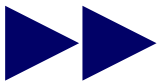
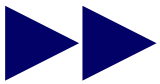
Theoretische Informatik
Kap 3: Komplexitätstheorie



Gliederung der Vorlesung

Theoretische Informatik
Kap 3: Komplexitätstheorie



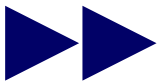
Motivation/Einordnung

Zentrale Fragestellungen

- Anhand welcher Kriterien beurteilt man die Güte von Algorithmen zur Lösung eines Problems?
- Wie gut können Algorithmen zur Lösung eines Problems sein?
- Kann man einem Problem ansehen, ob es einen effizienten Algorithmus zu seiner Lösung gibt?
- ...

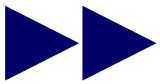
... wir konzentrieren uns auf die Ressource „Rechenzeit“

Theoretische Informatik
Kap 3: Komplexitätstheorie



Gliederung der Vorlesung

Theoretische Informatik
Kap 3: Komplexitätstheorie



Nicht-deterministische Turing-Maschinen

→ im Kapitel „Formale Sprachen/Automatentheorie“ wurde der Aspekt deterministische vs. nicht-deterministische „Maschinenmodelle“ bereits ausführlich diskutiert

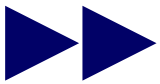
Schwerpunkte der bisherigen Diskussion

- Leistungsfähigkeit der verwendeten „Maschinenmodelle“
- Größe der Beschreibungen von „Maschinen“ für dieselbe Aufgabe

Schwerpunkt der Diskussion für Turing-Maschinen

- Effizienz von auf diesen „Maschinenmodellen“ realisierbaren Algorithmen

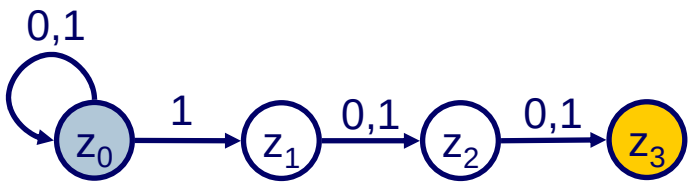
Theoretische Informatik
Kap 3: Komplexitätstheorie



Nicht-deterministische Turing-Maschinen

Zur Erinnerung

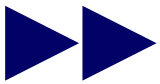
ein NFA für die Sprache L = { u1v | u,v ∈ {0,1}*, |v| = 2 }



ein Wort w ∈ {0,1}* wird akzeptiert, falls

- der NFA, gestartet im Anfangszustand, bei Eingabe von w in den Endzustand „gelangen“ kann

Theoretische Informatik
Kap 3: Komplexitätstheorie



Nicht-deterministische Turing-Maschinen

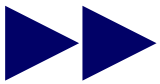
für nicht-deterministische „Maschinenmodelle“ ist typisch, daß

- nicht eindeutig festgelegt ist, was im nächsten „Rechenschritt“ passiert
- in bestimmten Situationen mehrere Aktionen möglich sind (/* es gibt mehrere Möglichkeiten, die Arbeit fortzusetzen */)

Konsequenz

zu jeder Eingabe gibt es mehrere Berechnungen, die zu unterschiedlichen Ergebnissen führen können

Theoretische Informatik
Kap 3: Komplexitätstheorie

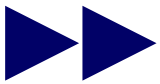


Nicht-deterministische Turing-Maschinen

Vor- und Nachteile nicht-deterministischer „Maschinenmodelle“

- sie sind technisch nicht realisierbar (/* im besten Fall auf deterministischen „Maschinen“ simulierbar */)
- auf ihnen lassen sich grundlegende algorithmische Ideen ziemlich einfach realisieren (/* kompakte Beschreibung der algorithmischen Ideen */)
- sie sind zur Formulierung grundlegender Konzepte in der Komplexitätstheorie wichtig

Theoretische Informatik
Kap 3: Komplexitätstheorie

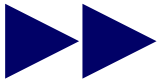


Nicht-deterministische Turing-Maschinen

Fokus

- betrachten deterministische und nicht-deterministische Turing-Maschinen zum Akzeptieren von formalen Sprachen
- untersuchen die Effizienz von sprachakzeptierenden Turing-Maschinen mit Blick auf die Ressource „Rechenzeit“

Theoretische Informatik
Kap 3: Komplexitätstheorie

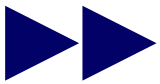


Nicht-deterministische Turing-Maschinen

Beispiel eine det. sprachakzeptierende TM M für die Sprache L mit
L = { w$\underline{w}$ | w $\in$ { a,b }* \ { ε }, $\underline{w}$ ist die gespiegelte Version von w }

	a	b	B
z_0	z_1,B,R	z_3,B,R	
z_1	z_1,a,R	z_1,b,R	z_2,B,L
z_2	z_5,B,L		
z_3	z_3,a,R	z_3,b,R	z_4,B,L
z_4		z_5,B,L	
z_5	z_6,a,L	z_6,b,L	z_e,B,N
z_6	z_6,a,L	z_6,b,L	z_0,B,R

Theoretische Informatik
Kap 3: Komplexitätstheorie



Nicht-deterministische Turing-Maschinen

Bestimmungsstücke einer det. sprachakzeptierenden TM M

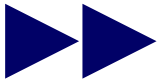
- endliche Menge Z von Zuständen
- Anfangszustand z_0, Endzustand z_e
- Bandalphabet Γ mit Blanksymbol B
- Eingabealphabet $\Sigma \subseteq \Gamma \setminus \{ B \}$
- Zustandsüberführungsfunktion

$$\delta: (Q \setminus \{ z_e \}) \times \Gamma \rightarrow Q \times \Gamma \times \{ L,R,N \}$$

die von M akzeptierte Sprache L(M)

- $w \in L(M)$, falls $w \in \Sigma^*$ und M die Anfangskonfiguration z_0w in eine akzeptierende Konfiguration uz_ev überführt

Theoretische Informatik
Kap 3: Komplexitätstheorie

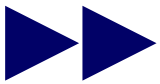


Nicht-deterministische Turing-Maschinen

Beispiel eine nicht-det. sprachakzeptierende TM N für die Sprache L mit
L = { w | w $\in$ { a,b }*, w enthält das Teilwort baba }

	a	b	B
z_0	z_0,a,R	z_0,b,R z_1,b,R	
z_1	z_2,a,R		
z_2		z_3,b,R	
z_3	z_e,a,N		

Theoretische Informatik
Kap 3: Komplexitätstheorie



Nicht-deterministische Turing-Maschinen

Beispiel eine nicht-det. sprachakzeptierende TM N für die Sprache L mit
L = { bin(q) | q ist keine Primzahl }

Arbeitsweise von N

Schritt 1:

starte eine nicht-det. TM N‘, die die Binärdarstellung einer Zahl t größer als 1 produziert

Schritt 2:

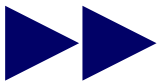
starte auf der Eingabe bin(q)Bbin(t) eine det. TM M‘, die genau dann in den Endzustand geht, wenn
t <q und t ein Teiler von q ist

Nicht-deterministische Turing-Maschinen

Beschreibung der nicht-det. TM N‘

	0	1	B
z_0			z_1,1,R
z_1			z_1,0,R z_1,1,R z_2,B,L
z_2	z_2,0,L	z_2,1,L	z_e,B,R

Theoretische Informatik
Kap 3: Komplexitätstheorie



Nicht-deterministische Turing-Maschinen

Bestimmungsstücke einer nicht-det. sprachakzeptierenden TM N

- endliche Menge Z von Zuständen
- Anfangszustand z_0, Endzustand z_e
- Bandalphabet Γ mit Blanksymbol B
- Eingabealphabet $\Sigma \subseteq \Gamma \setminus \{ B \}$
- Zustandsüberführungsfunktion

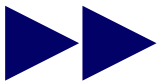
$$\delta: (Q \setminus \{ z_e \}) \times \Gamma \rightarrow 2^{Q \times \Gamma \times \{ L,R,N \}}$$

die von N akzeptierte Sprache L(N)

- $w \in L(N)$, falls $w \in \Sigma^*$ und es eine Möglichkeit gibt, daß N die Anfangskonfiguration z_0w in eine akzeptierende Konfiguration uz_ev überführt

Hinweis: es kann mehrere akzeptierende Konfigurationen geben

Theoretische Informatik
Kap 3: Komplexitätstheorie



Nicht-deterministische Turing-Maschinen

es sei Σ das zugrunde liegende Alphabet

es sei $L \subseteq \Sigma$

Satz

Wenn es eine sprachakzeptierende nicht-det. Turing-Maschine N mit L(N) = L gibt, so gibt es auch eine sprachakzeptierende det. Turing-Maschine M mit L(M) = L.

... beide „Maschinenmodelle“ haben dieselbe Leistungsfähigkeit