Negation

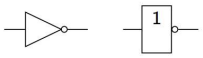
Ich gehe nicht ins Kino.

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', $!x$, $\overline{x}$, Nx, ...

Logikgatter: 

Konjunktion

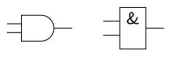
Der Himmel ist blau und die Sonne scheint.

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, $x\&y$, Kxy, ...

Logikgatter: 

Disjunktion, Alternative

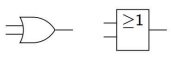
Ich trinke zum Essen Wein oder Bier (oder auch beides).

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: $x \vee y$, $x + y$, $x \mid y$, Axy

Logikgatter: 

Ausschließende Disjunktion (Antivalenz)

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

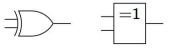
Trifft zu, wenn ich in genau einer der Stimmungslagen bin (die sich ausschließen).

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.

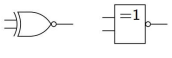
Trifft zu, wenn beide springen oder keiner.

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.

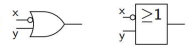
x	y	x implies y
1	1	1
1	0	0
0	1	1
0	0	1

„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, ...

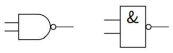
Logikgatter: 

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, ...

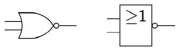
Logikgatter: 

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.

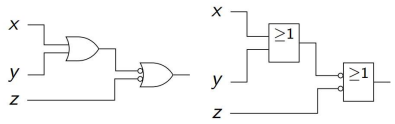
Logische Struktur: „Wenn x oder y, dann nicht z.“

Logische Funktion: $(x \text{ or } y)$ implies not z implies $(\text{or}(x, y), \text{not}(z))$

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: 

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

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.

Semantik aussagenlogischer Formeln

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

(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.

(v4) ist eine Abkürzung für:

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

(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	$((A \wedge \neg B) \supset \bot)$	bedeutet:
1	1	1 0 0 1 1 0	$I(A) = 1$, $I(B) = 1$: $\mathrm{val}_I(\cdots) = \cdots = 1$
1	0	1 1 1 0 0 0	$I(A) = 1$, $I(B) = 0$: $\mathrm{val}_I(\cdots) = \cdots = 0$
0	1	0 0 0 1 1 0	$I(A) = 0$, $I(B) = 1$: $\mathrm{val}_I(\cdots) = \cdots = 1$
0	0	0 0 1 0 1 0	$I(A) = 0$, $I(B) = 0$: $\mathrm{val}_I(\cdots) = \cdots = 1$

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

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.

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

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

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

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 $\supset)$.

$\langle \mathbb{B}, \text{and}, \text{or}, \text{not}, 0, 1 \rangle$ 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 $\neg$

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

Verschieben der Negation

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

Äquivalenzen für $\top$ und $\bot$

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

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

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

$I_{\vec{b}}$ ... 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.

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

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

$\text{DNF}_f = \bigvee\{ K_{\vec{b}} \mid f(\vec{b}) = 1,\ \vec{b} \in \mathbb{B}^n \}$ repräsentiert die Funktion f, d.h.:
$\text{val}_{I_{\vec{b}}}(\text{DNF}_f) = f(\vec{b})$ 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 $\vec{b} = (b_1, \dots, b_n) \in \mathbb{B}^n$:
$D_{\vec{b}} = \bigvee\{ A_i \mid b_i = 0,\ i = 1..n \} \vee \bigvee\{ \neg A_i \mid b_i = 1,\ i = 1..n \}$

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

$I_{\vec{b}}$ ... 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.

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

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

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

{not, and, or} ist funktional vollständig.

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:
$(^{(\neg)}A_{1,1} \vee {}^{(\neg)}A_{1,2} \vee {}^{(\neg)}A_{1,3} \vee \cdots) \wedge (^{(\neg)}A_{2,1} \vee {}^{(\neg)}A_{2,2} \vee {}^{(\neg)}A_{2,3} \vee \cdots) \wedge \cdots$

Konstruktion von DNFs/KNFs – Algebraische Methode

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

1. Ersetze alle Junktoren durch $\wedge$, $\vee$ und $\neg$.
   $A \uparrow B = \neg A \vee \neg B \quad A \downarrow B = \neg A \wedge \neg B \quad A \supset B = \neg A \vee B \quad A \subset B = A \vee \neg B$
   $A \equiv B = (\neg A \vee B) \wedge (A \vee \neg B) = (A \wedge B) \vee (\neg A \wedge \neg B)$
   $A \not\equiv B = (\neg A \vee \neg B) \wedge (A \vee B) = (A \wedge \neg B) \vee (\neg A \wedge B)$
2. Verschiebe Negationen nach innen, eliminiere Doppelnegationen.
   $\neg(A \wedge B) = \neg A \vee \neg B \quad \neg(A \vee B) = \neg A \wedge \neg B \quad \neg\neg A = A$
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 $\bot$.
   $A \wedge \top = A \quad A \wedge \bot = \bot \quad A \wedge \neg A = \bot \quad \neg\top = \bot$
   $A \vee \bot = A \quad A \vee \top = \top \quad A \vee \neg A = \top \quad \neg\bot = \top$

(Äquivalenzen werden hier von links nach rechts angewendet.)

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:

G gültig $\iff \neg G$ unerfüllbar
G widerlegbar $\iff \neg G$ erfüllbar
$G = H \iff G \not\equiv H$ unerfüllbar
$F_1, \dots, F_n \models G \iff F_1 \wedge \cdots \wedge F_n \wedge \neg G$ unerfüllbar

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!

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.

Formale Sprachen

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

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

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^*

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“

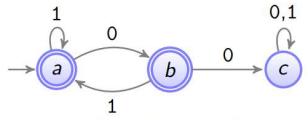
Erweiterte Übergangsfunktion $\delta^*\colon Q \times \Sigma^* \mapsto Q$

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

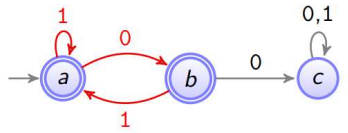
$\mathcal{A} = \langle Q, \Sigma, \delta, q_0, F \rangle$, wobei

- $Q = \{a, b, c\}$ ... 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.

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 $(q_1, ws, q_3) \in \delta^*$.

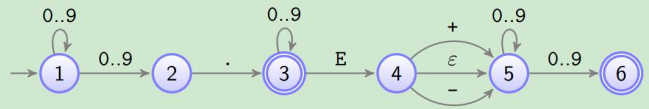
DEA: $\delta^*(q, \varepsilon) = q, \quad \delta^*(q, sw) = \delta^*(\delta(q, s), w)$

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

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	⋯	9	.	E	+	−	ε
AZ	1	{1,2}	⋯	{1,2}	{}	{}	{}	{}	{}
	2	{}	⋯	{}	{3}	{}	{}	{}	{}
EZ	3	{3}	⋯	{3}	{}	{4}	{}	{}	{}
	4	{}	⋯	{}	{}	{}	{5}	{5}	{5}
	5	{5,6}	⋯	{5,6}	{}	{}	{}	{}	{}
EZ	6	{}	⋯	{}	{}	{}	{}	{}	{}

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

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:

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

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

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 $q \in Q$
- $(q_1, w, w', q_2) \in \delta^*,\ (q_2, s, s', q_3) \in \delta \implies (q_1, ws, w's', q_3) \in \delta^*$

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

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

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:

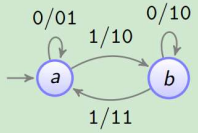
- Nur ein Anfangszustand: $I = \{q_0\}$
- Die Übergangsrelation ist deterministisch:
  - Der Folgezustand $\delta(q, s)$ ist eindeutig durch q und s bestimmt.
  - Keine ε-Übergänge
- Relationstupel: $(q, s, \gamma(q, s), \delta(q, s))$
- Alle Zustände sind Endzustände: $F = Q$

$(1:2)$-$(0,1)$-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	...

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

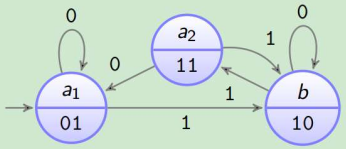
- Die Ausgabe erfolgt
  - ... bei Moore-Automaten durch den momentanen Zustand.
  - ... bei Mealy-Automaten beim Zustandswechsel, der durch Ursprungszustand und Eingabe festgelegt ist.
- Moore- und Mealy-Automaten besitzen dieselbe Ausdrucksstärke, sind aber schwächer als Transducer.
- Moore-Automaten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

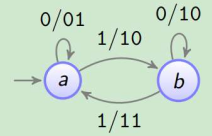


w:	ε	0	1	00	10	01	...	100	...
$[\mathcal{A}](w)$:	01	0101	0110	010101	011010	0110	...	01101010	...

Zum Vergleich: Mealy-Automat

δ	0	1
a	a	b
b	b	a

γ	0	1
a	01	10
b	10	11



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

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

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

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

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.

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

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

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

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

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

Verkettung

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

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

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

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

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

Beobachtungen:

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

Potenzen von $\{a, 42\}$

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

Potenzen eines Alphabets Σ

$\Sigma^0 = \{\varepsilon\}$ $\quad \Sigma^1 = \Sigma$
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 *:

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

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 $\Sigma = \{0, \dots, 9, ., E, +, -\}$

$real = digit \cdot digit^* \cdot \{.\} \cdot digit^* \cdot (\{\varepsilon\} \cup scale)$
$scale = \{E\} \cdot \{+, -, \varepsilon\} \cdot digit \cdot digit^*$
$digit = \{0, \dots, 9\} = \{0\} \cup \cdots \cup \{9\}$

Wichtig: Unterscheide Symbole des Alphabets von Meta-Symbolen!

$0, \dots, 9, ., E, +, -$ … Symbole des Alphabets
ε, *real*, *scale*, *digit* … Meta-Symbole, Abkürzungen

Reguläre Ausdrücke

Ausdrücke wie $digit \cdot digit^*$ 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:

$$\mathcal{L}(\emptyset) = \{\} \qquad \mathcal{L}(\varepsilon) = \{\varepsilon\} \qquad \mathcal{L}(s) = \{s\} \text{ für } s \in \Sigma$$

$$\mathcal{L}(r + r') = \mathcal{L}(r) \cup \mathcal{L}(r') \qquad \mathcal{L}(rr') = \mathcal{L}(r) \cdot \mathcal{L}(r') \qquad \mathcal{L}(r^*) = (\mathcal{L}(r))^*$$

Regulärer Ausdruck für die reellen Numerale

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

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

Die zugehörigen Sprachen:

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

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

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

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

$((a + b)^* + \varepsilon)^* = (a + b)^*$

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

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"
```

Posix Extended Regular Expressions (Ere)

regexp	trifft zu auf	*regexp*	trifft zu auf
\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

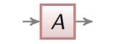
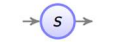
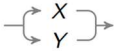
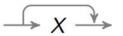
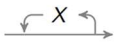
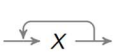
$digit = \{0, \ldots, 9\}$ `[0-9]`

$scale = \{\mathrm{E}\} \cdot \{+, -, \varepsilon\} \cdot digit \cdot digit^*$ `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

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	$X^i \cup X^{i+1} \cup \cdots \cup X^j$

Wiederholung: Reguläre Sprachen

Regulären Sprachen über einem Alphabet

Die Menge der regulären Sprachen über Σ, $\mathcal{L}_{\mathrm{reg}}(\Sigma)$, 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 $\Sigma = \{0, \ldots, 9, ., \mathrm{E}, +, -\}$

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

$scale = \{\mathrm{E}\} \cdot \{+, -, \varepsilon\} \cdot digit \cdot digit^*$

$digit = \{0, \ldots, 9\} = \{0\} \cup \cdots \cup \{9\}$

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.

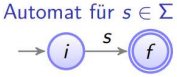
Nicht regulär sind Sprachen, deren Analyse ein unbegrenztes Gedächtnis erfordert:

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

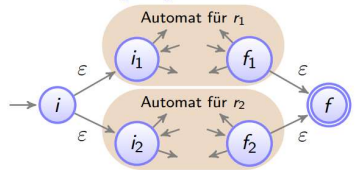
Vom regulären Ausdruck zum Automaten

Automat für $\emptyset$:

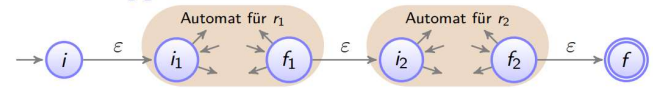
Automat für $s \in \Sigma \cup \{\varepsilon\}$:



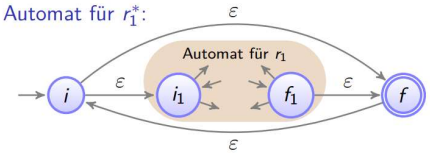
Automat für $r_1 + r_2$:



Automat für $r_1 r_2$:

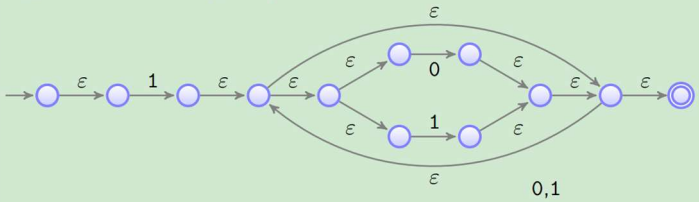


Automat für r_1^*:

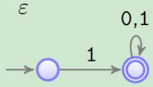


Automat für Binärnumerale ohne führende Null

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.

Ausdruckskraft regulärer Sprachen

Die regulären Sprachen sind genau jene, die von endlichen Automaten akzeptiert 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.

Deterministischer endlicher Automat (DEA)

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

- Q ... endliche Zustandsmenge
- Σ ... Eingabealphabet
- $\delta\colon Q \times \Sigma \mapsto Q$ ... Übergangsfunktion
- $i \in Q$ ... Anfangszustand
- $F \subseteq Q$ ... Endzustände

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

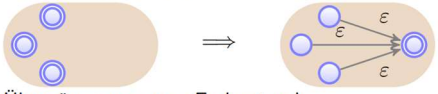
Unterschiede zu „normalen“ Automaten:

- 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.

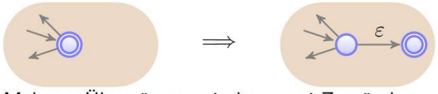
Umwandlung eines endlichen Automaten in einen verallgemeinerten

- Übergänge in den Anfangszustand oder Anfangszustand ist Endzustand:

- Mehrere Endzustände:



- Übergänge weg vom Endzustand:



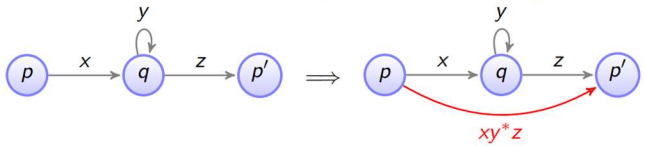
- Mehrere Übergänge zwischen zwei Zuständen:

Vom verallgemeinerten Automaten zum regulären Ausdruck

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

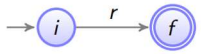
Für jeden Zustand $q \in Q - \{i, f\}$ führe folgende Schritte durch:

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



(x, y und z bezeichnen reguläre Ausdrücke.)

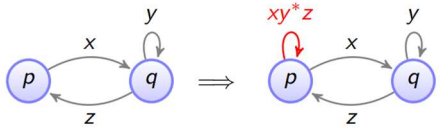
2. Entferne q und alle Kanten von und nach q aus dem Automaten.

Restautomat: 

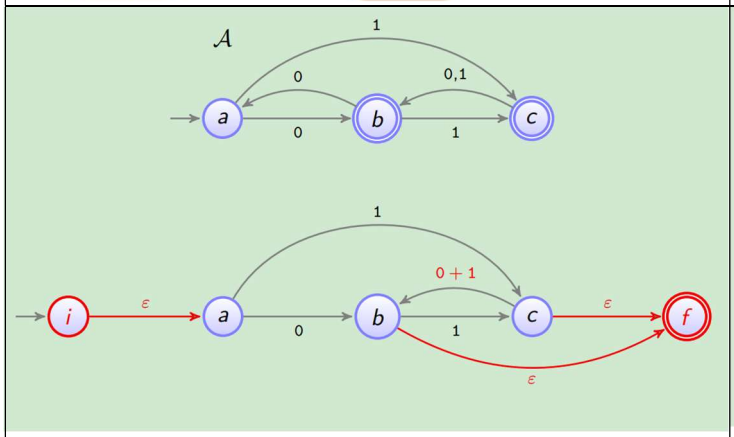
Ergebnis: r ist ein regulärer Ausdruck mit $\mathcal{L}(r) = \mathcal{L}(\mathcal{A})$.

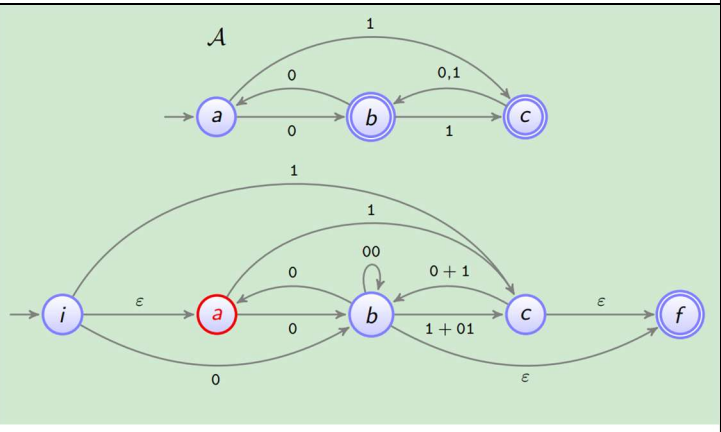
Anmerkungen:

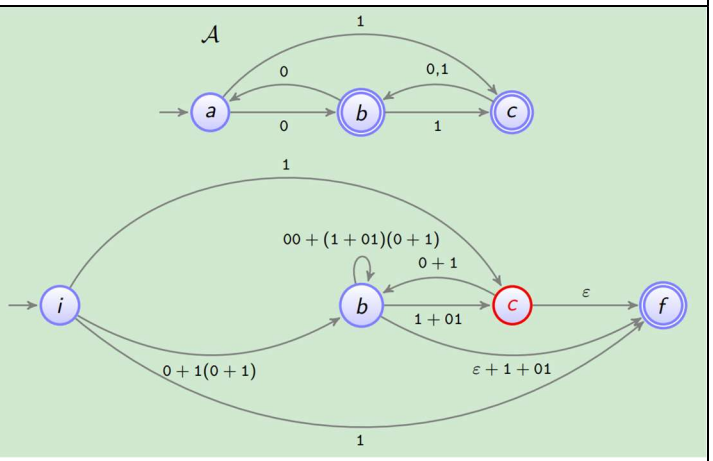
- Falls p und p' derselbe Knoten sind, erhält man:

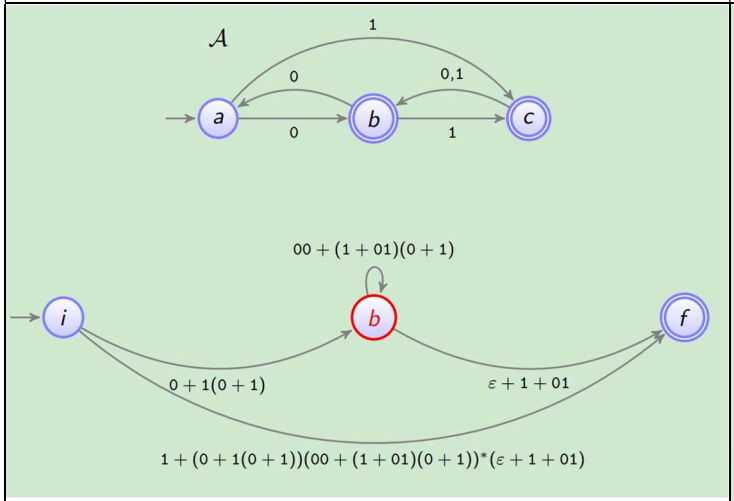


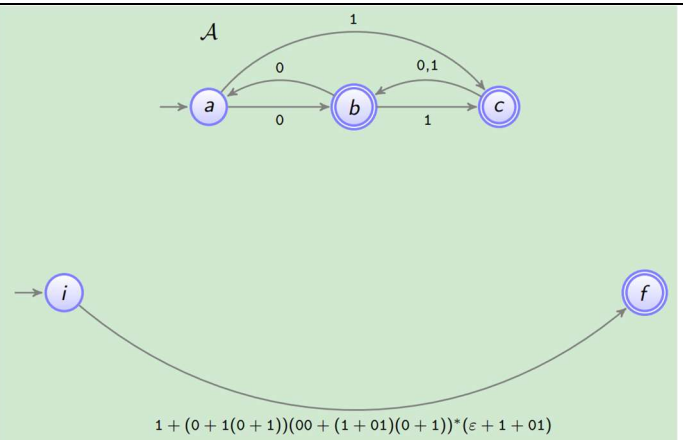
- Falls die Schleife mit dem Ausdruck y nicht existiert, entfällt y^*.
- Falls der Übergang p–p' bereits existiert, wird xy^*z addiert.











$r = 1 + (0 + 10 + 11)(00 + 10 + 11 + 010 + 011)^*(\varepsilon + 1 + 01)$

Es gilt: $\mathcal{L}(r) = \mathcal{L}(\mathcal{A})$.

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

$\{(),\ (()),\ ()(),\ ((())),\ (())(),\ ()()(),\ \ldots\}$

$\{\,\mathtt{a}^n\mathtt{b}^n \mid n \geq 0\,\}$ ist nicht regulär

$\{\varepsilon, \mathtt{ab}, \mathtt{aabb}, \mathtt{aaabbb}, \mathtt{aaaabbbb}, \mathtt{aaaaabbbbb}, \ldots\}$

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.

$G = \langle \{S\}, \{\mathtt{a}, \mathtt{b}\}, \{S \to \varepsilon \mid \mathtt{a}\,S\,\mathtt{b}\}, S\rangle$

$S \overset{*}{\Rightarrow} \mathtt{aabb}$, weil $\quad S \Rightarrow \mathtt{a}\,S\,\mathtt{b} \Rightarrow \mathtt{aa}\,S\,\mathtt{bb} \Rightarrow \mathtt{aa}\,\varepsilon\,\mathtt{bb} = \mathtt{aabb}$

$\mathcal{L}(G) = \{\, \mathtt{a}^n\mathtt{b}^n \mid n \geq 0 \,\} = \{\varepsilon, \mathtt{ab}, \mathtt{aabb}, \mathtt{aaabbb}, \ldots\}$

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. $S \overset{*}{\Rightarrow}_P w$

$\mathcal{L}(G) = \{\, w \in T^* \mid S \overset{*}{\Rightarrow} w \,\} = \{\, w \in T^* \mid S \overset{*}{\Rightarrow}_L w \,\}$
$= \{\, w \in T^* \mid S \overset{*}{\Rightarrow}_R w \,\} = \{\, w \in T^* \mid S \overset{*}{\Rightarrow}_P w \,\}$

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:

$$
\begin{aligned}
Fm &\Rightarrow_L (\ Fm\ Op\ Fm\ ) && \Rightarrow_L ((\ Fm\ Op\ Fm\ )\ Op\ Fm\ )\\
&\Rightarrow_L ((\ Var\ Op\ Fm\ )\ Op\ Fm\ ) && \Rightarrow_L ((\mathtt{A}\ Op\ Fm\ )\ Op\ Fm\ )\\
&\Rightarrow_L ((\mathtt{A} \uparrow Fm\ )\ Op\ Fm\ ) && \Rightarrow_L ((\mathtt{A} \uparrow Var\ )\ Op\ Fm\ )\\
&\Rightarrow_L ((\mathtt{A} \uparrow \mathtt{B})\ Op\ Fm\ ) && \Rightarrow_L ((\mathtt{A} \uparrow \mathtt{B}) \uparrow Fm\ )\\
&\Rightarrow_L ((\mathtt{A} \uparrow \mathtt{B}) \uparrow (\ Fm\ Op\ Fm\ )) && \Rightarrow_L ((\mathtt{A} \uparrow \mathtt{B}) \uparrow (\ Var\ Op\ Fm\ ))\\
&\Rightarrow_L ((\mathtt{A} \uparrow \mathtt{B}) \uparrow (\mathtt{A}\ Op\ Fm\ )) && \Rightarrow_L ((\mathtt{A} \uparrow \mathtt{B}) \uparrow (\mathtt{A} \uparrow Fm\ ))\\
&\Rightarrow_L ((\mathtt{A} \uparrow \mathtt{B}) \uparrow (\mathtt{A} \uparrow Var\ )) && \Rightarrow_L ((\mathtt{A} \uparrow \mathtt{B}) \uparrow (\mathtt{A} \uparrow \mathtt{B}))
\end{aligned}
$$

Wohlgeformte Klammerausdrücke

WKA ist die kleinste Menge, sodass

- $\varepsilon \in WKA$
- $(w) \in WKA$, wenn $w \in WKA$
- $w_1 w_2 \in WKA$, wenn $w_1, w_2 \in WKA$

$G = \langle \{W\},\quad \{(,)\},\quad \{W \to \varepsilon \mid (\ W\ ) \mid W\,W\},\quad W\ \rangle$

Beispiel einer Ableitung:

$$
\begin{aligned}
W &\Rightarrow_P W\,W\\
&\Rightarrow_P (W)(W)\\
&\Rightarrow_P ()(W\,W)\\
&\Rightarrow_P ()((W)(W))\\
&\Rightarrow_P ()(()())
\end{aligned}
$$

$\mathcal{L}(G) = \{\varepsilon, (), (()), ()(), ((())), (())(), ()(()), ()()(), \ldots\}$
$= WKA$

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

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

$G = \langle V, T, P, \mathit{Tabelle}\rangle$, wobei:

$P = \{$ *Tabelle* → `"\begin{tabular}"` [*Position*] *Spalten*
Zeilen
`"\end{tabular}"` ,

Position → `"[b]"` | `"[t]"` ,
Spalten → `"{"` *Spalte* { *Spalte* } `"}"` ,
Spalte → `"l"` | `"c"` | `"r"` ,

Zeilen → *Zeile* { `"\\"` *Zeile* } ,
Zeile → *Eintrag* { `"&"` *Eintrag* } ,

Eintrag → { *Tabelle* | *Zeichen* } ,
Zeichen → `"0"` | ... | `"9"` | `"a"` | ... | `"Z"` | `"."` | `"␣"` $\}$

$V = \{$*Tabelle*, *Position*, *Spalten*, *Spalte*, *Zeilen*, *Zeile*, *Eintrag*, *Zeichen*$\}$
$T = \{$ `"0"`, ..., `"9"`, `"a"`, ..., `"z"`, `"A"`, ..., `"Z"`,
`"."`, `"␣"`, `"{"`, `"}"`, `"["`, `"]"`, `"&"`, `"\"` $\}$

Nur eine Rekursion für Schachtelung der Tabellen notwendig!

Sokrates aus Sicht 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

- 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.	$\forall x\,(Mensch(x) \supset Sterblich(x))$
Sokrates ist ein Mensch.	$Mensch(sokrates)$
Sokrates ist sterblich.	$Sterblich(sokrates)$

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.
$\forall x(Mensch(x) \supset Sterblich(x)) \models Mensch(sokrates) \supset Sterblich(sokrates)$

$A,\ A \supset B \models B$ (Modus ponens)
Wenn A gilt und wenn B aus A folgt, dann gilt auch B.
$Mensch(sokrates),\ Mensch(sokrates) \supset Sterblich(sokrates) \models Sterblich(sokrates)$

$$\dfrac{Mensch(sokrates) \qquad \dfrac{\forall x(Mensch(x) \supset Sterblich(x))}{Mensch(sokrates) \supset Sterblich(sokrates)}}{Sterblich(sokrates)}$$

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

$Mensch(mutter(sokrates)),\quad Sterblich(x)$

Funktionssymbole: $mutter/1$, $sokrates/0$
Prädikatensymbole: $Mensch/1$, $Sterblich/1$
Variablensymbol: x

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) $f(t_1, \dots, t_n) \in \mathcal{T}$, falls $f/n \in \mathcal{F}$ und $t_1, \dots, t_n \in \mathcal{T}$. — Funktionssymbole mit der passenden Zahl an Termargumenten sind Terme.

Notation:

- Pre-/Postfixnotation für manche unären ($n = 1$) Funktionssymbole
- Infixnotation für manche binären ($n = 2$) Funktionssymbole
- Klammerneinsparungen durch Prioritäten

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:

$$I(f) \in \mathcal{U} \qquad \text{für } f/0 \in \mathcal{F}$$
$$I(f) : \mathcal{U}^n \mapsto \mathcal{U} \qquad \text{für } f/n \in \mathcal{F},\ n > 0$$

$\sigma : \mathcal{V} \mapsto \mathcal{U}$ ... 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) $\mathrm{val}_{I,\sigma}(f(t_1, \dots, t_n)) = I(f)(\mathrm{val}_{I,\sigma}(t_1), \dots, \mathrm{val}_{I,\sigma}(t_n))$ für $f/n \in \mathcal{F}$, $n > 0$.

Wert von (x+1)*(x+1+1) für $\sigma(x) = 0$

Arithmetik: $\mathcal{U} = \mathbb{Z}$, $I_1(1) = 1$, $I_1(+) = +$, $I_1(*) = \cdot$

$\mathrm{val}_{I_1,\sigma}((x+1)*(x+1+1)) = (0+1) \cdot (0+1+1) = 2$

Aussagenlogik: $\mathcal{U} = \mathbb{B}$, $I_2(1) = 1$, $I_2(+) = \text{or}$, $I_2(*) = \text{and}$

$\mathrm{val}_{I_2,\sigma}((x+1)*(x+1+1)) = \text{and}\,(\text{or}\,(0,1), \text{or}\,(0, \text{or}\,(1,1))) = 1$

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) $P(t_1, \dots, t_n) \in \mathcal{PF}$, wenn $P/n \in \mathcal{P}$ und $t_1, \dots, t_n \in \mathcal{T}(\mathcal{F}, \mathcal{V})$;
(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, $P(t_1, \dots, t_n)$ ... „Atomformeln“

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:

$$I(f) \in \mathcal{U} \qquad \text{für } f/0 \in \mathcal{F}$$
$$I(f) : \mathcal{U}^n \mapsto \mathcal{U} \qquad \text{für } f/n \in \mathcal{F},\ n > 0$$

$$I(P) \in \{0,1\} \qquad \text{für } P/0 \in \mathcal{P}$$
$$I(P) \subseteq \mathcal{U}^n \qquad \text{für } P/n \in \mathcal{P},\ n > 0$$
$$I(P) : \mathcal{U}^n \mapsto \{0,1\} \qquad \text{(alternative Sichtweise)}$$

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) $\mathrm{val}_{I,\sigma}(P(t_1, \dots, t_n)) = I(P)(\mathrm{val}_{I,\sigma}(t_1), \dots, \mathrm{val}_{I,\sigma}(t_n))$ für $P/n \in \mathcal{P}$;
(v3) $\mathrm{val}_{I,\sigma}(\top) = 1$ und $\mathrm{val}_{I,\sigma}(\bot) = 0$;
(v4) $\mathrm{val}_{I,\sigma}(\neg F) = \text{not}\,\mathrm{val}_{I,\sigma}(F)$;
(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):

$\mathrm{val}_{I,\sigma}(\forall x \exists y\, P(x, s(y))) = 1$

$\iff \mathrm{val}_{I,\sigma'}(\exists y\, P(x, s(y))) = 1$ für alle $\sigma' \overset{x}{\sim} \sigma$

$\iff \mathrm{val}_{I,\sigma''}(P(x, s(y))) = 1$ für mind. ein $\sigma'' \overset{y}{\sim} \sigma'$ und alle $\sigma' \overset{x}{\sim} \sigma$

$\iff I(P)(\mathrm{val}_{I,\sigma''}(x), \mathrm{val}_{I,\sigma''}(s(y))) = 1$ für mind. ein $\sigma'' \overset{y}{\sim} \sigma'$ und alle $\sigma' \overset{x}{\sim} \sigma$

$\iff \sigma''(x) = \sigma''(y) - 1$ 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:

$\sigma''(x) = (\sigma''(x) + 1) - 1$ für alle $\sigma''(x) = \sigma'(x) \in \mathbb{N}$

Gilt, daher ist $\forall x \exists y\, P(x, s(y))$ wahr in dieser Interpretation.

10

$\forall x \exists y\, P(x, s(y))$ ist falsch

... falls wir $\mathcal{U} = \mathbb{N}$, $I(s)(n) := n + 1$ und $I(P)(m, n) := (m = n)$ wählen (Variablenbelegungen σ beliebig):

$\mathrm{val}_{I,\sigma}(\forall x \exists y\, P(x, s(y))) = 1$

$\iff \mathrm{val}_{I,\sigma'}(\exists y\, P(x, s(y))) = 1$ für alle $\sigma' \overset{x}{\sim} \sigma$

$\iff \mathrm{val}_{I,\sigma''}(P(x, s(y))) = 1$ für mind. ein $\sigma'' \overset{y}{\sim} \sigma'$ und alle $\sigma' \overset{x}{\sim} \sigma$

$\iff I(P)(\mathrm{val}_{I,\sigma''}(x), \mathrm{val}_{I,\sigma''}(s(y))) = 1$ für mind. ein $\sigma'' \overset{y}{\sim} \sigma'$ und alle $\sigma' \overset{x}{\sim} \sigma$

$\iff \sigma''(x) = \sigma''(y) + 1$ 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.

11

Auf dem Spielplatz

$\mathcal{P} = \{Vater/1, Kind/1, SpieltMit/2\}$
$\mathcal{F} = \{albrecht/0, frieda/0\}$

$\mathcal{U} = \{\text{Albrecht, Bogdan, Erich, Frieda, Kathrin, Nina, Tamara}\}$
$I(Vater) = \{\text{Bogdan, Erich}\}$
$I(Kind) = \{\text{Albrecht, Frieda, Nina}\}$
$I(SpieltMit) = \{(\text{Albrecht, Frieda}), (\text{Bogdan, Albrecht}), (\text{Erich, Albrecht}), (\text{Erich, Frieda}), (\text{Frieda, Nina}), (\text{Frieda, Kathrin}), (\text{Frieda, Albrecht}), (\text{Nina, Frieda}), (\text{Tamara, Erich}), (\text{Tamara, Bogdan}), (\text{Tamara, Albrecht})\}$,
$I(albrecht) = \text{Albrecht}$
$I(frieda) = \text{Frieda}$

$\forall x\,(Vater(x) \supset \exists y\,(Kind(y) \wedge SpieltMit(x, y)))$

„Alle Väter spielen mit (mindestens) einem Kind."

Wahr in I, da Vater Bogdan mit Kind Albrecht und Vater Erich mit Kind Frieda spielt.

$\forall x\,(Vater(x) \wedge SpieltMit(x, albrecht))$

„Alle sind Väter und spielen mit Albrecht."

Falsch in I, da etwa Albrecht kein Vater ist.

$\exists x\,(Kind(x) \wedge \forall y\,(Kind(y) \supset SpieltMit(x, y)))$

„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.

$\forall x\,(SpieltMit(x, frieda) \not\equiv SpieltMit(x, albrecht))$

„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:

$\forall x\, F = F \qquad \exists x\, F = F$	
$\forall x\,(F \vee G) = F \vee \forall x\, G$	Distributivität $\forall/\vee$
$\exists x\,(F \wedge G) = F \wedge \exists x\, G$	Distributivität $\exists/\wedge$

15

Beispiel: Sport

$\mathcal{P} = \{Mensch/1, Sportart/1, Anstrengend/1, Betreibt/2\}$
$\mathcal{F} = \{handball/0, laufen/0\}$

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 ..." $\quad \forall x(Mensch(x) \supset \cdots)$
„Es gibt eine Sportart y ..." $\quad \exists y(Sportart(y) \wedge \cdots)$
„y ist anstrengend und x betreibt y" $\quad Anstrengend(y) \wedge Betreibt(x, y)$

$\forall x(Mensch(x) \supset \exists y(Sportart(y) \wedge Anstrengend(y) \wedge Betreibt(x, y)))$
$\forall x \exists y(Mensch(x) \supset (Sportart(y) \wedge Anstrengend(y) \wedge Betreibt(x, y)))$

$\exists x(Mensch(x) \wedge Anstrengend(x) \wedge Betreibt(x, handball) \wedge Betreibt(x, laufen))$

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)

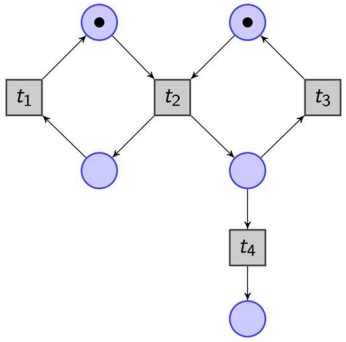
$FoolAt(x, y)$ ... you can fool x at time y
$Person(x)$ ... x is a person / x is one of the people
$PointInTime(y)$ ... y is a point in time

$$\begin{aligned} & \exists x(Person(x) \wedge \forall y(PointInTime(y) \supset FoolAt(x, y))) \\ \wedge\ & \forall x(Person(x) \supset \exists y(PointInTime(y) \wedge FoolAt(x, y))) \\ \wedge\ & \neg\forall x(Person(x) \supset \forall y(PointInTime(y) \supset FoolAt(x, y))) \end{aligned}$$

„You can fool some of the people all of the time,
and all of the people some of the time,
but you can make a fool of yourself anytime." [Unix fortune utility]

13

Petrinetze: Motivation

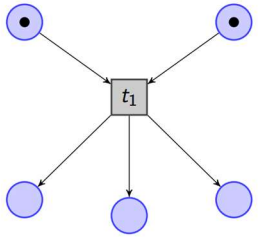


Notation:

- Stellen (dargestellt als Kreise): mögliche Plätze für Ressourcen
- Marken (dargestellt als kleine gefüllte Kreise): Ressourcen
- Transitionen (dargestellt als Rechtecke oder Balken): Systemübergänge

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.

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.

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.

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