

FGI-1 Repetitorium (2. Termin):

Logik

[Tag 1]

Jan Henrik Röwekamp
(8roeweka)

18.09., 19.09. & 20.09.2013

Organisation

- 3 Termine (Mi., Do. & Fr.,
der 18.09., 19.09. & 20.09.13)
- 13:00 – 17:00 Uhr in Raum B-201
- 30 Minuten Pause in der Mitte
(richtet sich nach dem Stoff)
- Folien online im Wiki unter:
<http://tinyurl.com/FGI1Rep>

Zu mir

- Jan Henrik Röwekamp
- M.Sc. Informatik (10. Hochschul-Semester)
- Seit 2008 als Student an der Uni Hamburg
- Mail/Jabber: 8roeweka (@inf...)
- Diverse Semester als Gruppenleiter gearbeitet (3x DM, 2x FGI-1, 1x FGI-2 & 1x AD)
- Autor der OLAT-Selbsttest-Fragen

Allgemein zum Ablauf

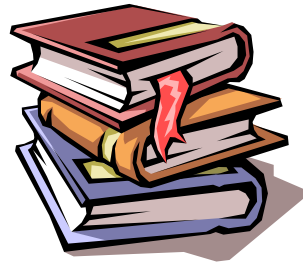
- Fragen zum aktuell vorgestellten Thema können jederzeit gestellt werden.
- Fragen zu nicht (in diesem Moment oder aber gar nicht) besprochenen Themen bitte am Ende des jeweiligen Tages
- Ich neige ein wenig zu schnell reden. Wenn es zu schnell wird, sagt bitte was. 😊

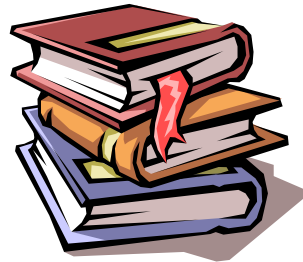
Was bringt das hier?

- Die wichtigsten Inhalte der Vorlesung werden wiederholt.
- Das Repetitorium hat nicht den Anspruch den gesamten Inhalt des Moduls zu wiederholen!
- Insbesondere weiterführende Fragestellungen, Beweise und Exkurse werden weggelassen.
- Das Repetitorium deckt nicht alles ab, was in der Klausur gefragt werden könnte, jedoch einen guten Teil dessen.

Was ist neu?

- „Bugfixes“
- Zusätzliche Folien zu komplizierten Themen



- Das Symbol  steht für Teile des Stoffs, die generell sehr wichtig sind. Das Repetitorium beschränkt sich zwar auf die wichtigsten Inhalte, jedoch gibt es auch hier Abstufungen. Das Symbol steht immer neben den wichtigen Inhalten. Ist es an der Überschrift zu finden, betrifft es das gesamte Thema.

Ablauf – Tag 1

- Organisation, Vorstellung, etc.
- Logik: Allgemeine Motivation
- AS_{AL} und L_{AL}
- Strukturelle Induktion in der Aussagenlogik
- Semantik, Äquivalenz und Normalformen
- Folgerbarkeit, Ableitungen und Inferenzregeln
- <Pause>
- Hornformeln und der Markierungsalgorithmus
- Resolution
- Prädikatenlogik allgemein
- Strukturen und Auswertungen von PL-Formeln

Ablauf – Tag 2

- Wdh.: Prädikatenlogik & Strukturen
- Prädikatenlogische Resolution
 - Formel schließen und bereinigte Pränexform (BPF)
 - Skolemisierung, Unifikation und Resolution
- Warum Automaten?
- Die Chomsky-Hierarchie
- Reguläre Mengen, DFA's und NFA's
- <Pause>
- Der Potenzautomat: Vom NFA zum DFA
- ϵ -FA & Überführung zum NFA (ϵ -frei machen)
- Kleene-Verfahren

Ablauf – Tag 3

- Pumping Lemma (uvw und uvwxy-Theorem)
- Grammatiken (CFG; Typ 2 und Typ 3)
- Grammatik-Normalformen
- Kellerautomaten
- Andere Grammatiken (Typ 0 und Typ 1)
- Turing-Maschinen
- <Pause>
- Entscheidbarkeit & Berechenbarkeitstheorie
- Komplexitätstheorie & O-Notation
- Komplexitätsklassen
- NP-Vollständigkeit

Logik: Allgemeine Motivation

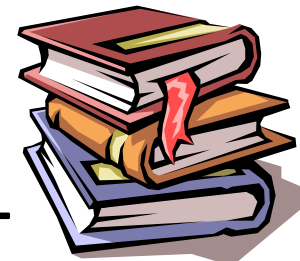
- „Wenn 16 Nägel und 4 Bretter vorhanden sind, können wir einen Rahmen bauen. Wenn 4 Pfosten und ein Rahmen und Leim vorhanden sind, können wir ein Bettgestell bauen. Aus einem Bettgestell und 20 Leisten kann ein Bett gebaut werden.“
- Bekannt: Vorhandene Rohmaterialien.
- Frage: Können wir ein Bett bauen?
- Modellierung der Realwelt mit Logik.
- Logik ist allgegenwärtig.

Ablauf – Tag 1

- Organisation, Vorstellung, etc.
- Logik: Allgemeine Motivation
- **AS_{AL} und L_{AL}**
- Strukturelle Induktion in der Aussagenlogik
- Semantik, Äquivalenz und Normalformen
- Folgerbarkeit, Ableitungen und Inferenzregeln
- <Pause>
- Hornformeln und der Markierungsalgorithmus
- Resolution
- Prädikatenlogik allgemein
- Strukturen und Auswertungen von PL-Formeln

Atomare Aussagen

- Elementare, nicht trennbare, Einzelaussagen
- Entsprechen **einem nicht weiter zerlegbaren Satz**, der wahr oder falsch sein kann.
- Beispiele für atomare Aussagen:
 - „Das Auto ist rot.“
 - „Es regnet.“
 - „7 ist keine Zahl.“
- Notiert mit einem Großbuchstaben: z.B.: A, B, C
- Die *Menge aller atomaren Aussagen* wird notiert als: **AS_{AL}**

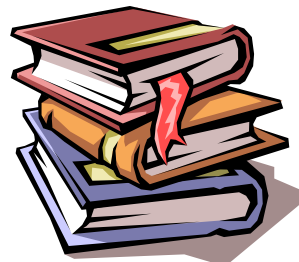


Die aussagenlogische Sprache L_{AL}

- Atomare Formeln sind nicht sehr aussagekräftig.
- Kombination von Aussagen zu (komplexen) Formeln durch sog. Junktoren:
 - Zweistellig:
 - Konjunktion: $\wedge$ („und“)
 - Disjunktion: $\vee$ („oder“)
 - Implikation: $\Rightarrow$ („wenn“)
 - Biimplikation: $\Leftrightarrow$ („genau dann, wenn“)
 - Einstellig:
 - Negation: $\neg$ („nicht“)

Die aussagenlogische Sprache L_{AL}

- Formeln werden notiert mit Großbuchstaben ab F. Z.B.: F, G oder H
- Junktoren lassen sich erneut auf Formeln anwenden.
- Auf diese Weise lassen sich alle aussagenlogischen Formeln erzeugen.
- Die *Menge aller Formeln*, die so erzeugt werden können wird als L_{AL} bezeichnet.



Ablauf – Tag 1

- Organisation, Vorstellung, etc.
- Logik: Allgemeine Motivation
- AS_{AL} und L_{AL}
- **Strukturelle Induktion in der Aussagenlogik**
- Semantik, Äquivalenz und Normalformen
- Folgerbarkeit, Ableitungen und Inferenzregeln
- <Pause>
- Hornformeln und der Markierungsalgorithmus
- Resolution
- Prädikatenlogik allgemein
- **Strukturen und Auswertungen von PL-Formeln**

Die natürlichen Zahlen mal anders

- $\mathbb{N} = \{1, 2, \dots\}$
- Geht diese Definition auch anders?
 - Sei 1 eine natürliche Zahl
 - Wenn k eine natürliche Zahl ist, dann ist $k+1$ auch eine.
- Trifft alle natürlichen Zahlen und ist *rekursiv*.
- Können Funktionen auch rekursiv definiert werden? Beispiel: $f: \mathbb{N} \rightarrow \mathbb{N}$ mit $f(x) = 3x+5$:
 - $f(1) = 8$
 - $f(n+1) = f(n) + 3$

Die natürlichen Zahlen mal anders

- $\mathbb{N} = \{1, 2, \dots\}$
- Geht diese Definition auch anders?

– Sei 1 eine natürliche Zahl	Basis
– Wenn k eine natürliche Zahl ist, dann ist $k+1$ auch eine.	Rekursiver Aufruf

- Trifft alle natürlichen Zahlen und ist *rekursiv*.
- Können Funktionen auch rekursiv definiert werden? Beispiel: $f: \mathbb{N} \rightarrow \mathbb{N}$ mit $f(x) = 3x+5$:
 - $f(1) = 8$
 - $f(n+1) = f(n) + 3$

Dieses mal mit L_{AL} statt $\mathbb{N}$

- Jedes atomare Aussagensymbol $A \in \mathbf{AS}_{AL}$ ist eine Formel aus L_{AL} .
- Seien F und $G \in L_{AL}$.
Dann ist auch:
 - $\neg F \in L_{AL}$.
 - $(F \wedge G) \in L_{AL}, (F \vee G) \in L_{AL}, (F \Rightarrow G) \in L_{AL}, (F \Leftrightarrow G) \in L_{AL}$

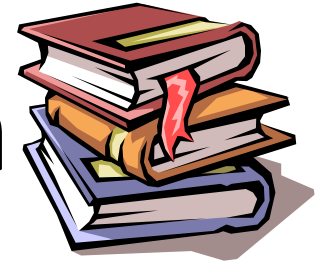
Dieses mal mit L_{AL} statt $\mathbb{N}$

- Jedes atomare Aussagensymbol $A \in \mathbf{AS}_{AL}$ ist eine Formel aus L_{AL} . Basis
- Seien F und $G \in L_{AL}$.
Dann ist auch: Rekursiver Aufruf
 - $\neg F \in L_{AL}$.
 - $(F \wedge G) \in L_{AL}, (F \vee G) \in L_{AL}, (F \Rightarrow G) \in L_{AL}, (F \Leftrightarrow G) \in L_{AL}$

Funktionen auf L_{AL} : analog zu $\mathbb{N}$

- Der Grad g einer Formel ist eine Funktion der Signatur $g : L_{AL} \rightarrow \mathbb{N}$ und wie folgt definiert:
- Sei $A \in \mathbf{AS}_{AL}$, dann ist:
 - $g(A) = 0$
- Seien F und $G \in L_{AL}$.
Dann ist:
 - $g(\neg F) = g(F) + 1$
 - $g(F \wedge G) = g(F \vee G) = g(F \Rightarrow G) = g(F \Leftrightarrow G) = g(F) + g(G) + 1$

Strukturelle Induktion



- Gegeben: These zu allen Formeln aus L_{AL} .
- Gesucht: Beweis der Gültigkeit der These.
- Idee:
 - Formeln können nur mit atomaren Aussagen und Junktoren erzeugt werden.
 - These zeigen für atomare Formeln
 - These zeigen für Junktoranwendung, falls die These bereits je für die zusammengeführten Formeln galt.
- Analogie zur vollständigen Induktion, nur basierend auf der Struktur der Formeln.

Vollst. vs. Strukturelle Induktion

Natürliche Zahlen:

Aussagenlog. Formeln:

1

$A \in \mathbf{AS}_{AL}$

Basierend auf n:

Basierend auf F und G:

n+1

$\neg F,$

$(F \wedge G), (F \vee G),$

$(F \Rightarrow G), (F \Leftrightarrow G)$

Vollst. vs. Strukturelle Induktion

Natürliche Zahlen:

Aussagenlog. Formeln:

Induktionsanfang

1

$A \in \mathbf{AS}_{AL}$

Induktionsannahme

Basierend auf n:

Basierend auf F und G:

n+1

Induktionsschritt

$\neg F,$
 $(F \wedge G), (F \vee G),$
 $(F \Rightarrow G), (F \Leftrightarrow G)$

Strukturelle Induktion: Beispiel

- Weisen Sie mittels struktureller Induktion nach, dass in jeder Formel die Zahl der öffnenden Klammern gleich der Zahl der schließenden Klammern ist.

Formal:

$$\forall F \in L_{AL}: |F|_{(} = |F|_{)}$$

Strukturelle Induktion: Lösung

- Zu Zeigen: $\forall F \in L_{AL}: |F|_{(} = |F|_{)}$

- Für $A \in AS_{AL}$ gilt offensichtlich:
 $|A|_{(} = 0 = |A|_{)}$ (*Ind.anfang*)

- Gelte für **zwei Formeln F und G** nun bereits:
 $|F|_{(} = |F|_{)}$ und $|G|_{(} = |G|_{)}$ (*Ind.annahme*)

- Sei $H = \neg F$. Es gilt: $|H|_{(} = |F|_{(} =_{IA} |F|_{)} = |H|_{)}$
- Sei $H = (F \wedge G)$ (*Ind.schritt*)
 - Es folgt: $|H|_{(} = |F|_{(} + |G|_{(} + 1$
 $=_{IA} |F|_{)} + |G|_{)} + 1 = |H|_{)}$
- Analog für $\vee, \Rightarrow$ und $\Leftrightarrow$.

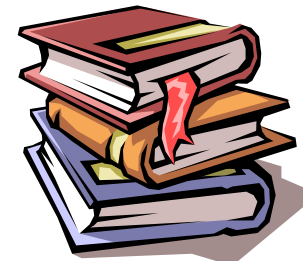


Ablauf – Tag 1

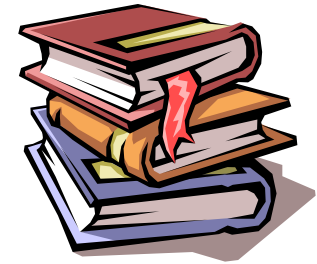
- Organisation, Vorstellung, etc.
- Logik: Allgemeine Motivation
- AS_{AL} und L_{AL}
- Strukturelle Induktion in der Aussagenlogik
- **Semantik**, Äquivalenz und Normalformen
- Folgerbarkeit, Ableitungen und Inferenzregeln
- <Pause>
- Hornformeln und der Markierungsalgorithmus
- Resolution
- Prädikatenlogik allgemein
- Strukturen und Auswertungen von PL-Formeln

Belegungen und Modelle

- Um eine Formel auszuwerten, müssen allen Aussagensymbolen Wahrheitswerte zugeordnet werden.
- Eine solche Zuordnung wird Belegung genannt und A geschrieben.
- Eine Belegung kann eine Formel F wahr machen: $A(F) = 1$ oder aber falsch: $A(F) = 0$
- Falls $A(F) = 1$ gilt, wird A auch „**Modell von F** “ genannt.
- Wie berechnet sich $A(F)$?

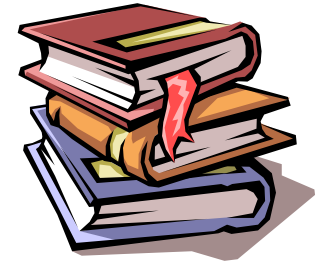


Wahrheitsverläufe der Junktoren



A	B	$(A \vee B)$	$(A \wedge B)$	$(A \Rightarrow B)$	$(A \Leftrightarrow B)$	$\neg A$
0	0	0	0	1	1	1
0	1	1	0	1	0	1
1	0	1	0	0	0	0
1	1	1	1	1	1	0

Erfüllbarkeit und Bezeichnungen



- 5 Begriffe basierend auf Belegungen zu Formeln: Eine Formel kann sein:
 - **Unerfüllbar:** *Alle Belegungen sind keine Modelle*
 - **Falsifizierbar:** *Mind. eine Belegung ist kein Modell*
 - **Kontingent:** *Mind. je eine Belegung ist ein Modell und eine kein Modell. Die Formel ist also sowohl falsifizierbar als auch erfüllbar*
 - **Erfüllbar:** *Mind. Eine Belegung ist ein Modell*
 - **Tautologisch/(Allgemein)gültig:** *Alle Belegungen sind Modelle*

Beispiele: Erfüllbarkeit

- **Unerfüllbar:** $F = (\neg B \wedge B)$
 - Sowohl $A(B) = 0$, als auch $A(B) = 1$ führen zu $A(F) = 0$
- **Falsifizierbar:** $G = (B \Rightarrow C)$
 - $A(B) = 1, A(C) = 0$ führt zu $A(G) = 0$
- **Kontingent:** $H = (B \Rightarrow C)$
 - $A(B) = 1, A(C) = 0$ führt zu $A(H) = 0$
 - $A(B) = 0, A(C) = 0$ führt zu $A(H) = 1$
- **Erfüllbar:** $I = (B \Rightarrow C)$
 - $A(B) = 1, A(C) = 1$ führt zu $A(I) = 1$
- **Tautologisch/(Allgemein)gültig:** $K = (\neg B \vee B)$
 - Sowohl $A(B) = 0$, als auch $A(B) = 1$ führen zu $A(K) = 1$

Ablauf – Tag 1

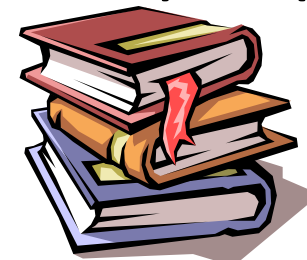
- Organisation, Vorstellung, etc.
- Logik: Allgemeine Motivation
- AS_{AL} und L_{AL}
- Strukturelle Induktion in der Aussagenlogik
- Semantik, Äquivalenz und Normalformen
- Folgerbarkeit, Ableitungen und Inferenzregeln
- <Pause>
- Hornformeln und der Markierungsalgorithmus
- Resolution
- Prädikatenlogik allgemein
- Strukturen und Auswertungen von PL-Formeln

Äquivalenz von Formeln

- Betrachte die Formeln: $(A \Rightarrow B)$ und $(\neg A \vee B)$

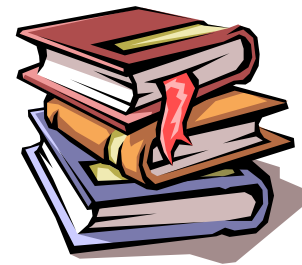
A	B	$\neg A$	$(A \Rightarrow B)$	$(\neg A \vee B)$
0	0	1	1	1
0	1	1	1	1
1	0	0	0	0
1	1	0	1	1

- Offensichtlich haben die beiden Formeln den gleichen Wahrheitsverlauf: $(A \Rightarrow B) \equiv (\neg A \vee B)$
- Mehr davon: vgl. **Satz 4.7**
(den sollte man können!)



Normalformen: KNF und DNF

- Alle Formeln aus L_{AL} können in Normalformen überführt werden.
- Konjunktive Normalform (KNF):
 - $((... \vee ... \vee ...) \wedge (... \vee ... \vee ...) \wedge ...)$
- Disjunktive Normalform (DNF):
 - $((... \wedge ... \wedge ...) \vee (... \wedge ... \wedge ...) \vee ...)$
- Erzeugung in 3 Schritten:
 - Implikationen auflösen
 - Negationen nach Innen bringen
 - Distributivität anwenden bis KNF/DNF entsteht



KNF und DNF Beispiel

- Bringen Sie die folgende Formel sowohl in **KNF** als auch in **DNF**:

$$\neg(A \Rightarrow B) \vee C$$

KNF und DNF Beispiel: Lösung

- $\neg(A \Rightarrow B) \vee C$
- Implikationen auflösen:
 - $\neg(A \Rightarrow B) \vee C \equiv \neg(\neg A \vee B) \vee C$
- Negationen nach Innen bringen:
 - $\neg(\neg A \vee B) \vee C \equiv (\neg\neg A \wedge \neg B) \vee C \equiv (A \wedge \neg B) \vee C$
- Distributivität anwenden:
 - $(A \wedge \neg B) \vee C \equiv \underline{(A \vee C) \wedge (\neg B \vee C)}$ KNF
 - $\underline{(A \wedge \neg B) \vee C}$ DNF

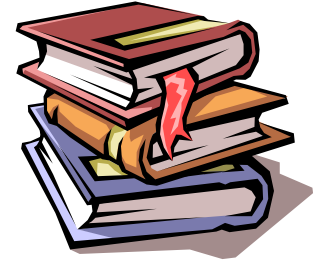
Ablauf – Tag 1

- Organisation, Vorstellung, etc.
- Logik: Allgemeine Motivation
- AS_{AL} und L_{AL}
- Strukturelle Induktion in der Aussagenlogik
- Semantik, Äquivalenz und Normalformen
- **Folgerbarkeit**, Ableitungen und Inferenzregeln
- <Pause>
- Hornformeln und der Markierungsalgorithmus
- Resolution
- Prädikatenlogik allgemein
- Strukturen und Auswertungen von PL-Formeln

Folgerung

- Einführungsbeispiel:
- „Wenn 16 Nägel und 4 Bretter vorhanden sind, können wir einen Rahmen bauen. Wenn 4 Pfosten und ein Rahmen und Leim vorhanden sind, können wir ein Bettgestell bauen. Aus einem Bettgestell und 20 Leisten kann ein Bett gebaut werden.“
- „Es kann ein Bett gebaut werden.“
- Lässt sich diese Aussage aus dem gegebenen Material und den Regeln folgern?

Folgerung



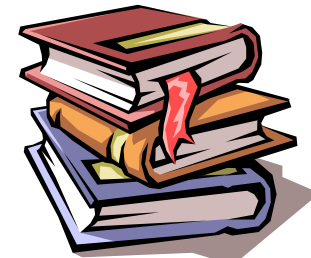
- Eine Formel G folgt aus einer Formel F , **wenn alle Modelle von F auch Modelle von G sind.**
- Notiert als: $F \models G$ („ G folgt aus F “)
- Das Folgerungszeichen hat noch 4 weitere Bedeutungen (**M** : Formelmenge, A : Belegung):
 - $M \models G$ („ G folgt aus der Formelmenge M “)
 - $A \models G$ („Die Belegung A erfüllt G “)
 - $\models G$ („ G ist eine Tautologie“)
 - $G \models$ („ G ist unerfüllbar“)

Das Symbol $\models$ im Detail

- Seien **M** Formelmenge und A eine Belegung.
- **M** $\models$ G („G folgt aus der Formelmenge M“)
 - Alle Belegungen, die *Modell für alle Formeln* aus M sind, sind auch Modell von G.
- A $\models$ G („Die Belegung A erfüllt G“)
 - Die Belegung A ist ein Modell von G.
- $\models$ G („G ist eine Tautologie“)
 - G lässt sich aus der leeren Menge folgern (s.Fol. 45f)
- G $\models$ („G ist unerfüllbar“)
 - (Selbst) die (unerfüllbare) leere Menge folgt aus G.

Folgerung

- Was bedeutet es, dass eine Formel G aus einer Formelmenge $\mathbf{M}$ folgt? Sei $\mathbf{M} = \{F_1, F_2, \dots, F_n\}$.
 - Wenn eine Belegung A gleichzeitig alle Formeln der Menge $\mathbf{M}$ erfüllt, muss auch $A(G) = 1$ sein.
 - $\mathbf{M} \models G$ gdw. $F_1 \wedge F_2 \wedge \dots \wedge F_n \models G$
- Wie weist man Folgerung nach?
 - Wahrheitstafel
 - Mittels dem Theorem: $\mathbf{M} \models G$ gdw. $\mathbf{M} \cup \{\neg G\} \models$
 - Sehr hilfreich da wir diverse Methoden zum Nachweis von Unerfüllbarkeit kennenlernen werden.



Ablauf – Tag 1

- Organisation, Vorstellung, etc.
- Logik: Allgemeine Motivation
- AS_{AL} und L_{AL}
- Strukturelle Induktion in der Aussagenlogik
- Semantik, Äquivalenz und Normalformen
- Folgerbarkeit, **Ableitungen und Inferenzregeln**
- <Pause>
- Hornformeln und der Markierungsalgorithmus
- Resolution
- Prädikatenlogik allgemein
- **Strukturen und Auswertungen von PL-Formeln**

Inferenzregeln

- Gesucht: Verfahren um aus einer Formelmengemenge daraus folgende Formeln abzuleiten.

- Methode dazu: Inferenzregeln

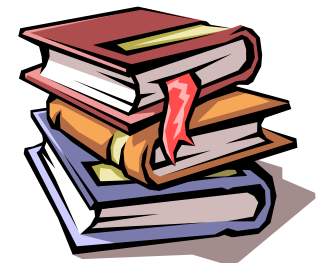
- Inferenzregeln besitzen:

- (Formel)mengenan Prämissen
- Eine Konklusion

$$\frac{A, (A \Rightarrow B)}{B}$$

$$A, (A \Rightarrow B) \therefore B$$

- Damit die Inferenzregel korrekt ist, muss **die Konklusion aus den Prämissen folgen** (andernfalls wäre die „Inferenzregel“ nicht sehr hilfreich oder sinnvoll)



Substitution

- Ersetzung von atomaren Aussagesymbolen mit (komplexen) Formeln. (**Keine äq. Umf.!**)
- Wenn uniform innerhalb und auch formelübergreifend substituiert wird (also **alle** Vorkommen eines Aussagesymbols gleichartig ersetzt werden), bleibt erhalten:
 - (Un)erfüllbarkeit und Allgemeingültigkeit
 - Äquivalenz: Wenn $F \equiv G$, dann $\text{sub}(F) \equiv \text{sub}(G)$
 - Folgerungsbeziehungen !
 - Anwendung auf Inferenzregeln

Korrektheit von Inferenzregeln

- Gegeben: *MyRule*: $(A \vee B), (B \Rightarrow C) \therefore (A \vee C)$
- Wir wollen nachweisen: $\{(A \vee B), (B \Rightarrow C)\} \models (A \vee C)$

A	B	C	$(A \vee B)$	$(B \Rightarrow C)$	$(A \vee C)$
0	0	0	0	1	0
0	0	1	0	1	1
0	1	0	1	0	0
0	1	1	1	1	1
1	0	0	1	1	1
1	0	1	1	1	1
1	1	0	1	0	1
1	1	1	1	1	1

Folgerungsbedingung ist erfüllt $\rightarrow$ Inferenzregel ist korrekt!

Ableitung (im Kalkül)

- Ein Kalkül $C = (L_{AL}, Ax, R)$ besteht aus:
 - Der aussagenlogischen Sprache L_{AL}
 - Einer Axiommenge Ax
 - Einer Inferenzregelmenge R
- Mit Ableitung können aus einer Formelmenge daraus folgende Formeln gefunden werden (sofern ein a-korrektes Kalkül vorliegt).
- Beispiel: Leiten Sie mittels $C = (L_{AL}, Ax, R)$ ab:
$$\{(A \vee B), B, (B \Rightarrow (B \Rightarrow D))\} \vdash_C (A \vee D)$$

Dabei sei: $Ax = \emptyset$, $R = \{MP, MyRule\}$

Ableitung (im Kalkül): Lösung

- $M = \{(A \vee B), B, (B \Rightarrow (B \Rightarrow D))\} \vdash_C (A \vee D)$

$$\text{MP: } \frac{A, (A \Rightarrow B)}{B} \quad \text{MyRule: } \frac{(A \vee B), (B \Rightarrow C)}{(A \vee C)}$$

Nr.	Formel	Begründung	
(1)	$F_1 := B$	aus M	
(2)	$F_2 := (B \Rightarrow (B \Rightarrow D))$	aus M	
(3)	$F_3 := (B \Rightarrow D)$	(1), (2), MP	$\text{sub}_3(A) = B, \text{sub}_3(B) = (B \Rightarrow D)$
(4)	$F_4 := (A \vee B)$	aus M	
(5)	$F_5 := (A \vee D)$	(3), (4), MyRule	$\text{sub}_5(A) = A, \text{sub}_5(B) = B,$ $\text{sub}_5(C) = D$

Ableitung (im Kalkül): Bsp. 2

- $M = \{(B \wedge D)\} \vdash_C ((B \wedge D) \vee A)$

$C = (L_{AL}, Ax, R)$
 $R = \{MP\}$

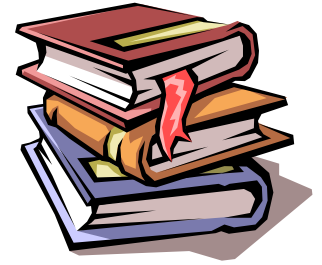
$$MP: \frac{A, (A \Rightarrow B)}{B} \quad Ax = \{ (A \Rightarrow (A \vee B)) \} \quad (H_{4a})$$

Nr.	Formel	Begründung	
(1)	$F_1 := (B \wedge D)$	aus M	
(2)	$F_2 := ((B \wedge D) \Rightarrow ((B \wedge D) \vee A))$	H_{4a}	$sub_2(A) = (B \wedge D), sub_2(B) = A$
(3)	$F_3 := ((B \wedge D) \vee A)$	(1), (2), MP	$sub_3(A) = (B \wedge D),$ $sub_3(B) = ((B \wedge D) \vee A)$

Pause!

30 Minuten

Kleines Quiz



- Was kann die Formel **F** in den folgenden Beispielen jeweils möglicherweise sein? Gültig, Kontingent und/oder Unerfüllbar?
- $\neg F$ unerfüllbar
- G unerfüllbar, $(F \vee G)$ gültig
- G gültig, $(F \Rightarrow G)$ gültig
- G kontingent, $F \models G$
- G kontingent, $(F \vee G)$ gültig
- G gültig, $(F \wedge G)$ kontingent

Kleines Quiz: Lösungen

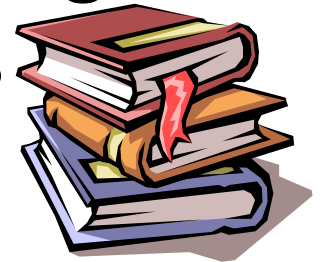
- Was kann die Formel **F** in den folgenden Beispielen jeweils möglicherweise sein? Gültig, Kontingent und/oder Unerfüllbar?
- $\neg F$ unerfüllbar (g)
- G unerfüllbar, $(F \vee G)$ gültig (g)
- G gültig, $(F \Rightarrow G)$ gültig (g k u)
- G kontingent, $F \models G$ (k u)
- G kontingent, $(F \vee G)$ gültig (g k)
- G gültig, $(F \wedge G)$ kontingent (k)

Ablauf – Tag 1

- Organisation, Vorstellung, etc.
- Logik: Allgemeine Motivation
- AS_{AL} und L_{AL}
- Strukturelle Induktion in der Aussagenlogik
- Semantik, Äquivalenz und Normalformen
- Folgerbarkeit, Ableitungen und Inferenzregeln
- <Pause>
- **Hornformeln und der Markierungsalgorithmus**
- Resolution
- Prädikatenlogik allgemein
- **Strukturen und Auswertungen von PL-Formeln**

Hornformeln

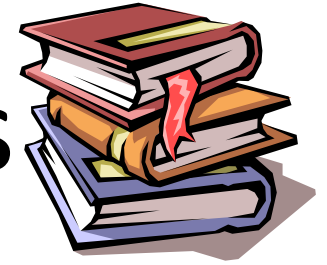
- Teilmenge der aussagenlogischen Sprache
- Formeln in KNF mit maximal einem Aussagensymbol ohne Negation davor („positives Literal“) in jeder Klausel.
- Beispiel für eine Hornformel:
$$(A \vee \neg B \vee \neg C) \wedge A \wedge (B \vee \neg D) \wedge (\neg E \vee \neg D \vee \neg A) \wedge E$$
- Jede Horn-Klausel kann als Implikation geschrieben werden, sofern 2 neue *logische Konstanten* eingeführt werden: $\top$ („top“) und $\perp$ („bottom“)
- $\top$ ist tautologisch und $\perp$ unerfüllbar



Implikationsschreibweise

- Offenbar gilt:
 - $F \equiv (\top \Rightarrow F)$ und $\neg F \equiv (F \Rightarrow \perp)$ für alle Formeln F .
- Jede Klausel wird nun wie in den folgenden Beispielen umgeschrieben:
 - $(A \vee \neg B \vee \neg C \vee \neg D) \equiv (A \vee \neg B \vee \neg(C \wedge D)) \equiv (A \vee \neg(B \wedge C \wedge D)) \equiv (\neg(B \wedge C \wedge D) \vee A) \equiv ((B \wedge C \wedge D) \Rightarrow A)$
 - $A \equiv (\top \Rightarrow A)$
 - $(\neg A \vee \neg B \vee \neg C) \equiv (\neg A \vee \neg(B \wedge C)) \equiv \neg(A \wedge B \wedge C) \equiv ((A \wedge B \wedge C) \Rightarrow \perp)$

Der Markierungsalgorithmus



- Erfüllbarkeitstest-Verfahren für Hornformeln
- Idee:
 - Formel ist in KNF, d.h. alle Klauseln müssen erfüllt sein, damit die Formel erfüllt ist.
 - Positive Literale, die alleine eine Klausel ausmachen, müssen 1 sein.
 - Wenn alle negativen Literale einer Klausel 1 sind, muss das einzig positive Literal der Klausel 1 sein.
 - Falls es kein positives Literal gibt in einer Klausel, aber alle negativen Literale auf 1 gesetzt wurden, ist die Formel automatisch unerfüllbar.

Markierungsalgorithmus: Beispiel

- Wenden Sie den Markierungsalgorithmus auf die folgende Formel an:

$$\neg A \wedge (B \vee \neg A \vee \neg C) \wedge C \wedge D \wedge (\neg C \vee \neg D \vee B) \wedge (\neg B \vee E) \wedge (\neg E \vee \neg A)$$

Markierungsalgorithmus: Lösung

- $\neg A \wedge (B \vee \neg A \vee \neg C) \wedge C \wedge D \wedge (\neg C \vee \neg D \vee B) \wedge$
 $(\neg B \vee E) \wedge (\neg E \vee \neg A) \equiv$
 $(A \Rightarrow \perp) \wedge ((A \wedge C) \Rightarrow B) \wedge (\top \Rightarrow C) \wedge (\top \Rightarrow D) \wedge$
 $((C \wedge D) \Rightarrow B) \wedge (B \Rightarrow E) \wedge ((E \wedge A) \Rightarrow \perp)$
- $(A \Rightarrow \perp) \wedge ((A \wedge C^1) \Rightarrow B) \wedge (\top \Rightarrow C^1) \wedge (\top \Rightarrow D^1) \wedge$
 $((C^1 \wedge D^1) \Rightarrow B) \wedge (B \Rightarrow E) \wedge ((E \wedge A) \Rightarrow \perp)$
- $(A \Rightarrow \perp) \wedge ((A \wedge C^1) \Rightarrow B^2) \wedge (\top \Rightarrow C^1) \wedge (\top \Rightarrow D^1) \wedge$
 $((C^1 \wedge D^1) \Rightarrow B^2) \wedge (B^2 \Rightarrow E) \wedge ((E \wedge A) \Rightarrow \perp)$
- $(A \Rightarrow \perp) \wedge ((A \wedge C^1) \Rightarrow B^2) \wedge (\top \Rightarrow C^1) \wedge (\top \Rightarrow D^1) \wedge$
 $((C^1 \wedge D^1) \Rightarrow B^2) \wedge (B^2 \Rightarrow E^3) \wedge ((E^3 \wedge A) \Rightarrow \perp)$

Markierungsalgorithmus: Lösung

- $(A \Rightarrow \perp) \wedge ((A \wedge C^1) \Rightarrow B^2) \wedge (\top \Rightarrow C^1) \wedge (\top \Rightarrow D^1) \wedge ((C^1 \wedge D^1) \Rightarrow B^2) \wedge (B^2 \Rightarrow E^3) \wedge ((E^3 \wedge A) \Rightarrow \perp)$
- Formel ist erfüllbar!
- Eine Erfüllende Belegung ist:

x	A	B	C	D	E
A(x)	0	1	1	1	1

Ablauf – Tag 1

- Organisation, Vorstellung, etc.
- Logik: Allgemeine Motivation
- AS_{AL} und L_{AL}
- Strukturelle Induktion in der Aussagenlogik
- Semantik, Äquivalenz und Normalformen
- Folgerbarkeit, Ableitungen und Inferenzregeln
- <Pause>
- Hornformeln und der Markierungsalgorithmus
- **Resolution**
- Prädikatenlogik allgemein
- Strukturen und Auswertungen von PL-Formeln

Resolution

- Einschränkung auf Hornformeln ist unhandlich
- Gibt es ein ähnlich einfaches Verfahren, wie den Markierungsalgorithmus, für allgemeine Formeln aus L_{AL} ?
- Vermutlich nicht! Diese Frage ist bisher ungeklärt und hat tiefgreifende Zusammenhänge zur Komplexitätstheorie (welche am Ende vom Automatenteil behandelt wird)!
- Aber: *Unerfüllbarkeit* lässt sich gut mit Hilfe des ***Resolutionskalküls*** nachweisen.

Einschub: Klauselmengen- Darstellung einer Formel in KNF

- Die Klauselmengen-Darstellung ist eine äquivalente Repräsentation einer Formel in KNF.
- Am besten zu verstehen am Beispiel:
 - Formel in KNF:
 $(A \vee B \vee \neg C) \wedge A \wedge (B \vee \neg D \vee B) \wedge (C \vee D \vee A)$
 - Formel in Klauselmengen-Darstellung:
 $\{ \{A, B, \neg C\}, \{A\}, \{B, \neg D\}, \{C, D, A\} \}$

Resolution

- Gegeben: Formel in KNF in Klauselmengen-Darstellung. Ablauf:
 - Suche nach ***einem*** Aussagensymbol, welches in 2 Klauseln, einmal als positives Literal und einmal als negatives Literal vorkommt.
 - Streiche beide Vorkommen, Vereinige die verbleibenden Inhalte der beiden Klauseln und füge sie der Klauselmenge als neue Klausel hinzu.
 - Wiederhole den Vorgang, bis entweder keine neuen Klauseln entstehen, oder aber die leere Klausel abgeleitet wurde.

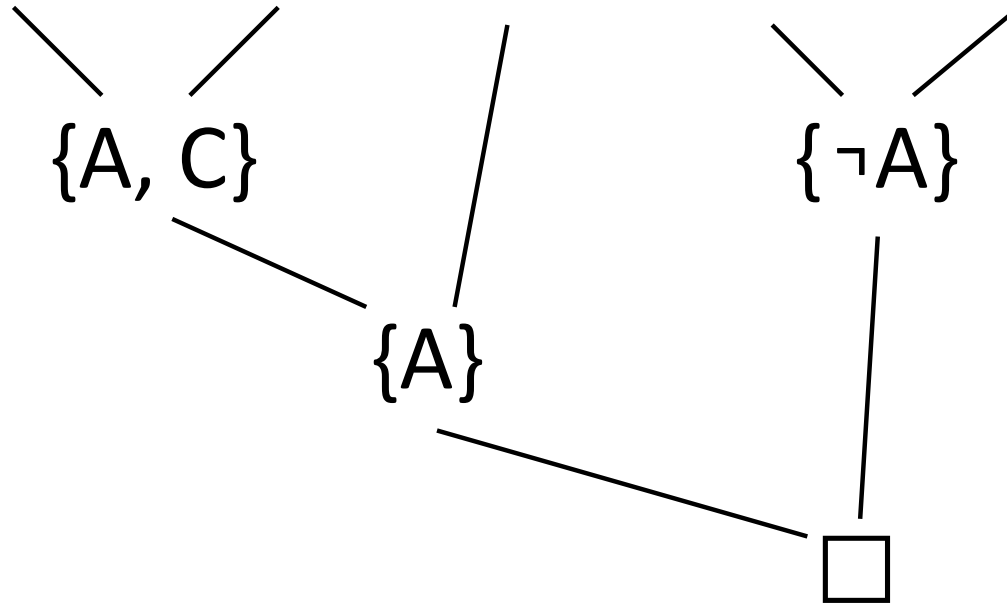
Resolution: Beispiel

- Weisen Sie folgendes mittels Resolution nach:
 $(A \vee \neg B) \wedge (B \vee C) \wedge \neg C \wedge (\neg A \vee D) \models D$

Resolution: Beispiel (Lösung)

- $(A \vee \neg B) \wedge (B \vee C) \wedge \neg C \wedge (\neg A \vee D) \models D$ gdw.
 $\{(A \vee \neg B) \wedge (B \vee C) \wedge \neg C \wedge (\neg A \vee D)\} \cup \{\neg D\} \models$
- Klauselmengendarstellung:

$\{ \{A, \neg B\}, \{B, C\}, \{\neg C\}, \{\neg A, D\}, \{\neg D\} \}$



Resolution: Alternativen

- P-Resolution:
 - Resolventenbildung nur möglich, wenn in einer der beiden Klauseln nur positive Literale vorkommen.
- N-Resolution:
 - Resolventenbildung nur möglich, wenn in einer der beiden Klauseln nur negative Literale vorkommen.
- P- und N-Resolution genau so mächtig, wie die normale Resolution!
- Weitere Resolutionsformen:
 - Lineare Resolution, Einheitsresolution, ...

Ablauf – Tag 1

- Organisation, Vorstellung, etc.
- Logik: Allgemeine Motivation
- AS_{AL} und L_{AL}
- Strukturelle Induktion in der Aussagenlogik
- Semantik, Äquivalenz und Normalformen
- Folgerbarkeit, Ableitungen und Inferenzregeln
- <Pause>
- Hornformeln und der Markierungsalgorithmus
- Resolution
- **Prädikatenlogik allgemein**
- **Strukturen und Auswertungen von PL-Formeln**

Prädikatenlogik

- Idee:
 - Aufsplitten atomarer Aussagensymbole in Bestandteile, die keine Aussagen mehr sind.
- Mächtiger als aussagenlogische Formeln.

Prädikatenlogik

- 5 neue Konzepte „unter der Ebene der atomaren Aussagen“:
 - **Konstanten:** entsprechen Worten
 - **Variablen:** wie Konst., aber mit Sonderfunktion
 - **Terme:** Oberbegriff: Konstanten / Variablen
 - **Funktionen:** entsp. Satzfragmenten. Anwendung erzeugt „*komplexe Terme*“. Werden angewendet auf (komplexe) Terme.
 - **Prädikate:** Werden angewendet auf (komplexe) Terme und *erzeugen atomare Aussagen*.

Notationskonventionen

- Folgende Konventionen zur Notation gibt es:
 - **Konstanten:** Kleinbuchstaben ab „a“
 - $a, b, c, d, \dots$
 - **Variablen:** Kleinbuchstaben um „x“ herum
 - $x, y, z, w, \dots$
 - **Funktionen:** Kleinbuchstaben ab „f“ + Klammern je nach Stelligkeit
 - $f() \quad g(, ,) \quad h(,)$
 - **Prädikate:** Großbuchstaben ab „P“ + Klammern je nach Stelligkeit
 - $P() \quad Q(, ,) \quad R(,)$

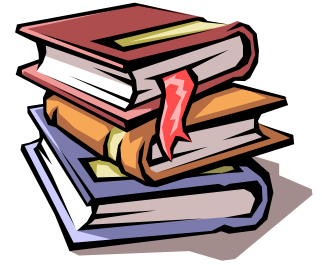
Bsp.: Prädikatenlogische Formel

- $P(x,a) \Rightarrow (P(b, f(c)) \wedge Q(g(y, z)))$
- Zuordnung der Bestandteile:
 - **Konstanten:** a, b, c
 - **Variablen:** x, y, z
 - **Terme:** a, b, c, x, y, z
 - **Funktionen:** f, g
 - **Komplexe Terme:** $f(c), g(y, z)$
 - **Prädikate:** P, Q
 - **Atomare Aussagen:** $P(x,a), P(b, f(c)), Q(g(y))$

Ablauf – Tag 1

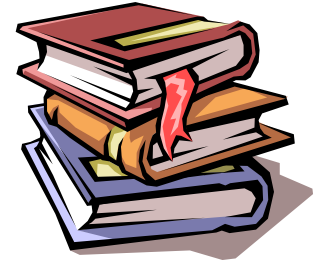
- Organisation, Vorstellung, etc.
- Logik: Allgemeine Motivation
- AS_{AL} und L_{AL}
- Strukturelle Induktion in der Aussagenlogik
- Semantik, Äquivalenz und Normalformen
- Folgerbarkeit, Ableitungen und Inferenzregeln
- <Pause>
- Hornformeln und der Markierungsalgorithmus
- Resolution
- Prädikatenlogik allgemein
- **Strukturen und Auswertungen von PL-Formeln**

Universen und Strukturen



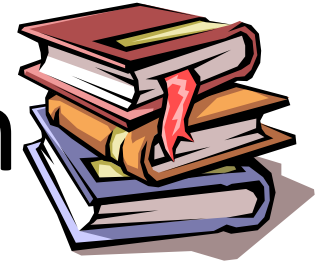
- Auswertung von prädikatenlogischen Formeln ist deutlich komplexer, als von a.l. Formeln.
- Eine PL-Formel kann viele Interpretationen haben.
- Zu Grunde liegt aber immer eine Domäne bzw. ein Universum U .
- Eine Struktur $S = (U, I)$ zu einer PL-Formel legt Universum U und Interpretation I fest und erlaubt die Berechnung eines Wahrheitswertes

Universen und Strukturen formal



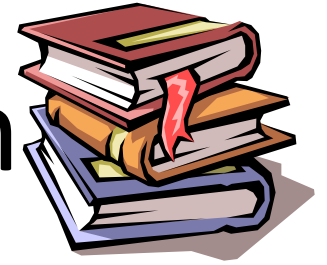
- Formal ist ein Universum U eine Menge.
- Bei der Interpretation unterscheidet sich dies:
 - Die Interpretation von Konstanten und Variablen ist formal ein Element aus U .
 - Die Interpretation von Funktionen f ist eine Abbildung $f : U \times U \times \dots \rightarrow U$
 - Insbesondere ist damit jeder komplexe Term auch nur ein Element aus U .
 - Die Interpretation von Prädikatsymbolen ist Formal ein Test auf Zugehörigkeit zu einer Teilmenge von U . Ergibt 0 oder 1.

Prädikatenlogik: Quantoren



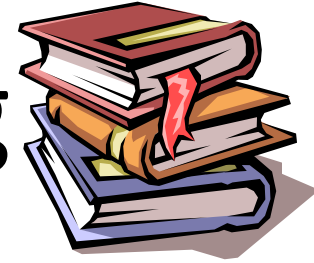
- Sonderfunktion von Variablen:
Sie können quantifiziert werden.
- Allquantor $\forall$ und Existenzquantor $\exists$
- Quantoren sitzen außerhalb von Prädikaten
(also auf/über der atomaren Aussagen-Ebene).
- Quantoren haben einen Gültigkeitsbereich, der
sich auf das nächste Symbol (bzw. Klammer)
bezieht. Dieser wird als Skopus bezeichnet.
- Quantoren überschreiben jede Interpretation
der an sie gebundenen Variablen!

Prädikatenlogik: Quantoren



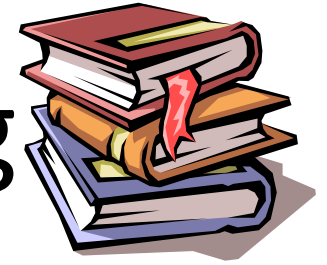
- Bei der Auswertung werden **alle Elemente** des Universums an Stelle der gebundenen Variablen x überprüft (alle sog. **x -Varianten**) und der jeweilige Wahrheitswert des Quantorenskopus wird ermittelt.
- Wenn die Formel bei wenigstens einer Variante zu 1 ausgewertet wurde, wird ein Existenzquantor zu 1 ausgewertet.
- Wenn bei jeder Variante die Formel zu 1 ausgewertet wurde, wird ein Allquantor zu 1 ausgewertet (Ein Existenzquantor natürl. auch).

Beispiel-Auswertung



- Werten Sie mit der Struktur $S = (U, I)$ die Formel $F = \forall x (\exists y P(f(x, y)) \vee Q(x, c))$ aus!
- $U = \{\mathbf{a}, \mathbf{b}\}$
- I :
 - $x \mapsto \mathbf{a}$
 - $y \mapsto \mathbf{b}$
 - $c \mapsto \mathbf{a}$
 - f :
 - $(\mathbf{a}, \mathbf{a}) \mapsto \mathbf{a}$
 - $(\mathbf{a}, \mathbf{b}) \mapsto \mathbf{a}$
 - $(\mathbf{b}, \mathbf{a}) \mapsto \mathbf{a}$
 - $(\mathbf{b}, \mathbf{b}) \mapsto \mathbf{b}$
 - $P = \{\mathbf{b}\}$
 - $Q = \{(x, y) \in U \times U \mid x = y\}$

Beispiel-Auswertung: Lösung



- $F = \forall x (\exists y P(f(x, y)) \vee Q(x, c))$
- Der Übersichtlichkeit halber sei:
 $G = (\exists y P(f(x, y)) \vee Q(x, c))$ (Formel ohne $\forall x$)
 $H = \exists y P(f(x, y))$

x	y	f(x, y)	P(f(x, y))	H	c	Q(x, c)	G	F
a	a	a	0	0	a	1	1	1
	b	a	0					
b	a	a	0	1	a	0	1	
	b	b	1					

Das wars für heute!

Bis morgen! 😊

FGI-1 Repetitorium (2. Termin): Logik / Automatentheorie [Tag 2]

Jan Henrik Röwekamp
(8roeweka)

18.09., 19.09. & 20.09.2013

Ablauf – Tag 2

- Wdh.: Prädikatenlogik & Strukturen
- Prädikatenlogische Resolution
 - Formel schließen und bereinigte Pränexform (BPF)
 - Skolemisierung, Unifikation und Resolution
- <Pause>
- Warum Automaten?
- Die Chomsky-Hierarchie
- Reguläre Mengen, DFA's und NFA's
- Der Potenzautomat: Vom NFA zum DFA
- ϵ -FA & Überführung zum NFA (ϵ -frei machen)
- Kleene-Verfahren

Wiederholung: Prädikatenlogik

- Aufsplitten von Aussagensymbolen in:
 - (komplexe) Terme, Funktionen und Prädikate
- Quantoren gebunden an Variablen
 - Allquantor $\forall$ und Existenzquantor $\exists$
- Strukturen S bestehen aus Universum U und Interpretation I
 - Auswertung einer PL-Formel
 - Berechnung eines Wahrheitswertes

Ablauf – Tag 2

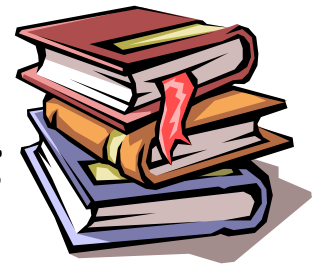
- Wdh.: Prädikatenlogik & Strukturen
- **Prädikatenlogische Resolution**
 - Formel schließen und bereinigte Pränexform (BPF)
 - Skolemisierung, Unifikation und Resolution
- <Pause>
- Warum Automaten?
- Die Chomsky-Hierarchie
- Reguläre Mengen, DFA's und NFA's
- Der Potenzautomat: Vom NFA zum DFA
- ϵ -FA & Überführung zum NFA (ϵ -frei machen)
- Kleene-Verfahren

Prädikatenlogische Resolution

- Unerfüllbarkeits-Nachweis unabhängig von Strukturen und Interpretationen.
- 6 Schritte:
 - Formel bereinigen
 - Formel schließen
 - Pränexform erzeugen
 - Skolemisierung
 - Matrix in KNF umformen
 - Unifikation + Resolution

Erfüllbarkeitsäquivalenz

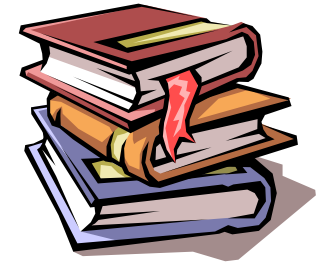
- Resolution: Nachweis von Unerfüllbarkeit
- Exakte Äquivalenz wird nicht benötigt zum Nachweis der Erfüllbarkeit. (Zu Äquivalenzen in der Prädikatenlogik siehe: **Satz 10.2!**)
- Wenn die (Un)erfüllbarkeit einer Formel F gleich der einer Formel G ist, sind G und F *erfüllbarkeitsäquivalent*.
- Wird hier ab sofort als „eä“ abgekürzt.
- Einige Schritte der PL-Resolution verletzen die Äquivalenz, *nicht aber die Eä*.



Formeln schließen: Beispiel

- Gegeben: Die Formel $(Q(y) \Rightarrow \forall x P(x))$
- Gesucht: Geschlossene Form
- 1) $(\exists y Q(y) \Rightarrow \forall x P(x)) \equiv \forall y (Q(y) \Rightarrow \forall x P(x))$
oder
- 2) $\exists y (Q(y) \Rightarrow \forall x P(x))$
- (Version 1 macht nur die Teilformel $Q(y)$ eä.)
- **Version 2** hingegen die ganze Formel, wie es beabsichtigt war.
- 1) und 2) sind nicht äq., da das Schließen der Formel so oder so nicht die Äq. erhält!

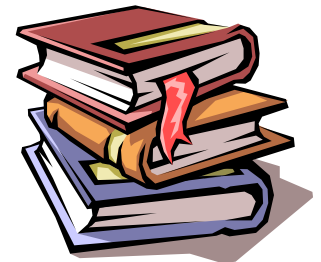
Bereinigte Pränexform (BPF)



- Jede PL-Formel kann in BPF gebracht werden.
- Eine Formel ist *bereinigt*, wenn:
 - Alle Quantoren unterschiedliche Variablen aufweisen
 - Keine Variable sowohl frei, als auch gebunden vorkommt.
- Eine Formel ist in *Pränexform*, wenn:
 - Alle Quantoren vorne in der Formel stehen.
 - Der Rest der Formel wird in diesem Fall *Matrix* genannt.

Freie und gebundene Variablen

- Variablen können frei (ohne Quantor) oder gebunden an einen Quantor vorkommen.
- Gebundene Variablen können umbenannt werden („Gebundene Umbenennung“)
- Freie Variablen dürfen **nicht** umbenannt werden!
 - Strukturen interpretieren freie Variablen. Umbenennung führt zu anderer Interpretation!
 - Gebundene Variablen werden über den Quantor ausgewertet, daher ist der Variablenname egal.

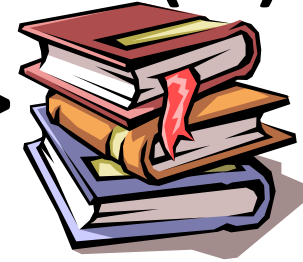


PL-Resolution: Aufgabe (1)

- Seien die folgenden Formeln gegeben:
 - $F = \forall x \neg P(x,x)$
 - $G = \forall x \forall y \forall z ((P(x,y) \wedge P(y,z)) \Rightarrow P(x,z))$
 - $H = \forall x \forall y (P(x,y) \Rightarrow \neg P(y,x))$
- Weisen Sie mittels prädikatenlogischer Resolution nach, dass $\{F, G\} \models H$ gilt!
- Wir erinnern uns:
 - $\{F, G\} \models H$ gdw. $\{F, G\} \cup \{\neg H\} \models$

PL-Resolution: Aufgabe (2)

<BPF erzeugen>



- Formel bisher: $\forall x \neg P(x,x) \wedge$
 $\forall x \forall y \forall z ((P(x,y) \wedge P(y,z)) \Rightarrow P(x,z)) \wedge$
 $\neg \forall x \forall y (P(x,y) \Rightarrow \neg P(y,x))$
- 1) Formel bereinigen (geb. Umben.) [**Äq**]:
 $\forall x \neg P(x,x) \wedge$
 $\forall u \forall y \forall z ((P(u,y) \wedge P(y,z)) \Rightarrow P(u,z)) \wedge$
 $\neg \forall v \forall w (P(v,w) \Rightarrow \neg P(w,v))$
- 2) Formel schließen. [**Eä**]
 - Da keine freien Variablen vorhanden sind, ist nichts zu tun. (Andernfalls Bindung an Existenzquantor)

PL-Resolution: Aufgabe (3)

<BPF erzeugen>

- 3) Pränexform erzeugen: [Äq]

$$\forall x \neg P(x,x) \wedge$$

$$\forall u \forall y \forall z ((P(u,y) \wedge P(y,z)) \Rightarrow P(u,z)) \wedge$$

$$\neg \forall v \forall w (P(v,w) \Rightarrow \neg P(w,v))$$

$$\equiv \forall x \neg P(x,x) \wedge \forall u \forall y \forall z ((P(u,y) \wedge P(y,z)) \Rightarrow P(u,z)) \wedge$$
$$\underline{\exists v} \underline{\exists w} \underline{\neg} (P(v,w) \Rightarrow \neg P(w,v))$$

$$\equiv \forall x \neg P(x,x) \wedge \underline{\exists v} \underline{\exists w} \forall u \forall y \forall z ($$
$$((P(u,y) \wedge P(y,z)) \Rightarrow P(u,z)) \wedge \neg (P(v,w) \Rightarrow \neg P(w,v)))$$

$$\equiv \underline{\exists v} \underline{\exists w} \underline{\forall u} \underline{\forall y} \underline{\forall z} \forall x (\neg P(x,x) \wedge$$
$$((P(u,y) \wedge P(y,z)) \Rightarrow P(u,z)) \wedge \neg (P(v,w) \Rightarrow \neg P(w,v)))$$

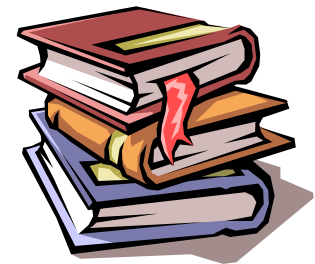
Skolemisierung

- Skolemisierung wird verwendet um Existenzquantoren aus der Formel zu entfernen.
- Grundlegende Idee:
 - Resolution ist ein Unerfüllbarkeits-Test-Verfahren
 - Erhaltung der genauen Äquivalenz der Formel ist nicht unbedingt nötig.
 - Lediglich die Erfüllbarkeit muss erhalten bleiben.
 - Wenn die Formel erfüllbar war, gab es zumindest ein Element des Universums für jeden Existenzquantor, um diesen zu erfüllen.

Skolemisierung

- Ersetzung der existenzquantifizierten Variablen mit einem festen Term (z.B. einer Konstanten).
- Abhängigkeit von Allquantoren **vor** dem Existenzquantor muss berücksichtigt werden:

- $\exists x \dots x \quad \rightarrow \quad \dots a$
- $\forall y \exists x \dots x \quad \rightarrow \quad \forall y \dots f(y)$
- $\forall y \forall z \exists x \dots x \quad \rightarrow \quad \forall y \forall z \dots f(y, z)$
- $\forall y \exists x \forall z \dots x \quad \rightarrow \quad \forall y \forall z \dots f(y)$



- Wichtig: Neue Konstanten- und Funktionssymbole dürfen vorher noch nicht vorkommen.

PL-Resolution: Aufgabe (4)

<4) Skolemisierung [Eä]>

- $\exists v \exists w \forall u \forall y \forall z \forall x (\neg P(x,x) \wedge ((P(u,y) \wedge P(y,z)) \Rightarrow P(u,z)) \wedge \neg(P(v,w) \Rightarrow \neg P(w,v)))$
- Sei I die Matrix der obigen Formel.
- Eä. Formel $\forall u \forall y \forall z \forall x I_{[v/a][w/b]}$:
 $\forall u \forall y \forall z \forall x (\neg P(x,x) \wedge ((P(u,y) \wedge P(y,z)) \Rightarrow P(u,z)) \wedge \neg(P(a, b) \Rightarrow \neg P(b, a)))$

PL-Resolution: Aufgabe (5)

<5) Matrix in KNF bringen [Äq]>

- $\forall u \forall y \forall z \forall x \left(\neg P(x,x) \wedge ((P(u,y) \wedge P(y,z)) \Rightarrow P(u,z)) \wedge \neg(P(a, b) \Rightarrow \neg P(b, a)) \right)$
- $\equiv \forall u \forall y \forall z \forall x \left(\neg P(x,x) \wedge (\neg(P(u,y) \wedge P(y,z)) \vee P(u,z)) \wedge \neg(\neg P(a, b) \vee \neg P(b, a)) \right)$
- $\equiv \forall u \forall y \forall z \forall x \left(\neg P(x,x) \wedge (\neg P(u,y) \vee \neg P(y,z) \vee P(u,z)) \wedge P(a, b) \wedge P(b, a) \right)$
- Klauselmengendarstellung:

$$\{ \{ \neg P(x,x) \}, \{ \neg P(u,y), \neg P(y,z), P(u,z) \}, \{ P(a, b) \}, \{ P(b, a) \} \}$$

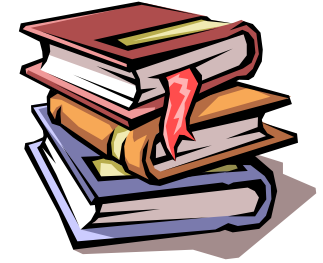
Unifikation

- Stand bis jetzt: PL-Formel in BPF, die vorne nur Allquantoren aufweist.
- Resolution testet Unerfüllbarkeit und Allquantoren werden zu „falsch“ ausgewertet, wenn ihr Skopus für wenigstens eine Variante zu „falsch“ wird.
- Idee:
 - Gezielte Suche nach einem solchen Element.
 - Element verwenden, um damit die leere Klausel im letzten Schritt abzuleiten.

Allgemeinster Unifikator

- Die Unifikation kann in mehreren Schritten (mit mehreren Substitutionen) erfolgen:
 - $y \rightarrow f(a)$
 - $y \rightarrow f(x) \rightarrow f(a)$
- Die zweite Unifikation ist „allgemeiner“ als die erste.
- Allgemeinheit: Wenn eine Substitution in 2 Schritte aufgeteilt werden kann, ist der erste Teilschritt *allgemeiner* als die gesamte Subst.
- Gesucht: der/die **allgemeinste(n) Unifikator(en)**

Unifikationsalgorithmus



- Ablauf: <Loop> Laufe in allen Klauseln (von links nach rechts) zur ersten Position, in der sich zwei Zeichen unterscheiden.
 - Falls der Unterschied durch einen Junktor oder Prädikat verursacht wird: termin.: „nicht unifizerbar“
 - Falls keiner der Terme eine Variable x ist, termin.: „nicht unifizierbar“, sonst sei der andere Term t .
 - Falls x in t vorkommt: termin.: „nicht unifizierbar“
 - Nimm $[x/t]$ in die Subst. σ auf.
 - Wenn nicht alle Pointer am Ende sind: goto <Loop>
 - Gib σ als **allgemeinsten Unifikator** aus.

Unifikationsalgorithmus: Beispiele

$$L = \{ P(x, y), P(y, f(a)) \}$$

$$L_1 = P(x, y) \quad L_2 = P(y, f(a))$$

$$\uparrow \quad \uparrow$$

$$\sigma_1 = [x/y]$$

$$L_1\sigma_1 = P(y, y) \quad L_2\sigma_1 = P(y, f(a))$$

$$\uparrow \quad \uparrow$$

$$\sigma_2 = [x/y] [y/f(a)]$$

$$L_1\sigma_2 = P(f(a), f(a)) \quad L_2\sigma_2 = P(f(a), f(a))$$

erfolgreich unifiziert!
 allgemeinsten Unifikator:
 $\sigma = [x/y] [y/f(a)]$

$$L = \{ P(x, y), P(y, f(y)) \}$$

$$L_1 = P(x, y) \quad L_2 = P(y, f(y))$$

$$\uparrow \quad \uparrow$$

$$\sigma_1 = [x/y]$$

$$L_1\sigma_1 = P(y, y) \quad L_2\sigma_1 = P(y, f(y))$$

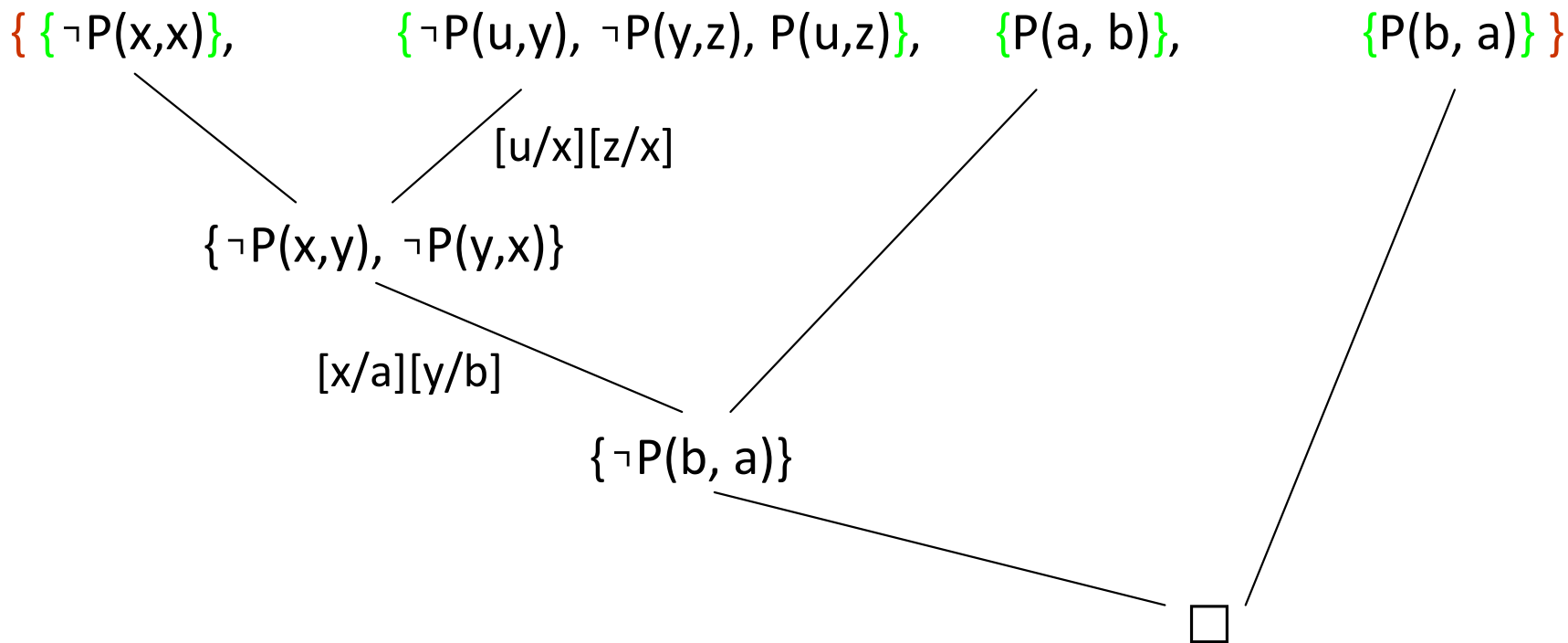
nicht unifizierbar, da
 $f(y)$ y als Teilterm enthält.

Resolution

- Unifikation erfolgt während der Resolution.
- Resolution im Kern wie in der Aussagenlogik.
- Wichtig:
 - Keine Konstanten ersetzen
 - Keine Funktionssymbole ersetzen
 - Stelligkeiten (von Funktionen und Prädikaten) können nicht geändert werden.
 - Prädikatensymbole müssen passen.

PL-Resolution: Aufgabe (6)

<6) Die Resolution selbst>



Resumee:

$\square \in \text{Res}^* \Rightarrow$

$\{F, G\} \cup \{\neg H\} \models \Rightarrow$

$\{F, G\} \models H$

Das wars zu Logik!

Auf zum Automatenteil! 😊

Aber erstmal:

30 Minuten PAUSE

Ablauf – Tag 2

- Wdh.: Prädikatenlogik & Strukturen
- Prädikatenlogische Resolution
 - Formel schließen und bereinigte Pränexform (BPF)
 - Skolemisierung, Unifikation und Resolution
- <Pause>
- **Warum Automaten?**
- Die Chomsky-Hierarchie
- Reguläre Mengen, DFA's und NFA's
- Der Potenzautomat: Vom NFA zum DFA
- ϵ -FA & Überführung zum NFA (ϵ -frei machen)
- Kleene-Verfahren

Warum Automaten?

- Gegeben: Eine Obermenge, z.B. die Menge aller 0 und 1 Folgen (binäre Codierung)
- Gesucht: Teilmenge, die bestimmte Eigenschaften erfüllt, bzw. Identifikation der Elemente der Teilmenge in der Obermenge
- Je nach Teilmenge sehr leicht bis sehr schwer bis hin zu unmöglich
- Idee: Abstrakte Automaten mit unterschiedlicher Mächtigkeit

Ablauf – Tag 2

- Wdh.: Prädikatenlogik & Strukturen
- Prädikatenlogische Resolution
 - Formel schließen und bereinigte Pränexform (BPF)
 - Skolemisierung, Unifikation und Resolution
- <Pause>
- Warum Automaten?
- **Die Chomsky-Hierarchie**
- Reguläre Mengen, DFA's und NFA's
- Der Potenzautomat: Vom NFA zum DFA
- ϵ -FA & Überführung zum NFA (ϵ -frei machen)
- Kleene-Verfahren

Die Chomsky-Hierarchie



Überabzählbare Mengen

$\neq$

Abzählbare Mengen

$\supseteq$

Typ 0

Typ 0 -
Grammatik

DTM NTM

Aufzählbare Mengen

$\supseteq$

DTM NTM

Entscheidbare Mengen

$\supseteq$

Typ 1

Kontextsensitive
Grammatik

LBA

Kontextsensitive Mengen

– uvwxy-Theorem

$\supseteq$

PDA

Kontextfreie Mengen

Typ 2

Kontextfreie
Grammatik

DPDA

Deterministisch
Kontextfreie Mengen

– uvw-Theorem

$\supseteq$

Typ 3

Rechts- bzw.
linkslinere
Grammatik

Potenzautomat

DFA

NFA

ϵ -FA

GFA

ϵ -frei machen

Kleene

trivial

Reguläre Mengen

$\supseteq$

Endliche Mengen

Spezialfälle

Ablauf – Tag 2

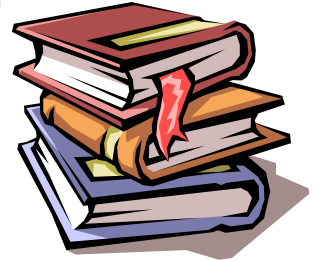
- Wdh.: Prädikatenlogik & Strukturen
- Prädikatenlogische Resolution
 - Formel schließen und bereinigte Pränexform (BPF)
 - Skolemisierung, Unifikation und Resolution
- <Pause>
- Warum Automaten?
- Die Chomsky-Hierarchie
- **Reguläre Mengen, DFA's und NFA's**
- Der Potenzautomat: Vom NFA zum DFA
- ε -FA & Überführung zum NFA (ε -frei machen)
- Kleene-Verfahren

Reguläre Mengen

- Klasse an Mengen
- Lassen sich durch bestimmte Strukturen beschreiben (reguläre Ausdrücke)
- Praxisanwendung: Z.B. Parsing
- Typ 3 Mengen der Chomsky-Hierarchie

Reguläre Ausdrücke

- Beschreiben reguläre Mengen
- 2 Schreibweisen:
 - Reguläre Ausdruck – Schreibweise $(a + bc)^*$
 - (Mengen – Schreibweise) $(\{a\} \cup \{b\} \cdot \{c\})^*$
- Bestehen aus den folgenden Komponenten:
 - Konkatenation: $(a \cdot b)$ beschreibt die Menge $\{ab\}$
 - Auswahl: $(a+b)$ beschreibt die Menge $\{a, b\}$
 - Sternbildung: a^* beschr. die Menge $\{\epsilon, a, aa, aaa, \dots\}$
 - Plusbildung: a^+ beschreibt die Menge $\{a, aa, aaa, \dots\}$



Ablauf – Tag 2

- Wdh.: Prädikatenlogik & Strukturen
- Prädikatenlogische Resolution
 - Formel schließen und bereinigte Pränexform (BPF)
 - Skolemisierung, Unifikation und Resolution
- <Pause>
- Warum Automaten?
- Die Chomsky-Hierarchie
- Reguläre Mengen, DFA's und NFA's
- Der Potenzautomat: Vom NFA zum DFA
- ϵ -FA & Überführung zum NFA (ϵ -frei machen)
- Kleene-Verfahren

Die Chomsky-Hierarchie



Überabzählbare Mengen

$\neq$

Abzählbare Mengen

$\supseteq$

Typ 0

Typ 0 -
Grammatik

DTM NTM

Aufzählbare Mengen

$\supseteq$

DTM NTM

Entscheidbare Mengen

$\supseteq$

Typ 1

Kontextsensitive
Grammatik

LBA

Kontextsensitive Mengen

– uvwxy-Theorem

$\supseteq$

PDA

Kontextfreie Mengen

Typ 2

Kontextfreie
Grammatik

DPDA

Deterministisch
Kontextfreie Mengen

– uvw-Theorem

$\supseteq$

Typ 3

Rechts- bzw.
linkslinere
Grammatik

Potenzautomat

DFA NFA

ϵ -FA

GFA

ϵ -frei machen

Kleene

trivial

Reguläre Mengen

$\supseteq$

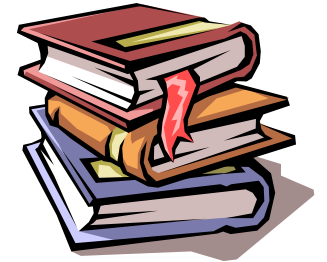
Endliche Mengen

Spezialfälle

DFA/NFA: (Nicht-)deterministische endliche Automaten

- Formal Quintupel (5-Tupel): $A = (Q, \Sigma, \delta, q_0/S_0, F)$
- Q ist eine endliche Menge an Zuständen.
- Σ ein endliches Eingabealphabet
- F ist eine Endzustandsmenge, es gilt $F \subseteq Q$.
- q_0 ist ein Startzustand (DFA) bzw. S_0 eine Startzustandsmenge (NFA).
- δ ist eine Überföhrungsfunktion:
 - DFA: $\delta : Q \times \Sigma \rightarrow Q$
 - NFA: $\delta : Q \times \Sigma \rightarrow 2^Q$

Akzeptierte Sprache von NFAs/DFAs



- <Tafel>
- Initial zusammenhängende DFAs (izDFAs)
 - Jeder Zustand ist vom Startzustand aus erreichbar
- Vollständige DFAs (vDFAs)
 - Aus jedem Zustand ist für jedes Eingabesymbol ein Folgezustand definiert.

Akzeptierte Sprache DFA: Beispiel

- $A = (Q, \Sigma, \delta, q_0, F)$

- $Q = \{q_0, q_1\}$

- $\Sigma = \{a, b\}$

- $F = \{q_1\}$

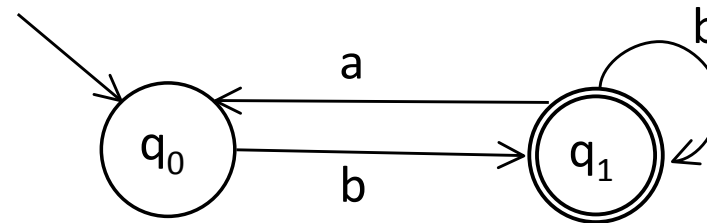
- δ :

- $\delta(q_0, b) = q_1$

- $\delta(q_1, b) = q_1$

- $\delta(q_1, a) = q_0$

- $L(A) = b (b^* + ab)^* = b (b + ab)^*$



Ablauf – Tag 2

- Wdh.: Prädikatenlogik & Strukturen
- Prädikatenlogische Resolution
 - Formel schließen und bereinigte Pränexform (BPF)
 - Skolemisierung, Unifikation und Resolution
- <Pause>
- Warum Automaten?
- Die Chomsky-Hierarchie
- Reguläre Mengen, DFA's und NFA's
- **Der Potenzautomat: Vom NFA zum DFA**
- ϵ -FA & Überführung zum NFA (ϵ -frei machen)
- Kleene-Verfahren

Die Chomsky-Hierarchie



Überabzählbare Mengen

$\neq$

Abzählbare Mengen

$\supseteq$

Typ 0

Typ 0 -
Grammatik

DTM NTM

Aufzählbare Mengen

$\supseteq$

DTM NTM

Entscheidbare Mengen

$\supseteq$

Typ 1

Kontextsensitive
Grammatik

LBA

Kontextsensitive Mengen

– uvwxy-Theorem

$\supseteq$

PDA

Kontextfreie Mengen

Typ 2

Kontextfreie
Grammatik

DPDA

Deterministisch
Kontextfreie Mengen

– uvw-Theorem

$\supseteq$

Typ 3

Rechts- bzw.
linkslinere
Grammatik

Potenzautomat

DFA

NFA

ϵ -FA

GFA

Reguläre Mengen

$\supseteq$

Endliche Mengen

114

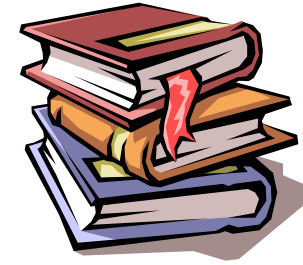
Spezialfälle

ϵ -frei machen

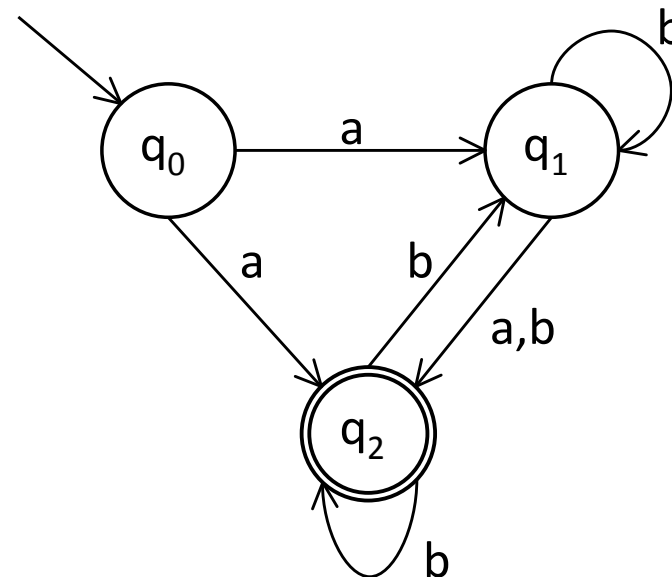
Kleene

trivial

Potenzautomat

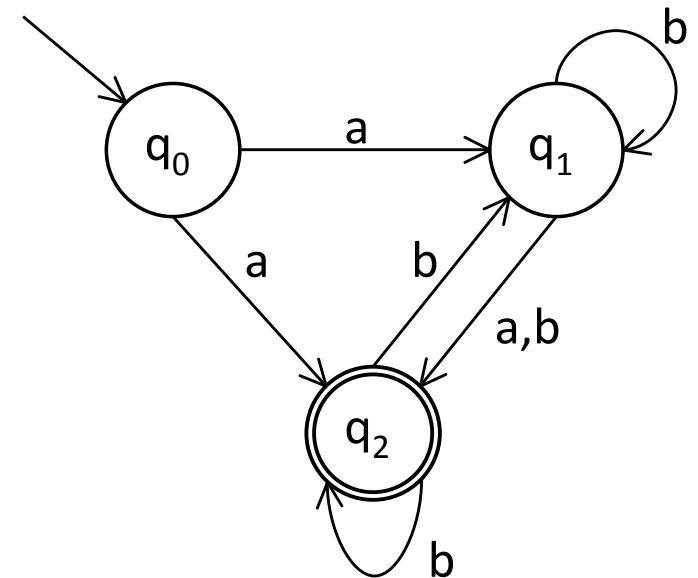
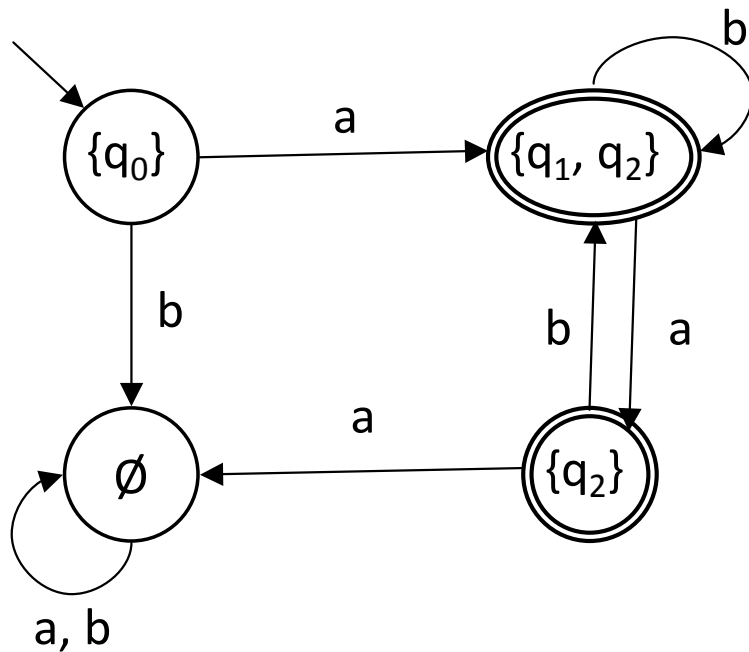


- NFA's lassen sich in (exponentiell größere) DFA's überführen, ohne dabei die akzeptierte Sprache zu ändern.
- Aufgabe: Wandeln Sie den folgenden NFA B in einen DFA um:
- Welche Sprache akzeptiert der NFA B?



Potenzautomat: Lösung

- $L(B) = (a + ab^*(a+b)) (b + (bb^*(a+b)))^*$



Ablauf – Tag 2

- Wdh.: Prädikatenlogik & Strukturen
- Prädikatenlogische Resolution
 - Formel schließen und bereinigte Pränexform (BPF)
 - Skolemisierung, Unifikation und Resolution
- <Pause>
- Warum Automaten?
- Die Chomsky-Hierarchie
- Reguläre Mengen, DFA's und NFA's
- Der Potenzautomat: Vom NFA zum DFA
- ϵ -FA & Überführung zum NFA (ϵ -frei machen)
- Kleene-Verfahren

Die Chomsky-Hierarchie



Überabzählbare Mengen

$\neq$

Abzählbare Mengen

$\supseteq$

Typ 0

Typ 0 -
Grammatik

DTM NTM

Aufzählbare Mengen

$\supseteq$

DTM NTM

Entscheidbare Mengen

$\supseteq$

Typ 1

Kontextsensitive
Grammatik

LBA

Kontextsensitive Mengen

– uvwxy-Theorem

$\supseteq$

PDA

Kontextfreie Mengen

Typ 2

Kontextfreie
Grammatik

DPDA

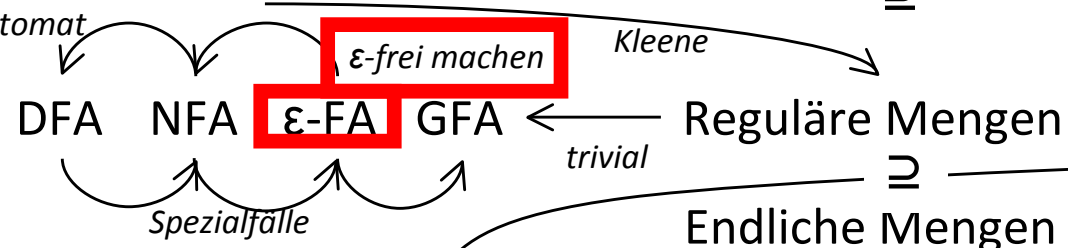
Deterministisch
Kontextfreie Mengen

– uvw-Theorem

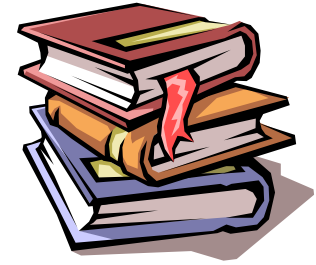
$\supseteq$

Typ 3

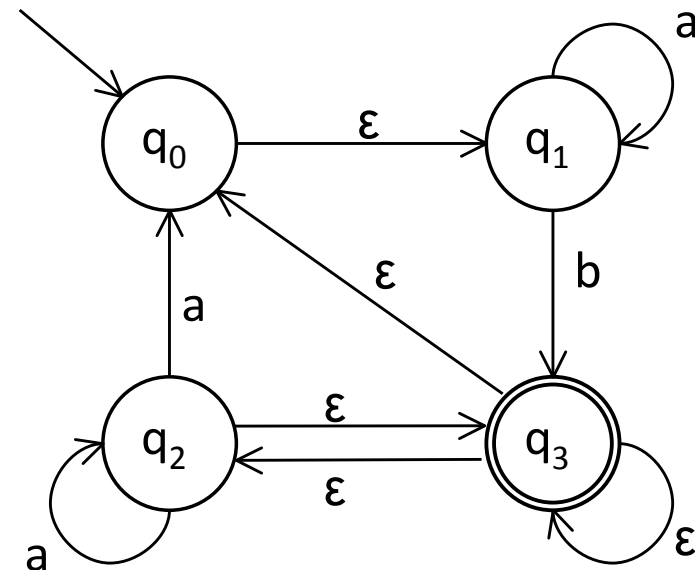
Rechts- bzw.
linkslinere
Grammatik



ϵ -FA & ϵ -frei machen



- ϵ -FA: Endlicher Automat, der das leere Wort an Kanten erlaubt.
- Kann zu gleich großem NFA umgebaut werden.
- Aufgabe: Wandeln Sie den folgenden ϵ -FA in einen NFA um:



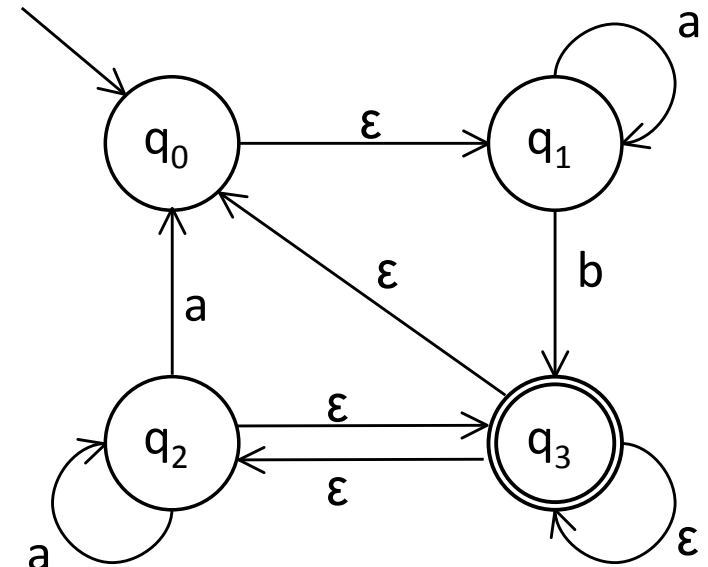
ϵ -frei machen: Lösung (1)

R_0	q_0	q_1	q_2	q_3
q_0	1			
q_1		1		
q_2			1	
q_3				1

R_1	q_0	q_1	q_2	q_3
q_0		1		
q_1				
q_2				1
q_3	1		1	1

R_2	q_0	q_1	q_2	q_3
q_0				
q_1				
q_2	1		1	1
q_3	1	1	1	1

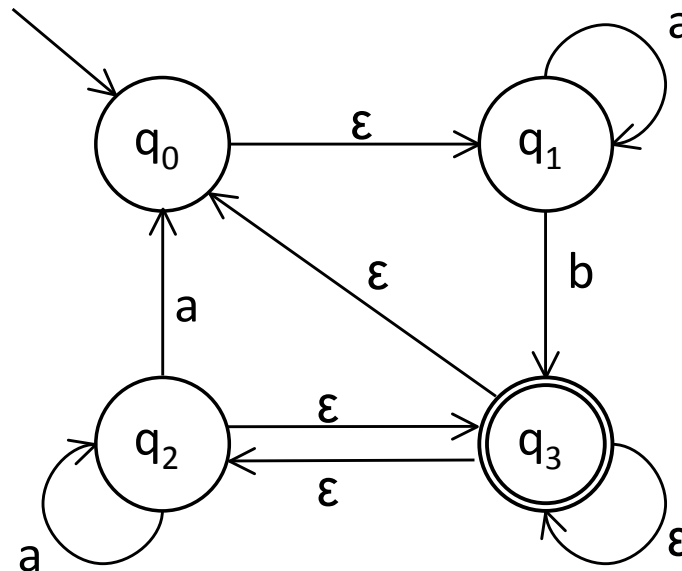
R_3	q_0	q_1	q_2	q_3
q_0				
q_1				
q_2	1	1	1	1
q_3	1	1	1	1



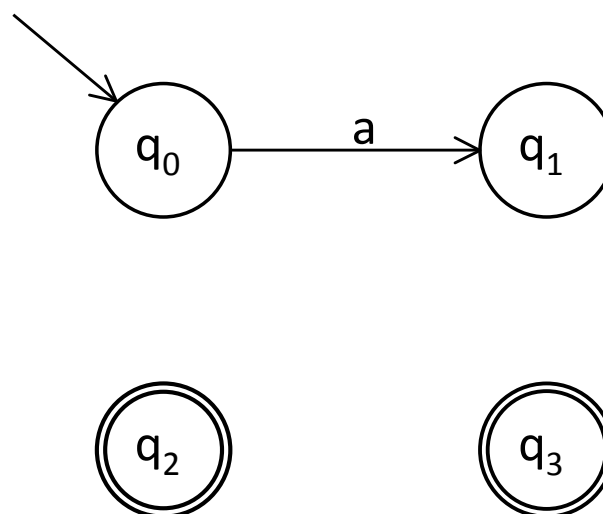
R	q_0	q_1	q_2	q_3
q_0	1	1		
q_1		1		
q_2	1	1	1	1
q_3	1	1	1	1

(Die Tabellen sind persönliche Hilfen, und nicht zwangs-

ϵ -frei machen: Lösung (2)



Füge Kante von q_0 zu q_1 mit a hinzu



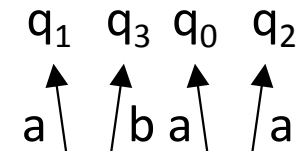
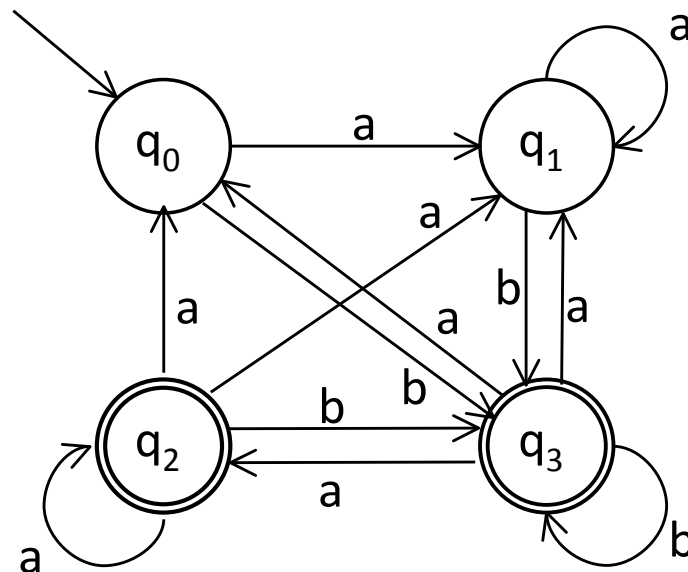
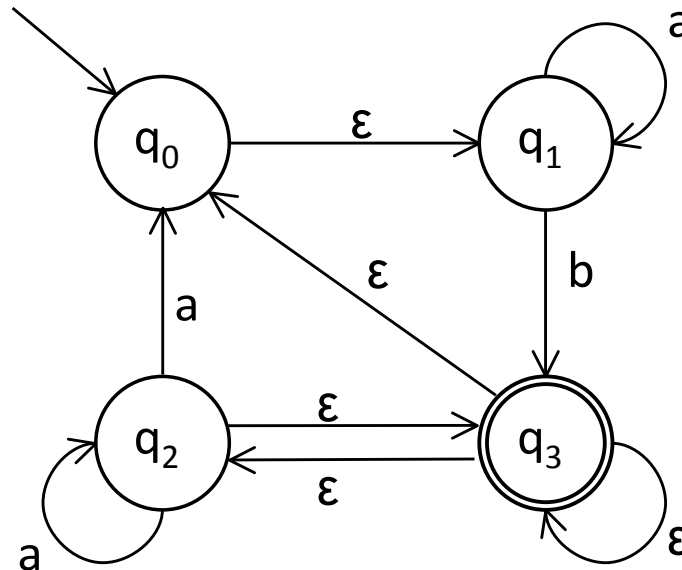
		q₁	q ₃	q ₀	q ₂
R	q ₀	q ₁	q ₂	q ₃	
q₀	1	1			
q ₁		1			
q ₂	1	1	1	1	
q ₃	1	1	1	1	

ϵ -frei machen: Lösung (3)

ϵ -Hülle als Relation:

$R = \{ (q_0, q_0), (q_0, q_1), (q_1, q_1), (q_2, q_0), (q_2, q_1), (q_2, q_2), (q_2, q_3), (q_3, q_0), (q_3, q_1), (q_3, q_2), (q_3, q_3) \}$

$= \{ (q_0, q_1), (q_2, q_0), (q_2, q_1), (q_2, q_3), (q_3, q_0), (q_3, q_1), (q_3, q_2) \} \cup \text{Id}_Q$



R	q_0	q_1	q_2	q_3
q_0	1	1		
q_1		1		
q_2	1	1	1	1
q_3	1	1	1	1

Ablauf – Tag 2

- Wdh.: Prädikatenlogik & Strukturen
- Prädikatenlogische Resolution
 - Formel schließen und bereinigte Pränexform (BPF)
 - Skolemisierung, Unifikation und Resolution
- <Pause>
- Warum Automaten?
- Die Chomsky-Hierarchie
- Reguläre Mengen, DFA's und NFA's
- Der Potenzautomat: Vom NFA zum DFA
- ϵ -FA & Überführung zum NFA (ϵ -frei machen)
- **Kleene-Verfahren**

Die Chomsky-Hierarchie



Überabzählbare Mengen

$\neq$

Abzählbare Mengen

$\supseteq$

Typ 0

Typ 0 -
Grammatik

DTM NTM

Aufzählbare Mengen

$\supseteq$

DTM NTM

Entscheidbare Mengen

$\supseteq$

Typ 1

Kontextsensitive
Grammatik

LBA

Kontextsensitive Mengen

– uvwxy-Theorem

$\supseteq$

PDA

Kontextfreie Mengen

Typ 2

Kontextfreie
Grammatik

DPDA

Deterministisch
Kontextfreie Mengen

– uvw-Theorem

$\supseteq$

Typ 3

Rechts- bzw.
linkslinere
Grammatik

Potenzautomat

DFA

NFA

ϵ -FA

GFA

Kleene

trivial

Reguläre Mengen

$\supseteq$

Endliche Mengen

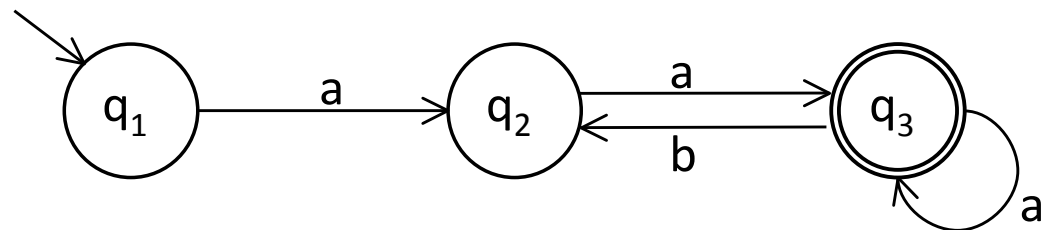
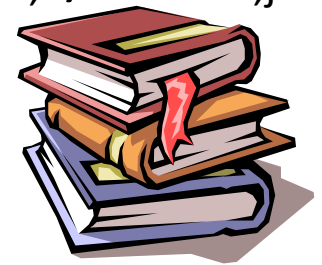
Spezialfälle

ϵ -frei machen

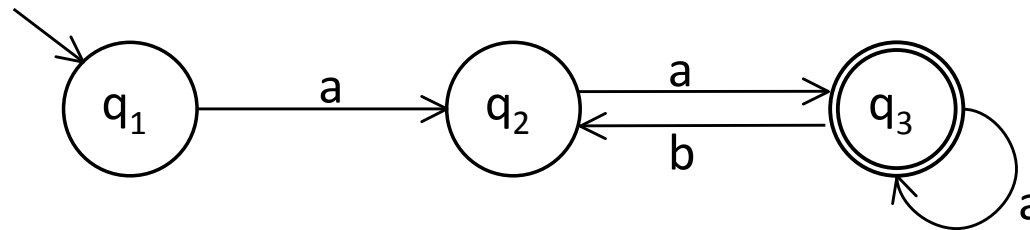
Kleene-Verfahren

- Rekursives Verfahren zum Ableiten des regulären Ausdrucks aus einem Automaten
- Exponentiell aufwändig!
- Aufgabe: Leiten Sie den zugehörigen regulären Ausdruck aus diesem Automaten ab:

Kleene's Rekursionsformel:
$$R^k_{i,j} = R^{k-1}_{i,j} \cup R^{k-1}_{i,k} (R^{k-1}_{k,k})^* R^{k-1}_{k,j}$$



Kleene-Verfahren: Anwendung



- Wir leiten aus den Übergängen ab:

$$R^0_{1,1} = \{\epsilon\}$$

$$R^0_{2,1} = \emptyset$$

$$R^0_{3,1} = \emptyset$$

$$R^0_{1,2} = \{a\}$$

$$R^0_{2,2} = \{\epsilon\}$$

$$R^0_{3,2} = \{b\}$$

$$R^0_{1,3} = \emptyset$$

$$R^0_{2,3} = \{a\}$$

$$R^0_{3,3} = \{a, \epsilon\}$$

Kleene-Verfahren: Lösung

$$\begin{aligned} R^1_{1,3} &= R^0_{1,3} \cup R^0_{1,1} (R^0_{1,1})^* R^0_{1,3} \\ &= (R^0_{1,1})^* R^0_{1,3} = \{\epsilon\}^* \emptyset = \emptyset \end{aligned}$$

$$\begin{aligned} R^1_{1,2} &= R^0_{1,2} \cup R^0_{1,1} (R^0_{1,1})^* R^0_{1,2} \\ &= (R^0_{1,1})^* R^0_{1,2} = \{\epsilon\}^* \{a\} = \{a\} \end{aligned}$$

$$R^1_{2,2} = R^0_{2,2} \cup R^0_{2,1} \dots = \{\epsilon\} \cup \emptyset = \{\epsilon\}$$

$$R^1_{2,3} = R^0_{2,3} \cup R^0_{2,1} \dots = \{a\} \cup \emptyset = \{a\}$$

$$R^1_{3,3} = R^0_{3,3} \cup R^0_{3,1} \dots = \{a, \epsilon\} \cup \emptyset = \{a, \epsilon\}$$

$$R^1_{3,2} = R^0_{3,2} \cup R^0_{3,1} \dots = \{b\} \cup \emptyset = \{b\}$$

Kleene-Verfahren: Lösung

$$\begin{aligned} R^3_{1,3} &= R^2_{1,3} \cup R^2_{1,3} (R^2_{3,3})^* R^2_{3,3} \\ &= R^2_{1,3} \cup R^2_{1,3} (R^2_{3,3})^+ \\ &= R^2_{1,3} (\{\epsilon\} \cup (R^2_{3,3})^+) = R^2_{1,3} (R^2_{3,3})^* \\ &= (R^1_{1,3} \cup R^1_{1,2} (R^1_{2,2})^* R^1_{2,3}) (R^2_{3,3})^* \\ &= (\emptyset \cup \{a\} \{\epsilon\}^* \{a\}) (R^2_{3,3})^* \\ &= \{a\} \{a\} (R^1_{3,3} \cup R^1_{3,2} (R^1_{2,2})^* R^1_{2,3})^* \\ &= \{a\} \{a\} (\{a, \epsilon\} \cup \{b\} \{\epsilon\}^* \{a\})^* \\ &= \{a\} \{a\} (\{a\} \cup \{b\} \{a\})^* \simeq \mathbf{aa (a + ba)^*} \end{aligned}$$

Das wars für heute!

Bis morgen zum Finale! 😊

FGI-1 Repetitorium (2. Termin): Automatentheorie [Tag 3]

Jan Henrik Röwekamp
(8roeweka)

18.09., 19.09. & 20.09.2013

Ablauf – Tag 3

- Pumping Lemma (uvw und uvwxy-Theorem)
- Grammatiken (CFG; Typ 2 und Typ 3)
- Grammatik-Normalformen
- Kellerautomaten
- Andere Grammatiken (Typ 0 und Typ 1)
- <Pause>
- Turing-Maschinen
- Entscheidbarkeit & Berechenbarkeitstheorie
- Komplexitätstheorie & O-Notation
- Komplexitätsklassen
- NP-Vollständigkeit

Abschlusseigenschaften

- Reguläre Mengen sind abgeschlossen gegenüber:
 - Vereinigung
 - Komplement
 - Produkt
 - Schnittbildung
 - Sternbildung
 - (Homomorphismen)

Ablauf – Tag 3

- Pumping Lemma (uvw und uvwxy-Theorem)
- Grammatiken (CFG; Typ 2 und Typ 3)
- Grammatik-Normalformen
- Kellerautomaten
- Andere Grammatiken (Typ 0 und Typ 1)
- <Pause>
- Turing-Maschinen
- Entscheidbarkeit & Berechenbarkeitstheorie
- Komplexitätstheorie & O-Notation
- Komplexitätsklassen
- NP-Vollständigkeit

Die Chomsky-Hierarchie



Überabzählbare Mengen

$\neq$

Abzählbare Mengen

$\supseteq$

Typ 0

Typ 0 -
Grammatik

DTM NTM

Aufzählbare Mengen

$\supseteq$

DTM NTM

Entscheidbare Mengen

$\supseteq$

Typ 1

Kontextsensitive
Grammatik

LBA

Kontextsensitive Mengen

uvwxy-Theorem

$\supseteq$

PDA

Kontextfreie Mengen

Typ 2

Kontextfreie
Grammatik

DPDA

Deterministisch
Kontextfreie Mengen

uvw-Theorem

$\supseteq$

Typ 3

Rechts- bzw.
linkslinere
Grammatik

Potenzautomat

DFA

NFA

ϵ -FA

GFA

ϵ -frei machen

Kleene

trivial

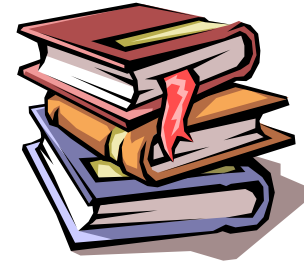
Reguläre Mengen

$\supseteq$

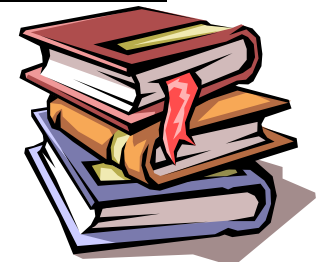
Endliche Mengen

Spezialfälle

Pumping Lemma



- Motivation: <Tafel>
- 2 Pumping Lemmas: uvw- und uvwxy-Theorem
- Beide sind sehr ähnlich.
- **Achtung: Obwohl das uvw-Theorem etwas über reguläre Mengen aussagt, kann damit nicht nachgewiesen werden, dass eine Menge regulär ist !!! Sondern nur, dass sie es nicht ist.**
- **Das Analoge gilt für das uvwxy-Theorem und kontextfreie Mengen!**



Pumping Lemma: uvw

- uvw-Theorem formal (Sei $L \subseteq \Sigma^*$) :
 $\forall L \in Reg \quad \exists n \in \mathbb{N} \quad \forall z \in L : \quad |z| \geq n \quad \Rightarrow$
 $\exists u, v, w \in \Sigma^*, z = uvw:$
 $|uv| \leq n$
 $|v| \geq 1$
 $\{u\}\{v\}^*\{w\} \subseteq L$
- Anwendung:
 - Annahme: Sprache ist regulär.
 - Auftrag: Führe das PL zum Widerspruch.

PL: Anwendung im Detail (1)

- $\exists n \in \mathbb{N} \quad \forall z \in L: |z| \geq n \Rightarrow$
 $\exists u, v, w \in \Sigma^*, z = uvw:$
 $|uv| \leq n$
 $|v| \geq 1$
 $\{u\}\{v\}^*\{w\} \subseteq L$
- 1) Existenzquantor widerlegen: Zeigen, dass es für **alle** $n \in \mathbb{N}$ **nicht** gilt. Also ein allgemeines n wählen (und damit insb. *kein* Beispiel für n !).

PL: Anwendung im Detail (2)

- $\exists n \in \mathbb{N} \quad \forall z \in L : |z| \geq n \Rightarrow$
 $\exists u, v, w \in \Sigma^*, z = uvw:$
 $|uv| \leq n$
 $|v| \geq 1$
 $\{u\}\{v\}^*\{w\} \subseteq L$
- 2) Allquantor widerlegen: Gegenbeispiel genügt. Wähle ein Beispiel-Wort in Abhängigkeit von n .

PL: Anwendung im Detail (3)

- $\exists n \in \mathbb{N} \quad \forall z \in L: \quad |z| \geq n \Rightarrow$
 $\exists u, v, w \in \Sigma^*, z = uvw:$
 $|uv| \leq n$
 $|v| \geq 1$
 $\{u\}\{v\}^*\{w\} \subseteq L$
- 3) Implikation falsifizieren: „ $1 \Rightarrow 0$ “ Fall finden:
Davon ausgehen, dass $|z| \geq n$ wahr ($=1$) ist
und zeigen, dass die rechte Seite zu 0 wird.

PL: Anwendung im Detail (4)

- $\exists n \in \mathbb{N} \quad \forall z \in L: |z| \geq n \Rightarrow$
 $\exists u, v, w \in \Sigma^*, z = uvw:$
 $|uv| \leq n$
 $|v| \geq 1$
 $\{u\}\{v\}^*\{w\} \subseteq L$
- 4) Existenzquantor widerlegen: Zeigen, dass alle theoretisch möglichen Zerlegungen zum Wahrheitswert 0 führen.

PL: Anwendung im Detail (5)

- $\exists n \in \mathbb{N} \quad \forall z \in L: \quad |z| \geq n \quad \Rightarrow$
 $\exists u, v, w \in \Sigma^*, z = uvw:$

$$|uv| \leq n$$

$$|v| \geq 1$$

$$\{u\}\{v\}^*\{w\} \subseteq L$$

- 5) Konjunktion widerlegen: Wird 0, wenn eine der Bedingungen 0 ist. Am einfachsten:
 $|uv| \leq n$ und $|v| \geq 1$ als wahr annehmen und zeigen, dass dann $\{u\}\{v\}^*\{w\} \subseteq L$ falsch ist.

PL: Beispiel-Anwendung für $a^n b^n$

- 1) Sei $k \in \mathbb{N}$ die Zahl des Pumping Lemmas.
- 2) Sei z in Abhängigkeit von k das Wort $a^k b^k \in L$
- 3) Offensichtlich gilt $|z| \geq k$
- 4+5) Alle möglichen Zerlegung des Wortes in Teile u , v und w (welche die ersten beiden Bedingungen nicht verletzen) bewirken, dass v aus a 's besteht.
- 5) Da offensichtlich $uv^2w \notin L$, folgt $L \notin \text{Reg}$

Pumping Lemma: $uvwxy$

- $uvwxy$ -Theorem formal (Sei $L \subseteq \Sigma^*$) :
 $\forall L \in Cf \quad \exists n \in \mathbb{N} \quad \forall z \in L : |z| \geq n \Rightarrow$
 $\exists u, v, w, x, y \in \Sigma^*, z = uvwxy:$
 $|vwx| \leq n$
 $|vx| \geq 1$
 $\{u\}\{v\}^i\{w\}\{x\}^i\{y\} \subseteq L \quad i \in \mathbb{N}$

Ablauf – Tag 3

- Pumping Lemma (uvw und uvwxy-Theorem)
- **Grammatiken (CFG; Typ 2 und Typ 3)**
- Grammatik-Normalformen
- Kellerautomaten
- Andere Grammatiken (Typ 0 und Typ 1)
- <Pause>
- Turing-Maschinen
- Entscheidbarkeit & Berechenbarkeitstheorie
- Komplexitätstheorie & O-Notation
- Komplexitätsklassen
- **NP-Vollständigkeit**

Die Chomsky-Hierarchie



Überabzählbare Mengen

$\neq$

Abzählbare Mengen

$\supseteq$

Typ 0

Typ 0 -
Grammatik

DTM NTM

Aufzählbare Mengen

$\supseteq$

DTM NTM

Entscheidbare Mengen

$\supseteq$

Typ 1

Kontextsensitive
Grammatik

LBA

Kontextsensitive Mengen

– uvwxy-Theorem

$\supseteq$

PDA

Kontextfreie Mengen

Typ 2

Kontextfreie
Grammatik

$\supseteq$

DPDA

Deterministisch
Kontextfreie Mengen

– uvw-Theorem

$\supseteq$

Typ 3

Rechts- bzw.
linkslinere
Grammatik

Potenzautomat

DFA

NFA

ϵ -FA

GFA

ϵ -frei machen

Kleene

trivial

Reguläre Mengen

$\supseteq$

Endliche Mengen

Spezialfälle

Kontextfreie Grammatiken

- Formal Quadrupel (4-Tupel): $G = (\Sigma, N, P, S)$
- Umgekehrter Ansatz zu Automaten:
Wort wird generiert, statt eingelesen.
- Ableitungen und Ableitungsbäume
- Praxisanwendung: Bspw. Compilerbau
- Typ 2 und Typ 3 Grammatiken
 - Rechts- und linkslineare Grammatik

Ablauf – Tag 3

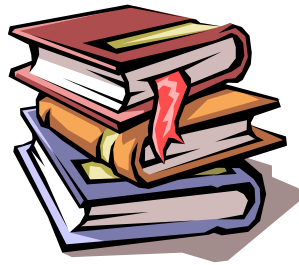
- Pumping Lemma (uvw und uvwxy-Theorem)
- Grammatiken (CFG; Typ 2 und Typ 3)
- **Grammatik-Normalformen**
- Kellerautomaten
- Andere Grammatiken (Typ 0 und Typ 1)
- <Pause>
- Turing-Maschinen
- Entscheidbarkeit & Berechenbarkeitstheorie
- Komplexitätstheorie & O-Notation
- Komplexitätsklassen
- **NP-Vollständigkeit**

Normalformen

- 2 Normalformen betrachtet:
Chomsky und (Greibach)
- Jede Grammatik kann in beide Normalformen überführt werden.
- Chomsky-Normalform (CNF) wird in 6 Schritten erzeugt.

Chomsky-Normalform erzeugen

- 1. ε -frei machen
- 2. Reduzieren
- 3. Kettenregeln entfernen
- 4. Lange Terminalregeln ersetzen
- 5. Verkürzen zu langer Regeln
- 6. Wiederherstellen der Sprache (ε)



- Bringen Sie die folgende Grammatik in CNF:
 $\Sigma = \{a, b, d\}$

$$S \rightarrow AB \mid aA$$

$$A \rightarrow B \mid A \mid bb$$

$$B \rightarrow DC \mid d \mid AB \mid \varepsilon$$

$$C \rightarrow AC \mid CB \mid C$$

$$D \rightarrow d \mid \varepsilon \mid S$$

ε -frei machen

- $M_0 := \{A' \in N \mid A' \rightarrow \varepsilon \in P\}$
 $M_{i+1} := M_i \cup \{A' \in N \mid \exists w \in M_i^* : A' \rightarrow w \in P\}$

ε -frei machen (Lösung)

- $M_0 = \{B, D\}$
- $M_1 = \{B, D\} \cup \{B, D, A\} = \{B, D, A\}$
- $M_2 = \{S, B, D, A\}$
- $M_3 = \{S, B, D, A\} = M_2 = N_\varepsilon$

$$S \rightarrow AB \mid aA$$

$$A \rightarrow B \mid A \mid bb$$

$$B \rightarrow DC \mid d \mid AB \mid \varepsilon$$

$$C \rightarrow AC \mid CB \mid C$$

$$D \rightarrow d \mid \varepsilon \mid S$$

$$S \rightarrow AB \mid aA \mid A \mid B \mid \cancel{\varepsilon} \mid a$$

$$A \rightarrow B \mid A \mid bb \mid \cancel{\varepsilon}$$

$$B \rightarrow DC \mid d \mid AB \mid \cancel{\varepsilon} \mid C \mid A \mid B$$

$$C \rightarrow AC \mid CB \mid C$$

$$D \rightarrow d \mid \cancel{\varepsilon} \mid S$$

$$S \in N_\varepsilon !$$

Das Startsymbol S lässt sich direkt oder indirekt auf ε ableiten, die Sprache beinhaltet also das leere Wort! Dieses muss in Schritt 6 wieder hinzugefügt werden!

Produktive Nonterminale

- $M_0 := \Sigma$
 $M_{i+1} := M_i \cup \{A' \in N \mid \exists w \in M_i^* : A' \rightarrow w \in P\}$

Produktive Nonterminale (Lösung)

- $M_0 = \{a, b, d\}$
- $M_1 = \{a, b, d, S, A, B, D\}$
- $M_2 = \{a, b, d, S, A, B, D\} = M_1$

$S \rightarrow AB \mid aA \mid A \mid B \mid a$

$A \rightarrow B \mid A \mid bb$

$B \rightarrow DC \mid d \mid AB \mid C \mid A \mid B$

$C \rightarrow AC \mid CB \mid C$

$D \rightarrow d \mid S$

$S \rightarrow AB \mid aA \mid A \mid B \mid a$

$A \rightarrow B \mid A \mid bb$

$B \rightarrow \cancel{DC} \mid d \mid AB \mid \cancel{C} \mid A \mid B$

~~$C \rightarrow AC \mid CB \mid C$~~

$D \rightarrow d \mid S$

Erreichbare Nonterminale

- $M_0 := \{S\}$
 $M_{i+1} := M_i \cup$
 $\{B' \in N \mid \exists A' \in M_i \ \exists u, v \in (N \cup \Sigma)^* :$
 $A' \rightarrow uB'v \in P\}$

Erreichbare Nonterminale (Lösung)

- $M_0 = \{S\}$
- $M_1 = \{S, A, B\}$
- $M_2 = \{S, A, B\} = M_1$

$S \rightarrow AB \mid aA \mid A \mid B \mid a$

$A \rightarrow B \mid A \mid bb$

$B \rightarrow d \mid AB \mid A \mid B$

$D \rightarrow d \mid S$

$S \rightarrow AB \mid aA \mid A \mid B \mid a$

$A \rightarrow B \mid A \mid bb$

$B \rightarrow d \mid AB \mid A \mid B$

~~$D \rightarrow d \mid S$~~

Schritte 3-6 Lösung

$S \rightarrow AB \mid aA \mid A \mid B \mid a$

$A \rightarrow B \mid A \mid bb$

$B \rightarrow d \mid AB \mid A \mid B$

$S \rightarrow AB \mid aA \mid d \mid bb \mid a$

$A \rightarrow d \mid AB \mid bb$

$B \rightarrow d \mid AB \mid bb$

3

$S \rightarrow AB \mid \langle a \rangle A \mid d \mid \langle b \rangle \langle b \rangle \mid a$

$A \rightarrow d \mid AB \mid \langle b \rangle \langle b \rangle$

$B \rightarrow d \mid AB \mid \langle b \rangle \langle b \rangle$

$\langle a \rangle \rightarrow a$

$\langle b \rangle \rightarrow b$

4

6

$S \rightarrow AB \mid \langle a \rangle A \mid d \mid \langle b \rangle \langle b \rangle \mid a \mid \epsilon$

$A \rightarrow d \mid AB \mid \langle b \rangle \langle b \rangle$

$B \rightarrow d \mid AB \mid \langle b \rangle \langle b \rangle$

$\langle a \rangle \rightarrow a$

$\langle b \rangle \rightarrow b$

Ablauf – Tag 3

- Pumping Lemma (uvw und uvwxy-Theorem)
- Grammatiken (CFG; Typ 2 und Typ 3)
- Grammatik-Normalformen
- **Kellerautomaten**
- Andere Grammatiken (Typ 0 und Typ 1)
- <Pause>
- Turing-Maschinen
- Entscheidbarkeit & Berechenbarkeitstheorie
- Komplexitätstheorie & O-Notation
- Komplexitätsklassen
- **NP-Vollständigkeit**

Die Chomsky-Hierarchie



Überabzählbare Mengen

$\neq$

Abzählbare Mengen

$\supseteq$

Typ 0

Typ 0 -
Grammatik

DTM NTM

Aufzählbare Mengen

$\supseteq$

DTM NTM

Entscheidbare Mengen

$\supseteq$

Typ 1

Kontextsensitive
Grammatik

LBA

Kontextsensitive Mengen

– uvwxy-Theorem

$\supseteq$

Typ 2

Kontextfreie
Grammatik

PDA

Kontextfreie Mengen

DPDA

$\supseteq$
Deterministisch
Kontextfreie Mengen

– uvw-Theorem

$\supseteq$

Typ 3

Rechts- bzw.
linkslinere
Grammatik

Potenzautomat

DFA

NFA

ϵ -FA

GFA

ϵ -frei machen

Kleene

trivial

Reguläre Mengen

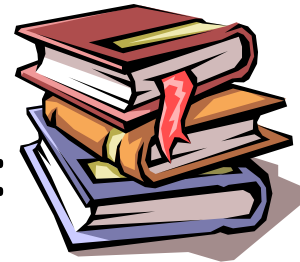
$\supseteq$

Endliche Mengen

Spezialfälle

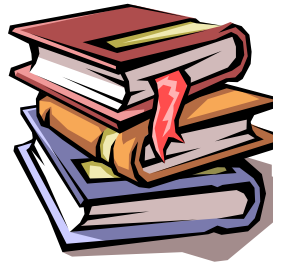
Kellerautomaten (PDAs)

- Formalismus analog zu endlichen Automaten.
- Aber:
 - Zusätzlicher Stack-Speicher
 - Nichtdeterminismus nicht mehr so offensichtlich
 - DPDAs und NPDAs akzeptieren nicht die gleiche Sprachfamilie!
 - 2 Möglichkeiten der Akzeptanz:
Leerer Keller und Endzustand



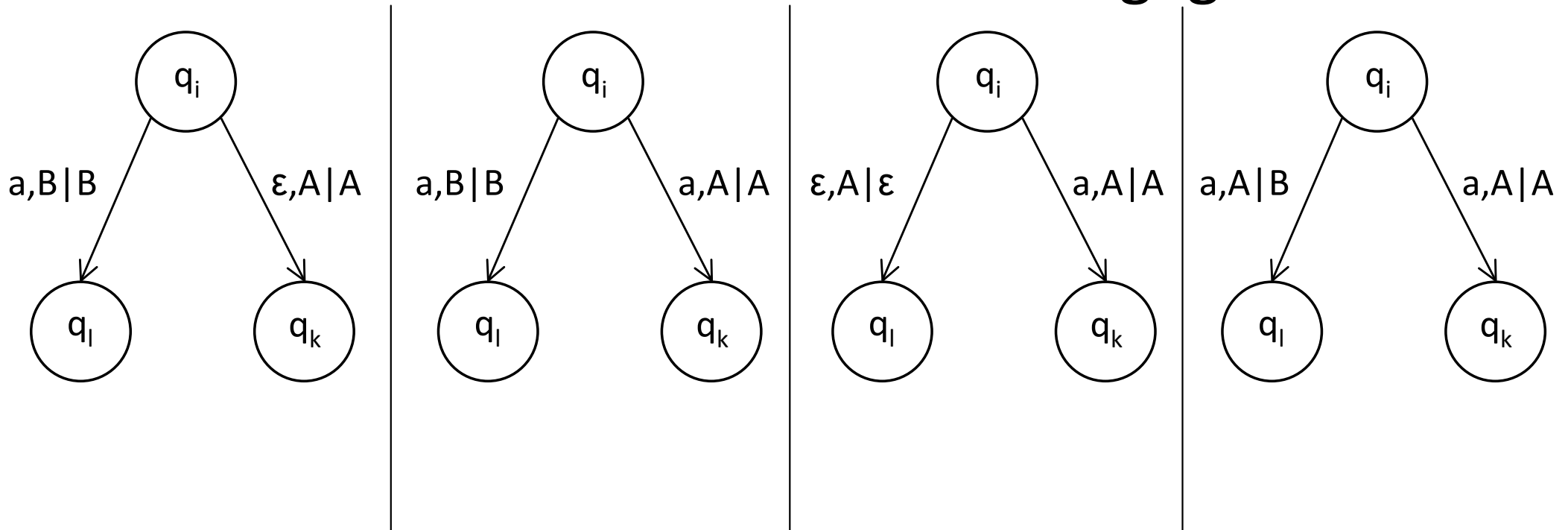
PDA: Akzeptanz & Konfiguration

- Akzeptanz mit leerem Keller ($L_\epsilon(A)$):
 - Wort wurde komplett gelesen, und der Keller ist komplett leer (auch kein Kellerbodensymbol!)
- Akzeptanz mit Endzustand ($L(A)$):
 - Wort wurde komplett gelesen, und befindet sich im Endzustand.
- Konfiguration eines PDA:
 - Element aus $Q \times \Sigma^* \times \Gamma^*$ also z.B.: $(q_0, abb, A\perp)$



PDA: Nichtdeterminismus

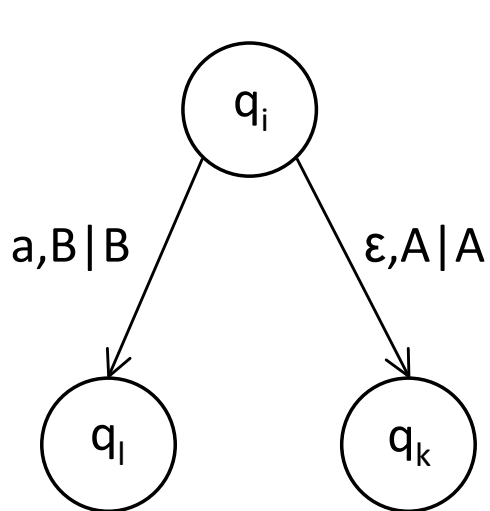
- Die folgenden Ausschnitte seien aus einem ansonsten deterministischen PDA gegeben:



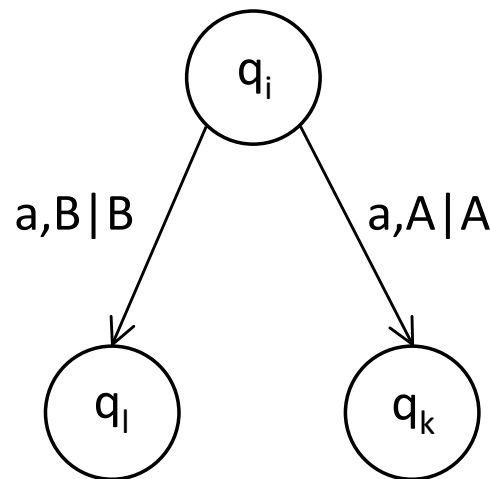
- Welche Automaten sind DPDA's? Welche nicht?

PDA: Nichtdet.: Lösung

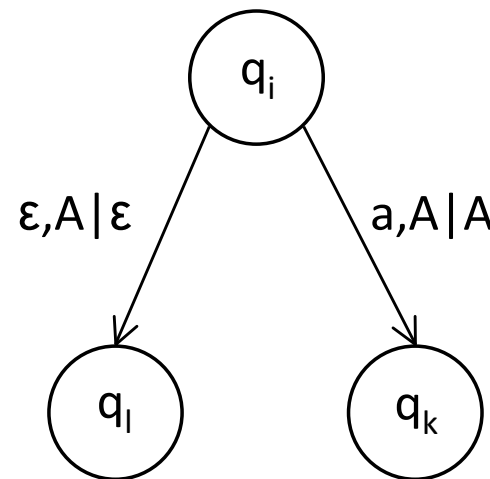
- Die folgenden Ausschnitte seien aus einem ansonsten deterministischen PDA gegeben:



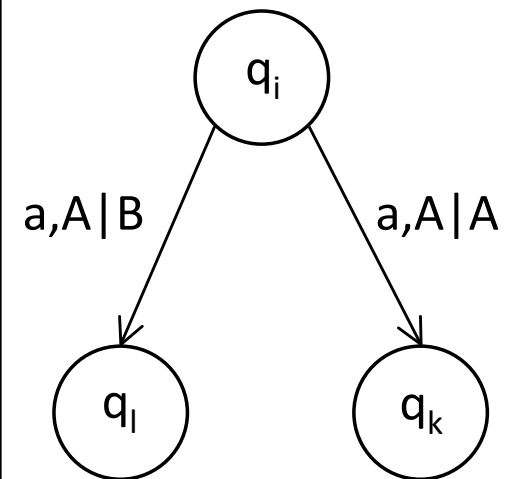
DPDA



DPDA



NPDA



NPDA

- Welche Automaten sind DPDA's? Welche nicht?

Ablauf – Tag 3

- Pumping Lemma (uvw und uvwxy-Theorem)
- Grammatiken (CFG; Typ 2 und Typ 3)
- Grammatik-Normalformen
- Kellerautomaten
- **Andere Grammatiken (Typ 0 und Typ 1)**
- <Pause>
- Turing-Maschinen
- Entscheidbarkeit & Berechenbarkeitstheorie
- Komplexitätstheorie & O-Notation
- Komplexitätsklassen
- **NP-Vollständigkeit**

Die Chomsky-Hierarchie



Überabzählbare Mengen

$\neq$

Abzählbare Mengen

$\supseteq$

Typ 0

Typ 0 -
Grammatik

DTM NTM

Aufzählbare Mengen

$\supseteq$

DTM NTM

Entscheidbare Mengen

$\supseteq$

Typ 1

Kontextsensitive
Grammatik

LBA

Kontextsensitive Mengen

– uvwxy-Theorem

$\supseteq$

PDA

Kontextfreie Mengen

Typ 2

Kontextfreie
Grammatik

DPDA

Deterministisch
Kontextfreie Mengen

– uvw-Theorem

$\supseteq$

Typ 3

Rechts- bzw.
linkslinere
Grammatik

Potenzautomat

DFA

NFA

ϵ -FA

GFA

ϵ -frei machen

Kleene

trivial

Reguläre Mengen

$\supseteq$

Endliche Mengen

Spezialfälle

Typ 0 und Typ 1 Grammatiken

- Typ 1 Grammatiken werden auch „kontextsensitive Grammatiken“ oder „monotone Grammatiken“ genannt.
- Existenz von Grammatiken, die Typ 2 (CFG), aber nicht Typ 1 sind.
- Typ 0 Grammatiken unterliegen nicht mehr der Monotonie-Einschränkung.
- Typ 0 Grammatiken sind sehr mächtig.

PAUSE
(30 Minuten)

Ablauf – Tag 3

- Pumping Lemma (uvw und uvwxy-Theorem)
- Grammatiken (CFG; Typ 2 und Typ 3)
- Grammatik-Normalformen
- Kellerautomaten
- Andere Grammatiken (Typ 0 und Typ 1)
- <Pause>
- **Turing-Maschinen**
- Entscheidbarkeit & Berechenbarkeitstheorie
- Komplexitätstheorie & O-Notation
- Komplexitätsklassen
- **NP-Vollständigkeit**

Die Chomsky-Hierarchie



Überabzählbare Mengen

$\neq$

Abzählbare Mengen

$\supseteq$

Typ 0

Typ 0 -
Grammatik

DTM NTM

Aufzählbare Mengen

$\supseteq$

DTM NTM

Entscheidbare Mengen

$\supseteq$

Typ 1

Kontextsensitive
Grammatik

LBA

Kontextsensitive Mengen

– uvwxy-Theorem

$\supseteq$

PDA

Kontextfreie Mengen

Typ 2

Kontextfreie
Grammatik

DPDA

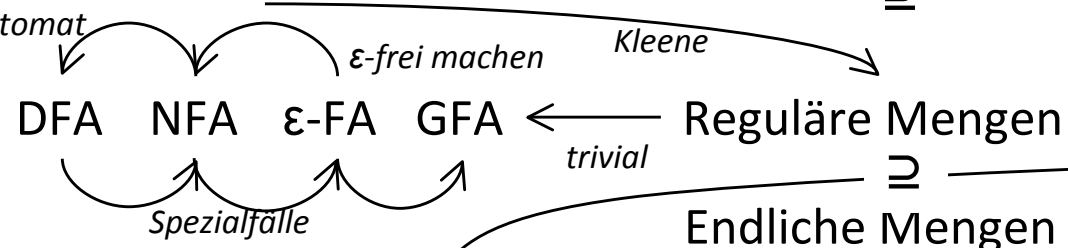
Deterministisch
Kontextfreie Mengen

– uvw-Theorem

$\supseteq$

Typ 3

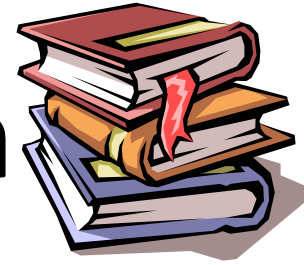
Rechts- bzw.
linkslinere
Grammatik



Turing-Maschinen

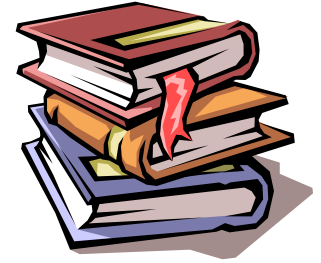
- DTM formal 7-Tupel: $T = (Q, \Sigma, \Gamma, \delta, q_0, \#, F)$
 - Q ist eine Menge an Zuständen
 - Σ ist eine Eingabealphabet
 - Γ ein Bandalphabet
 - $\delta : (Q \times \Gamma) \rightarrow (Q \times \Gamma \times \{l, r, -\})$ eine Überföhrungsfkt.
 - $q_0 \in Q$ ist ein Startzustand
 - $\# \in \Gamma$ ist das Bandendsymbol
 - $F \subseteq Q$ ist eine Menge an Endzuständen

Turing-Maschinen



- Variabel beweglicher Lese-Schreibkopf
- NTM und DTM akzeptieren gleiche Sprachfamilie (Typ 0)
- Turing-Mächtigkeit & Satz von Turing
- Rechnungen und Erfolgsrechnungen
- Abbruch beim Erreichen des ersten Endzustands
- Nicht alles muss gelesen werden!
- Mehrere Bänder möglich (k-Band-TM)

Konfiguration & Rechnung



- Eine Konfiguration einer TM ist ein Element aus:
 $\Gamma^* \cdot Q \cdot \Gamma^+$ also z.B. $\#q_0abb\#$
- Q gibt den Zustand an, in dem sich die TM befindet.
- Γ^* und Γ^+ den Inhalt des Arbeitsbandes links vom Schreibkopf (Γ^*) und auf und rechts vom SK (Γ^+).
- Eine **Rechnung** ist eine Folge von Konfigurationen
- Endet die Folge in einem Endzustand wird von einer **Erfolgsrechnung** gesprochen.

Ablauf – Tag 3

- Pumping Lemma (uvw und uvwxy-Theorem)
- Grammatiken (CFG; Typ 2 und Typ 3)
- Grammatik-Normalformen
- Kellerautomaten
- Andere Grammatiken (Typ 0 und Typ 1)
- <Pause>
- Turing-Maschinen
- **Entscheidbarkeit & Berechenbarkeitstheorie**
- Komplexitätstheorie & O-Notation
- Komplexitätsklassen
- **NP-Vollständigkeit**

Die Chomsky-Hierarchie



Überabzählbare Mengen

$\neq$

Abzählbare Mengen

$\supseteq$

Aufzählbare Mengen

$\supseteq$

Entscheidbare Mengen

$\supseteq$

Kontextsensitive Mengen

$\supseteq$

Kontextfreie Mengen

$\supseteq$

Deterministisch
Kontextfreie Mengen

$\supseteq$

Reguläre Mengen

$\supseteq$

Endliche Mengen

Typ 0

Typ 0 -
Grammatik

DTM NTM

DTM NTM

Typ 1

Kontextsensitive
Grammatik

LBA

– uvwxy-Theorem

PDA

Typ 2

Kontextfreie
Grammatik

DPDA

– uvw-Theorem

Typ 3

Rechts- bzw.
linkslinere
Grammatik

Potenzautomat

DFA

NFA

ϵ -FA

GFA

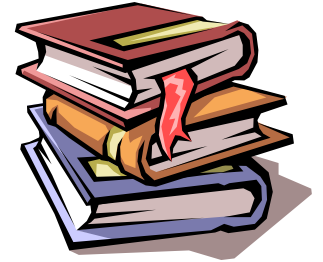
ϵ -frei machen

Kleene

trivial

Spezialfälle

Berechenbarkeitstheorie: Begriffe



- **Abzählbare Menge:**

Es existiert eine Bijektion aus den natürlichen Zahlen auf die Menge.

- **Aufzählbare Menge:**

Die Menge ist abzählbar und Elemente der Menge sind in endlicher Zeit durch eine TM als solche identifizierbar.

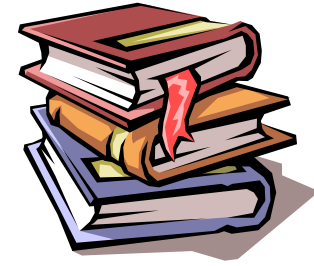
- **Entscheidbare Menge:**

Die Menge und ihr Komplement sind aufzählbar

Berechenbarkeitstheorie: Begriffe

- **Überabzählbare Menge:**
 - Es existiert keine Bijektion aus den natürlichen Zahlen auf die Menge BZW.
 - Jede Liste bzw. Struktur einer Liste von Elementen der Menge ist unvollständig BZW.
 - Die Menge kann nicht geordnet werden (In dem Sinne, dass es ein benennbares „erstes“, „zweites“, „drittes“, usw. Element gäbe).
- Überabzählbare Mengen existieren: z.B. $\mathbb{R}$
- Beweis: Diagonalisierung

Das Halteproblem H

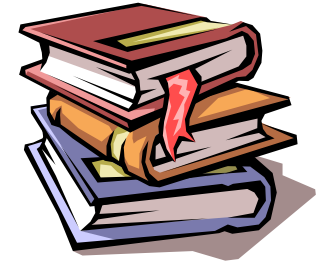


- Bezeichnet das Problem, ob ein Programm bei gegebener Eingabe nach endlich vielen Schritten hält, oder nicht.
- Das Halteproblem ist aufzählbar (Das gegebene Programm kann, wenn es hält, einfach ausgeführt werden bis es hält.
→ „Ja“-Instanzen können erkannt werden.)
- Das Halteproblem ist jedoch nicht entscheidbar!
- Sehr wichtig in der Informatik

Charakteristische Funktion

- Die charakteristische Funktion
 $\chi_L : \Sigma^* \rightarrow \{0, 1\}$
der Menge L gibt an, ob ein Element in der Menge liegt (1), oder aber nicht (0).
- Ist die charakteristische Funktion berechnbar, so ist die Menge entscheidbar. (und umgekehrt!)
- Bindeglied zwischen Berechenbarkeit (von Fkt.) und Entscheidbarkeit (von Mengen)

Beispiele zu den Mengen



- Überabzählbar: $\mathbb{R}, 2^{\mathbb{N}}$
- Abzählbar, aber nicht aufzählbar: L_d
- Aufzählbar (Typ 0), aber nicht entscheidbar: H
- Entscheidbar, aber nicht Typ 1: L_e
- Typ 1, aber nicht Typ 2: $a^n b^n c^n$
- N.det. Typ 2, aber nicht det. Typ 2: ww^{rev}
- Det. Typ 2, aber nicht Typ 3: $a^n b^n$
- Typ 3, aber nicht endlich: a^*

Ablauf – Tag 3

- Pumping Lemma (uvw und uvwxy-Theorem)
- Grammatiken (CFG; Typ 2 und Typ 3)
- Grammatik-Normalformen
- Kellerautomaten
- Andere Grammatiken (Typ 0 und Typ 1)
- <Pause>
- Turing-Maschinen
- Entscheidbarkeit & Berechenbarkeitstheorie
- **Komplexitätstheorie & O-Notation**
- Komplexitätsklassen
- **NP-Vollständigkeit**

Komplexitätstheorie:

Jenseits der Chomsky-Hierarchie

- Chomsky-Hierarchie legt keinen Wert auf die **Geschwindigkeit** der Berechnung.
- In der Praxis: Geschwindigkeit hochgradig relevant
- → (strukturelle) Komplexitätstheorie
- Wichtiges Hilfsmittel der Komplexitätstheorie: O-Notation

O-Notation

- Etwaige asymptotische Abschätzung der Laufzeit in Abhängigkeit von der Eingabelänge
- Konstante Faktoren werden ignoriert
- Relevant ist nur der „schwerwiegendste“ Anteil der Laufzeitfunktion
- $O(f(n))$ bedeutet frei: Maximal so schlimm, wie $f(n)$ (ohne konstanten Faktor)
- Z.B.: $3n^3 + 5n^2 + 3n + 7$ liegt in $O(n^3)$. Aber auch in $O(n^5)$ und $O(2^n)$. Aber nicht in $O(n^2)$.

O-Notation

- Zusätzliche Mengen:

- $o(f(n))$ „ $<$ “

- $\Omega(f(n))$ „ $\geq$ “

- $\omega(f(n))$ „ $>$ “

- $\Theta(f(n))$ „ $=$ “

- Mathematischer Nachweis:

$g(n) \in O(f(n))$ gdw. $\lim_{n \rightarrow \infty} g(n)/f(n) < \infty$

$g(n) \in o(f(n))$ gdw. $\lim_{n \rightarrow \infty} g(n)/f(n) = 0$

$g(n) \in \Omega(f(n))$ gdw. $\lim_{n \rightarrow \infty} f(n)/g(n) < \infty$

Zeit- und Platzmaße

- Zeitmessung in Anzahl der Zustandsübergänge in Abhängigkeit von der Eingabelänge n :
 - Obergrenze $t(n)$
- Platzmessung in Anzahl der verwendeten Bandzellen in Abhängigkeit von der Eingabelänge n :
 - Obergrenze $s(n)$

Einschub: LBA

- LBA = „Linear beschränkter Automat“
- TM mit der Einschränkung:
 - $s(n) \in O(n)$
- Äquivalent zu Typ 1 Sprachen, bzw. kontextsensitiven Mengen C_s .

Ablauf – Tag 3

- Pumping Lemma (uvw und uvwxy-Theorem)
- Grammatiken (CFG; Typ 2 und Typ 3)
- Grammatik-Normalformen
- Kellerautomaten
- Andere Grammatiken (Typ 0 und Typ 1)
- <Pause>
- Turing-Maschinen
- Entscheidbarkeit & Berechenbarkeitstheorie
- Komplexitätstheorie & O-Notation
- **Komplexitätsklassen**
- **NP-Vollständigkeit**

Komplexitätsklassen formal

- $DTime(t(n))$: Alle Sprachen, die eine DTM in maximal $t(n)$ Zeitschritten entscheidet
- $NTime(t(n))$: -- “ -- NTM
- $DSpace(s(n))$: Alle Sprachen, die eine DTM mit maximal $s(n)$ Bandzellen entscheidet
- $NSpace(s(n))$: -- “ -- NTM

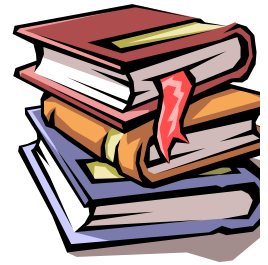
-
- $P = \bigcup_{i \geq 1} DTime(n^i)$ Gilt
 - $NP = \bigcup_{i \geq 1} NTime(n^i)$ $\bigcup_{i \geq 1} DTime(n^i)$
 - $PSPACE = \bigcup_{i \geq 1} DSpace(n^i)$ =
 - $NPSpace = \bigcup_{i \geq 1} NSpace(n^i)$ $\bigcup_{i \geq 1} DTime((n^i)^2) ?$

Lösung zur Frage

- Ja, die Aussage ist korrekt.
- DTime gibt Auskunft über **obere Schranken** von Anzahlen von Berechnungsschritten.
- Quadriert man ein Polynom, ist es immernoch ein Polynom!
- Die Mengen sind exakt die gleichen.
- Bspw. Liegt n in $O(n)$,
aber auch in $O(n^2)$.
- Man verliert also keine Informationen durch die Quadratbildung.

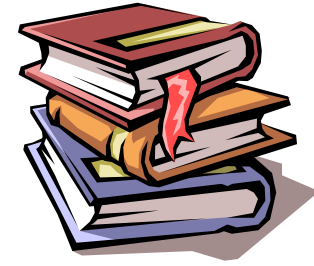
$$\begin{array}{l} \text{Gilt} \\ \bigcup_{i \geq 1} \text{DTime}(n^i) \\ = \\ \bigcup_{i \geq 1} \text{DTime}((n^i)^2) \quad ? \end{array}$$

Komplexitätsklassen informell



- **P:** Menge aller Sprachen, die von einer deterministischen TM mit polynomiellem Zeitbedarf entschieden werden können.
- **NP:** Menge aller Sprachen, die von einer nichtdeterministischen TM mit polynomiellem Zeitbedarf entschieden werden können.
- **PSpace:** Menge aller Sprachen, die von einer deterministischen TM mit polynomiellem Platzbedarf entschieden werden können.
- **NPSpace:** Menge aller Sprachen, die von einer nichtdeterministischen TM mit polynomiellem Platzbedarf entschieden werden können.

Komplexitätsklassen



- $P \subseteq NP$ und $PSpace \subseteq NPSPACE$:
 - Gilt, da jede DTM eine spezielle NTM ist.
- Satz von Savitch besagt: $NPSPACE \subseteq PSpace$
 - Beweis zu umfangreich fürs Repetitorium.
- $NP \subseteq PSpace$ gilt.
- Insgesamt: **$P \subseteq NP \subseteq PSpace = NPSPACE$**
- $NP \subseteq PSpace$ Begründung:
 - Sei $L \in NP$.
 - Folglich gibt es eine NTM, die L in poly. Zeit entscheidet.

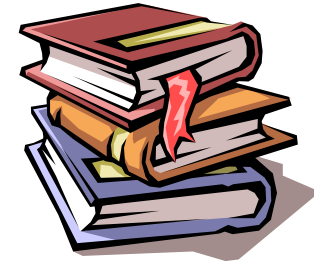
$NP \subseteq PSpace$ (Fortsetzung)

- Damit die NTM in poly. Zeit laufen kann, muss die theoretisch längste (zeitintensivste) Rechnung auch noch polynomiell sein.
- In t Zeitschritten können **nur maximal t Bandzellen** verwendet werden (*Eine neue Zelle in jedem Schritt*).
- Wird die **NTM durch eine DTM simuliert**, so muss jede mögliche Rechnung durchlaufen werden, bis das Wort akzeptiert wird, oder aber alle möglichen Rechnungen betrachtet wurden.
- Das Ergebnis der letzten Rechnung spielt keine Rolle für die nächste Rechnung, der Platz kann also (gelöscht und) **wiederverwendet werden**.
- Im Worst Case braucht die DTM also t Bandzellen um L zu entscheiden. t ist polynomiell.
- Es folgt direkt $L \in PSpace \quad \rightarrow NP \subseteq PSpace$

Ablauf – Tag 3

- Pumping Lemma (uvw und uvwxy-Theorem)
- Grammatiken (CFG; Typ 2 und Typ 3)
- Grammatik-Normalformen
- Kellerautomaten
- Andere Grammatiken (Typ 0 und Typ 1)
- <Pause>
- Turing-Maschinen
- Entscheidbarkeit & Berechenbarkeitstheorie
- Komplexitätstheorie & O-Notation
- Komplexitätsklassen
- **NP-Vollständigkeit**

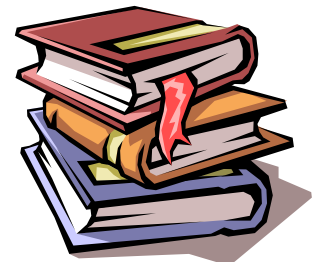
NP-Vollständigkeit



- Schwerwiegendste Probleme in NP.
- Mindestens so schwer, wie alle anderen Probleme in NP.
- Alle anderen Probleme in NP lassen sich als Spezialfälle eines NP-vollständigen Problem modellieren.
- Existenz lange Zeit ungeklärt.
- Erstes NP-vollständige Problem entdeckt durch Cook und Levin

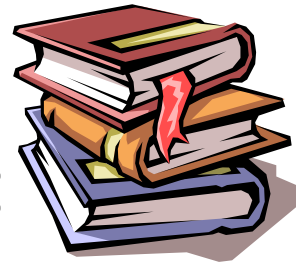
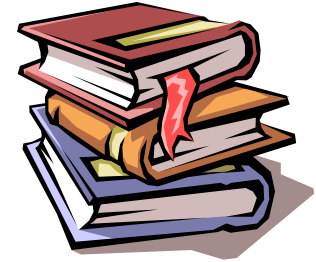
Reduktionen

- Überführung eines Problems in einen Spezialfall eines schwerwiegenderen (oder mindestens so schweren) Problems.
- Reduktion von A (kleineres Problem) nach B (schwereres Problem) notiert als: $A \leq B$
 - Achtung: $\leq$ trifft keine Aussage über eine Richtung, sondern über die Schwere der Probleme
- Ist die Reduktion deterministisch in Polynomialzeit möglich, schreibt man: $A \leq_{\text{POL}} B$



NP-Vollständigkeit

- Ein Problem L ist NP-vollständig gdw.:
 - L in NP liegt. UND
 - Alle anderen Probleme A aus NP in deterministischer Polynomialzeit auf L reduzierbar sind.
(L „NP-schwer“ ist)
- NP-Schwere ist transitiv:
 - NP-Schwere kann durch Reduktion eines NP-schweren Problem auf das neue Problem gezeigt werden



E N D E

Vielen Dank für eure Aufmerksamkeit!
Viel Erfolg bei der Klausur!

Ggf. Hinweis für die Zukunft:
Ich gebe ich auch privat Nachhilfe! (25€ / 60 Min.).
Einfach an 8roeweka@inf... mailen.

Wenn ihr **Feedback** habt (negativ/positiv egal), könnt
ihr mir auch gern mailen, ich freue mich über jede
Rückmeldung! 😊