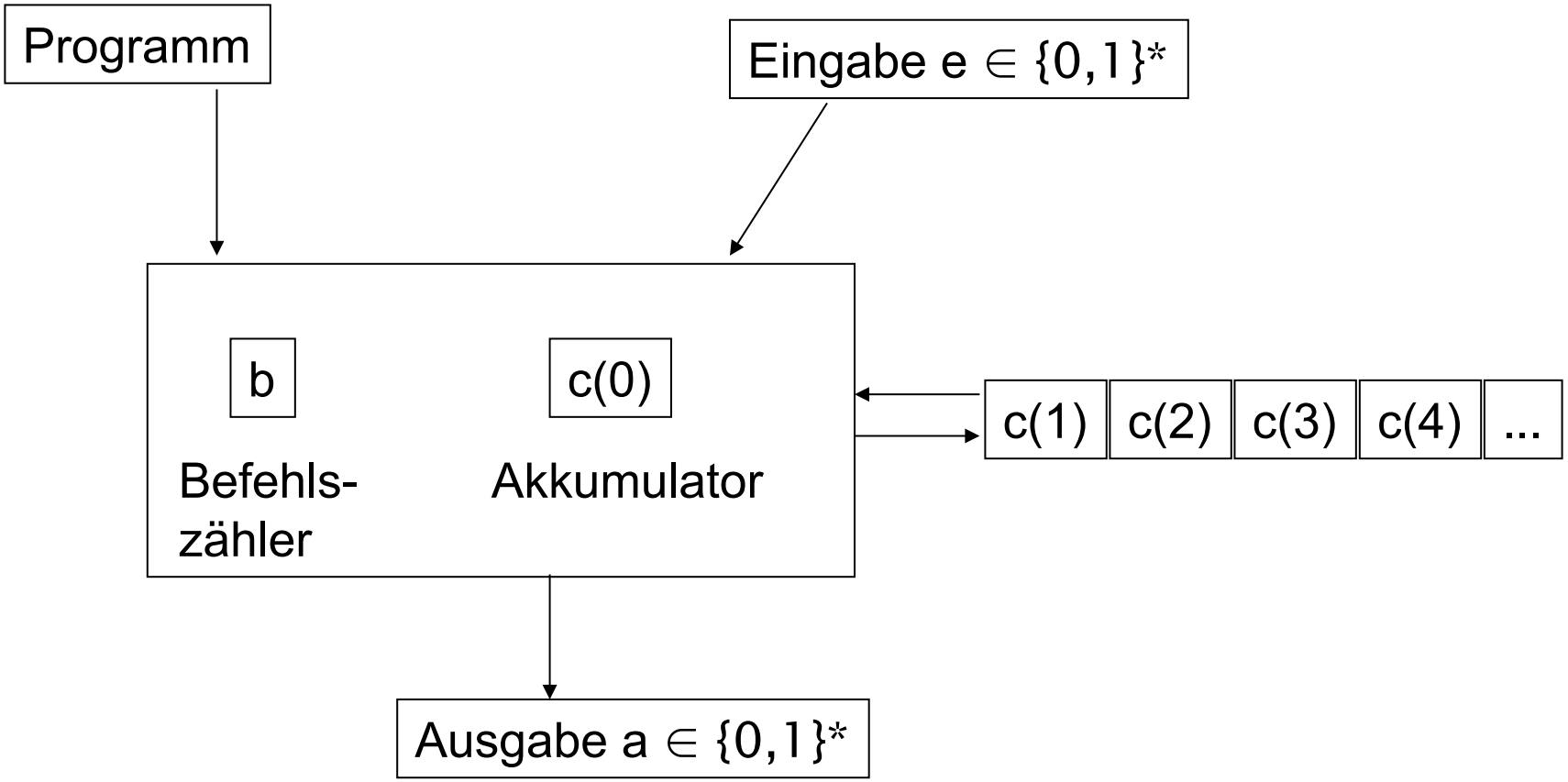
Random Access Maschinen

TECHNISCHE UNIVERSITÄT DARMSTADT



Random Access Maschinen

TECHNISCHE UNIVERSITÄT DARMSTADT

Ein/Ausgabe

read	c(0) := head(e); e := e \ head(e); b := b+1; falls |e| > 0
	c(0) := EOF; b := b + 1; sonst
write	a := a c(0); b := b + 1;

Arithmetik

add x	c(0) := c(0) + c(x); b := b + 1;
sub x	c(0) := c(0) – c(x); b := b + 1; falls c(x) < c(0)
	c(0) := 0; b := b + 1; sonst
c *add* x	c(0) := c(0) + x; b := b + 1;
c *sub* x	analog: Operation mit Konstante c

Random Access Maschinen

TECHNISCHE UNIVERSITÄT DARMSTADT

Sprünge

goto j b := j;

if (c(0) R i) *then goto* j; b := $\begin{cases} j & \text{falls } c(0)\ R\ i \\ b+1 & \text{sonst} \end{cases}$, $R \in \{<, >, =, \leq, \geq\}$

end Programm hält

Speicherzugriffe

direkt

load x c(0) := c(x); b := b + 1;

store x c(i) := c(0); b := b + 1;

indirekt

iload x c(0) := c(c(x)); b := b + 1;

istore x c(c(i)) := c(0); b := b + 1;

Sprachbeschreibungen und Maschinen

TECHNISCHE UNIVERSITÄT DARMSTADT

- Die Frage, ob ein $w \in \Sigma^*$ ein Wort aus einer Sprache $L \subseteq \Sigma^*$ ist, kann unterschiedlich schwierig zu lösen sein

  – **Bsp 2.: In einem sehr komplizierten Fall ist sie nicht entscheidbar**:

    - **geg**: Codierung einer Random Access Machine (RAM, das entspricht in etwa einem herkömmlicher Computer mit unendlich viel Speicher), sowie ein $w \in \Sigma^*$
      Frage: Hält die RAM bei Eingabe w?

“nicht entscheidbar” heisst: es gibt keinen Algorithmus, der für alle Instanzen das Problem lösen kann. (faszinierende Nebeneffekte, Busy Beaver)

Turingmaschinen

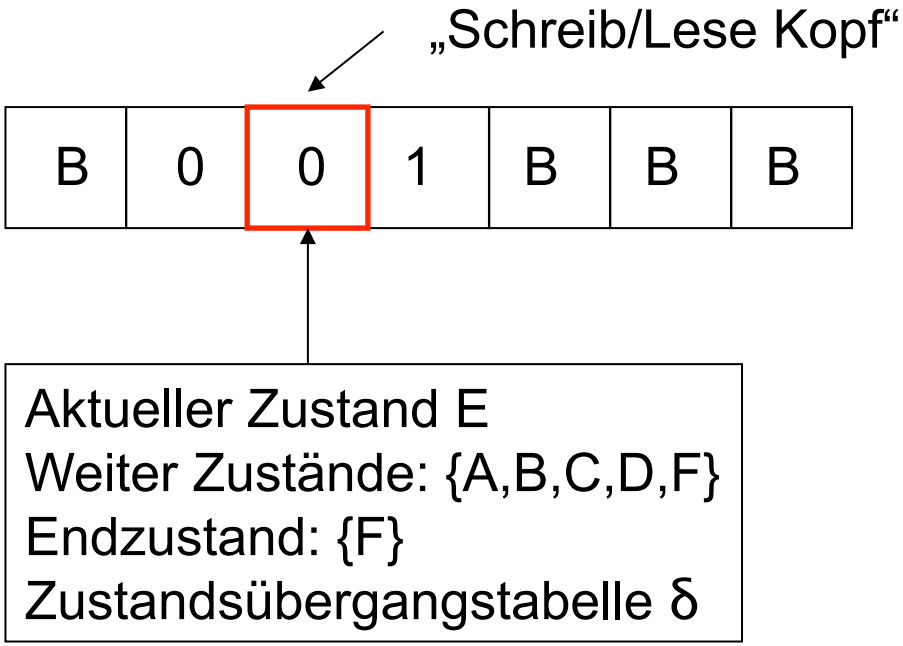
TECHNISCHE UNIVERSITÄT DARMSTADT

- **Formal ist eine (1-Band) Turingmaschine ein 6-Tupel**

- **Def:** Eine (deterministische 1-Band) Turingmaschine ist ein 6-Tupel $(Q,\Sigma,\Gamma,\delta,q_0,F)$, wobei
- Q eine endliche Menge von Zuständen ist,
- Σ ein endliches Alphabet
- $\Gamma := \Sigma \cup \{B\}$, B das so genannte Blank-Symbol
- δ: $Q \times \Gamma \rightarrow Q \times \Gamma \times \{R,N,L\}$ die (partielle) Übergangsfunktion,
- q_0 der Startzustand und
- $F \subseteq Q$ die Menge akzeptierender Endzustände.

Turingmaschinen

TECHNISCHE UNIVERSITÄT DARMSTADT



„Programm“: Falls die Turingmaschine im Zustand q ist und das Zeichen a liest,
dann gehe in den Zustand q‘, überschreibe a durch a‘, und bewege den Kopf nach rechts, links oder gar nicht.

Schreibweise: $\delta(q,a) = (q',a',R)$

Turingmaschinen, Beispiel

TECHNISCHE UNIVERSITÄT DARMSTADT

Geg: #x$y mit x,y $\in \Sigma^*$ **Frage**: Ist x Präfix von y?

$A := (Q, \Sigma, \delta, q_0, F)$,
$Q := \{q_0, q_1, q_2, q_3, q_4, q_5, q_e, q_f\}$,
$\Sigma := \{0,1,\#,\$\}$,
$\Gamma := \Sigma \cup \{B\}$
$F := \{q_f\}$
q_0

δ	0	1	B	#	$
q_0	$(q_0,0,L)$	$(q_0,1,L)$	(q_1,B,R)	$(q_0,\#,L)$	$(q_0,\$,L)$
q_1	$(q_2,\#,R)$	$(q_4,\#,R)$	(q_f,B,N)	$(q_1,\#,R)$	$(q_f,\$,N)$
q_2	$(q_2,0,R)$	$(q_2,1,R)$			$(q_3,\$,R)$
q_3	$(q_0,\$,L)$	$(q_e,1,N)$	(q_e,B,N)	$(q_e,\#,N)$	$(q_3,\$,R)$
q_4	$(q_4,0,R)$	$(q_4,1,R)$			$(q_5,\$,R)$
q_5	$(q_e,1,N)$	$(q_0,\$,L)$	(q_e,B,N)	$(q_e,\#,N)$	$(q_5,\$,R)$
q_e					
q_f					

q_0: Laufe nach links, bis ein B kommt, gehe dann 1 nach rechts
q_1: lauf nach rechts, bis kein # mehr kommt. B oder $ sind gut, gehe dann in q_f.
sonst merke das nächste Zeichen und ersetze es durch #. Gehe nach q_2 oder q_4.
q_2: das gemerkte Zeichen war eine 0. Lauf nach rechts zum ersten $. Gehe dann in q_3.
q_3: lauf durch die $-Zeichen. Wenn am Ende der $ eine 0 steht, ist das ok., gehe nach q_0, sonst gehe nach q_e.
q_4: das gemerkte Zeichen war eine 1. Lauf nach rechts zum ersten $. Gehe Dann in q_5.
q_5: lauf durch die $-Zeichen. Wenn am Ende der $ eine 1 steht, ist das ok., gehe nach q_0, sonst gehe nach q_e.

Turingmaschinen

TECHNISCHE UNIVERSITÄT DARMSTADT

- **Eine Mehrband- Turingmaschine ist ein 6-Tupel**

- **Def:** Eine (deterministische Mehrband) Turingmaschine ist ein 6-Tupel $(Q,\Sigma,\Gamma,\delta,q_0,F)$, wobei
- Q eine endliche Menge von Zuständen ist,
- Σ ein endliches Alphabet
- $\Gamma := \Sigma \cup \{B\}$, B das so genannte Blank-Symbol
- δ: $Q \times \Gamma^k \rightarrow Q \times \Gamma^k \times \{R,N,L\}^k$ die (partielle) Übergangsfunktion,
- q_0 der Startzustand und
- $F \subseteq Q$ die Menge akzeptierender Endzustände.

Nichtdeterministische Turingmaschinen

TECHNISCHE UNIVERSITÄT DARMSTADT

- Eine nichtdeterministische Turingmaschine (NTM) ist definiert, wie eine deterministische Turingmaschine, nur dass δ eine Übergangsrelation und keine Funktion ist.

- $\delta: Q \times \Gamma \rightarrow 2^{Q \times \Gamma \times \{R,N,L\}}$ ist die Übergangsrelation.

- Bsp.: Wenn die TM in Zustand q ist, und ein a liest, und $\delta(q,a) = \{(q', b, R), (q'', a, L)\}$ ist, dann ist die nichtdeterministische TM im nächsten Schritt entweder in Zustand q‘, nachdem sie ein b geschrieben hat, oder sie ist in Zustand q‘‘ nachdem sie ein a schrieb.

- Die Laufzeit einer NTM ist definiert als die Länge des kürzesten Berechnungsweges, der in einem akzeptierenden Endzustand endet.

Theorem 1: RAM und Turingmaschine können sich gegenseitig simulieren.

Theorem 2:

Church-Turing Hypothese:

Die von jeglicher Maschine berechenbaren
Funktionen sind genau die,
die von Turingmaschinen berechenbar sind.

Turingmaschinen

TECHNISCHE UNIVERSITÄT DARMSTADT

Def.:
Eine Sprache L heißt entscheidbar, wenn es eine Turingmaschine gibt, die zu jeder Eingabe w $\in \Sigma^*$ nach endlicher Zeit anhält, und genau dann in einem akzeptierenden Zustand endet, wenn w $\in$ L gilt.

Eine Sprache L heißt semi-entscheidbar, wenn es eine Turingmaschine gibt, die zu jeder Eingabe w $\in$ L nach endlicher Zeit in einem akzeptierenden Endzustand anhält.

Eine Funktion f heißt berechenbar, wenn es eine Turingmaschine gibt, die für alle Eingaben x, die aus dem Definitionsbereich von f stammen nach endlich vielen Schritten anhält und f(x) auf das Band schreibt.

Unendscheidbarkeit

TECHNISCHE UNIVERSITÄT DARMSTADT

Gibt es unendscheidbare Sprachen?

Ja, denn es gibt nur abzählbar unendlich viele Turingmaschinen, aber überabzählbar viele Sprachen $L \subseteq \{0,1\}^*$

Begründung mit Hilfe des Cantor‘schen Diagonalisierungsverfahrens:

	M_1	M_2	M_3	M_4	... →
0	n	j	j	n	
1	n	n	n	j	
01	j	j	j	n	
...					
x_i ↓					

Eintrag (M_i,x_k)=„j“ bedeutet, dass x_k aus der Sprache $L(M_i)$ ist. Sei nun L die Sprache, die genau aus den Wörtern besteht, bei denen beim Eintrag (M_i,x_i) „n“ steht. L gehört zu keiner der aufgeführten TMs.

Berechenbarkeit

TECHNISCHE UNIVERSITÄT DARMSTADT

Gibt es Funktionen, die nicht von einer Turingmaschine berechnet werden können?
Ja.

Die Busy-Beaver Funktion $\Sigma(n)$ ist definiert als die Anzahl der Einsen, die eine Champion-Turingmaschine auf ein zu Beginn leeres Band ausgibt, wobei n die Anzahl der erlaubten Zustände darstellt. Die TM muss irgendwann halten. Wir gehen weiterhin davon aus, dass diese Einsen alle zusammenhängend sein müssen.

Beweis:

Berechenbarkeit

TECHNISCHE UNIVERSITÄT DARMSTADT

Annahme: Die Busy-Beaver Funktion $\Sigma(n)$ ist berechnbar, und EVALΣ ist die TM, die $\Sigma(n)$ berechnet. Bei einer Eingabe von n Einsen schreibt sie $\Sigma(n)$ Einsen auf das Band und hält dann an.

Im folgenden definieren wir 4 Hilfs-TMs.

Sei INC eine TM, die bis zum ersten B nach rechts läuft, dort eine 1 schreibt und dann hält.

DOUBLE sein eine andere TM, die die Anzahl Einsen, die sich auf dem Band befinden verdoppelt. DOUBLE berechnet also zu Eingabe n n+n.

Wir bilden nun eine neue TM: DOUBLE | EVALΣ | INC
Die Anzahl der Zustände dieser Maschine sei n_0

Berechenbarkeit

TECHNISCHE UNIVERSITÄT DARMSTADT

Sei CREATE_n_0 eine weitere TM, welche n_0 Einsen auf ein leeres Band schreibt. Diese TM gibt es, trivialerweise eine mit n_0 vielen Zuständen.

Sei nun $N := n_0 + n_0$

Das Finale: Sei BADΣ folgende TM:

$$\underbrace{\text{CREATE\_}n_0}_{n_0} \mid \underbrace{\text{DOUBLE} \mid \text{EVAL}\Sigma(N) \mid \text{INC}}_{n_0}$$

Diese Maschine hat N Zustände. Sie startet auf leerem Band, schreibt n_0 Einsen, verdoppelt diese, berechnet $\Sigma(N)$ und schreibt eine weitere 1.

BADΣ hat also eine 1 mehr als $\Sigma(N)$ geschrieben! Es folgt, dass die Annahme falsch war.

Busy Beaver

TECHNISCHE UNIVERSITÄT DARMSTADT

Interessanterweise sind einige Busy-Beaverwerte bekannt. Z.B. für TMs mit 2 Symbolen :

#Zustände	Anzahl Einsen des Siegers
1	1
2	4
3	6
4	13
5	>= 4098
6	>= 95.524.079

Damit beenden wir den Ausflug in die Turingmaschinen und gehen Wieder zur RAM über.

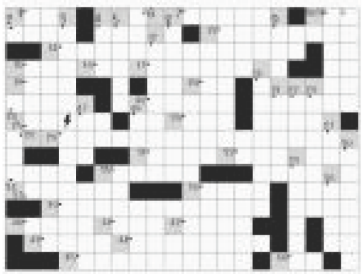
Probleme des täglichen Lebens

TECHNISCHE UNIVERSITÄT DARMSTADT

Im folgenden sind die Probleme lösbar. Die Frage ist nur in welcher Zeit und mit wieviel Speicherplatz.

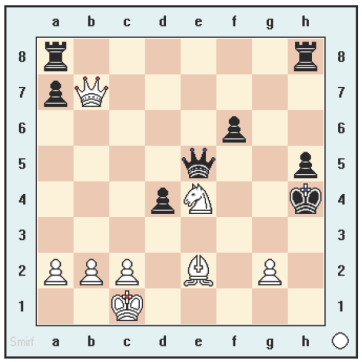
- Was ist schwieriger?

??



– Kopfrechnen

– Kreuzworträtsel



– Schach

– Sokoban

– Puzzle

Probleme und Problembeschreibungen, Wdh

TECHNISCHE UNIVERSITÄT DARMSTADT

- Eine Sprache $L \subseteq \Sigma^*$ muss nun irgendwie beschrieben werden.
  - z.B. durch einen *regulären Ausdruck*: (0*10*)
    - $\varnothing$ ist ein regulärer Ausdruck.
    - ε ist ein regulärer Ausdruck.
    - $\forall\ a_i \in \Sigma$ ist a_i ein regulärer Ausdruck.
    - Sind *x* und *y* reguläre Ausdrücke, so auch $x \cup y$, (*xy*) und *x* *.
    - Es gibt keine weiteren regulären Ausdrücke.

  - z.B. durch eine **Problembeschreibung**:
    - **Definition**: Ein *Entscheidungsproblem* ist ein input-output Tupel mit

      geg.: Kodierung eines Inputs einer Instanz, mittels Alphabet $\sum$

      ges.: ja/nein
    - Die Teilmenge aller Inputs, für die die Antwort “ja” ist, ist offenbar eine Sprache

Algorithmus und Rechenmodell für RAM

TECHNISCHE UNIVERSITÄT DARMSTADT

Zusätzliche Unterscheidung: unit-cost vs. log-cost Modell

***Unit-cost Modell**: jeder Befehl der RAM wird in einem Schritt abgearbeitet*

Typischer Befehlssatz:
+, -, *, /, vergleichen, löschen, schreiben und lesen von rationalen Zahlen, Programmfluß mittels if ... else Verzweigung, Schleifen
Dieses Modell werden wir vorwiegend benutzen.

***Log-cost Modell**: jeder Befehl benötigt Θ(k) Zeit, wobei k die Anzahl der Bits der Operanden ist.*

Typischer Befehlssatz:
laden, speichern, goto, branch on zero, addiere, subtrahiere, bitweises und, bitweises oder, bit-Komplement
Dieses Modell ist realistischer und wird u.a. in der Optimierung relevant. Z.B. bei der so genannten Ellipsoidmethode zur Lösung linearer Programme

Algorithmus und Rechenmodell

TECHNISCHE UNIVERSITÄT DARMSTADT

Effizienzmaße (Algorithmus A): worst-case, average-case, best-case

$T_A(x)$ = Anzahl Befehle, die *A* bei Eingabe *x* ausführt.
$S_A(x)$ = größte Adresse im Speicher, die *A* bei Eingabe *x* benutzt.

- **Worst Case Laufzeit:** $T_A^{wc}(n) := max\ \{T_A(x) \mid <x> \leq n\}$

- **Average Case Laufzeit:** $T_A^{ac}(n) := \sum_{\{x \mid <x> = n\}} p_x\ T_A(x)$,
  erfordert die Kenntnis von Auftrittswahrscheinlichkeiten, bzw.
  Annahme von Gleichverteilung

- **Best Case Laufzeit:** $T_A^{bc}(n) := min\ \{T_A(x) \mid <x> \leq n\}$

- **Platzbedarf**: $S_A^{wc}(n) := max\ \{S_A(x) \mid <x> \leq n\}$

Ein Zeit-Komplexitätsmaß

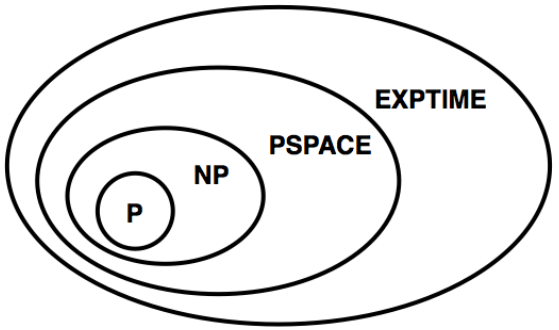
TECHNISCHE UNIVERSITÄT DARMSTADT

- Definition: Komplexität eines Algorithmus
  - Sei A ein deterministischer (RAM-)Algorithmus, der auf allen Eingaben hält.
  - Die **Laufzeit (Zeitkomplexität)** von A ist eine Funktion f_A: N→N,
    - wobei $f_A(n)$ die maximale Anzahl von Schritten von A beschreibt, **über alle Eingaben der Länge n**.
    - Linear-Zeit-Algorithmus: $f_A(n) \leq c\ n$ für eine Konstante c
    - Polynom-Zeit-Algorithmus: $f_A(n) \leq c\ n^k$ für Konstanten c und k
- Definition: Komplexität eines Problems
  - Die Zeit- (Platz-) Komplexität eines Problems p ist die Laufzeit des schnellsten (am wenigsten Speicherplatz benötigenden) Algorithmus, der Problem p löst.
  - Ein Problem p ist "in Polynomzeit lösbar", wenn es Algorithmus A, Polynom Π und $n_0 \in \mathbb{N}$ gibt, so dass für alle $n > n_0$ **gilt : $f_A(n) <= \Pi(n)$**

P, NP, PSPACE

TECHNISCHE UNIVERSITÄT DARMSTADT

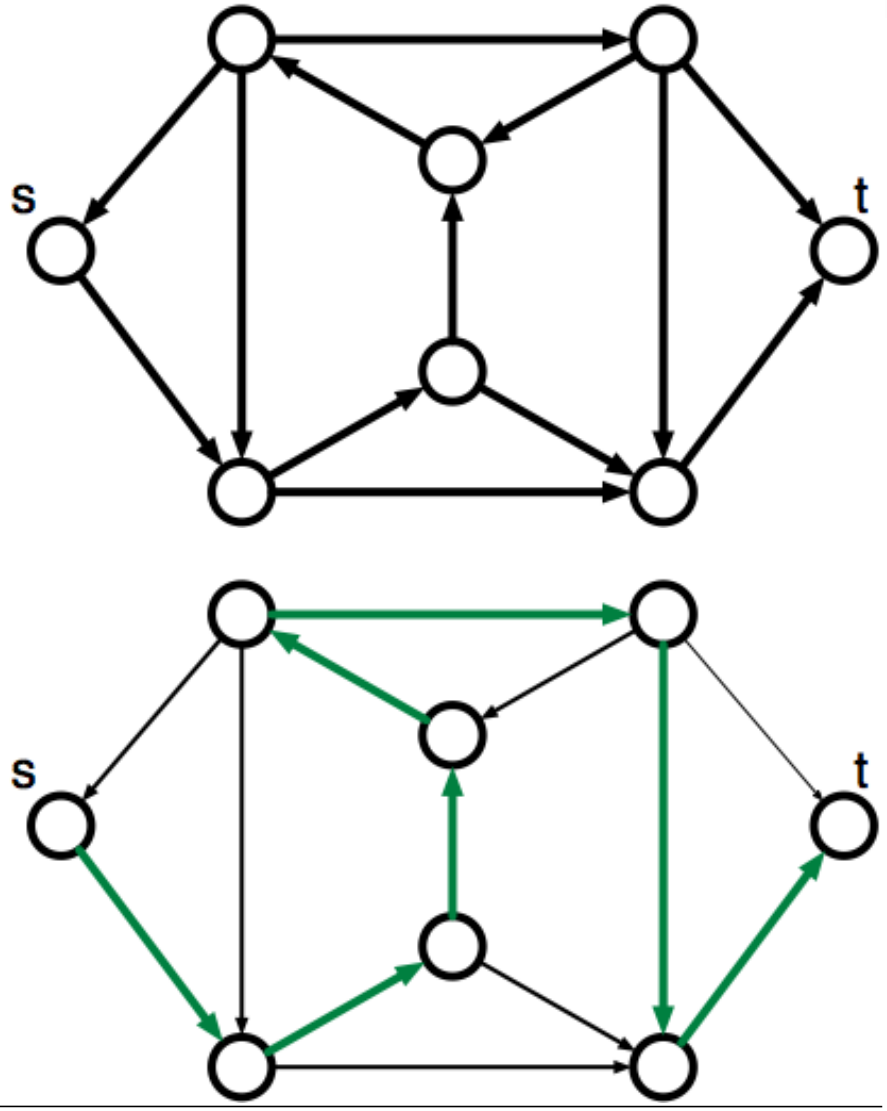
- **P:** Klasse aller Probleme, die von einer deterministischen RAM in Polynomzeit gelöst werden können
- **NP:** Klasse aller Probleme, die von einer nichtdeterministischen TM in Polynomzeit gelöst werden können.
- **PSPACE :** Klasse aller Probleme, die von einer deterministischen RAM mit polynomiell viel Platz gelöst werden können
- Man weiß nur, dass P ≠ EXPTIME und $\mathbf{P} \subseteq \mathbf{NP} \subseteq \mathbf{PSPACE} \subseteq \mathbf{EXPTIME}$
- EXPTIME = $\bigcup_k \mathbf{TIME}(2^{n^k})$
- Allgemein wird aber vermutet, dass alle Inklusionen echt sind, d.h.



Beispiele

TECHNISCHE UNIVERSITÄT DARMSTADT

- **Definition: *HAMPATH***
  - Das Hamiltonsche Pfadproblem
    - Geg.:
      - ein gerichteter Graph
      - Zwei Knoten s,t
    - Ges.: existiert ein Hamitonscher Pfad von s nach t
      - d.h. ein gerichteter Pfad, der alle Knoten besucht, aber keine Kante zweimal benutzt
- **Algorithmus für Hamiltonscher Pfad:**
  - Rate eine Permutation $(s,v_1,v_2,...,v_{n-2},t)$
  - Teste, ob Permutation ein Pfad ist
    - falls ja, akzeptiere
    - falls nein, verwerfe
- **Also: HamPath $\in$ NP**



Das SAT Problem

TECHNISCHE UNIVERSITÄT DARMSTADT

- Eine Boolesche Funktion $f(x_1,x_2,..,x_n)$ ist erfüllbar, wenn es eine Wertebelegung für $x_1,x_2,..,x_n$ gibt, so dass $f(x_1,x_2,..,x_n) = 1$
  - $(x \vee y) \wedge (z \vee \neg x \vee \neg y) \wedge (x \vee \neg z)$ ist erfüllbar, da
    - die Belegung $x = 1, y = 0, z = 0$
  - $(1 \vee 0) \wedge (0 \vee 0 \vee 1) \wedge (1 \vee 1) = 1 \wedge 1 \wedge 1 = 1$ liefert.
- Definition (SAT Problem, die Mutter aller NPc Probleme)
  - **Gegeben**:
    - Boolesche Funktion ϕ
  - **Gesucht**:
    - Gibt es $x_1,x_2,..,x_n$ so dass $\phi(x_1,x_2,..,x_n) = 1$
- SAT ist in NP. Man vermutet, dass SAT nicht in P ist.

Das QSAT Problem

TECHNISCHE UNIVERSITÄT DARMSTADT

- Eine quantifizierte Boolesche Formel (QBF) besteht aus
  - Einer Folge von Quantoren $\exists x$, $\forall y$ mit daran gebundenen Variablen; obdA seien genau alle x_i mit ungeradem i existenzquantifiziert
  - Einer Booelschen Funktion $F(x_1,x_2,...,x_m)$
  - Jede Variable der Funktion ist genau einmal an einem Quantor gebunden
- Die quantifizierte Boolesche Formel ist erfüllbar falls
  - Im Falle eines Existenzquantors: $\exists x\ F(x) \Leftrightarrow F(0) \vee F(1)$
  - Im Falle eines Allquantors: $\forall x\ F(x) \Leftrightarrow F(0) \wedge F(1)$
- Definition (QSAT Problem, die Mutter aller PSPACEc Probleme)
  - **Gegeben**: Quantifizierte Boolesche Funktion ϕ
  - **Frage**: Gibt es x_1, so dass es für alle x_2 ein x_3 gibt, so dass ... so dass $\phi(x_1,x_2,..,x_n) = 1$
- QSAT ist in PSPACE. Man vermutet, dass QSAT nicht in NP ist. QSAT ist PSPACE-schwer.

Beispiele:

TECHNISCHE UNIVERSITÄT DARMSTADT

- $\exists x\ \forall y\ (x \wedge y) \vee (\neg x \wedge \neg y)$

  $= (\forall y\ (0 \wedge y) \vee (\neg 0 \wedge \neg y)) \quad \vee (\forall y\ (1 \wedge y) \vee (\neg 1 \wedge \neg y))$

  $= (\forall y: \neg y) \quad \vee (\forall y: y)$

  $= (\neg 0 \wedge \neg 1) \quad \vee (0 \wedge 1)$

  $= 0 \vee 0$

  $= 0$

- $\forall y\ \exists x\ (x \wedge y) \vee (\neg x \wedge \neg y)$

  $= (\exists x: (x \wedge 0) \vee (\neg x \wedge \neg 0)) \wedge (\exists x: (x \wedge 1) \vee (\neg x \wedge \neg 1))$

  $= (\exists x: : \neg x) \quad \wedge (\exists x: x)$

  $= (\neg 0 \vee \neg 1) \quad \wedge (0 \vee 1)$

  $= 1 \wedge 1$

  $= 1$