P, Fm \rangle$, wobei

$T = \{\top, \bot, \neg, (,), \wedge, \uparrow, \vee, \downarrow, \equiv, \not\equiv, \supset, \subset\} \cup \mathcal{V}$

$P = \{\ Fm \to Var \mid \top \mid \bot \mid \neg Fm \mid (\ Fm\ Op\ Fm\ )\ ,$
$\quad Op \to \wedge \mid \uparrow \mid \vee \mid \downarrow \mid \equiv \mid \not\equiv \mid \supset \mid \subset\ ,$
$\quad Var \to \mathtt{A} \mid \mathtt{B} \mid \mathtt{C} \mid \cdots\ \}$

$((\mathtt{A} \uparrow \mathtt{B}) \uparrow (\mathtt{A} \uparrow \mathtt{B}))$ ist eine aussagenlogische Formel, weil:

$$
\begin{array}{l}
Fm \Rightarrow_P (\ Fm\ Op\ Fm\ ) \\
\quad \Rightarrow_P (\ (\ Fm\ Op\ Fm\ ) \uparrow (\ Fm\ Op\ Fm\ )\ ) \\
\quad \Rightarrow_P (\ (\ Var \uparrow Var\ ) \uparrow (\ Var \uparrow Var\ )\ ) \\
\quad \Rightarrow_P (\ (\mathtt{A} \uparrow \mathtt{B}\ ) \uparrow (\mathtt{A} \uparrow \mathtt{B}\ )\ )
\end{array}
$$

Reelle Numerale

$G = \langle V, T, P, \mathit{Real} \rangle$, wobei

$V = \{\mathit{Real}, \mathit{Scale}, \mathit{Sign}, \mathit{Digits}, \mathit{Digit}\}$
$T = \{0, \ldots, 9, \ ., \ \mathtt{E}, \ +, \ -\}$

und P folgende Produktionen sind:

$$
\begin{aligned}
\mathit{Real} &\to \mathit{Digit\ Digits\ .\ Digits\ Scale} \\
\mathit{Scale} &\to \varepsilon \mid \mathtt{E}\ \mathit{Sign\ Digit\ Digits} \\
\mathit{Sign} &\to \varepsilon \mid + \mid - \\
\mathit{Digits} &\to \varepsilon \mid \mathit{Digit\ Digits} \\
\mathit{Digit} &\to 0 \mid \cdots \mid 9
\end{aligned}
$$

Optionalität:
$A \to \varepsilon \mid B$ (A steht für optionales B)

Wiederholung:
$A \to \varepsilon \mid B\,A$ (A steht für wiederholtes B, auch 0 Mal)
$A \to B \mid B\,A$ (A steht für wiederholtes B, mind. 1 Mal)

Wohlgeformte Klammerausdrücke

WKA ist die kleinste Menge, sodass

- $\varepsilon \in \mathit{WKA}$
- $(w) \in \mathit{WKA}$, wenn $w \in \mathit{WKA}$
- $w_1 w_2 \in \mathit{WKA}$, wenn $w_1, w_2 \in \mathit{WKA}$

$G = \langle \{W\}, \quad \{(,)\}, \quad \{W \to \varepsilon \mid (\,W\,) \mid W\,W\}, \quad W \rangle$

Beispiel einer Ableitung:
$$
\begin{aligned}
W &\Rightarrow_P W\,W \\
&\Rightarrow_P (W)(W) \\
&\Rightarrow_P ()(W\,W) \\
&\Rightarrow_P ()((W)(W)) \\
&\Rightarrow_P ()(()())
\end{aligned}
$$

$$
\begin{aligned}
\mathcal{L}(G) &= \{\varepsilon, \ (), \ (()), \ ()(), \ ((())), \ (())(), \ ()(()), \ ()()(), \ldots\} \\
&= \mathit{WKA}
\end{aligned}
$$

Induktive Definition vs. kontextfreie Grammatik

Induktive Definition für $\mathcal{M}$	kontextfreie Grammatik für $\mathcal{M}$
Mengen $\mathcal{M}$, $\mathcal{M}_0$, A_1, B_1, ...	Var. $\langle\mathcal{M}\rangle$, $\langle\mathcal{M}_0\rangle$, $\langle A_1\rangle$, $\langle B_1\rangle$, ...
$\mathcal{M}$ ist die kleinste Menge, sodass:	$\mathcal{M} = \mathcal{L}(\langle \ldots, P, \langle\mathcal{M}\rangle\rangle)$, wobei P folgende Produktionen sind:
$\mathcal{M}_0 \subseteq \mathcal{M}$	$\langle\mathcal{M}\rangle \to \langle\mathcal{M}_0\rangle$
$f(x_1, \ldots, x_m) \in \mathcal{M}$, falls $x_1 \in A_1, \ldots, x_m \in A_m$	$\langle\mathcal{M}\rangle \to f(\langle A_1\rangle, \ldots, \langle A_m\rangle)$
$g(x_1, \ldots, x_n) \in \mathcal{M}$, falls $x_1 \in B_1, \ldots, x_n \in B_n$	$\langle\mathcal{M}\rangle \to g(\langle B_1\rangle, \ldots, \langle B_n\rangle)$
$\cdots \in \mathcal{M}$, falls ...	$\langle\mathcal{M}\rangle \to \ldots$
$h(x_1, x_2) \in \mathcal{M}$, falls $x_1, x_2 \in \mathcal{M}$	$\langle\mathcal{M}\rangle \to h(\langle\mathcal{M}\rangle, \langle\mathcal{M}\rangle)$
$h(x, x) \in \mathcal{M}$, falls $x \in \mathcal{M}$	keine Entsprechung, nicht kontextfrei

Was Sie heute erwartet

Backus-Naur-Form (BNF)

Synonym für kontextfreie Produktionen

Historische Notation:

- Nonterminale in spitzen Klammern
- Terminale unter Anführungszeichen
- ::= als Trennsymbol

Durch diese Konvention entfällt die Notwendigkeit, die Mengen der Nonterminale (V) und der Terminale (T) anzugeben.

$$
\begin{aligned}
\langle \mathit{Real} \rangle &::= \langle \mathit{Digit} \rangle \, \langle \mathit{Digits} \rangle \, \texttt{"."} \, \langle \mathit{Digits} \rangle \, \langle \mathit{Scale} \rangle \\
\langle \mathit{Digit} \rangle &::= \texttt{"0"} \mid \cdots \mid \texttt{"9"} \\
\langle \mathit{Digits} \rangle &::= \varepsilon \mid \langle \mathit{Digit} \rangle \, \langle \mathit{Digits} \rangle \\
\langle \mathit{Scale} \rangle &::= \varepsilon \mid \texttt{"E"} \, \langle \mathit{Sign} \rangle \, \langle \mathit{Digit} \rangle \, \langle \mathit{Digits} \rangle \\
\langle \mathit{Sign} \rangle &::= \varepsilon \mid \texttt{"+"} \mid \texttt{"-"}
\end{aligned}
$$

Erweiterte Backus-Naur-Form (EBNF)

„Regular Right Part Grammar“ (RRPG)

- erlaubt reguläre Ausdrücke auf rechter Seite der Produktionen
- Bessere Lesbarkeit durch Vermeidung von Rekursionen und Leerwörtern
- Kompaktere Spezifikationen
- keine echte Erweiterung: EBNF-Notationen lassen sich eliminieren

Elimination der EBNF-Notation:

$A \to w_1 \, (w_2) \, w_3$ entspricht $A \to w_1 \, B \, w_3$, $B \to w_2$

$A \to w_1 \, \{w_2\} \, w_3$ entspricht $A \to w_1 \, B \, w_3$, $B \to \varepsilon \mid w_2 \, B$

$A \to w_1 \, [w_2] \, w_3$ entspricht $A \to w_1 \, B \, w_3$, $B \to \varepsilon \mid w_2$

Listen von Bezeichnern

EBNF: $IdList \rightarrow Id\ \{\ \texttt{","}\ Id\ \}$

Ohne EBNF: $IdList \rightarrow Id\ B$
$B \rightarrow \varepsilon \mid \texttt{","}\ Id\ B$

oder $IdList \rightarrow Id \mid Id\ \texttt{","}\ IdList$

Skalierungsfaktor der reellen Numerale

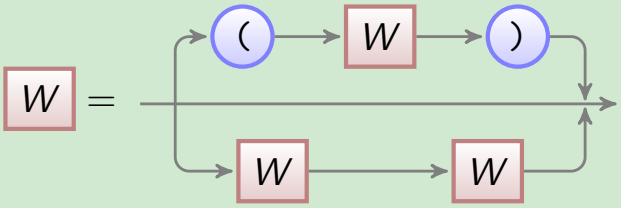
EBNF: $Scale \rightarrow \texttt{"E"}\ [\texttt{"+"} \mid \texttt{"-"}]\ Digit\ \{Digit\}$

Ohne EBNF: $Scale \rightarrow \texttt{"E"}\ B_1\ Digit\ B_2$
$B_1 \rightarrow \varepsilon \mid \texttt{"+"} \mid \texttt{"-"}$
$B_2 \rightarrow \varepsilon \mid Digit\ B_2$

Syntaxdiagramme

Mit rekursiven Diagrammen können beliebige kontextfreie Grammatiken in EBNF dargestellt werden.

Wohlgeklammerte Ausdrücke



$W \rightarrow (\ W\ )$
$\mid \varepsilon$
$\mid W\ W$

Was Sie heute erwartet

Style Guide für Grammatiken

Wiederholungsoperator statt Rekursion (wenn möglich).

Schlecht: $A \to a \mid A\,A$ oder $A \to a \mid a\,A$

Besser: $a\,\{\,a\,\}$ oder $a+$ oder ... (je nach gewählter Notation)

Optionsoperator statt Leerwort.

Schlecht: $\varepsilon \mid A$

Besser: $[\,A\,]$ oder $A?$ oder ... (je nach gewählter Notation)

Modularisierung statt Duplizierung.

Schlecht: $\mathit{Identifier} \to (\texttt{"a"} \mid \cdots \mid \texttt{"z"})\,\{\,\texttt{"a"} \mid \cdots \mid \texttt{"z"} \mid \texttt{"0"} \mid \cdots \mid \texttt{"9"}\,\}$

Besser:
$$\begin{aligned} \mathit{Identifier} &\to \mathit{Letter}\,\{\,\mathit{Letter} \mid \mathit{Digit}\,\} \\ \mathit{Letter} &\to \texttt{"a"} \mid \cdots \mid \texttt{"z"} \\ \mathit{Digit} &\to \texttt{"0"} \mid \cdots \mid \texttt{"9"} \end{aligned}$$

Verwendung sprechender Namen.

Schlecht:
$$\begin{aligned} X &\to Y\,\{\,Y \mid Z\,\} \\ Y &\to \texttt{"a"} \mid \cdots \mid \texttt{"z"} \\ Z &\to \texttt{"0"} \mid \cdots \mid \texttt{"9"} \end{aligned}$$

Besser: Siehe oben.

Klare Unterscheidung zwischen Terminalen, Nonterminalen und Metanotationen.

Schlecht: `\begin{tabular}[[Position]]{Spalte{Spalte}}`

Besser: $\texttt{"\textbackslash begin\{tabular\}"}\ [\ \texttt{"["}\ \mathit{Position}\ \texttt{"]"}\ ]\ \texttt{"\{"}\ \mathit{Spalte}\,\{\,\mathit{Spalte}\,\}\ \texttt{"\}"}$

$\texttt{"\textbackslash begin\{tabular\}"}\ [\ \texttt{"["}\ \langle \mathit{Position} \rangle\ \texttt{"]"}\ ]\ \texttt{"\{"}\ \langle \mathit{Spalte} \rangle\,\{\,\langle \mathit{Spalte} \rangle\,\}\ \texttt{"\}"}$

Inhaltliche Strukturierung statt Minimierung der Nonterminale und Regeln.

Schlecht: $\texttt{"\textbackslash begin\{tabular\}"}[\texttt{"["}(\texttt{"t"} \mid \texttt{"b"})\texttt{"]"}]\texttt{"\{"}(\texttt{"l"} \mid \texttt{"c"} \mid \texttt{"r"})\{\texttt{"l"} \mid \texttt{"c"} \mid \texttt{"r"}\}\texttt{"\}"}$

Besser:
$$\begin{aligned} \ldots &\to \texttt{"\textbackslash begin\{tabular\}"}\ [\mathit{Position}]\ \mathit{Spalten} \\ \mathit{Position} &\to \texttt{"[b]"} \mid \texttt{"[t]"} \\ \mathit{Spalten} &\to \texttt{"\{"}\ \mathit{Spalte}\,\{\,\mathit{Spalte}\,\}\ \texttt{"\}"} \\ \mathit{Spalte} &\to \texttt{"l"} \mid \texttt{"c"} \mid \texttt{"r"} \end{aligned}$$

Beispiel: Tabellen in LaTeX

TeX ... Textsatzsystem von Donald E. Knuth
LaTeX ... TeX-Makros für „logisches Markup“ von Leslie Lamport

```
\begin{tabular}[t]{lc}
Eintrag 11 & Eintrag 12 \\
Eintrag 21 &
  \begin{tabular}{rr}
  Eintrag 22 & ist selber \\
  eine       & Tabelle.
  \end{tabular}\\
Eintrag 31 &  Eintrag 32
\end{tabular}
```

Eintrag 11	Eintrag 12
Eintrag 21	Eintrag 22 ist selber eine Tabelle.
Eintrag 31	Eintrag 32

Gesucht: Kontextfreie Grammatik G in EBNF für die Sprache der LaTeX-Tabellen

Beispiel: Tabellen in LaTeX

$G = \langle V, T, P, \mathit{Tabelle} \rangle$, wobei:

$$
\begin{aligned}
P = \{\ \mathit{Tabelle} &\to \texttt{"\textbackslash begin\{tabular\}"}\ [\,\mathit{Position}\,]\ \mathit{Spalten}\\
&\quad\ \mathit{Zeilen}\\
&\quad\ \texttt{"\textbackslash end\{tabular\}"}\ ,\\
\mathit{Position} &\to \texttt{"[b]"} \mid \texttt{"[t]"}\ ,\\
\mathit{Spalten} &\to \texttt{"\{"}\ \mathit{Spalte}\ \{\,\mathit{Spalte}\,\}\ \texttt{"\}"}\ ,\\
\mathit{Spalte} &\to \texttt{"l"} \mid \texttt{"c"} \mid \texttt{"r"}\ ,\\
\mathit{Zeilen} &\to \mathit{Zeile}\ \{\ \texttt{"\textbackslash\textbackslash"}\ \mathit{Zeile}\ \}\ ,\\
\mathit{Zeile} &\to \mathit{Eintrag}\ \{\ \texttt{"\&"}\ \mathit{Eintrag}\ \}\ ,\\
\mathit{Eintrag} &\to \{\,\mathit{Tabelle} \mid \mathit{Zeichen}\,\}\ ,\\
\mathit{Zeichen} &\to \texttt{"0"} \mid \cdots \mid \texttt{"9"} \mid \texttt{"a"} \mid \cdots \mid \texttt{"Z"} \mid \texttt{"."} \mid \texttt{"␣"}\ \}
\end{aligned}
$$

$V = \{\mathit{Tabelle}, \mathit{Position}, \mathit{Spalten}, \mathit{Spalte}, \mathit{Zeilen}, \mathit{Zeile}, \mathit{Eintrag}, \mathit{Zeichen}\}$

$T = \{\texttt{"0"}, \ldots, \texttt{"9"}, \texttt{"a"}, \ldots, \texttt{"z"}, \texttt{"A"}, \ldots, \texttt{"Z"},$
$\texttt{"."}, \texttt{"␣"}, \texttt{"\{"}, \texttt{"\}"}, \texttt{"["}, \texttt{"]"}, \texttt{"\&"}, \texttt{"\textbackslash"}\}$

Nur eine Rekursion für Schachtelung der Tabellen notwendig!

Was Sie heute erwartet

N. Wirth: Programming in Modula-2

Appendix 1

The Syntax of Modula–2

```
ident = letter {letter | digit}.
number = integer | real.
integer = digit {digit} | octalDigit {octalDigit} ("B"|"C")|
    digit {hexDigit} "H".
real = digit {digit} "." {digit} [ScaleFactor].
ScaleFactor = "E" ["+"|"-"] digit {digit}.
hexDigit = digit |"A"|"B"|"C"|"D"|"E"|"F".
digit = octalDigit | "8"|"9".
octalDigit = "0"|"1"|"2"|"3"|"4"|"5"|"6"|"7".
string = "'" {character} "'" | '"' {character} '"' .
qualident = ident {"." ident}.
ConstantDeclaration = ident "=" ConstExpression.
ConstExpression = SimpleConstExpr [relation SimpleConstExpr].
relation = "=" | "#" | "<>" | "<" | "<=" | ">" | ">=" | IN .
SimpleConstExpr = ["+"|"-"] ConstTerm {AddOperator ConstTerm}.
AddOperator = "+" | "-" | OR .
ConstTerm = ConstFactor {MulOperator ConstFactor}.
MulOperator = "*" | "/" | DIV | MOD | AND | "&" .
ConstFactor = qualident | number | string | set |
    "(" ConstExpression ")" | NOT ConstFactor.
set = [qualident] "{" [element {"," element}] "}".
element = ConstExpression [".." ConstExpression].
TypeDeclaration = ident "=" type.
type = SimpleType | ArrayType | RecordType | SetType |
    PointerType | ProcedureType.
SimpleType = qualident | enumeration | SubrangeType.
enumeration = "(" IdentList ")".
IdentList = ident {"," ident}.
SubrangeType = "[" ConstExpression ".." ConstExpression "]".
ArrayType = ARRAY SimpleType {"," SimpleType} OF type.
RecordType = RECORD FieldListSequence END.
FieldListSequence = FieldList {";" FieldList}.
FieldList = [IdentList ":" type |
    CASE [ident ":"] qualident OF variant {"|" variant}
    [ELSE FieldListSequence] END].
variant = CaseLabelList ":" FieldListSequence.
CaseLabelList = CaseLabels {"," CaseLabels}.
CaseLabels = ConstExpression [".." ConstExpression].
SetType = SET OF SimpleType.
PointerType = POINTER TO type.
ProcedureType = PROCEDURE [FormalTypeList].
FormalTypeList = "(" [(VAR] FormalType
    {"," [VAR] FormalType)} ")" [":" qualident].
VariableDeclaration = IdentList ":" type.
```



```
designator = qualident {"." ident | "[" ExpList "]" | "↑"}.
ExpList = expression {"," expression}.
expression = SimpleExpression [relation SimpleExpression].
SimpleExpression = ["+"|"-"] term {AddOperator term}.
term = factor {MulOperator factor}.
factor = number | string | set | designator [ActualParameters] |
    "(" expression ")" | NOT factor.
ActualParameters = "(" [ExpList] ")" .
statement = [assignment | ProcedureCall |
    IfStatement | CaseStatement | WhileStatement |
    RepeatStatement | LoopStatement | ForStatement |
    WithStatement | EXIT | RETURN [expression] ].
assignment = designator ":=" expression.
ProcedureCall = designator [ActualParameters].
StatementSequence = statement {";" statement}.
IfStatement = IF expression THEN StatementSequence
    {ELSIF expression THEN StatementSequence}
    [ELSE StatementSequence] END.
CaseStatement = CASE expression OF case {"|" case}
    [ELSE StatementSequence] END.
case = CaseLabelList ":" StatementSequence.
WhileStatement = WHILE expression DO StatementSequence END.
RepeatStatement = REPEAT StatementSequence UNTIL expression.
ForStatement = FOR ident ":=" expression TO expression
    [BY ConstExpression] DO StatementSequence END.
LoopStatement = LOOP StatementSequence END.
WithStatement = WITH designator DO StatementSequence END .
ProcedureDeclaration = ProcedureHeading ";" block ident.
ProcedureHeading = PROCEDURE ident [FormalParameters].
block = {declaration} [BEGIN StatementSequence] END.
declaration = CONST {ConstantDeclaration ";"} |
    TYPE {TypeDeclaration ";"} |
    VAR {VariableDeclaration ";"} |
    ProcedureDeclaration ";" | ModuleDeclaration ";".
FormalParameters =
    "(" [FPSection {";" FPSection}] ")" [":" qualident].
FPSection = [VAR] IdentList ":" FormalType.
FormalType = [ARRAY OF] qualident.
ModuleDeclaration =
    MODULE ident [priority] ";" {import} [export] block ident.
priority = "[" ConstExpression "]".
export = EXPORT [QUALIFIED] IdentList ";".
import = [FROM ident] IMPORT IdentList ";".
DefinitionModule = DEFINITION MODULE ident ";" {import}
    [export] {definition} END ident ".".
definition = CONST {ConstantDeclaration ";"} |
    TYPE {ident ["=" type] ";"} |
    VAR {VariableDeclaration ";"} |
    ProcedureHeading ";" .
ProgramModule =
    MODULE ident [priority] ";" {import} block ident "." .
```

N. Wirth: Programming in Modula-2

Appendix 1

The Syntax of Modula-2

```
1  ident = letter {letter | digit}.
2  number = integer | real.
3  integer = digit {digit} | octalDigit {
4     digit {hexDigit} "H".
5  real = digit {digit} "." {digit} [Scale
6  ScaleFactor = "E" ["+"|"-"] digit {d
7  hexDigit = digit |"A"|"B"|"C"|"D"|
8  digit = octalDigit | "8"|"9".
9  octalDigit = "0"|"1"|"2"|"3"|"4"|"
10 string = "'" {character} "'" | '"' {cha
11 qualident = ident {"." ident}.
12 ConstantDeclaration = ident "=" Co
13 ConstExpression = SimpleConstExp
14 relation = "=" | "#" | "<>" | "<" | "
15 SimpleConstExpr = ["+"|"-"] Cons
16 AddOperator = "+" | "-" | OR .
17 ConstTerm = ConstFactor {MulOpe
18 MulOperator = "*" | "/" | DIV | M
19 ConstFactor = qualident | number
20    "(" ConstExpression ")" | NOT (
21 set = [qualident] "{" [element {","
22 element = ConstExpression [".." ConstExpression].
23 TypeDeclaration = ident "=" type.
24 type = SimpleType | ArrayType | RecordType | SetType |
25    PointerType | ProcedureType.
26 SimpleType = qualident | enumeration | SubrangeType.
27 enumeration = "(" IdentList ")".
28 IdentList = ident {"," ident}.
29 SubrangeType = "[" ConstExpression ".." ConstExpression "]".
30 ArrayType = ARRAY SimpleType {"," SimpleType} OF type.
31 RecordType = RECORD FieldListSequence END.
32 FieldListSequence = FieldList {";" FieldList}.
33 FieldList = [IdentList ":" type |
34    CASE [ident ":"] qualident OF variant {"|" variant}
35    [ELSE FieldListSequence] END].
36 variant = CaseLabelList ":" FieldListSequence.
37 CaseLabelList = CaseLabels {"," CaseLabels}.
38 CaseLabels = ConstExpression [".." ConstExpression].
39 SetType = SET OF SimpleType.
40 PointerType = POINTER TO type.
41 ProcedureType = PROCEDURE [FormalTypeList].
42 FormalTypeList = "(" [[VAR] FormalType
43    {"," [VAR] FormalType}] ")" [":" qualident].
44 VariableDeclaration = IdentList ":" type.
```

158 A1. The Syntax of Modula-2

```
1  ident = letter {letter | digit}.
2  number = integer | real.
3  integer = digit {digit} | octalDigit {octalDigit} ("B"|"C") |
4     digit {hexDigit} "H".
5  real = digit {digit} "." {digit} [ScaleFactor].
6  ScaleFactor = "E" ["+"|"-"] digit {digit}.
7  hexDigit = digit |"A"|"B"|"C"|"D"|"E"|"F".
8  digit = octalDigit | "8"|"9".
9  octalDigit = "0"|"1"|"2"|"3"|"4"|"5"|"6"|"7".
```

```
72 ProcedureDeclaration = ProcedureHeading ";" block ident.
73 ProcedureHeading = PROCEDURE ident [FormalParameters].
74 block = {declaration} [BEGIN StatementSequence] END.
75 declaration = CONST {ConstantDeclaration ";"} |
76    TYPE {TypeDeclaration ";"} |
77    VAR {VariableDeclaration ";"} |
78    ProcedureDeclaration ";" | ModuleDeclaration ";".
79 FormalParameters =
80    "(" [FPSection {";" FPSection}] ")" [":" qualident].
81 FPSection = [VAR] IdentList ":" FormalType.
82 FormalType = [ARRAY OF] qualident.
83 ModuleDeclaration =
84    MODULE ident [priority] ";" {import} [export] block ident.
85 priority = "[" ConstExpression "]".
86 export = EXPORT [QUALIFIED] IdentList ";".
87 import = [FROM ident] IMPORT IdentList ";".
88 DefinitionModule = DEFINITION MODULE ident ";" {import}
89    [export] {definition} END ident ".".
90 definition = CONST {ConstantDeclaration ";"} |
91    TYPE (ident ["=" type] ";") |
92    VAR {VariableDeclaration ";"} |
93    ProcedureHeading ";" .
94 ProgramModule =
95    MODULE ident [priority] ";" {import} block ident "." .
```

40

J. Gosling et al.: JAVA Language Specification (6.2.2012)

CHAPTER 18

Syntax

THIS chapter presents a grammar for the Java programming language.

The grammar presented piecemeal in the preceding chapters (§2.3) is much better for exposition, but it is not well suited as a basis for a parser. The grammar presented in this chapter is the basis for the reference implementation. Note that it is not an LL(1) grammar, though in many cases it minimizes the necessary look ahead.

The grammar below uses the following BNF-style conventions:

- *[x]* denotes zero or one occurrences of x.
- *{x}* denotes zero or more occurrences of x.
- *x* | *y* means one of either x or y.

Identifier:
  IDENTIFIER

QualifiedIdentifier:
  Identifier { `.` *Identifier* }

QualifiedIdentifierList:
  QualifiedIdentifier { `,` *QualifiedIdentifier* }

EnumBody:
  `{` *[EnumConstants] [,] [EnumBodyDeclarations]* `}`

EnumConstants:
  EnumConstant
  EnumConstants `,` *EnumConstant*

EnumConstant:
  [Annotations] Identifier [Arguments] [ClassBody]

EnumBodyDeclarations:
  `;` *{ClassBodyDeclaration}*

AnnotationTypeBody:
  `{` *[AnnotationTypeElementDeclarations]* `}`

AnnotationTypeElementDeclarations:
  AnnotationTypeElementDeclaration
  AnnotationTypeElementDeclarations AnnotationTypeElementDeclaration

AnnotationTypeElementDeclaration:
  {Modifier} AnnotationTypeElementRest

AnnotationTypeElementRest:
  Type Identifier AnnotationMethodOrConstantRest `;`
  ClassDeclaration
  InterfaceDeclaration
  EnumDeclaration
  AnnotationTypeDeclaration

AnnotationMethodOrConstantRest:
  AnnotationMethodRest
  ConstantDeclaratorsRest

AnnotationMethodRest:
  `()` *[`[]`]* *[*`default` *ElementValue]*

41

J. Gosling et al.: JAVA Language Specification (6.2.2012)

CHAPTER 18

The grammar below uses the following BNF-style conventions:

- *[x]* denotes zero or one occurrences of x.
- *{x}* denotes zero or more occurrences of x.
- *x | y* means one of either x or y.

THIS chapter presents a grammar fo

The grammar presented piecemeal in t
for exposition, but it is not well suited a
in this chapter is the basis for the refe
LL(1) grammar, though in many cases

The grammar below uses the followin

- *[x]* denotes zero or one occurrences
- *{x}* denotes zero or more occurrences of x.
- *x | y* means one of either x or y.

Identifier:
IDENTIFIER

QualifiedIdentifier:
Identifier { . *Identifier* }

QualifiedIdentifierList:
QualifiedIdentifier { , *QualifiedIdentifier* }

EnumBody:
{ *[EnumConstants] [,] [EnumBodyDeclarations]* }

EnumConstants:
EnumConstant

AnnotationTypeElementDeclarations AnnotationTypeElementDeclaration

AnnotationTypeElementDeclaration:
{Modifier} AnnotationTypeElementRest

AnnotationTypeElementRest:
Type Identifier AnnotationMethodOrConstantRest ;
ClassDeclaration
InterfaceDeclaration
EnumDeclaration
AnnotationTypeDeclaration

AnnotationMethodOrConstantRest:
AnnotationMethodRest
ConstantDeclaratorsRest

AnnotationMethodRest:
( ) *[*`[]`*]* *[*`default` *ElementValue]*

J. Gosling et al.: JAVA Language Specification (6.2.2012)

SYNTAX

CHAPTER 18

Syntax

THIS chapter presents a grammar for the Java programming language.

The grammar presented piecemeal in the preceding chapters (§2.3) is much better for exposition, but it is not well suited as a basis for a parser. The grammar presented in this chapter is the basis for the reference implementation. Note that it is not an LL(1) grammar, though in many cases it minimizes the necessary look ahead.

The grammar below uses the following BNF-style conventions:

- *[x]* denotes zero or one occurrences of x
- *{x}* denotes zero or more occurrence
- *x | y* means one of either x or y.

Identifier:
IDENTIFIER

QualifiedIdentifier:
Identifier { . *Identifier* }

QualifiedIdentifierList:
QualifiedIdentifier { , *Qualifiec*

QualifiedIdentifier:
Identifier { . *Identifier* }

QualifiedIdentifierList:
QualifiedIdentifier { , *QualifiedIdentifier* }

EnumBody:
{ *[EnumConstants] [,] [EnumBodyDeclarations]* }

EnumConstants:
EnumConstant
EnumConstants , *EnumConstant*

EnumConstant:
[Annotations] Identifier [Arguments] [ClassBody]

EnumBodyDeclarations:
; *{ClassBodyDeclaration}*

AnnotationTypeBody:
{ *[AnnotationTypeElementDeclarations]* }

AnnotationTypeElementDeclarations:

J. Gosling et al.: JAVA Language Specification (6.2.2012)

CHAPTER 18

Syntax

EnumBody:
{ *[EnumConstants] [,] [EnumBodyDeclarations]* }

EnumConstants:
EnumConstant
EnumConstants , *EnumConstant*

EnumConstant:
[Annotations] Identifier [Arguments] [ClassBody]

EnumConstants:
EnumConstant
EnumConstants , *EnumConstant*

EnumConstant:
[Annotations] Identifier [Arguments] [ClassBody]

EnumBodyDeclarations:
; *{ClassBodyDeclaration}*

eclarations] }

ations:
laration
larations AnnotationTypeElementDeclaration

ation:
lementRest

ethodOrConstantRest ;

Rest:

( ) *[*[ ]*] [*`default` *ElementValue]*

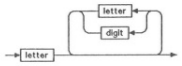
591 606

N. Wirth: Programming in Modula-2

Appendix 4

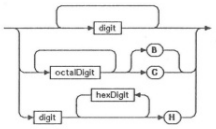
Syntax Diagrams

ident

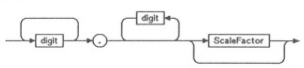


number

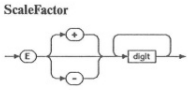
integer



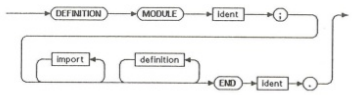
real



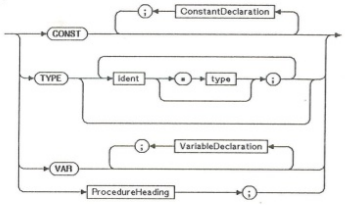
ScaleFactor



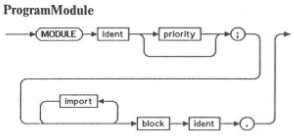
DefinitionModule



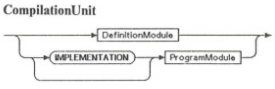
definition



ProgramModule



CompilationUnit

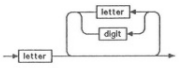


N. Wirth: Programming in Modula-2

Appendix 4

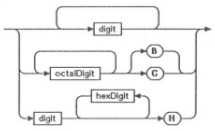
Syntax Diagrams

ident

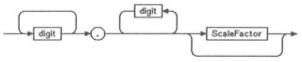


number

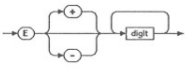
integer



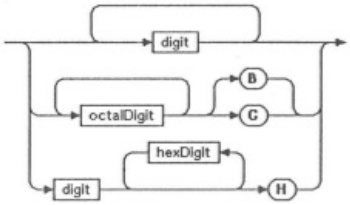
real



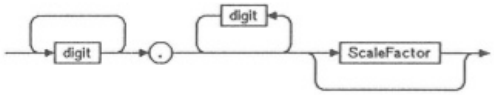
ScaleFactor



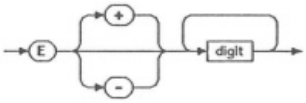
integer



real



ScaleFactor



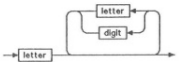
N. Wirth: Programming in Modula-2

180 A4. Syntax Diagrams

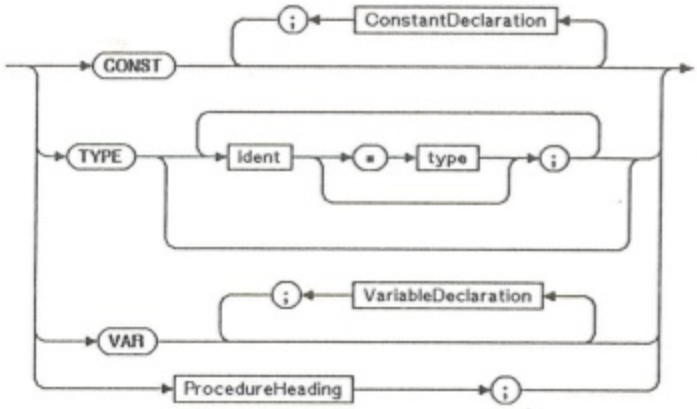
Appendix 4

Syntax Diagrams

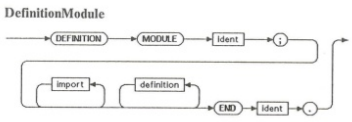
ident

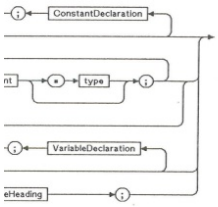


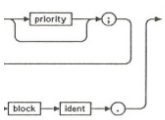
definition

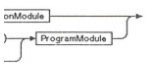


DefinitionModule









42

Was Sie heute erwartet

Jenseits der Kontextfreiheit

Formale Sprachen

$\{ a^n b^n c^n \mid n \geq 0 \}$ ist nicht kontextfrei.

Formale Sprachen

$\{ ww \mid w \in \{a, b, c\}^* \}$ ist nicht kontextfrei.

Programmiersprachen

Typen- und Deklarationsbedingungen nicht oder nur schwer kontextfrei darstellbar.

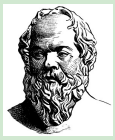
Natürliche Sprachen (Schweizer Dialekt)

Mer d'chind em Hans es huus
haend wele laa hälfe aastriiche.

Wir wollten die Kinder dem Hans
helfen lassen das Haus anzustreichen.

Was Sie heute erwartet

Sokrates aus Sicht der Aussagenlogik



Alle Menschen sind sterblich. A
Sokrates ist ein Mensch. B
Sokrates ist sterblich. C

$A, B \not\models C$

A	B	C
1	1	0

Clipart courtesy FCIT

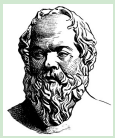
- Die Prämissen und die Konklusion sind für die Aussagenlogik atomar.
- Können nur durch (drei unabhängige) Variablen modelliert werden.
- Keine korrekte Inferenz!
- Keine Berücksichtigung der inneren Struktur der Aussagen:
  „alle Menschen“, „ist sterblich“, „ist Mensch“, . . .

Aussagenlogik zu ausdrucksschwach für eine adäquate Modellierung!

Prädikatenlogik: Erweiterung der Aussagenlogik um

- Quantoren: für alle ($\forall$), es gibt ($\exists$)
- Prädikatensymbole: $Mensch(x)$ . . . „x ist ein Mensch“
- Funktionsterme:
  $Mensch(mutter(sokrates))$ . . . „Sokrates Mutter ist ein Mensch“

Sokrates aus Sicht der Prädikatenlogik



Alle Menschen sind sterblich.	$\forall x\ (\mathit{Mensch}(x) \supset \mathit{Sterblich}(x))$
Sokrates ist ein Mensch.	$\mathit{Mensch}(\mathit{sokrates})$
Sokrates ist sterblich.	$\mathit{Sterblich}(\mathit{sokrates})$

Clipart courtesy FCIT

Warum ist das eine korrekte Inferenz?

$\forall x\ P(x) \models P(t)$ (Instanziierungsregel)

Wenn P für alle x gilt, dann gilt P für jedes spezielle Objekt t.

$\forall x(\mathit{Mensch}(x) \supset \mathit{Sterblich}(x)) \models \mathit{Mensch}(\mathit{sokrates}) \supset \mathit{Sterblich}(\mathit{sokrates})$

$A,\ A \supset B \models B$ (Modus ponens)

Wenn A gilt und wenn B aus A folgt, dann gilt auch B.

$\mathit{Mensch}(\mathit{sokrates}),\ \mathit{Mensch}(\mathit{sokrates}) \supset \mathit{Sterblich}(\mathit{sokrates}) \models \mathit{Sterblich}(\mathit{sokrates})$

$$\dfrac{\mathit{Mensch}(\mathit{sokrates}) \qquad \dfrac{\forall x(\mathit{Mensch}(x) \supset \mathit{Sterblich}(x))}{\mathit{Mensch}(\mathit{sokrates}) \supset \mathit{Sterblich}(\mathit{sokrates})}}{\mathit{Sterblich}(\mathit{sokrates})}$$

Funktions-, Prädikaten- und Variablensymbole

$\mathcal{F}$ ... Menge der Funktionssymbole; besitzen Stelligkeit (Arität)
$f/n \in \mathcal{F}$ ... f ist ein n-stelliges Funktionssymbol.
(f benötigt n Argumente.)
$f/0$ ... nullstelliges Funktionssymbol, Konstantensymbol

$\mathcal{P}$ ... Menge der Prädikatensymbole; besitzen Stelligkeit (Arität)
$P/n \in \mathcal{P}$ ... P ist ein n-stelliges Prädikatensymbol.
(P benötigt n Argumente.)
$P/0$ ... nullstelliges Prädikatensymbol, entspricht Aussagenvariable

$\mathcal{V} = \{x, y, z, x_0, x_1, \dots\}$ ... Individuenvariablensymbole

$\mathit{Mensch}(\mathit{mutter}(\mathit{sokrates})),\quad \mathit{Sterblich}(x)$

Funktionssymbole: $\mathit{mutter}/1$, $\mathit{sokrates}/0$
Prädikatensymbole: $\mathit{Mensch}/1$, $\mathit{Sterblich}/1$
Variablensymbol: x

Was Sie heute erwartet

Terme

- ... bestehen aus Variablen- und Funktionssymbolen.
- ... ermöglichen die Bildung von Ausdrücken wie $3 \cdot \sin(x + 2)$.
- ... sind eine allgemeine Repräsentationsform für hierarchische Strukturen.

Terme über $\mathcal{F}$ und $\mathcal{V}$

Die Menge $\mathcal{T}(\mathcal{F}, \mathcal{V})$, kurz $\mathcal{T}$, der Terme über $\mathcal{F}$ und $\mathcal{V}$ ist die kleinste Menge, für die gilt:

(t1) $\mathcal{V} \subseteq \mathcal{T}$ — Individuenvariablen sind Terme.

(t2) $f \in \mathcal{T}$, falls $f/0 \in \mathcal{F}$. — Konstantensymbole sind Terme.

(t3) $f(t_1, \ldots, t_n) \in \mathcal{T}$, falls $f/n \in \mathcal{F}$ und $t_1, \ldots, t_n \in \mathcal{T}$. — Funktionssymbole mit der passenden Zahl an Termargumenten sind Terme.

Notation:

- Pre-/Postfixnotation für manche unären ($n = 1$) Funktionssymbole
- Infixnotation für manche binären ($n = 2$) Funktionssymbole
- Klammerneinsparungen durch Prioritäten

Verschiedene Terme

$3 \cdot \sin(x+2)$ bzw. $\cdot(3, \sin(+(x,2)))$

$s(s(s(0)))$ Darstellung der Zahl $3 = 1 + (1 + (1 + 0))$

$f(a, f(b, f(c, nil)))$ Darstellung der Liste $[a, b, c]$

$g(f(a), g(a, (f(x))))$ ist ein Term, falls $a/0, f/1, g/2 \in \mathcal{F}$ und $x \in \mathcal{V}$.

$$\dfrac{g/2 \in \mathcal{F} \quad \dfrac{f/1 \in \mathcal{F} \quad \dfrac{a/0 \in \mathcal{F}}{a \in \mathcal{T}}\,\text{t2}}{f(a) \in \mathcal{T}}\,\text{t3} \quad \dfrac{g/2 \in \mathcal{F} \quad \dfrac{a/0 \in \mathcal{F}}{a \in \mathcal{T}}\,\text{t2} \quad \dfrac{f/1 \in \mathcal{F} \quad \dfrac{x \in \mathcal{V}}{x \in \mathcal{T}}\,\text{t1}}{f(x) \in \mathcal{T}}\,\text{t3}}{g(a, (f(x))) \in \mathcal{T}}\,\text{t3}}{g(f(a), g(a, (f(x)))) \in \mathcal{T}(\mathcal{F}, VS)}\,\text{t3}$$

Die Menge der Terme, $\mathcal{T}(\mathcal{F}, \mathcal{V})$, ist die kleinste Menge, sodass:

(t1) $v \in \mathcal{T}$, falls $v \in \mathcal{V}$.

(t2) $f \in \mathcal{T}$, falls $f/0 \in \mathcal{F}$.

(t3) $f(t_1, \ldots, t_n) \in \mathcal{T}$, falls $f/n \in \mathcal{F}$ und $t_1, \ldots, t_n \in \mathcal{T}$.

Aussagenlogische Formeln sind Terme.

Sei $\mathcal{V} = \{A, B, C, \ldots\}$ und $\mathcal{F} = \{\top/0, \bot/0, \neg/1, \wedge/2, \vee/2, \ldots\}$.

Dann ist $\mathcal{T}(\mathcal{F}, \mathcal{V})$ die Menge der aussagenlogischen Formeln (unter Verwendung der Infixnotation für binäre Operatoren).

$(A \wedge B) \supset C$ entspricht $\supset(\wedge(A, B), C)$.

$A \wedge (B \supset C)$ entspricht $\wedge(A, \supset(B, C))$.

Reguläre Ausdrücke über dem Alphabet Σ sind Terme.

Sei $\mathcal{V} = \{\}$ und $\mathcal{F} = \{\varepsilon/0, \emptyset/0, +/2, \cdot/2, {}^*/1\} \cup \{\, s/0 \mid s \in \Sigma \,\}$.

Dann ist $\mathcal{T}(\mathcal{F}, \mathcal{V})$ die Menge der regulären Ausdrücke über Σ.

($+$ und $\cdot$ werden infix notiert, $\cdot$ wird nicht angeschrieben, und $*$ ist ein Postfixoperator).

So ziemlich alles lässt sich als Term auffassen ...

Semantik von Termen

Variablensymbole: Wert abhängig von momentaner Wertebelegung

Konstanten- und Funktionssymbole:

- Vordefinierte Symbole: feste, unveränderliche Bedeutung (fix eingebaut in Termsemantik)
- Freie Symbole: Interpretation als Konstante bzw. Funktionen

Interpretation

Eine Interpretation I über einem Wertebereich $\mathcal{U}$ ordnet jedem Funktionssymbol aus $\mathcal{F}$ eine Funktion wie folgt zu:

$$I(f) \in \mathcal{U} \qquad \text{für } f/0 \in \mathcal{F}$$
$$I(f) : \mathcal{U}^n \mapsto \mathcal{U} \qquad \text{für } f/n \in \mathcal{F},\ n > 0$$

Arithmetische Ausdrücke (wie etwa `(x+1)*(x+1+1)`)

$\mathcal{F} = \{0, 1, +, -, *\}$

$\mathcal{U} = \mathbb{Z} = \{\ldots, -2, -1, 0, 1, 2, \ldots\}$
$I(0) = 0$
$I(1) = 1$

$I(+) = +$ (Addition)
$I(-) = -$ (Subtraktion)
$I(*) = \cdot$ (Multiplikation)

Aussagenlogik

$\mathcal{F} = \{\top/0, \bot/0, \neg/1, \wedge/2, \vee/2, \ldots\}$

$\mathcal{U} = \mathbb{B} = \{0, 1\}$

$I(\top) = 1$
$I(\bot) = 0$
$I(\neg) = \text{not}$
$I(\wedge) = \text{and}$
$\ldots$

Reguläre Ausdrücke

$\mathcal{F} = \{\varepsilon/0, \emptyset/0, +/2, \cdot/2, {}^*/1\} \cup \{\, s/0 \mid s \in \Sigma \,\}$

$\mathcal{U} = 2^{\Sigma^*}$

$I(\varepsilon) = \{\varepsilon\}$
$I(\emptyset) = \{\}$
$I(s) = \{s\}$
$I(+) = \cup$
$I(\cdot) = \cdot$
$I({}^*) = {}^*$

$\sigma : \mathcal{V} \mapsto \mathcal{U}$ ... Wertebelegung für die Variablensymbole

Semantik von Termen

Der Wert eines Terms in einer Interpretation I mit Variablenbelegung σ wird festgelegt durch die Funktion $\mathrm{val}_{I,\sigma}$, definiert als:

(v1) $\mathrm{val}_{I,\sigma}(v) = \sigma(v)$ für $v \in \mathcal{V}$;

(v2) $\mathrm{val}_{I,\sigma}(f) = I(f)$ für $f/0 \in \mathcal{F}$;

(v3) $\mathrm{val}_{I,\sigma}(f(t_1, \ldots, t_n)) = I(f)(\mathrm{val}_{I,\sigma}(t_1), \ldots, \mathrm{val}_{I,\sigma}(t_n))$ für $f/n \in \mathcal{F}$, $n > 0$.

Wert von `(x+1)*(x+1+1)` für $\sigma(x) = 0$

Arithmetik: $\mathcal{U} = \mathbb{Z}$, $I_1(1) = 1$, $I_1(+) = +$, $I_1(*) = \cdot$

$\mathrm{val}_{I_1,\sigma}(\texttt{(x+1)*(x+1+1)}) = (0 + 1) \cdot (0 + 1 + 1) = 2$

Aussagenlogik: $\mathcal{U} = \mathbb{B}$, $I_2(1) = 1$, $I_2(+) = \mathrm{or}$, $I_2(*) = \mathrm{and}$

$\mathrm{val}_{I_2,\sigma}(\texttt{(x+1)*(x+1+1)}) = \mathrm{and}\ (\mathrm{or}\ (0, 1), \mathrm{or}\ (0, \mathrm{or}\ (1, 1))) = 1$

3.0 VU Formale Modellierung

Gernot Salzer

Forschungsbereich Theory and Logic
Institut für Logic and Computation

15.5.2018

Was Sie letztes Mal hörten

1. Organisatorisches
2. Was bedeutet Modellierung?
3. Aussagenlogik
4. Endliche Automaten
5. Reguläre Sprachen
6. Kontextfreie Grammatiken
   - 6.1. Definition
   - 6.2. Beispiele
   - 6.3. Mehrdeutigkeit
   - 6.4. Andere Notationen
   - 6.5. Style Guide für Grammatiken
   - 6.6. Grammatiken in freier Wildbahn
   - 6.7. Jenseits der Kontextfreiheit
7. Prädikatenlogik

Kontextfreie Grammatiken

- Menge von Zeichenersetzungsregeln
- ausdrucksstärker als endliche Automaten und reguläre Ausdrücke
- können geschachtelte Strukturen darstellen

Wohlgeformte Klammerausdrücke

$G = \langle \{W\}, \quad \{(,)\}, \quad \{W \to \varepsilon \mid (\, W\,) \mid W\, W\}, \quad W \rangle$

Beispiel einer Ableitung:
$$\begin{aligned} W &\Rightarrow_P W\, W \\ &\Rightarrow_P (W)(W) \\ &\Rightarrow_P ()(W\, W) \\ &\Rightarrow_P ()((W)(W)) \\ &\Rightarrow_P ()(()()) \end{aligned}$$

$\mathcal{L}(G) = \{\varepsilon, \ (), \ (()), \ ()(), \ ((())), \ (())(), \ ()(()), \ ()()(), \ldots\}$

Kontextfreie Grammatik

... wird beschrieben durch ein 4-Tupel $G = \langle V, T, P, S \rangle$, wobei

- V ... Menge der Nonterminalsymbole (Variablen)
- T ... Menge der Terminalsymbole ($V \cap T = \{\}$)
- $P \subseteq V \times (V \cup T)^*$ ... Produktionen
- $S \in V$ ... Startsymbol

Schreibweise: $A \to w$ statt $(A, w) \in P$

$A \to w_1 \mid \cdots \mid w_n$ statt $A \to w_1, \ldots, A \to w_n$

Ableitbarkeit

... in einem Schritt:

$x\, A\, y \Rightarrow x\, w\, y$, falls $A \to w \in P$ und $x, y \in (V \cup T)^*$.

... in mehreren Schritten:

$u \overset{*}{\Rightarrow} v$, falls

- $u = v$, oder
- $u \Rightarrow u'$ und $u' \overset{*}{\Rightarrow} v$ für ein Wort $u' \in (V \cup T)^*$.

Von Grammatik G generierte Sprache

$\mathcal{L}(G) = \{ w \in T^* \mid S \overset{*}{\Rightarrow} w \}$

Grammatiken G_1 und G_2 heißen äquivalent, wenn $\mathcal{L}(G_1) = \mathcal{L}(G_2)$ gilt.

Grammatik für $\mathrm{a}^n\mathrm{b}^n$

$G = \langle \{S\}, \{\mathrm{a}, \mathrm{b}\}, \{S \to \varepsilon \mid \mathrm{a}\,S\,\mathrm{b}\}, S \rangle$

$\mathrm{aabb} \in \mathcal{L}(G)$, weil $S \overset{*}{\Rightarrow} \mathrm{aabb}$ gilt:
$S \Rightarrow \mathrm{a}\,S\,\mathrm{b} \Rightarrow \mathrm{aa}\,S\,\mathrm{bb} \Rightarrow \mathrm{aa}\,\varepsilon\,\mathrm{bb} \Rightarrow \mathrm{aabb}$

$\mathcal{L}(G) = \{ \mathrm{a}^n\mathrm{b}^n \mid n \geq 0 \} = \{\varepsilon, \mathrm{ab}\,\mathrm{aabb}\,\mathrm{aaabbb}, \ldots\}$

Kontextfreie Sprachen

Eine Sprache heißt kontextfrei, wenn es eine kontextfreie Grammatik gibt, die sie generiert.

Eingeschränkte Ableitungsrelationen

Linksableitungsrelation $\Rightarrow_L$: Nur die linkeste Variable wird ersetzt.

Rechtsableitungsrelation $\Rightarrow_R$: Nur die rechteste Variable wird ersetzt.

Parallelableitungsrelation $\Rightarrow_P$: Alle Variablen werden gleichzeitig ersetzt.

$G = \langle \{A, B, C\}, \{\mathrm{a}, \mathrm{b}, \mathrm{c}\}, \{A \to \mathrm{a},\ B \to \mathrm{b} \mid ABC,\ C \to \mathrm{c}\}, B \rangle$

abc besitzt 6 Ableitungen in G:

$B \Rightarrow ABC \Rightarrow \mathrm{a}BC \Rightarrow \mathrm{ab}C \Rightarrow \mathrm{abc}$
$B \Rightarrow ABC \Rightarrow \mathrm{a}BC \Rightarrow \mathrm{a}Bc \Rightarrow \mathrm{abc}$
$B \Rightarrow ABC \Rightarrow A\mathrm{b}C \Rightarrow \mathrm{ab}C \Rightarrow \mathrm{abc}$
$B \Rightarrow ABC \Rightarrow A\mathrm{b}C \Rightarrow A\mathrm{bc} \Rightarrow \mathrm{abc}$
$B \Rightarrow ABC \Rightarrow AB\mathrm{c} \Rightarrow \mathrm{a}B\mathrm{c} \Rightarrow \mathrm{abc}$
$B \Rightarrow ABC \Rightarrow AB\mathrm{c} \Rightarrow A\mathrm{bc} \Rightarrow \mathrm{abc}$

aber jeweils nur eine Links-, Rechts- und Parallelableitung:

$B \Rightarrow_L ABC \Rightarrow_L \mathrm{a}BC \Rightarrow_L \mathrm{ab}C \Rightarrow_L \mathrm{abc}$
$B \Rightarrow_R ABC \Rightarrow_R AB\mathrm{c} \Rightarrow_R A\mathrm{bc} \Rightarrow_R \mathrm{abc}$
$B \Rightarrow_P ABC \Rightarrow_P \mathrm{abc}$

$\mathcal{L}(G) = \{ w \in T^* \mid S \overset{*}{\Rightarrow} w \} = \{ w \in T^* \mid S \overset{*}{\Rightarrow}_L w \}$
$= \{ w \in T^* \mid S \overset{*}{\Rightarrow}_R w \} = \{ w \in T^* \mid S \overset{*}{\Rightarrow}_P w \}$

Was Sie letztes Mal hörten

Notationen für kontextfreie Produktionen

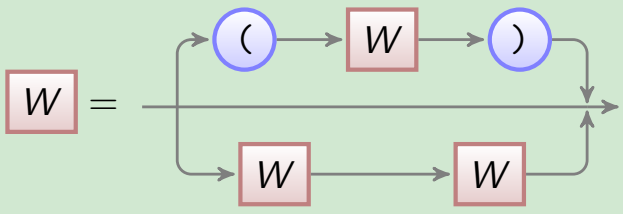
Backus-Naur-Form (BNF)

$\langle Real \rangle ::= \langle Digit \rangle \langle Digits \rangle \text{ " . " } \langle Digits \rangle \langle Scale \rangle$

Erweiterte Backus-Naur-Form (EBNF)

$IdList \rightarrow Id \{ \text{ " , " } Id \}$

Syntaxdiagramme



$$W \rightarrow (W) \mid \varepsilon \mid W\ W$$

Was Sie letztes Mal hörten

1. Organisatorisches
2. Was bedeutet Modellierung?
3. Aussagenlogik
4. Endliche Automaten
5. Reguläre Sprachen
6. Kontextfreie Grammatiken
   - 6.1. Definition
   - 6.2. Beispiele
   - 6.3. Andere Notationen
   - 6.4. Style Guide für Grammatiken
   - 6.5. Grammatiken in freier Wildbahn
   - 6.6. Jenseits der Kontextfreiheit
7. Prädikatenlogik

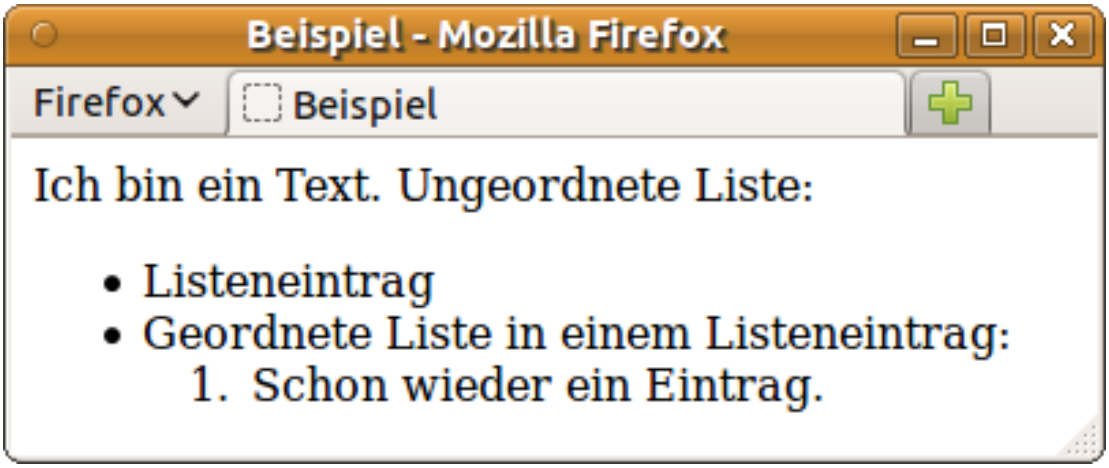
Style Guide für Grammatiken

- Wiederholungsoperator statt Rekursion
- Optionsoperator statt Leerwort
- Modularisierung statt Duplizierung
- Sprechende Namen
- Klare Kennzeichnung und Unterscheidung von Terminalen, Nonterminalen und Metanotationen
- Strukturierung statt minimale Anzahl von Nonterminalen und Regeln

Beispiel: einfache HTML-Dokumente

- Beginnen mit `<html>`, enden mit `</html>`.
  Dazwischen: Dokumentenkopf und -körper.
- Kopf: Dokumenttitel zwischen `<head>` und `</head>`.
  Titel: Text zwischen `<title>` und `</title>`.
- Körper beginnt mit `<body>`, endet mit `</body>`.
  Dazwischen: Texte, geordnete und ungeordnete Listen in beliebiger Reihenfolge und Zahl.
- Geordnete Liste: Listeneinträge zwischen `<ol>` und `</ol>`.
  Ungeordnete Liste: Listeneinträge zwischen `<ul>` und `</ul>`.
  Listeneintrag: Folge von Texten und Listen zwischen `<li>` und `</li>`.
- Text: Folge von Buchstaben, Ziffern, Leerzeichen, Sonderzeichen.


```
<html><head><title>Beispiel</title></head>
      <body>Ich bin ein Text. Ungeordnete Liste:
            <ul><li>Listeneintrag</li>
                <li>Geordnete Liste in einem Listeneintrag:
                    <ol><li>Schon wieder ein Eintrag.</li></ol>
                </li>
            </ul>
      </body>
</html>
```



Gesucht: Grammatik G in EBNF für einfache HTML-Dokumente

$G = \langle V, T, P, \mathit{Document} \rangle$, wobei:

$$
\begin{aligned}
P = \{\ & \mathit{Document} \to \texttt{"<html>"}\ \mathit{Head}\ \mathit{Body}\ \texttt{"</html>"},\\
& \mathit{Head} \to \texttt{"<head>"}\ \mathit{Title}\ \texttt{"</head>"},\\
& \mathit{Title} \to \texttt{"<title>"}\ \mathit{Text}\ \texttt{"</title>"},\\
& \mathit{Body} \to \texttt{"<body>"}\ \mathit{Contents}\ \texttt{"</body>"},\\
& \mathit{Contents} \to \{\ \mathit{Text} \mid \mathit{OList} \mid \mathit{UList}\ \},\\
& \mathit{OList} \to \texttt{"
"}\ \mathit{Item}\ \{\ \mathit{Item}\ \}\ \texttt{"
"},\\
& \mathit{UList} \to \texttt{"
"}\ \mathit{Item}\ \{\ \mathit{Item}\ \}\ \texttt{"
"},\\
& \mathit{Item} \to \texttt{"
"}\ \mathit{Contents}\ \texttt{"
"},\\
& \mathit{Text} \to \mathit{Char}\ \{\ \mathit{Char}\ \},\\
& \mathit{Char} \to \texttt{"a"} \mid \cdots \mid \texttt{"z"} \mid \texttt{"A"} \mid \cdots \mid \texttt{"Z"} \mid \texttt{"0"} \mid \cdots \mid \texttt{"9"} \mid\\
& \qquad \texttt{"␣"} \mid \texttt{","} \mid \texttt{";"} \mid \texttt{":"} \mid \texttt{"."} \mid \texttt{"!"} \mid \texttt{"?"}\ \}
\end{aligned}
$$

$V = \{\mathit{Document}, \mathit{Head}, \mathit{Title}, \mathit{Body}, \mathit{Contents}, \mathit{OList}, \mathit{UList}, \mathit{Item}, \mathit{Text}, \mathit{Char}\}$
$T = \{\cdots \text{alle Zeichen zwischen Anführungszeichen} \cdots\}$

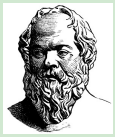
Nur eine Rekursion für die Schachtelung der Listen notwendig!

Was Sie letztes Mal hörten

1. Organisatorisches
2. Was bedeutet Modellierung?
3. Aussagenlogik
4. Endliche Automaten
5. Reguläre Sprachen
6. Kontextfreie Grammatiken
7. Prädikatenlogik
   - 7.1. Motivation
   - 7.2. Terme
   - 7.3. Syntax
   - 7.4. Semantik
   - 7.5. Modellierung

Sokrates

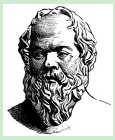
. . . in der Aussagenlogik



Alle Menschen sind sterblich.	A	$A, B \not\models C$
Sokrates ist ein Mensch.	B	1 1 0
Sokrates ist sterblich.	C	

Clipart courtesy FCIT

. . . in der Prädikatenlogik



Alle Menschen sind sterblich.	$\forall x\ (Mensch(x) \supset Sterblich(x))$
Sokrates ist ein Mensch.	$Mensch(sokrates)$
Sokrates ist sterblich.	$Sterblich(sokrates)$

Clipart courtesy FCIT

Prädikatenlogik erweitert die Aussagenlogik um

- Quantoren: für alle ($\forall$), es gibt ($\exists$)
- Prädikatensymbole: $Mensch(x)$ . . . „x ist ein Mensch“
- Funktionsterme:
  $Mensch(mutter(sokrates))$ . . . „Sokrates Mutter ist ein Mensch“

Was Sie letztes Mal hörten

Syntax der Terme

$\mathcal{V} = \{x, y, z, x_0, x_1, \dots\}$ ... Individuenvariablensymbole

$\mathcal{F}$ ... Menge der Funktionssymbole; besitzen Stelligkeit (Arität)
$f/n \in \mathcal{F}$ ... f ist ein n-stelliges Funktionssymbol.
(f benötigt n Argumente.)
$f/0$ ... nullstelliges Funktionssymbol, Konstantensymbol

Terme über $\mathcal{F}$ und $\mathcal{V}$

Die Menge $\mathcal{T}(\mathcal{F}, \mathcal{V})$, kurz $\mathcal{T}$, der Terme über $\mathcal{F}$ und $\mathcal{V}$ ist die kleinste Menge, für die gilt:

(t1) $\mathcal{V} \subseteq \mathcal{T}$ — Individuenvariablen sind Terme.

(t2) $f \in \mathcal{T}$, falls $f/0 \in \mathcal{F}$. — Konstantensymbole sind Terme.

(t3) $f(t_1, \dots, t_n) \in \mathcal{T}$, falls $f/n \in \mathcal{F}$ und $t_1, \dots, t_n \in \mathcal{T}$.
Funktionssymbole mit der passenden Zahl an Termargumenten sind Terme.

Pre-/In-/Postfix-Schreibweise für unäre bzw. binäre Funktionssymbole
Vereinbarung von Prioritäten zur Klammervereinfachung

Verschiedene Terme

$3 \cdot \sin(x + 2)$ bzw. $\cdot(3, \sin(+(x, 2)))$

$s(s(s(0)))$ Darstellung der Zahl $3 = 1 + (1 + (1 + 0))$

$f(a, f(b, f(c, nil)))$ Darstellung der Liste $[a, b, c]$

$g(f(a), g(a, (f(x))))$ Term über $\mathcal{F} = \{a/0, f/1, g/2\}$ and $\mathcal{V} = \{x\}$

Aussagenlogische Formeln

... sind Terme über $\mathcal{F} = \{\top/0, \bot/0, \neg/1, \wedge/2, \vee/2, \dots\}$ und $\mathcal{V} = \{A, B, C, \dots\}$.

Reguläre Ausdrücke über dem Alphabet Σ

... sind Terme über $\mathcal{F} = \{\varepsilon/0, \emptyset/0, +/2, \cdot/2, {}^*/1\} \cup \{\, s/0 \mid s \in \Sigma \,\}$ und $\mathcal{V} = \{\}$.

Semantik von Termen

$\sigma\colon \mathcal{V} \mapsto \mathcal{U}$ ... Wertebelegung für die Variablensymbole

Interpretation

Eine Interpretation I über einem Wertebereich $\mathcal{U}$ ordnet jedem Funktionssymbol aus $\mathcal{F}$ eine Funktion wie folgt zu:

$$I(f) \in \mathcal{U} \qquad \text{für } f/0 \in \mathcal{F}$$
$$I(f)\colon \mathcal{U}^n \mapsto \mathcal{U} \quad \text{für } f/n \in \mathcal{F},\ n > 0$$

Semantik von Termen

Der Wert eines Terms in einer Interpretation I mit Variablenbelegung σ wird festgelegt durch die Funktion $\mathrm{val}_{I,\sigma}$, definiert als:

(v1) $\mathrm{val}_{I,\sigma}(v) = \sigma(v)$ für $v \in \mathcal{V}$;

(v2) $\mathrm{val}_{I,\sigma}(f) = I(f)$ für $f/0 \in \mathcal{F}$;

(v3) $\mathrm{val}_{I,\sigma}(f(t_1, \ldots, t_n)) = I(f)(\mathrm{val}_{I,\sigma}(t_1), \ldots, \mathrm{val}_{I,\sigma}(t_n))$ für $f/n \in \mathcal{F}$, $n > 0$.

Wert von `(x+1)*(x+1+1)` für $\sigma(x) = 0$

Arithmetik: $\mathcal{U} = \mathbb{Z}$, $I_1(1) = 1$, $I_1(+) = +$, $I_1(*) = \cdot$

$\mathrm{val}_{I_1,\sigma}(\texttt{(x+1)*(x+1+1)}) = (0+1) \cdot (0+1+1) = 2$

Aussagenlogik: $\mathcal{U} = \mathbb{B}$, $I_2(1) = 1$, $I_2(+) = \mathrm{or}$, $I_2(*) = \mathrm{and}$

$\mathrm{val}_{I_2,\sigma}(\texttt{(x+1)*(x+1+1)}) = \mathrm{and}\ (\mathrm{or}\ (0,1), \mathrm{or}\ (0, \mathrm{or}\ (1,1))) = 1$

Was Sie heute erwartet

Prädikatenlogik – Syntax

$\mathcal{V}$ ... Individuenvariablensymbole
$\mathcal{F}$, $\mathcal{P}$ ... Funktions- bzw. Prädikatensymbole mit Stelligkeiten
$(\mathcal{F}, \mathcal{P})$ ... „Signatur“

Syntax prädikatenlogischer Formeln

Die Menge $\mathcal{PF}$ der prädikatenlogischen Formeln über der Signatur $(\mathcal{F}, \mathcal{P})$ ist die kleinste Menge, für die gilt:

(p1) $P \in \mathcal{PF}$, wenn $P/0 \in \mathcal{P}$;
(p2) $P(t_1, \ldots, t_n) \in \mathcal{PF}$, wenn $P/n \in \mathcal{P}$ und $t_1, \ldots, t_n \in \mathcal{T}(\mathcal{F}, \mathcal{V})$;
(p3) $\top, \bot \in \mathcal{PF}$;
(p4) $\neg F \in \mathcal{PF}$, wenn $F \in \mathcal{PF}$;
(p5) $(F * G) \in \mathcal{PF}$, wenn $F, G \in \mathcal{PF}$ und $* \in \{\wedge, \uparrow, \vee, \downarrow, \equiv, \not\equiv, \supset, \subset\}$.
(p6) $\forall x\, F \in \mathcal{PF}$, wenn $x \in \mathcal{V}$ und $F \in \mathcal{PF}$.
(p7) $\exists x\, F \in \mathcal{PF}$, wenn $x \in \mathcal{V}$ und $F \in \mathcal{PF}$.

P, $P(t_1, \ldots, t_n)$ ... „Atomformeln“

$\forall x\ (Mensch(x) \supset Sterblich(x))$ ist eine Formel

$Mensch/1$ und $Sterblich/1$ sind Prädikatensymbole.

$$\dfrac{\dfrac{Mensch/1 \in \mathcal{P} \quad \dfrac{x \in \mathcal{V}}{x \in \mathcal{T}}\ t1}{Mensch(x) \in \mathcal{PF}}\ p2 \qquad \dfrac{\dfrac{Sterblich/1 \in \mathcal{P} \quad \dfrac{x \in \mathcal{V}}{x \in \mathcal{T}}\ t1}{Sterblich(x) \in \mathcal{PF}}\ p2}{(Mensch(x) \supset Sterblich(x)) \in \mathcal{PF}}\ p5 \quad x \in \mathcal{V}}{\forall x\ (Mensch(x) \supset Sterblich(x)) \in \mathcal{PF}}\ p6$$

Was Sie heute erwartet

Prädikatenlogik – Semantik

Prädikatensymbole: repräsentieren Relationen bzw. Elementaraussagen

- Vordefinierte Symbole wie $\top$ und $\bot$: feste, unveränderliche Bedeutung (fix eingebaut in Formelsemantik)
- Freie Symbole: Interpretation durch Relationen bzw. Wahrheitswerte

(Prädikatenlogische) Interpretation

Eine Interpretation I über einem Wertebereich $\mathcal{U}$ ordnet jedem Symbol der Signatur $(\mathcal{F}, \mathcal{P})$ eine Funktion bzw. Relation wie folgt zu:

$I(f) \in \mathcal{U}$	für $f/0 \in \mathcal{F}$
$I(f) \colon \mathcal{U}^n \mapsto \mathcal{U}$	für $f/n \in \mathcal{F}$, $n > 0$
$I(P) \in \{0, 1\}$	für $P/0 \in \mathcal{P}$
$I(P) \subseteq \mathcal{U}^n$	für $P/n \in \mathcal{P}$, $n > 0$
$I(P) \colon \mathcal{U}^n \mapsto \{0, 1\}$	(alternative Sichtweise)

Nullstellige Prädikatensymbole = Aussagenvariablen

Zwei Interpretationsmöglichkeiten für $P/0 \in \mathcal{P}$: $I(P) = 0$ oder $I(P) = 1$.

Einstellige Prädikatensymbole = Typen, Klassen

Sei $Mann/1, Frau/1 \in \mathcal{P}$ und $\mathcal{U} = \{\text{Andrea}, \text{Anna}, \text{Hans}, \text{Maria}, \text{Tom}\}$.

$I(Mann)$ und $I(Frau)$ als Mengen:

$$I(Mann) = \{\text{Andrea}, \text{Hans}, \text{Tom}\}$$
$$I(Frau) = \{\text{Andrea}, \text{Anna}, \text{Maria}\}$$

... oder als Funktionen:

x	$I(Mann)(x)$	$I(Frau)(x)$
Andrea	1	1
Anna	0	1
Hans	1	0
Maria	0	1
Tom	1	0

Mehrstellige Prädikatensymbole = Relationen, Beziehungen

Sei $</2 \in \mathcal{P}$ und $\mathcal{U} = \mathbb{N}$. $I(<)$ als Menge:

$$I(<) = \{\, (m,n) \mid m < n \,\} = \{(0,1), (0,2), (1,2), (0,3), (1,3), \ldots\}$$

... oder als Funktion:

$$I(<)(m,n) = \begin{cases} 1 & \text{falls } m < n \\ 0 & \text{sonst} \end{cases}$$

Sei $Plus/3 \in \mathcal{P}$ und $\mathcal{U} = \mathbb{N}$. $I(Plus)$ als Menge:

$$\begin{aligned} I(Plus) &= \{\, (k,l,m) \mid k + l = m \,\} \\ &= \{(0,0,0), (1,0,1), (0,1,1), (2,0,2)\,(1,1,2)\,(0,2,2), \ldots\} \end{aligned}$$

... oder als Funktion:

$$I(Plus)(k,l,m) = \begin{cases} 1 & \text{falls } k + l = m \\ 0 & \text{sonst} \end{cases}$$

Semantik prädikatenlogischer Formeln

Der Wert einer Formel in einer Interpretation I mit Variablenbelegung σ wird festgelegt durch die Funktion $\mathrm{val}_{I,\sigma}$, definiert als:

(v1) $\mathrm{val}_{I,\sigma}(P) = I(P)$ für $P/0 \in \mathcal{P}$;

(v2) $\mathrm{val}_{I,\sigma}(P(t_1, \ldots, t_n)) = I(P)(\mathrm{val}_{I,\sigma}(t_1), \ldots, \mathrm{val}_{I,\sigma}(t_n))$ für $P/n \in \mathcal{P}$;

(v3) $\mathrm{val}_{I,\sigma}(\top) = 1$ und $\mathrm{val}_{I,\sigma}(\bot) = 0$;

(v4) $\mathrm{val}_{I,\sigma}(\neg F) = \mathrm{not}\, \mathrm{val}_{I,\sigma}(F)$;

(v5) $\mathrm{val}_{I,\sigma}(\,(F * G)\,) = \mathrm{val}_{I,\sigma}(F) \circledast \mathrm{val}_{I,\sigma}(G)$,
wobei $\circledast$ die logische Funktion zum Operator $*$ ist;

(v6) $$\mathrm{val}_{I,\sigma}(\,\forall x\, F\,) = \begin{cases} 1 & \text{falls } \mathrm{val}_{I,\sigma'}(F) = 1 \text{ für alle } \sigma' \overset{x}{\sim} \sigma \\ 0 & \text{sonst} \end{cases}$$

(v7) $$\mathrm{val}_{I,\sigma}(\,\exists x\, F\,) = \begin{cases} 1 & \text{falls } \mathrm{val}_{I,\sigma'}(F) = 1 \text{ für mind. ein } \sigma' \overset{x}{\sim} \sigma \\ 0 & \text{sonst} \end{cases}$$

$\sigma \overset{x}{\sim} \sigma'$ ... $\sigma(v) = \sigma'(v)$ für alle $v \in \mathcal{V}$ mit $v \neq x$

(σ und σ' sind identisch, nur $\sigma(x)$ und $\sigma'(x)$ können verschieden sein.)

$\forall x\, \exists y\, P(x, s(y))$ ist wahr

... falls wir $\mathcal{U} = \mathbb{N}$, $I(s)(n) := n - 1$ und $I(P)(m, n) := (m = n)$ wählen (Variablenbelegungen σ beliebig):

$\mathrm{val}_{I,\sigma}(\forall x\, \exists y\, P(x, s(y))) = 1$

$\iff \mathrm{val}_{I,\sigma'}(\exists y\, P(x, s(y))) = 1 \quad$ für alle $\sigma' \overset{x}{\sim} \sigma$

$\iff \mathrm{val}_{I,\sigma''}(P(x, s(y))) = 1 \quad$ für mind. ein $\sigma'' \overset{y}{\sim} \sigma'$ und alle $\sigma' \overset{x}{\sim} \sigma$

$\iff I(P)(\mathrm{val}_{I,\sigma''}(x), \mathrm{val}_{I,\sigma''}(s(y))) = 1 \quad$ für mind. ein $\sigma'' \overset{y}{\sim} \sigma'$ und alle $\sigma' \overset{x}{\sim} \sigma$

$\iff \sigma''(x) = \sigma''(y) - 1 \quad$ für mind. ein $\sigma'' \overset{y}{\sim} \sigma'$ und alle $\sigma' \overset{x}{\sim} \sigma$

Wir wählen jene Variablenbelegung σ'', für die $\sigma''(y) = \sigma''(x) + 1$ gilt:

$\sigma''(x) = (\sigma''(x) + 1) - 1 \quad$ für alle $\sigma''(x) = \sigma'(x) \in \mathbb{N}$

Gilt, daher ist $\forall x\, \exists y\, P(x, s(y))$ wahr in dieser Interpretation.

$\forall x\, \exists y\, P(x, s(y))$ ist falsch

... falls wir $\mathcal{U} = \mathbb{N}$, $I(s)(n) := n + 1$ und $I(P)(m, n) := (m = n)$ wählen (Variablenbelegungen σ beliebig):

$\mathrm{val}_{I,\sigma}(\forall x\, \exists y\, P(x, s(y))) = 1$

$\iff \mathrm{val}_{I,\sigma'}(\exists y\, P(x, s(y))) = 1 \quad$ für alle $\sigma' \overset{x}{\sim} \sigma$

$\iff \mathrm{val}_{I,\sigma''}(P(x, s(y))) = 1 \quad$ für mind. ein $\sigma'' \overset{y}{\sim} \sigma'$ und alle $\sigma' \overset{x}{\sim} \sigma$

$\iff I(P)(\mathrm{val}_{I,\sigma''}(x), \mathrm{val}_{I,\sigma''}(s(y))) = 1 \quad$ für mind. ein $\sigma'' \overset{y}{\sim} \sigma'$ und alle $\sigma' \overset{x}{\sim} \sigma$

$\iff \sigma''(x) = \sigma''(y) + 1 \quad$ für mind. ein $\sigma'' \overset{y}{\sim} \sigma'$ und alle $\sigma' \overset{x}{\sim} \sigma$

Problem: Für $\sigma''(x) = \sigma'(x) = 0$ besitzt die Gleichung keine Lösung in $\mathbb{N}$:

$$\sigma''(x) \neq \sigma''(y) + 1 \quad \text{für alle } \sigma'' \overset{y}{\sim} \sigma'$$

$\forall x\, \exists y\, P(x, s(y))$ ist daher falsch in dieser Interpretation.

Auf dem Spielplatz

$$\begin{aligned}
\mathcal{P} &= \{\mathit{Vater}/1, \mathit{Kind}/1, \mathit{SpieltMit}/2\} \\
\mathcal{F} &= \{\mathit{albrecht}/0, \mathit{frieda}/0\} \\
\\
\mathcal{U} &= \{\text{Albrecht}, \text{Bogdan}, \text{Erich}, \text{Frieda}, \text{Kathrin}, \text{Nina}, \text{Tamara}\} \\
I(\mathit{Vater}) &= \{\text{Bogdan}, \text{Erich}\} \\
I(\mathit{Kind}) &= \{\text{Albrecht}, \text{Frieda}, \text{Nina}\} \\
I(\mathit{SpieltMit}) &= \{(\text{Albrecht}, \text{Frieda}), (\text{Bogdan}, \text{Albrecht}), \\
&\quad (\text{Erich}, \text{Albrecht}), (\text{Erich}, \text{Frieda}), \\
&\quad (\text{Frieda}, \text{Nina}), (\text{Frieda}, \text{Kathrin}), (\text{Frieda}, \text{Albrecht}), (\text{Nina}, \text{Frieda}), \\
&\quad (\text{Tamara}, \text{Erich}), (\text{Tamara}, \text{Bogdan}), (\text{Tamara}, \text{Albrecht})\}, \\
I(\mathit{albrecht}) &= \text{Albrecht} \\
I(\mathit{frieda}) &= \text{Frieda}
\end{aligned}$$

$\forall x\,(\mathit{Vater}(x) \supset \exists y\,(\mathit{Kind}(y) \wedge \mathit{SpieltMit}(x, y)))$

„Alle Väter spielen mit (mindestens) einem Kind.“

Wahr in I, da Vater Bogdan mit Kind Albrecht und Vater Erich mit Kind Frieda spielt.

Auf dem Spielplatz

$$\begin{aligned}
\mathcal{P} &= \{\mathit{Vater}/1, \mathit{Kind}/1, \mathit{SpieltMit}/2\} \\
\mathcal{F} &= \{\mathit{albrecht}/0, \mathit{frieda}/0\} \\
\\
\mathcal{U} &= \{\text{Albrecht}, \text{Bogdan}, \text{Erich}, \text{Frieda}, \text{Kathrin}, \text{Nina}, \text{Tamara}\} \\
I(\mathit{Vater}) &= \{\text{Bogdan}, \text{Erich}\} \\
I(\mathit{Kind}) &= \{\text{Albrecht}, \text{Frieda}, \text{Nina}\} \\
I(\mathit{SpieltMit}) &= \{(\text{Albrecht}, \text{Frieda}), (\text{Bogdan}, \text{Albrecht}), \\
&\quad (\text{Erich}, \text{Albrecht}), (\text{Erich}, \text{Frieda}), \\
&\quad (\text{Frieda}, \text{Nina}), (\text{Frieda}, \text{Kathrin}), (\text{Frieda}, \text{Albrecht}), (\text{Nina}, \text{Frieda}), \\
&\quad (\text{Tamara}, \text{Erich}), (\text{Tamara}, \text{Bogdan}), (\text{Tamara}, \text{Albrecht})\}, \\
I(\mathit{albrecht}) &= \text{Albrecht} \\
I(\mathit{frieda}) &= \text{Frieda}
\end{aligned}$$

$\forall x\,(\mathit{Vater}(x) \wedge \mathit{SpieltMit}(x, \mathit{albrecht}))$

„Alle sind Väter und spielen mit Albrecht.“

Falsch in I, da etwa Albrecht kein Vater ist.

Auf dem Spielplatz

$$\begin{aligned}
\mathcal{P} &= \{\mathit{Vater}/1, \mathit{Kind}/1, \mathit{SpieltMit}/2\} \\
\mathcal{F} &= \{\mathit{albrecht}/0, \mathit{frieda}/0\} \\
\\
\mathcal{U} &= \{\text{Albrecht}, \text{Bogdan}, \text{Erich}, \text{Frieda}, \text{Kathrin}, \text{Nina}, \text{Tamara}\} \\
I(\mathit{Vater}) &= \{\text{Bogdan}, \text{Erich}\} \\
I(\mathit{Kind}) &= \{\text{Albrecht}, \text{Frieda}, \text{Nina}\} \\
I(\mathit{SpieltMit}) &= \{(\text{Albrecht}, \text{Frieda}), (\text{Bogdan}, \text{Albrecht}), \\
&\quad (\text{Erich}, \text{Albrecht}), (\text{Erich}, \text{Frieda}), \\
&\quad (\text{Frieda}, \text{Nina}), (\text{Frieda}, \text{Kathrin}), (\text{Frieda}, \text{Albrecht}), (\text{Nina}, \text{Frieda}), \\
&\quad (\text{Tamara}, \text{Erich}), (\text{Tamara}, \text{Bogdan}), (\text{Tamara}, \text{Albrecht})\}, \\
I(\mathit{albrecht}) &= \text{Albrecht} \\
I(\mathit{frieda}) &= \text{Frieda}
\end{aligned}$$

$\exists x\,(\mathit{Kind}(x) \wedge \forall y\,(\mathit{Kind}(y) \supset \mathit{SpieltMit}(x, y)))$

„Es gibt (mindestens) ein Kind, das mit allen Kindern spielt.“

Falsch in I, da Albrecht nicht mit Nina, Frieda nicht mit Frieda und Nina nicht mit Albrecht spielt.

Auf dem Spielplatz

$$\begin{aligned}
\mathcal{P} &= \{\mathit{Vater}/1, \mathit{Kind}/1, \mathit{SpieltMit}/2\} \\
\mathcal{F} &= \{\mathit{albrecht}/0, \mathit{frieda}/0\} \\
\\
\mathcal{U} &= \{\text{Albrecht}, \text{Bogdan}, \text{Erich}, \text{Frieda}, \text{Kathrin}, \text{Nina}, \text{Tamara}\} \\
I(\mathit{Vater}) &= \{\text{Bogdan}, \text{Erich}\} \\
I(\mathit{Kind}) &= \{\text{Albrecht}, \text{Frieda}, \text{Nina}\} \\
I(\mathit{SpieltMit}) &= \{(\text{Albrecht}, \text{Frieda}), (\text{Bogdan}, \text{Albrecht}), \\
&\quad (\text{Erich}, \text{Albrecht}), (\text{Erich}, \text{Frieda}), \\
&\quad (\text{Frieda}, \text{Nina}), (\text{Frieda}, \text{Kathrin}), (\text{Frieda}, \text{Albrecht}), (\text{Nina}, \text{Frieda}), \\
&\quad (\text{Tamara}, \text{Erich}), (\text{Tamara}, \text{Bogdan}), (\text{Tamara}, \text{Albrecht})\}, \\
I(\mathit{albrecht}) &= \text{Albrecht} \\
I(\mathit{frieda}) &= \text{Frieda}
\end{aligned}$$

$\forall x\,(\mathit{SpieltMit}(x, \mathit{frieda}) \not\equiv \mathit{SpieltMit}(x, \mathit{albrecht}))$

„Alle spielen entweder mit Frieda oder Albrecht (aber nicht mit beiden).“

Falsch in I, da Erich sowohl mit Frieda als auch mit Albrecht spielt.

Eine Formel F heißt

- erfüllbar, wenn $\mathrm{val}_{I,\sigma}(F) = 1$ für mindestens ein I und ein σ (I und σ nennt man Modell von F);
- widerlegbar, wenn $\mathrm{val}_{I,\sigma}(F) = 0$ für mindestens ein I und ein σ (I und σ nennt man Gegenbeispiel oder Gegenmodell zu F);
- unerfüllbar, wenn $\mathrm{val}_{I,\sigma}(F) = 0$ für alle I und alle σ;
- (allgemein)gültig, wenn $\mathrm{val}_{I,\sigma}(F) = 1$ für alle I und alle σ (F nennt man in diesem Fall auch Tautologie).

Äquivalenz, Konsequenz und Gültigkeit

Semantische Äquivalenz

Zwei Formeln F und G heißen äquivalent, geschrieben $F = G$, wenn $\mathrm{val}_{I,\sigma}(F) = \mathrm{val}_{I,\sigma}(G)$ für alle I und alle σ gilt.

$F_1, \ldots, F_n \models_{I,\sigma} G$:
„Aus $\mathrm{val}_{I,\sigma}(F_1) = \cdots = \mathrm{val}_{I,\sigma}(F_n) = 1$ folgt $\mathrm{val}_{I,\sigma}(G) = 1$.“

Logische Konsequenz

$F_1, \ldots, F_n \models G$: $F_1, \ldots, F_n \models_{I,\sigma} G$ gilt für alle I und σ.

Die Formeln F und G sind äquivalent ($F = G$) genau dann, wenn $F \equiv G$ eine gültige Formel ist.

$F_1, \ldots, F_n \models G$ genau dann, wenn $(F_1 \wedge \cdots \wedge F_n) \supset G$ gültig.

Wichtige Äquivalenzen

Neben den aussagenlogischen Äquivalenzen gelten auch noch folgende:

$\neg\forall x\, F = \exists x\, \neg F$ $\neg\exists x\, F = \forall x\, \neg F$	Dualität von $\forall$ und $\exists$
$\forall x\, F[x] = \forall y\, F[y]$ $\exists x\, F[x] = \exists y\, F[y]$	Umbenennung gebundener Variablen (sofern y nicht bereits in F vorkommt)
$\forall x\, \forall y\, F = \forall y\, \forall x\, F$ $\exists x\, \exists y\, F = \exists y\, \exists x\, F$	Vertauschung gleichartiger Quantoren
$\forall x\, (F \wedge G) = \forall x\, F \wedge \forall x\, G$	Distributivität $\forall/\wedge$
$\exists x\, (F \vee G) = \exists x\, F \vee \exists x\, G$	Distributivität $\exists/\vee$

Falls x nicht frei in F vorkommt, gilt außerdem:

$\forall x\, F = F$ $\quad$ $\exists x\, F = F$	
$\forall x\, (F \vee G) = F \vee \forall x\, G$	Distributivität $\forall/\vee$
$\exists x\, (F \wedge G) = F \wedge \exists x\, G$	Distributivität $\exists/\wedge$

Das Lügner-Paradoxon

Ich behaupte: „Ich lüge!“ – Sage ich die Wahrheit?

Wenn ich die Wahrheit sage, ist “Ich lüge“ richtig,
d.h., ich sage nicht die Wahrheit. – Widerspruch!

Wenn ich nicht die Wahrheit sage, ist „Ich lüge“ falsch,
d.h., ich sage die Wahrheit. – Widerspruch!

Das Kreter-Halbparadoxon

Der Kreter Epimenides behauptet: „Alle Kreter lügen!“.
Sagt Epimenides die Wahrheit?

Wenn ja, lügen alle Kreter, also auch er. – Widerspruch!

Wenn nein, ist es falsch, dass alle Kreter lügen.
Trugschluss: „Alle Kreter sagen die Wahrheit!“ (Also auch Epimenides?)
Richtig: Es gibt mindestens einen aufrichtigen Kreter. Kein Widerspruch.

Falsch: $\neg\forall = \forall\neg$ $\qquad$ Richtig: $\neg\forall = \exists\neg$

Das Gültigkeitsproblem der Prädikatenlogik

Gültigkeitsproblem

Gegeben: prädikatenlogische Formel F
Frage: Ist F gültig, d.h., gilt $\text{val}_{I,\sigma}(F) = 1$ für alle I und σ?

Viele praktische Aufgaben lassen sich als Probleme der Prädikatenlogik formulieren, wie z.B.

- Verifikation von Software
- Wissensrepräsentation

Die meisten prädikatenlogischen Fragen lassen sich als Gültigkeitsproblem formulieren.

Problem: Das Gültigkeitsproblem der Prädikatenlogik ist unentscheidbar.

Das bedeutet:

- Ist die Formel gültig, lässt sich mit verschiedenen Methoden ein Beweis finden (kann allerdings beliebig lange dauern).
- Ist die Formel nicht gültig, lässt sich das mit keiner Methode zuverlässig feststellen. Beweiser terminieren in diesem Fall oft nicht.

Was Sie heute erwartet

Wahl der Prädikatenstelligkeit

Max liest Zeitung.

Nullstelliges Prädikat (Aussagenvariable):

$Max\_liest\_Zeitung$

Einstelliges Prädikat:

$Liest\_Zeitung(max)$

Zweistelliges Prädikat:

$Liest(max, zeitung)$

Die beste Wahl hängt von den Umständen ab.

Wähle ein eigenes Symbol für Satzteile, die öfter auftreten (können).

Zeit-, Eigenschafts- und Hauptwörter

Sokrates ist sterblich.

Möglichkeit 1: Hauptwort wird zum Prädikat.

$Sokrates\_ist(sterblich)$

Möglichkeit 2: Zeit-/Eigenschaftswort wird zum Prädikat.

$Ist\_sterblich(sokrates)$

Die zweite Möglichkeit ist fast immer die richtige.

Mache Zeitwörter bzw. Eigenschaften zu Prädikatensymbole und Hauptwörter zu ihren Argumenten.

Fürwörter (Pronomen)

Persönliche Fürwörter (ich, du, er, sie, es, ...) oder bezügliche Fürwörter (der, welcher, ...) vermeiden Wiederholungen und stellen Bezüge her.

Mia liebt Max, der wiederum sie liebt.

Umformung in

Mia liebt Max und Max liebt Mia.

ergibt die Formel

$Liebt(mia, max) \wedge Liebt(max, mia)$

Ersetze Fürwörter durch Namen. Wiederhole Satzteile, die durch Fürwörter ersetzt wurden.

Unbestimmte Fürwörter wie „nichts“ ziehen Subjekt/Objekt und Negation zusammen.

Nichts währt ewig.

Trenne Verneinung und Subjekt:

Es existiert nicht etwas, das ewig währt.

Ergibt die Formel

$\neg\exists x\, Währt\_ewig(x)$

Ich kann nichts sehen.
$\neg\exists x\, Kann\_sehen(gernot, x)$

Führe Variablen für unbestimmte Fürwörter ein.
Mache Negationen explizit.

Fürwörter und Bindewörter

Vor mir ist etwas Großes, und es ist hungrig.

Die zwei Teilsätze können nicht getrennt behandet werden: „es“ bezieht sich auf „etwas Großes“. Die entsprechende Variable muss überall dieselbe sein.

Es gibt etwas, das vor mir ist, das groß ist, und das hungrig ist.
$\exists x\,(\mathit{Befindet\_sich\_vor(x,gernot)} \wedge \mathit{Groß}(x) \wedge \mathit{Hungrig}(x))$

Erstrecken sich die Bezüge von Fürwörtern über Bindewörter hinweg, behandle die Fürwörter vor den Bindewörtern.

Quantoren, Typen und Beziehungen

Jeder Student ist jünger als irgendein Professor.

Identifiziere Objekt-/Personenarten und stelle sie durch einstellige Typprädikate dar.

„ist Student“	$\Rightarrow \mathit{Stud}(\dots)$
„ist Professor“	$\Rightarrow \mathit{Prof}(\dots)$
Max ist Student.	$\Rightarrow \mathit{Stud}(\mathit{max})$
Gernot ist Professor.	$\Rightarrow \mathit{Prof}(\mathit{gernot})$
Jemand ist Student.	$\Rightarrow \mathit{Stud}(x)$

Identifiziere Beziehungen (Relationen) und formalisiere sie durch mehrstellige Prädikatensymbole.

„ist jünger als“	$\Rightarrow \mathit{Jünger}(\dots,\dots)$
Max ist jünger als Gernot.	$\Rightarrow \mathit{Jünger}(\mathit{max},\mathit{gernot})$

Quantoren, Typen und Beziehungen

Jeder Student ist jünger als irgendein Professor.

Verwende $\forall$-Quantoren für „alle“/„jede“ und $\exists$-Quantoren für „es gibt“/„jemand“/„irgendein“.

$\forall x\,(\mathit{Stud}(x) \supset \exists y\,(\mathit{Prof}(y) \wedge \mathit{Jünger}(x, y)))$

Für alle Individuen x gilt:
Falls x ein Student ist, gibt es mindestens ein Individuum y,
sodass y Professor ist und x jünger als y ist.

Funktionssymbole

Jedes Kind ist jünger als seine Mutter.

„x ist ein Kind“ $\Rightarrow \mathit{Kind}(x)$
„y ist Mutter von x“ $\Rightarrow \mathit{Mutter}(y, x)$

$\forall x\,\forall y((\mathit{Kind}(x) \wedge \mathit{Mutter}(y, x)) \supset \mathit{Jünger}(x, y))$

Diese Formalisierung berücksichtigt nicht, dass jedes Kind genau eine genetische Mutter hat.

Besser: Fasse „Mutter“ als Funktion auf, die jedem Kind seine eindeutig bestimmte Mutter zuordnet.

„Mutter von x“ $\Rightarrow \mathit{mutter}(x)$

$\forall x\,(\mathit{Kind}(x) \supset \mathit{Jünger}(x, \mathit{mutter}(x))$

Alternierende Quantoren

Vergleichen Sie:

$\forall x \exists y\, \mathit{Mutter}(y, x)$	$\exists y \forall x\, \mathit{Mutter}(y, x)$
„Jeder hat eine Mutter.“	„Jemand ist die Mutter von allen.“
y hängt vom gewählten x ab.	y ist unabhängig vom gewählten x.

Vertauschung unterschiedlicher Quantoren ändert die Bedeutung!

Quantoren und Eigenschaften

Alle vernünftigen Leute verabscheuen Gewalt.

$\forall$ vernünftige x (x verabscheut Gewalt) (Zwischenschritt, ist keine Formel!)

$\forall x\, (\mathit{Vernünftig}(x) \supset \mathit{Verabscheut}(x, \mathit{gewalt}))$

Eigenschaften $\forall$-quantifizierter Variablen werden zu Prämissen einer Implikation.

Manche vernünftige Leute verabscheuen Gewalt.
Es gibt vernünftige Leute, die Gewalt verabscheuen.

$\exists$ vernünftige x (x verabscheut Gewalt) (Zwischenschritt, ist keine Formel!)

$\exists x\, (\mathit{Vernünftig}(x) \wedge \mathit{Verabscheut}(x, \mathit{gewalt}))$

Eigenschaften $\exists$-quantifizierter Variablen werden zu Teilen einer Konjunktion.

Beispiel: Sport

$\mathcal{P} = \{\mathit{Mensch}/1, \mathit{Sportart}/1, \mathit{Anstrengend}/1, \mathit{Betreibt}/2\}$
$\mathcal{F} = \{\mathit{handball}/0, \mathit{laufen}/0\}$

Alle Menschen betreiben eine anstrengende Sportart.

Zu jedem Menschen x gibt es eine Sportart,
die anstrengend ist und von x betrieben wird.

Zu jedem Menschen x gibt es eine Sportart y,
sodass y anstrengend ist und x y betreibt.

„Zu jedem Menschen x ...“ $\forall x(\mathit{Mensch}(x) \supset \cdots)$
„Es gibt eine Sportart y ...“ $\exists y(\mathit{Sportart}(y) \wedge \cdots)$
„y ist anstrengend und x betreibt y“ $\mathit{Anstrengend}(y) \wedge \mathit{Betreibt}(x, y)$

$\forall x(\mathit{Mensch}(x) \supset \exists y(\mathit{Sportart}(y) \wedge \mathit{Anstrengend}(y) \wedge \mathit{Betreibt}(x, y)))$
$\forall x \exists y(\mathit{Mensch}(x) \supset (\mathit{Sportart}(y) \wedge \mathit{Anstrengend}(y) \wedge \mathit{Betreibt}(x, y)))$

Beispiel: Sport

$\mathcal{P} = \{\mathit{Mensch}/1, \mathit{Sportart}/1, \mathit{Anstrengend}/1, \mathit{Betreibt}/2\}$
$\mathcal{F} = \{\mathit{handball}/0, \mathit{laufen}/0\}$

Es gibt anstrengende Menschen, die Handball und Laufen betreiben.

Es gibt (mindestens) einen Menschen x,
der anstrengend ist, Handball betreibt und Laufen betreibt.

Es gibt (mindestens) einen Menschen x,
sodass x anstrengend ist, x Handball betreibt und x Laufen betreibt.

„Es gibt einen Menschen x“ $\exists x(\mathit{Mensch}(x) \wedge \cdots)$
„x ist anstrengend“ $\mathit{Anstrengend}(x)$
„x betreibt Handball“ $\mathit{Betreibt}(x, \mathit{handball})$
„x betreibt Laufen“ $\mathit{Betreibt}(x, \mathit{laufen})$

$\exists x(\mathit{Mensch}(x) \wedge \mathit{Anstrengend}(x) \wedge \mathit{Betreibt}(x, \mathit{handball}) \wedge \mathit{Betreibt}(x, \mathit{laufen}))$

Fooling people

„You can fool some of the people all of the time,
and all of the people some of the time,
but you cannot fool all of the people all of the time.“

(zugeschrieben Abraham Lincoln, 16. Präsident der USA, 1809–1865)

$FoolAt(x, y)$	... you can fool x at time y
$Person(x)$	... x is a person / x is one of the people
$PointInTime(y)$	... y is a point in time

$$\begin{aligned} & \exists x(Person(x) \wedge \forall y(PointInTime(y) \supset FoolAt(x, y))) \\ \wedge \quad & \forall x(Person(x) \supset \exists y(PointInTime(y) \wedge FoolAt(x, y))) \\ \wedge \neg & \forall x(Person(x) \supset \forall y(PointInTime(y) \supset FoolAt(x, y))) \end{aligned}$$

3.0 VU Formale Modellierung

Gernot Salzer

Forschungsbereich Theory and Logic
Institut für Logic and Computation

29.5.2018

Was Sie letztes Mal hörten

1. Organisatorisches
2. Was bedeutet Modellierung?
3. Aussagenlogik
4. Endliche Automaten
5. Reguläre Sprachen
6. Kontextfreie Grammatiken
7. Prädikatenlogik
   - 7.1. Motivation
   - 7.2. Terme
   - 7.3. Syntax
   - 7.4. Semantik
   - 7.5. Modellierung

Prädikatenlogik – Syntax

$\mathcal{V}$ ... Individuenvariablensymbole
$\mathcal{F}$, $\mathcal{P}$ ... Funktions- bzw. Prädikatensymbole mit Stelligkeiten
$(\mathcal{F}, \mathcal{P})$ ... „Signatur“

Syntax prädikatenlogischer Formeln

Die Menge $\mathcal{PF}$ der prädikatenlogischen Formeln über der Signatur $(\mathcal{F}, \mathcal{P})$ ist die kleinste Menge, für die gilt:

(p1) $P \in \mathcal{PF}$, wenn $P/0 \in \mathcal{P}$;

(p2) $P(t_1, \ldots, t_n) \in \mathcal{PF}$, wenn $P/n \in \mathcal{P}$ und $t_1, \ldots, t_n \in \mathcal{T}(\mathcal{F}, \mathcal{V})$;

(p3) $\top, \bot \in \mathcal{PF}$;

(p4) $\neg F \in \mathcal{PF}$, wenn $F \in \mathcal{PF}$;

(p5) $(F * G) \in \mathcal{PF}$, wenn $F, G \in \mathcal{PF}$ und $* \in \{\wedge, \uparrow, \vee, \downarrow, \equiv, \not\equiv, \supset, \subset\}$.

(p6) $\forall x\, F \in \mathcal{PF}$, wenn $x \in \mathcal{V}$ und $F \in \mathcal{PF}$.

(p7) $\exists x\, F \in \mathcal{PF}$, wenn $x \in \mathcal{V}$ und $F \in \mathcal{PF}$.

P, $P(t_1, \ldots, t_n)$ ... „Atomformeln“

Was Sie letztes Mal hörten

Prädikatenlogik – Semantik

Prädikatensymbole: repräsentieren Relationen bzw. Elementaraussagen

- Vordefinierte Symbole wie $\top$ und $\bot$: feste, unveränderliche Bedeutung (fix eingebaut in Formelsemantik)
- Freie Symbole: Interpretation durch Relationen bzw. Wahrheitswerte

(Prädikatenlogische) Interpretation

Eine Interpretation I über einem Wertebereich $\mathcal{U}$ ordnet jedem Symbol der Signatur $(\mathcal{F}, \mathcal{P})$ eine Funktion bzw. Relation wie folgt zu:

$I(f) \in \mathcal{U}$	für $f/0 \in \mathcal{F}$
$I(f) \colon \mathcal{U}^n \mapsto \mathcal{U}$	für $f/n \in \mathcal{F}$, $n > 0$
$I(P) \in \{0, 1\}$	für $P/0 \in \mathcal{P}$
$I(P) \subseteq \mathcal{U}^n$	für $P/n \in \mathcal{P}$, $n > 0$
$I(P) \colon \mathcal{U}^n \mapsto \{0, 1\}$	(alternative Sichtweise)

Semantik prädikatenlogischer Formeln

Der Wert einer Formel in einer Interpretation I mit Variablenbelegung σ wird festgelegt durch die Funktion $\mathrm{val}_{I,\sigma}$, definiert als:

(v1) $\mathrm{val}_{I,\sigma}(P) = I(P)$ für $P/0 \in \mathcal{P}$;

(v2) $\mathrm{val}_{I,\sigma}(P(t_1, \ldots, t_n)) = I(P)(\mathrm{val}_{I,\sigma}(t_1), \ldots, \mathrm{val}_{I,\sigma}(t_n))$ für $P/n \in \mathcal{P}$;

(v3) $\mathrm{val}_{I,\sigma}(\top) = 1$ und $\mathrm{val}_{I,\sigma}(\bot) = 0$;

(v4) $\mathrm{val}_{I,\sigma}(\neg F) = \mathrm{not}\, \mathrm{val}_{I,\sigma}(F)$;

(v5) $\mathrm{val}_{I,\sigma}(\,(F * G)\,) = \mathrm{val}_{I,\sigma}(F) \circledast \mathrm{val}_{I,\sigma}(G)$,
wobei $\circledast$ die logische Funktion zum Operator $*$ ist;

(v6) $$\mathrm{val}_{I,\sigma}(\,\forall x\, F\,) = \begin{cases} 1 & \text{falls } \mathrm{val}_{I,\sigma'}(F) = 1 \text{ für alle } \sigma' \overset{x}{\sim} \sigma \\ 0 & \text{sonst} \end{cases}$$

(v7) $$\mathrm{val}_{I,\sigma}(\,\exists x\, F\,) = \begin{cases} 1 & \text{falls } \mathrm{val}_{I,\sigma'}(F) = 1 \text{ für mind. ein } \sigma' \overset{x}{\sim} \sigma \\ 0 & \text{sonst} \end{cases}$$

$\sigma \overset{x}{\sim} \sigma' \ldots \sigma(v) = \sigma'(v)$ für alle $v \in \mathcal{V}$ mit $v \neq x$

(σ und σ' sind identisch, nur $\sigma(x)$ und $\sigma'(x)$ können verschieden sein.)

Eine Formel F heißt

- erfüllbar, wenn $\mathrm{val}_{I,\sigma}(F) = 1$ für mindestens ein I und ein σ (I und σ nennt man Modell von F);
- widerlegbar, wenn $\mathrm{val}_{I,\sigma}(F) = 0$ für mindestens ein I und ein σ (I und σ nennt man Gegenbeispiel oder Gegenmodell zu F);
- unerfüllbar, wenn $\mathrm{val}_{I,\sigma}(F) = 0$ für alle I und alle σ;
- (allgemein)gültig, wenn $\mathrm{val}_{I,\sigma}(F) = 1$ für alle I und alle σ (F nennt man in diesem Fall auch Tautologie).

Äquivalenz, Konsequenz und Gültigkeit

Semantische Äquivalenz

Zwei Formeln F und G heißen äquivalent, geschrieben $F = G$, wenn $\mathrm{val}_{I,\sigma}(F) = \mathrm{val}_{I,\sigma}(G)$ für alle I und alle σ gilt.

$F_1, \ldots, F_n \models_{I,\sigma} G$:
„Aus $\mathrm{val}_{I,\sigma}(F_1) = \cdots = \mathrm{val}_{I,\sigma}(F_n) = 1$ folgt $\mathrm{val}_{I,\sigma}(G) = 1$.“

Logische Konsequenz

$F_1, \ldots, F_n \models G$: $F_1, \ldots, F_n \models_{I,\sigma} G$ gilt für alle I und σ.

Die Formeln F und G sind äquivalent ($F = G$) genau dann, wenn $F \equiv G$ eine gültige Formel ist.

$F_1, \ldots, F_n \models G$ genau dann, wenn $(F_1 \wedge \cdots \wedge F_n) \supset G$ gültig.

Wichtige Äquivalenzen

Neben den aussagenlogischen Äquivalenzen gelten auch noch folgende:

$\neg\forall x\, F = \exists x\, \neg F$ $\neg\exists x\, F = \forall x\, \neg F$	Dualität von $\forall$ und $\exists$
$\forall x\, F[x] = \forall y\, F[y]$ $\exists x\, F[x] = \exists y\, F[y]$	Umbenennung gebundener Variablen (sofern y nicht bereits in F vorkommt)
$\forall x\, \forall y\, F = \forall y\, \forall x\, F$ $\exists x\, \exists y\, F = \exists y\, \exists x\, F$	Vertauschung gleichartiger Quantoren
$\forall x\, (F \wedge G) = \forall x\, F \wedge \forall x\, G$	Distributivität $\forall/\wedge$
$\exists x\, (F \vee G) = \exists x\, F \vee \exists x\, G$	Distributivität $\exists/\vee$

Falls x nicht frei in F vorkommt, gilt außerdem:

$\forall x\, F = F \qquad \exists x\, F = F$	
$\forall x\, (F \vee G) = F \vee \forall x\, G$	Distributivität $\forall/\vee$
$\exists x\, (F \wedge G) = F \wedge \exists x\, G$	Distributivität $\exists/\wedge$

Was Sie letztes Mal hörten

Wahl der Prädikate

Wähle ein eigenes Symbol für Satzteile, die öfter auftreten (können).

Max liest Zeitung.
Max_liest_Zeitung
Liest_Zeitung(*max*)
Liest(*max*, *zeitung*)

Mache Zeitwörter bzw. Eigenschaften zu Prädikatensymbole und Hauptwörter zu ihren Argumenten.

Sokrates ist sterblich.
Ist_sterblich(*sokrates*)

Fürwörter (Pronomen)

Ersetze Fürwörter durch Namen. Wiederhole Satzteile, die durch Fürwörter ersetzt wurden.
Führe Variablen für unbestimmte Fürwörter ein.
Mache Negationen explizit.

Mia liebt Max, der wiederum sie liebt.
$Liebt(mia, max) \wedge Liebt(max, mia)$

Nichts währt ewig.
$\neg\exists x\ Währt\_ewig(x)$

Ich kann nichts sehen.
$\neg\exists x\ Kann\_sehen(gernot, x)$

Vor mir ist etwas Großes, und es ist hungrig.
$\exists x\ (Befindet\_sich\_vor(x,gernot) \wedge Groß(x) \wedge Hungrig(x))$

Quantoren, Typen und Beziehungen

Jeder Student ist jünger als irgendein Professor.

Identifiziere Objekt-/Personenarten und stelle sie durch einstellige Typprädikate dar.

„ist Student“ $\Rightarrow Stud(\dots)$
„ist Professor“ $\Rightarrow Prof(\dots)$

Max ist Student. $\Rightarrow Stud(max)$
Gernot ist Professor. $\Rightarrow Prof(gernot)$
Jemand ist Student. $\Rightarrow Stud(x)$

Identifiziere Beziehungen (Relationen) und formalisiere sie durch mehrstellige Prädikatensymbole.

„ist jünger als“ $\Rightarrow Jünger(\dots,\dots)$

Max ist jünger als Gernot. $\Rightarrow Jünger(max, gernot)$

Quantoren, Typen und Beziehungen

Jeder Student ist jünger als irgendein Professor.

Verwende $\forall$-Quantoren für „alle“/„jede“ und $\exists$-Quantoren für „es gibt“/„jemand“/„irgendein“.

$\forall x\,(Stud(x) \supset \exists y\,(Prof(y) \wedge Jünger(x,y)))$

Für alle Individuen x gilt:
Falls x ein Student ist, gibt es mindestens ein Individuum y,
sodass y Professor ist und x jünger als y ist.

Funktionssymbole

Jedes Kind ist jünger als seine Mutter.

„x ist ein Kind“ $\Rightarrow \mathit{Kind}(x)$
„y ist Mutter von x“ $\Rightarrow \mathit{Mutter}(y, x)$

$\forall x \, \forall y ((\mathit{Kind}(x) \wedge \mathit{Mutter}(y, x)) \supset \mathit{Jünger}(x, y))$

Diese Formalisierung berücksichtigt nicht, dass jedes Kind genau eine genetische Mutter hat.

Besser: Fasse „Mutter“ als Funktion auf, die jedem Kind seine eindeutig bestimmte Mutter zuordnet.

„Mutter von x“ $\Rightarrow \mathit{mutter}(x)$

$\forall x \, (\mathit{Kind}(x) \supset \mathit{Jünger}(x, \mathit{mutter}(x))$

Alternierende Quantoren

Vergleichen Sie:

$\forall x \, \exists y \, \mathit{Mutter}(y, x)$	$\exists y \, \forall x \, \mathit{Mutter}(y, x)$
„Jeder hat eine Mutter.“	„Jemand ist die Mutter von allen.“
y hängt vom gewählten x ab.	y ist unabhängig vom gewählten x.

Vertauschung unterschiedlicher Quantoren ändert die Bedeutung!

Quantoren und Eigenschaften

Alle vernünftigen Leute verabscheuen Gewalt.

$\forall x\,(\mathit{Vernünftig}(x) \supset \mathit{Verabscheut}(x, \mathit{gewalt}))$

Eigenschaften $\forall$-quantifizierter Variablen werden zu Prämissen einer Implikation.

Manche vernünftige Leute verabscheuen Gewalt.
Es gibt vernünftige Leute, die Gewalt verabscheuen.

$\exists x\,(\mathit{Vernünftig}(x) \wedge \mathit{Verabscheut}(x, \mathit{gewalt}))$

Eigenschaften $\exists$-quantifizierter Variablen werden zu Teilen einer Konjunktion.

Fooling people

„You can fool some of the people all of the time,
and all of the people some of the time,
but you cannot fool all of the people all of the time.“

(zugeschrieben Abraham Lincoln, 16. Präsident der USA, 1809–1865)

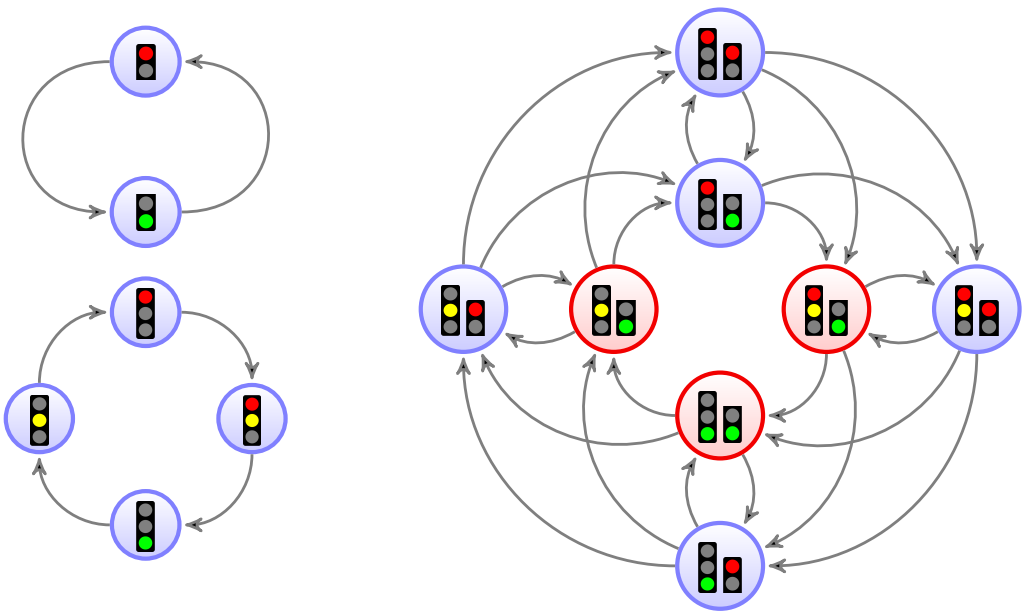
$\mathit{FoolAt}(x, y)$	... you can fool x at time y
$\mathit{Person}(x)$	... x is a person / x is one of the people
$\mathit{PointInTime}(y)$	... y is a point in time

$$\begin{aligned} & \exists x(\mathit{Person}(x) \wedge \forall y(\mathit{PointInTime}(y) \supset \mathit{FoolAt}(x, y))) \\ \wedge\ & \forall x(\mathit{Person}(x) \supset \exists y(\mathit{PointInTime}(y) \wedge \mathit{FoolAt}(x, y))) \\ \wedge\ & \neg\forall x(\mathit{Person}(x) \supset \forall y(\mathit{PointInTime}(y) \supset \mathit{FoolAt}(x, y))) \end{aligned}$$

Was Sie heute erwartet

1. Organisatorisches
2. Was bedeutet Modellierung?
3. Aussagenlogik
4. Endliche Automaten
5. Reguläre Sprachen
6. Kontextfreie Grammatiken
7. Prädikatenlogik
8. Petri-Netze
   - 8.1. Motivation
   - 8.2. Definitionen
   - 8.3. Modellierung

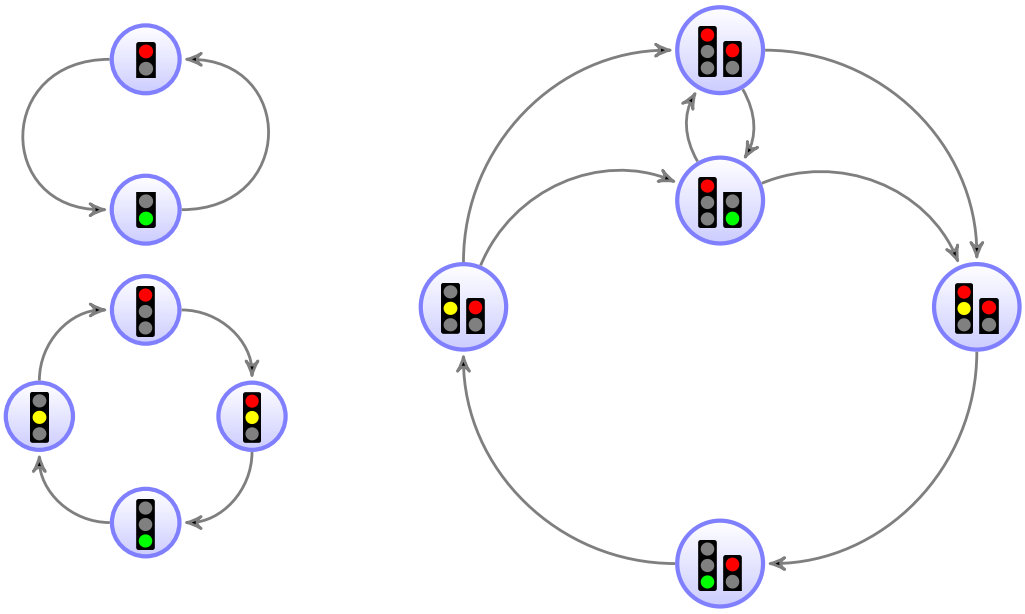
Fußgängerkreuzung als endlicher Automat



Modellierung des Gesamtsystems durch einen Produktautomaten:

- Bilde alle Kombinationszustände.
- Übergange dort, wo die ursprünglichen Automaten welche hatten.

Fußgängerkreuzung als endlicher Automat



Modellierung des Gesamtsystems durch einen Produktautomaten:

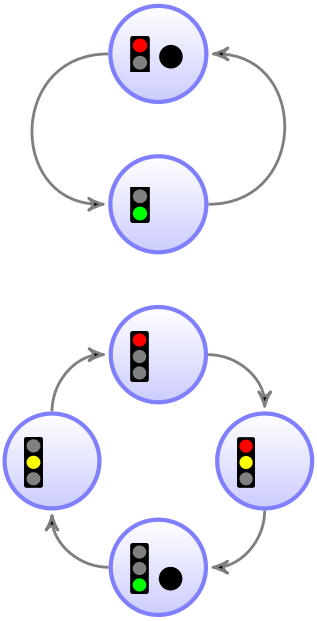
- Bilde alle Kombinationszustände.
- Übergange dort, wo die ursprünglichen Automaten welche hatten.
- Elimination unerwünschter Zustände und Übergänge.

Endliche Automaten ungeeignet für Modellierung verteilter Systeme:

- Anzahl der Kombinationszustände steigt exponentiell mit der Zahl der Komponenten.
- Anzahl der Übergänge steigt exponentiell.
- Es ist schwierig, Änderungen einer Komponente in den Gesamtautomaten zu übertragen.
- Endliche Automaten mit einem aktiven Zustand entsprechen nicht der Idee verteilter Systeme mit vielen unabhängigen Abläufen nebeneinander, die nur bei Bedarf synchronisiert werden.

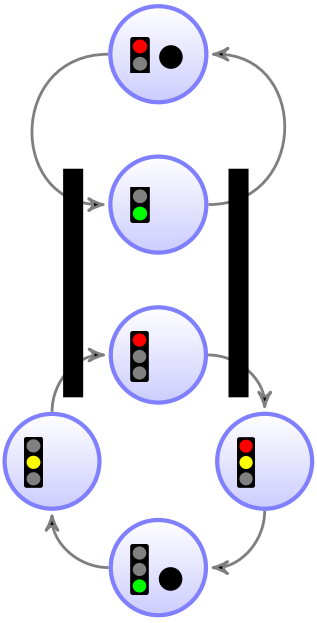
Petri-Netz = Automat mit mehreren aktiven Stellen + Synchronisation

Fußgängerkreuzung als Petri-Netz



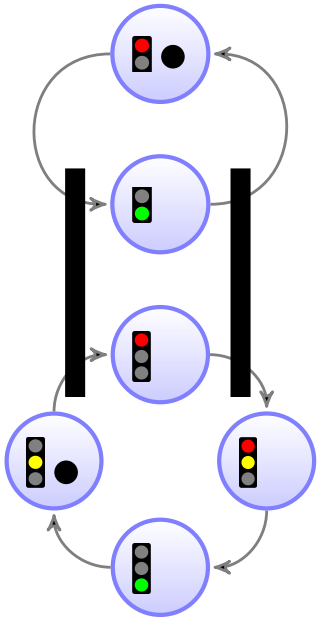
- Marken zeigen die aktiven Stellen an.

Fußgängerkreuzung als Petri-Netz



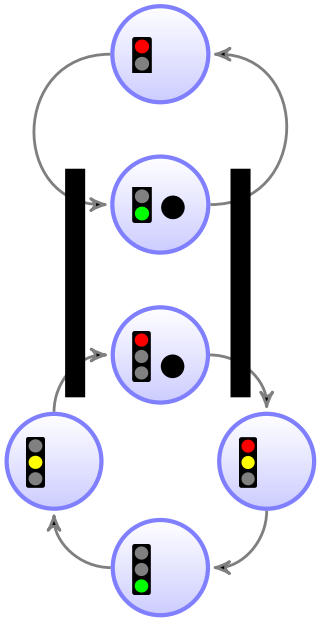
- Marken zeigen die aktiven Stellen an.
- Transitionen werden nur durchlässig („feuern“), wenn alle Eingangszustände Marken besitzen.
  Marken dürfen nur gleichzeitig weiterwandern.

Fußgängerkreuzung als Petri-Netz



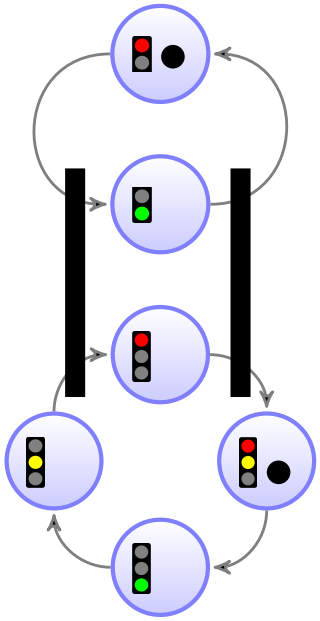
- Marken zeigen die aktiven Stellen an.
- Transitionen werden nur durchlässig („feuern“), wenn alle Eingangszustände Marken besitzen.
  Marken dürfen nur gleichzeitig weiterwandern.

Fußgängerkreuzung als Petri-Netz



- Marken zeigen die aktiven Stellen an.
- Transitionen werden nur durchlässig („feuern“), wenn alle Eingangszustände Marken besitzen.
  Marken dürfen nur gleichzeitig weiterwandern.

Fußgängerkreuzung als Petri-Netz



- Marken zeigen die aktiven Stellen an.
- Transitionen werden nur durchlässig („feuern“), wenn alle Eingangszustände Marken besitzen.
  Marken dürfen nur gleichzeitig weiterwandern.

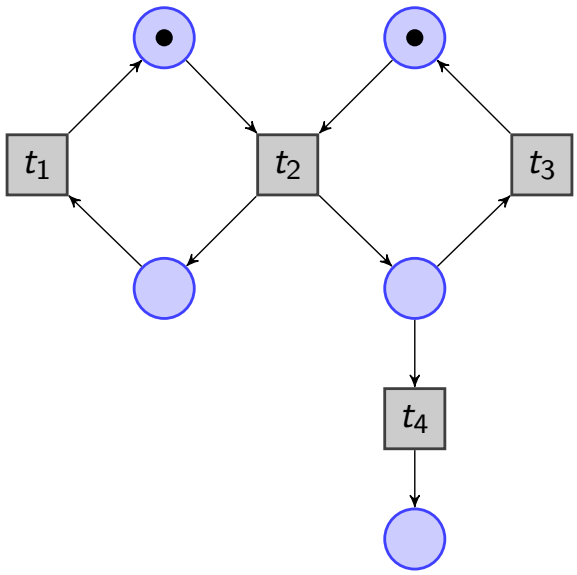
Petrinetze: Motivation

Formalismus zur Modellierung von nebenläufigen Systemen (concurrent/parallel systems).

- Bei Systemübergängen können Ressourcen konsumiert und neu erzeugt werden.
- Natürliche Modellierung der räumlichen Verteilung von Ressourcen, Nebenläufigkeit und (Zugriffs-)Konflikten.
- Intuitive graphische Darstellung.
- Weit verbreitet, z.B. Aktivitätsdiagramme (activity diagrams) in UML (Unified Modeling Language).

Übernommen von Barbara König, Vorlesung „Modellierung nebenläufiger Systeme“, Uni Duisburg-Essen, 2011

Petrinetze: Motivation



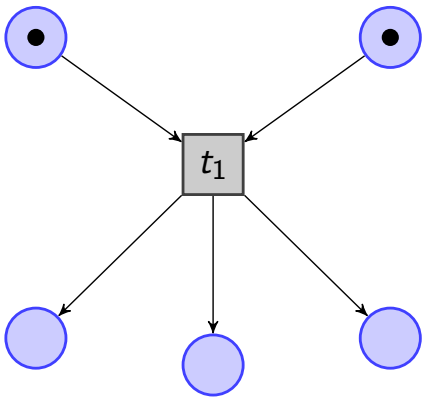
Notation:

- Stellen (dargestellt als Kreise): mögliche Plätze für Ressourcen
- Marken (dargestellt als kleine gefüllte Kreise): Ressourcen
- Transitionen (dargestellt als Rechtecke oder Balken): Systemübergänge

Übernommen von Barbara König, Vorlesung „Modellierung nebenläufiger Systeme“, Uni Duisburg-Essen, 2011

Petrinetze: Motivation

Darstellung einer Transition:

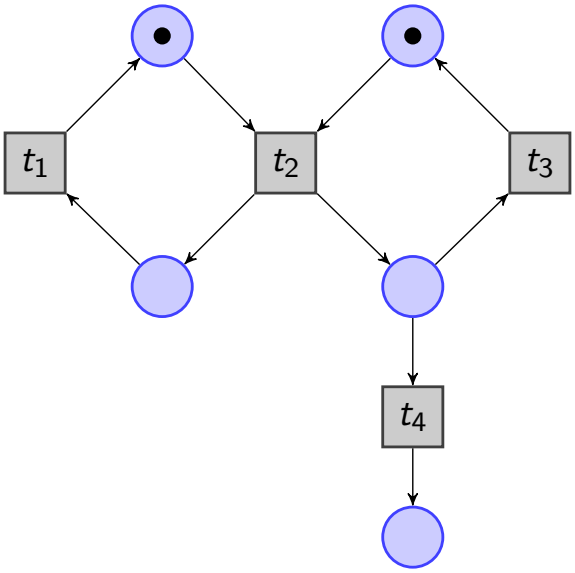


Vorbedingungen

Nachbedingungen

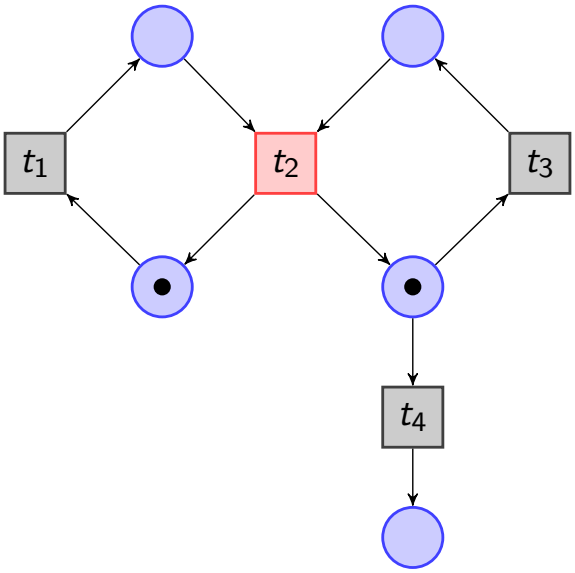
- Vorbedingungen sind die Marken, die konsumiert werden
- Nachbedingungen sind die Marken, die erzeugt werden
- Das Entfernen der Marken der Vorbedingungen und das Erzeugen der Marken der Nachbedingungen nennt man Schalten bzw. Feuern der Transition.

Petrinetze: Beispiel



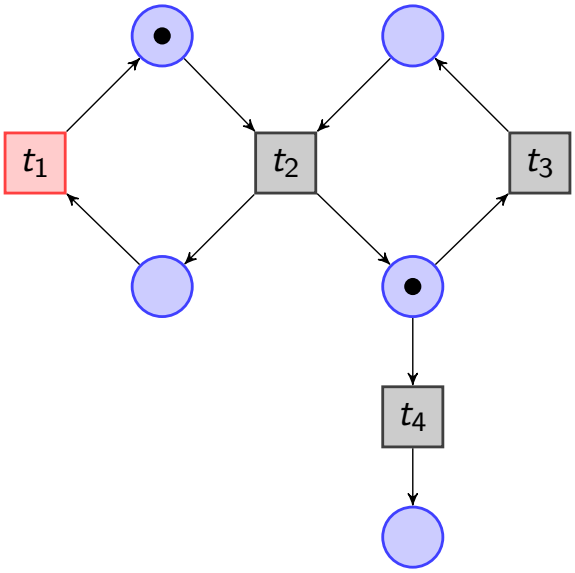
Übernommen von Barbara König, Vorlesung „Modellierung nebenläufiger Systeme“, Uni Duisburg-Essen, 2011

Petrinetze: Beispiel



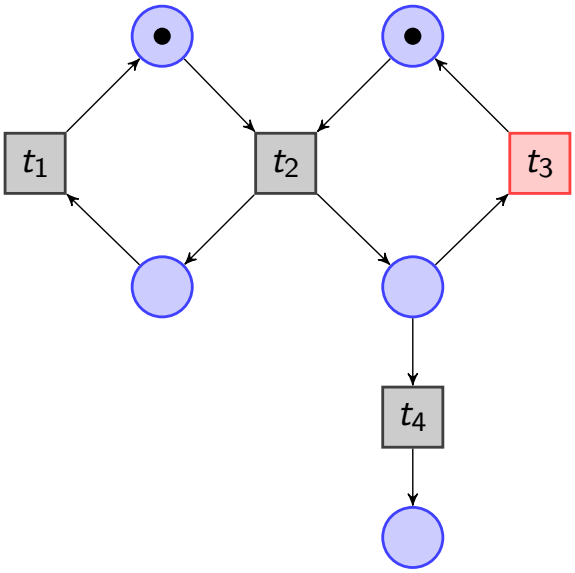
Übernommen von Barbara König, Vorlesung „Modellierung nebenläufiger Systeme“, Uni Duisburg-Essen, 2011

Petrinetze: Beispiel



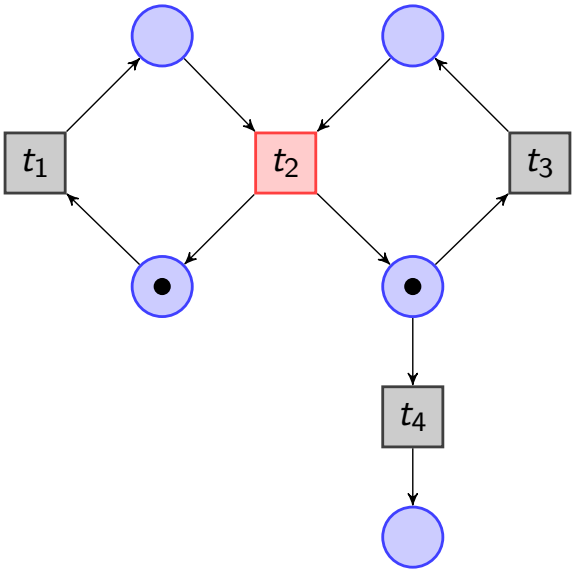
Übernommen von Barbara König, Vorlesung „Modellierung nebenläufiger Systeme“, Uni Duisburg-Essen, 2011

Petrinetze: Beispiel



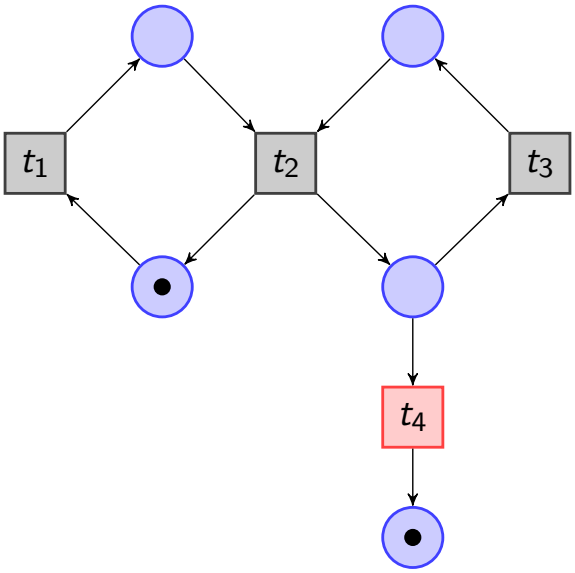
Übernommen von Barbara König, Vorlesung „Modellierung nebenläufiger Systeme“, Uni Duisburg-Essen, 2011

Petrinetze: Beispiel



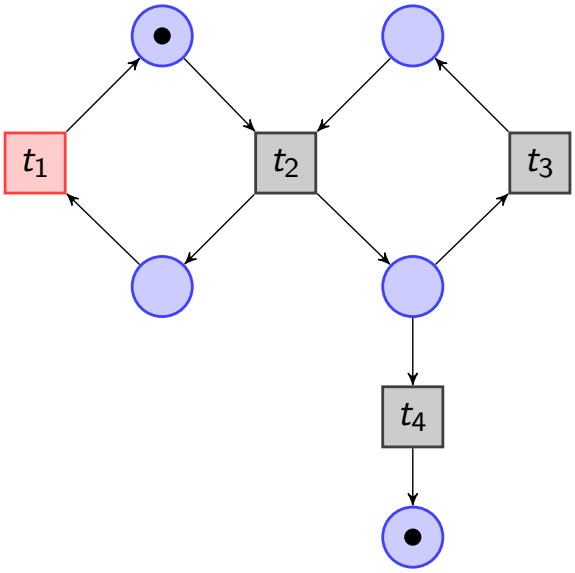
Übernommen von Barbara König, Vorlesung „Modellierung nebenläufiger Systeme“, Uni Duisburg-Essen, 2011

Petrinetze: Beispiel



Übernommen von Barbara König, Vorlesung „Modellierung nebenläufiger Systeme“, Uni Duisburg-Essen, 2011

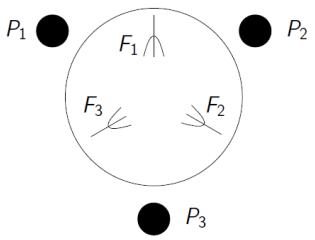
Petrinetze: Beispiel



Übernommen von Barbara König, Vorlesung „Modellierung nebenläufiger Systeme“, Uni Duisburg-Essen, 2011

Dining Philosophers

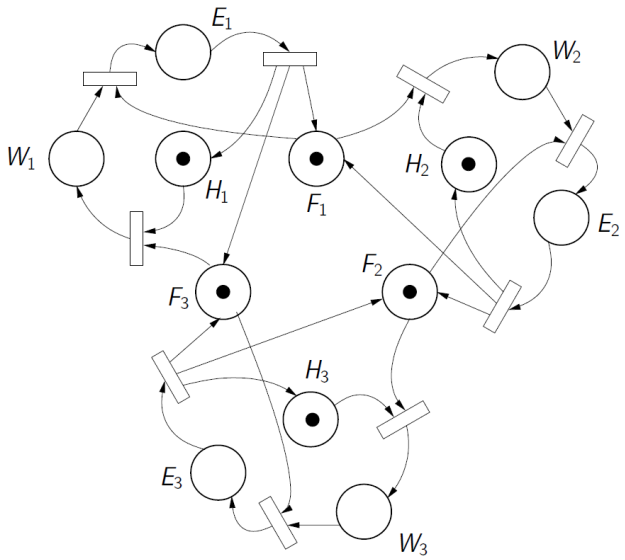
- Drei Philosophen sitzen um einen runden Tisch, zwischen je zwei Philosophen liegt eine Gabel.
- Wird ein Philosoph hungrig, hält er sich an folgenden Ablauf:
  1. Er nimmt die Gabel zu seiner Rechten.
  2. Er nimmt die Gabel zu seiner Linken.
  3. Er isst.
  4. Er legt beide Gabeln zurück.
- Sollte eine Gabel nicht an ihrem Platz liegen, wenn der Philosoph sie benötigt, so wartet er, bis die Gabel wieder verfügbar ist.



Übernommen von Barbara König, Vorlesung „Modellierung nebenläufiger Systeme“, Uni Duisburg-Essen, 2011

Dining Philosophers

Modellierung als Petrinetz:



Deadlock erreichbar, d.h., eine Markierung, bei der keine Transition mehr geschaltet werden kann.

Übernommen von Barbara König, Vorlesung „Modellierung nebenläufiger Systeme“, Uni Duisburg-Essen, 2011

Was Sie heute erwartet

Definitionen

Petrinetz

... wird beschrieben durch ein 5-Tupel $N = \langle S, T, {}^\bullet(), ()^\bullet, m_0 \rangle$, wobei

- S ... endliche Menge von Stellen
- T ... endliche Menge von Transitionen
- ${}^\bullet(): T \mapsto M$ ... Vorbedingungen
- $()^\bullet: T \mapsto M$ ... Nachbedingungen
- $m_0 \in M$ ... Anfangsmarkierung

M ... Menge der Markierungen, d.h., aller Abbildungen $S \mapsto \mathbb{N}$

$m_0 \in M$ legt fest, wieviele Marken zu Beginn in jeder Stelle liegen.

${}^\bullet t \in M$ legt fest, wieviele Marken die Transition t aus jeder Stelle entfernt.

$t^\bullet \in M$ legt fest, wieviele Marken die Transition t zu jeder Stelle hinzufügt.

Schalten und Erreichbarkeit

$m, m' \in M$ ... Markierungen

Ordnung: $m \leq m'$ falls $m(s) \leq m'(s)$ für alle $s \in S$

Addition: $m \oplus m' = m''$ falls $m''(s) = m(s) + m'(s)$ für alle $s \in S$.

Subtraktion: $m \ominus m' = m''$ falls $m''(s) = m(s) - m'(s)$ für alle $s \in S$.

- Eine Transition t ist für eine Markierung m aktiviert, wenn ${}^\bullet t \leq m$ gilt, d.h., wenn genug Marken vorhanden sind, um die Transition zu schalten.
- Sei t eine Transition, die für die Markierung m aktiviert ist. Dann kann t schalten (oder feuern), was zu der Nachfolgemarkierung $m' = m \ominus {}^\bullet t \oplus t^\bullet$ führt. Symbolisch $m[t\rangle m'$.
- Eine Markierung m_n heißt erreichbar in einem Netz, falls es eine Folge von Transitionen $t_1, \ldots, t_n$ gibt mit $m_0[t_1\rangle m_1 \ldots m_{n-1}[t_n\rangle m_n$, wobei m_0 die Anfangsmarkierung ist.

Übernommen von Barbara König, Vorlesung „Modellierung nebenläufiger Systeme“, Uni Duisburg-Essen, 2011

Graphische Notation

In der graphischen Notation werden ${}^\bullet t$ und $t^\bullet$ folgendermaßen dargestellt:

- Kein Pfeil zwischen s und t, falls ${}^\bullet t(s) = 0$ (bzw. $t^\bullet(s) = 0$).
- Ein Pfeil zwischen s und t, falls ${}^\bullet t(s) = 1$ (bzw. $t^\bullet(s) = 1$).
- Ein Pfeil mit Beschriftung n zwischen s und t, falls ${}^\bullet t(s) = n > 1$ (bzw. $t^\bullet(s) = n > 1$).

Der Wert ${}^\bullet t(s)$ bzw. $t^\bullet(s)$ wird auch als Gewicht bezeichnet.

Was Sie heute erwartet

Beispiel: Leser-Schreiber-Problem

Leser-Schreiber-Problem

Beim Leser-Schreiber-Problem operieren n Leserprozesse und m Schreiberprozesse auf ein und derselben Datei. Damit die Dateiinhalte nicht inkonsistent werden, müssen die folgenden Bedingungen beachtet werden:

- Es können zur gleichen Zeit mehrere Leserprozesse auf die Datei zugreifen.
- Ein Schreiberprozess darf nur dann auf die Datei zugreifen, wenn gerade kein anderer Prozess (lesend oder schreibend) auf die Datei zugreift.

Modellieren Sie das Leser-Schreiber-Problem als Petrinetz mit $n = 3$ und $m = 1$. Es können maximal 2 Leserprozesse gleichzeitig die Datei lesen.

Beispiel: Leser-Schreiber-Problem

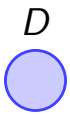
Leser-Schreiber-Problem

Beim Leser-Schreiber-Problem operieren n Leserprozesse und m Schreiberprozesse auf ein und derselben Datei. Damit die Dateiinhalte nicht inkonsistent werden, müssen die folgenden Bedingungen beachtet werden:

- Es können zur gleichen Zeit mehrere Leserprozesse auf die Datei zugreifen.
- Ein Schreiberprozess darf nur dann auf die Datei zugreifen, wenn gerade kein anderer Prozess (lesend oder schreibend) auf die Datei zugreift.

Modellieren Sie das Leser-Schreiber-Problem als Petrinetz mit $n = 3$ und $m = 1$. Es können maximal 2 Leserprozesse gleichzeitig die Datei lesen.

Beispiel: Leser-Schreiber-Problem



Beispiel: Leser-Schreiber-Problem

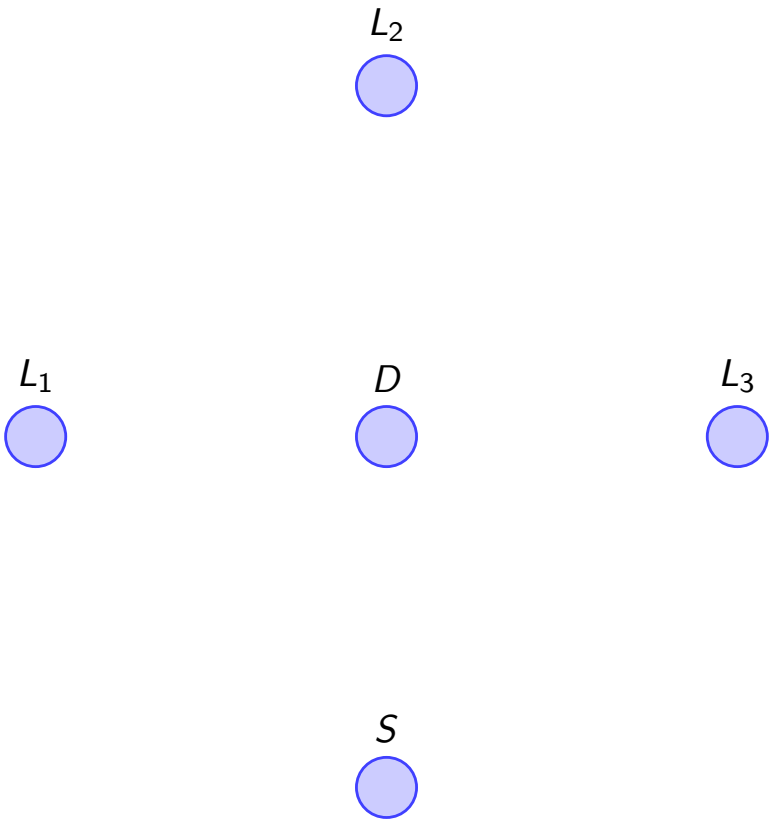
Leser-Schreiber-Problem

Beim Leser-Schreiber-Problem operieren n Leserprozesse und m Schreiberprozesse auf ein und derselben Datei. Damit die Dateiinhalte nicht inkonsistent werden, müssen die folgenden Bedingungen beachtet werden:

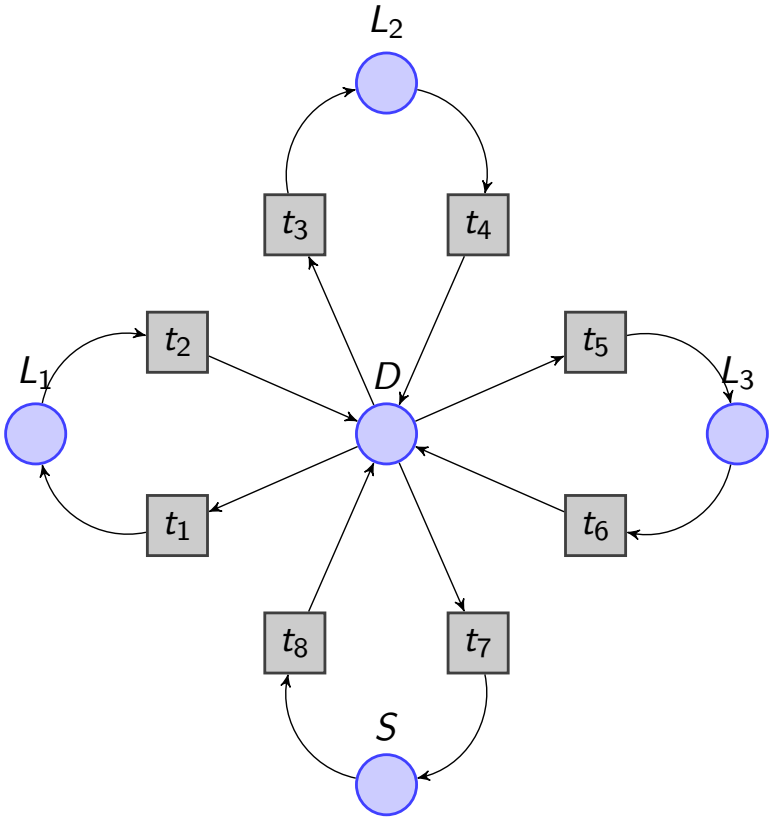
- Es können zur gleichen Zeit mehrere Leserprozesse auf die Datei zugreifen.
- Ein Schreiberprozess darf nur dann auf die Datei zugreifen, wenn gerade kein anderer Prozess (lesend oder schreibend) auf die Datei zugreift.

Modellieren Sie das Leser-Schreiber-Problem als Petrinetz mit $n = 3$ und $m = 1$. Es können maximal 2 Leserprozesse gleichzeitig die Datei lesen.

Beispiel: Leser-Schreiber-Problem



Beispiel: Leser-Schreiber-Problem



Beispiel: Leser-Schreiber-Problem

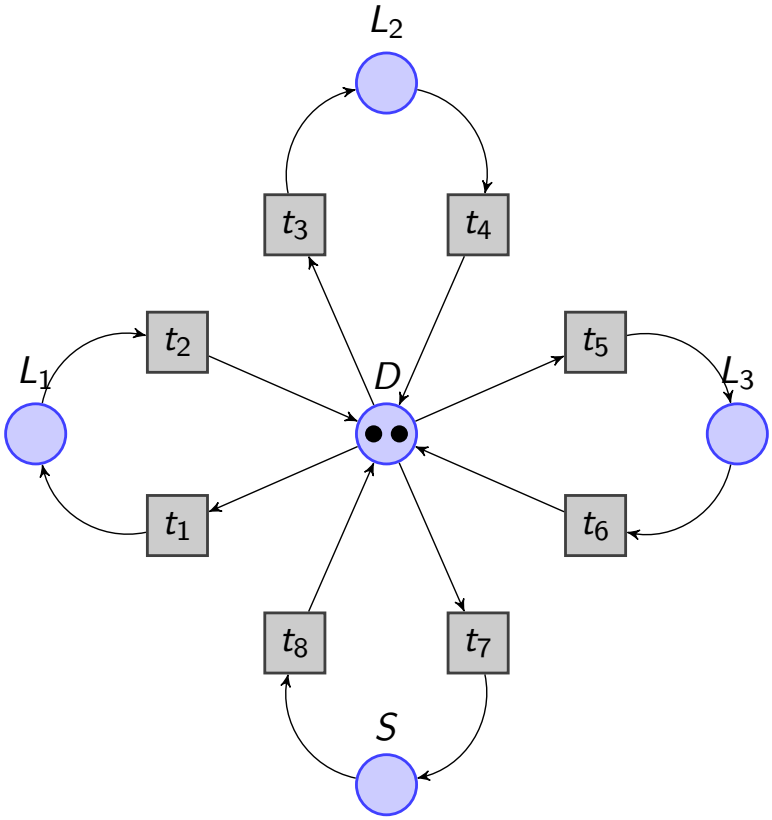
Leser-Schreiber-Problem

Beim Leser-Schreiber-Problem operieren n Leserprozesse und m Schreiberprozesse auf ein und derselben Datei. Damit die Dateiinhalte nicht inkonsistent werden, müssen die folgenden Bedingungen beachtet werden:

- Es können zur gleichen Zeit mehrere Leserprozesse auf die Datei zugreifen.
- Ein Schreiberprozess darf nur dann auf die Datei zugreifen, wenn gerade kein anderer Prozess (lesend oder schreibend) auf die Datei zugreift.

Modellieren Sie das Leser-Schreiber-Problem als Petrinetz mit $n = 3$ und $m = 1$. Es können maximal 2 Leserprozesse gleichzeitig die Datei lesen.

Beispiel: Leser-Schreiber-Problem



Beispiel: Leser-Schreiber-Problem

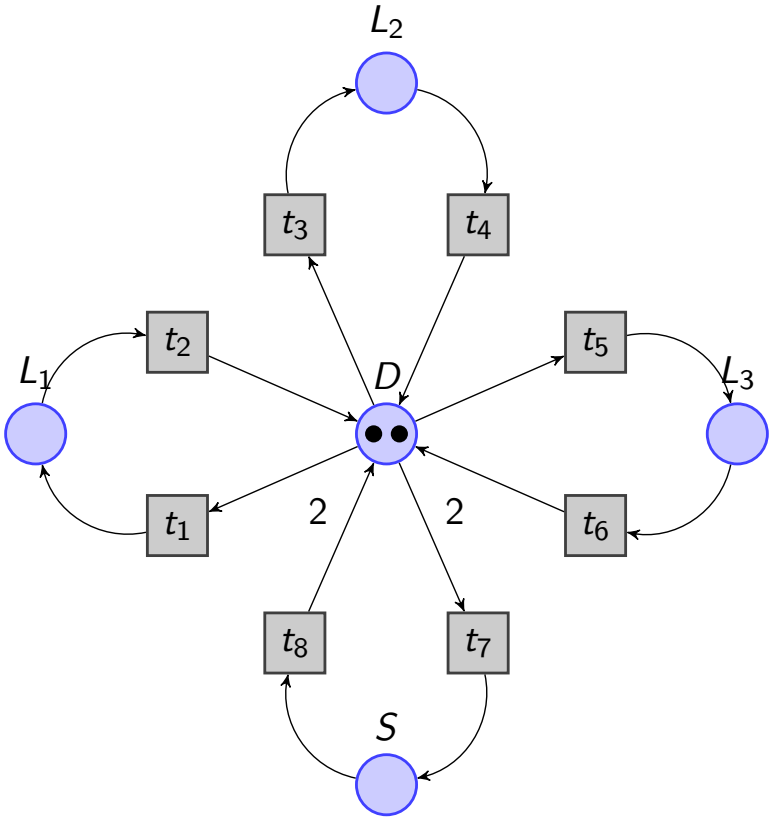
Leser-Schreiber-Problem

Beim Leser-Schreiber-Problem operieren n Leserprozesse und m Schreiberprozesse auf ein und derselben Datei. Damit die Dateiinhalte nicht inkonsistent werden, müssen die folgenden Bedingungen beachtet werden:

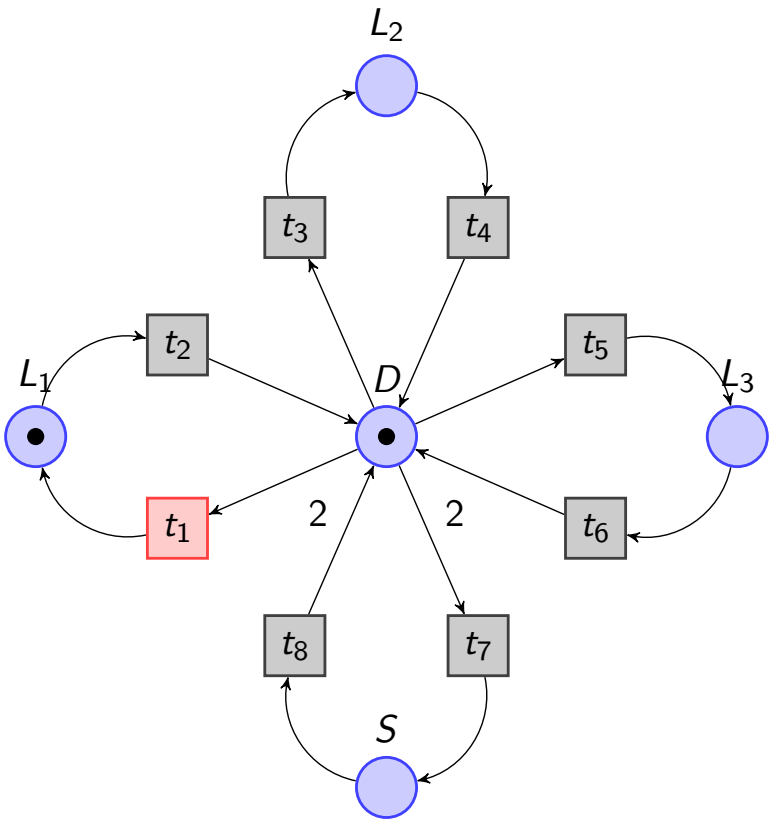
- Es können zur gleichen Zeit mehrere Leserprozesse auf die Datei zugreifen.
- Ein Schreiberprozess darf nur dann auf die Datei zugreifen, wenn gerade kein anderer Prozess (lesend oder schreibend) auf die Datei zugreift.

Modellieren Sie das Leser-Schreiber-Problem als Petrinetz mit $n = 3$ und $m = 1$. Es können maximal 2 Leserprozesse gleichzeitig die Datei lesen.

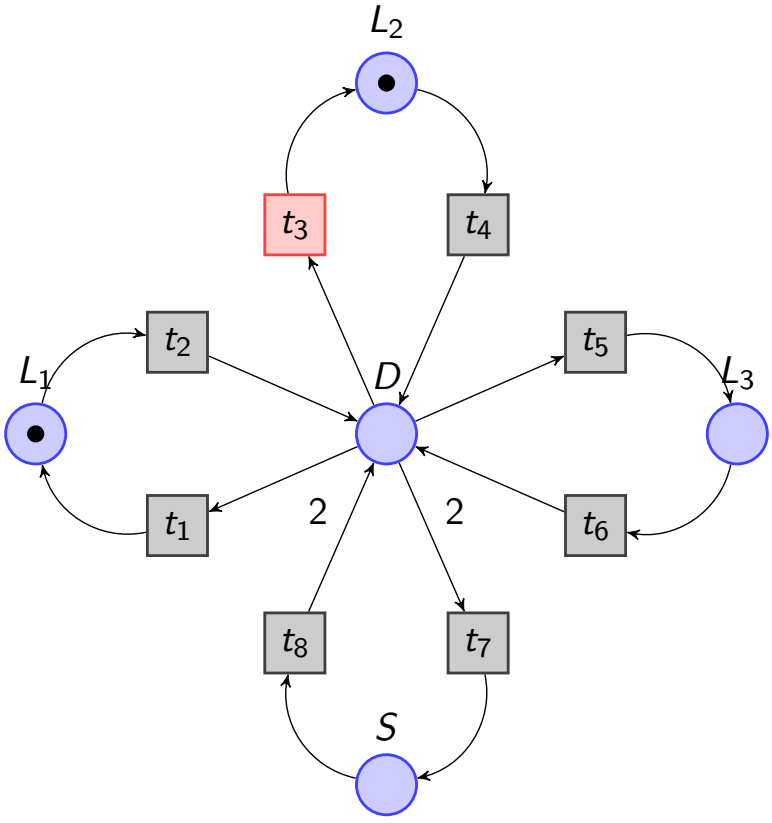
Beispiel: Leser-Schreiber-Problem



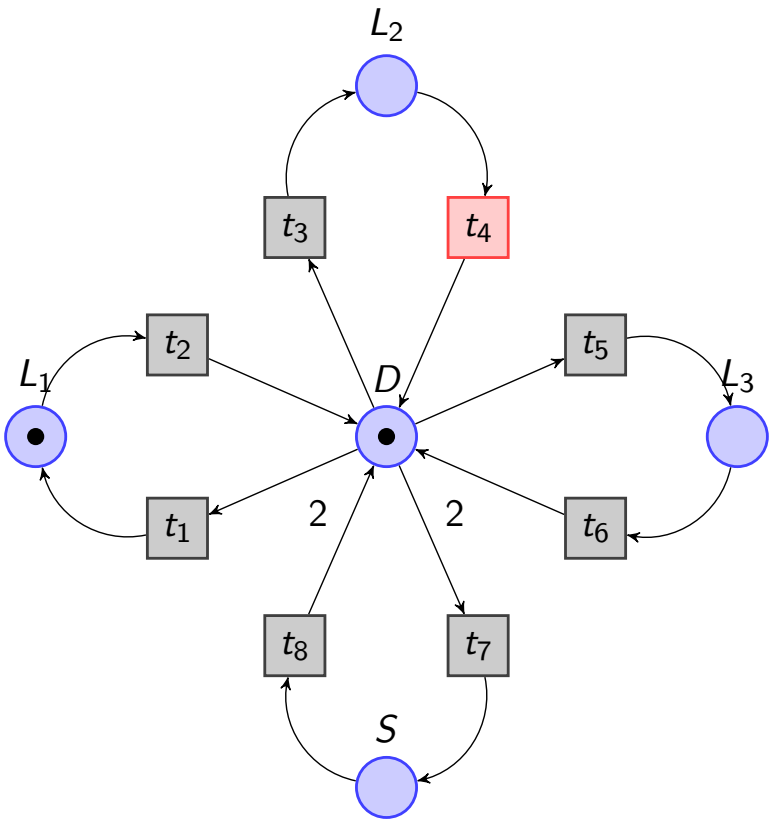
Beispiel: Leser-Schreiber-Problem



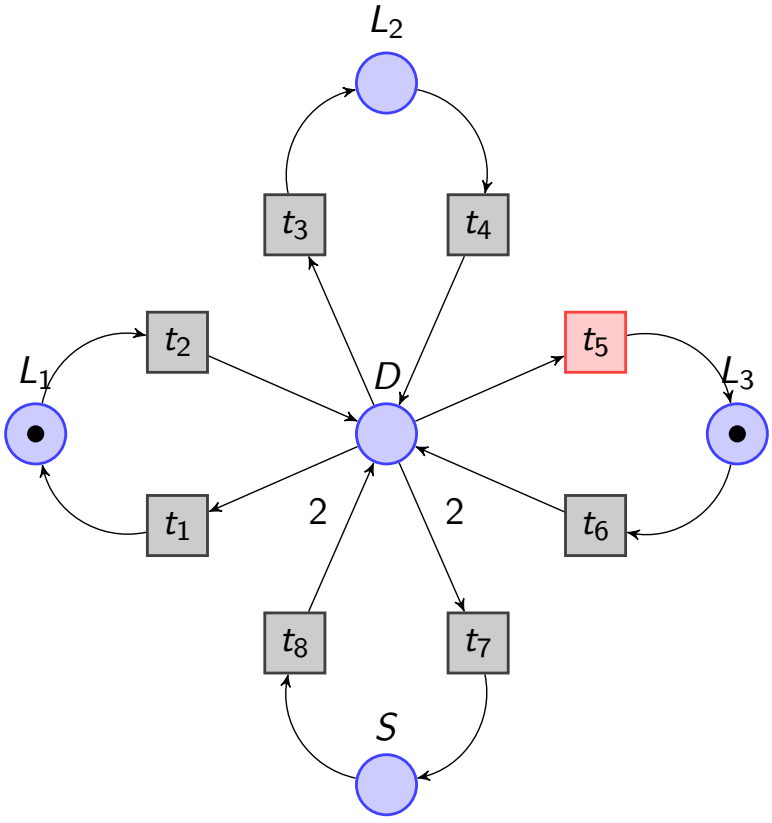
Beispiel: Leser-Schreiber-Problem



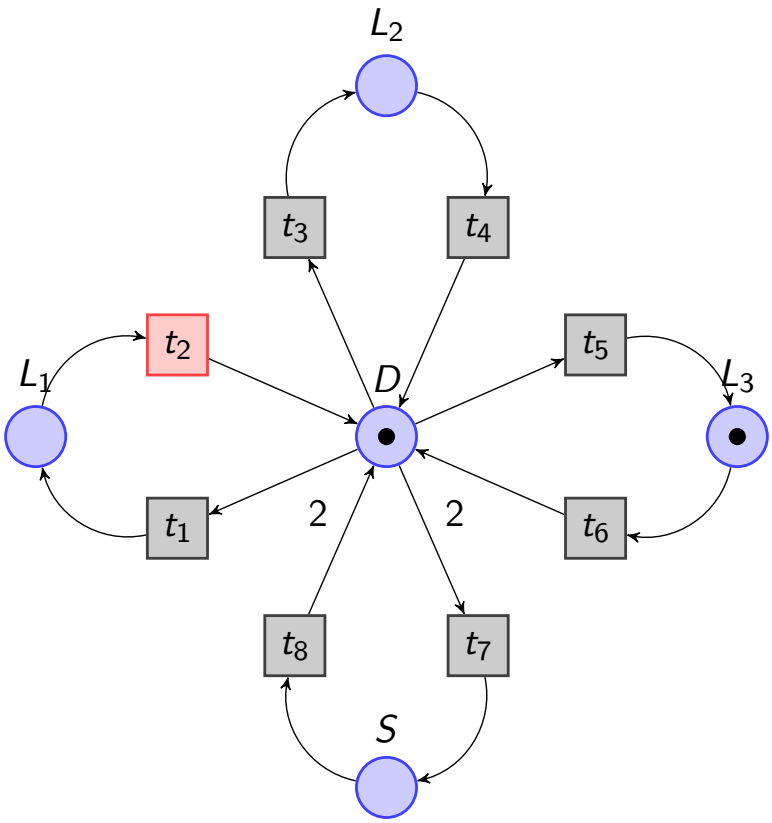
Beispiel: Leser-Schreiber-Problem



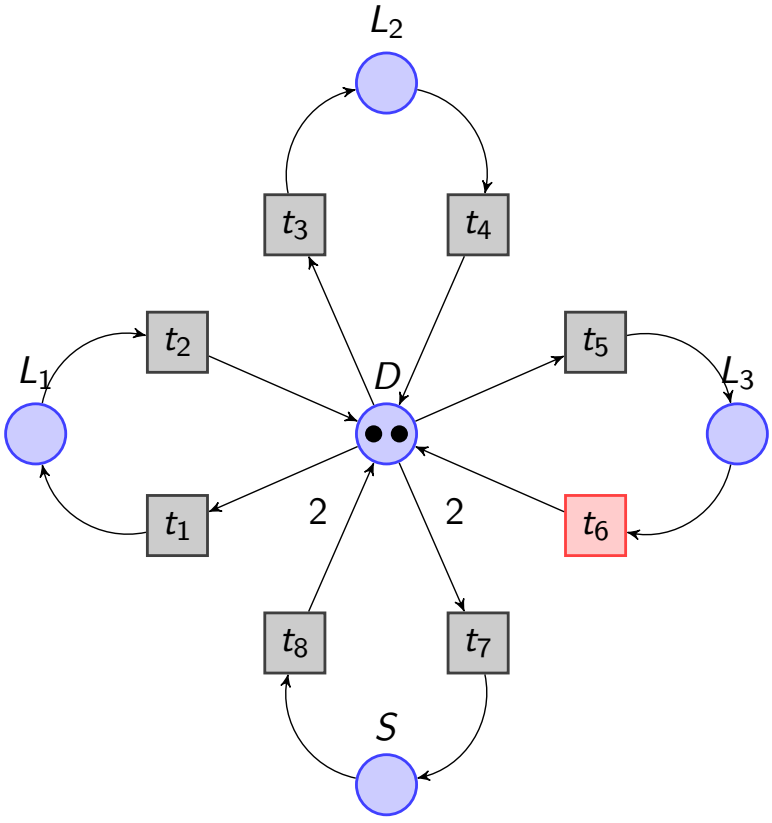
Beispiel: Leser-Schreiber-Problem



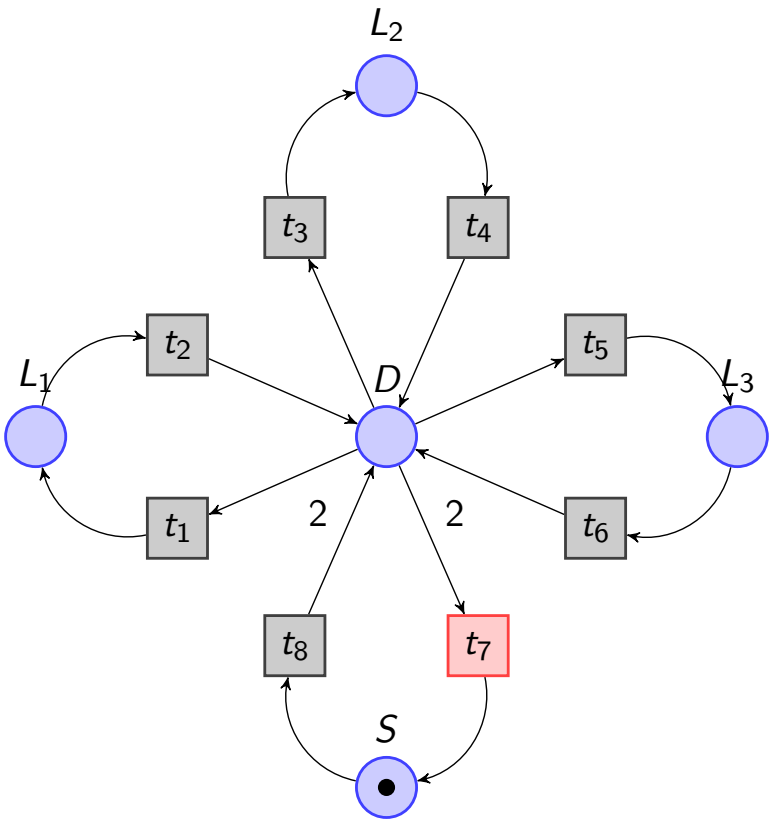
Beispiel: Leser-Schreiber-Problem



Beispiel: Leser-Schreiber-Problem



Beispiel: Leser-Schreiber-Problem



Beispiel: Leser-Schreiber-Problem