Informatik III

4.4 Platzkomplexitätsklassen

Christian Schindelhauer

Albert-Ludwigs-Universität Freiburg
Institut für Informatik
Rechnernetze und Telematik
Wintersemester 2007/08

Komplexitätstheorie

Platzklassen

2

Komplexitätstheorie - Platzklassen

- **Platzkomplexität**
  - Definition
  - Simulation mehrerer Bänder
  - Savitchs Theorem
- **PSPACE**
  - PSPACE-schwierig
  - Das quantifizierte Boolesche Erfüllbarkeitsproblem
  - Gewinnstrategien für Spiele
- **Chomsky-Hierachien**
  - Lineare platzbeschränkte TM
  - Das Wortproblem linear platzbeschränkter TMs

Ein Platzkomplexitätsmaß

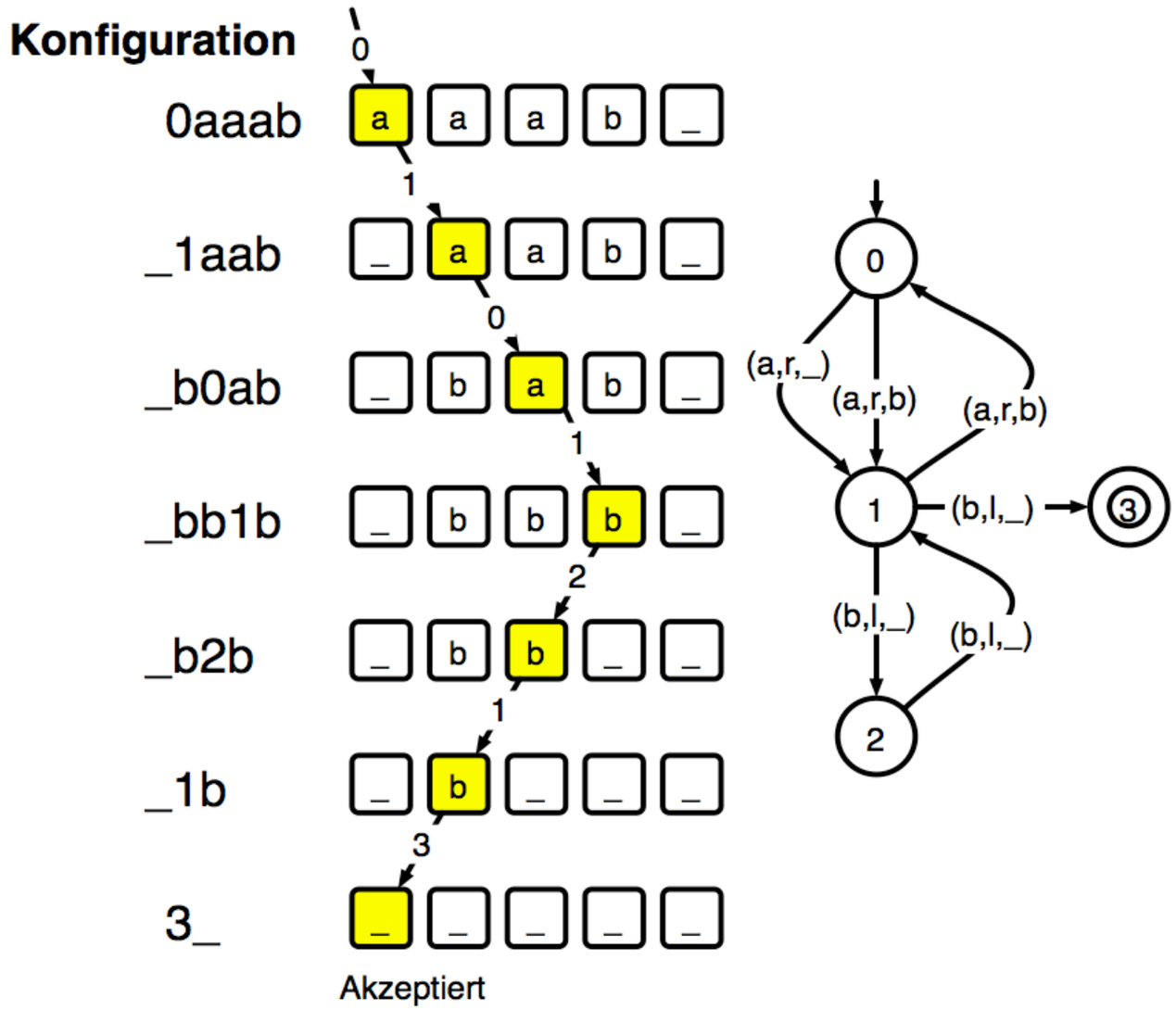
- **Definition**
  - Sei M eine deterministische Turing-Maschine, die auf allen Eingaben hält.
  - Der (Speicher-) **Platzbedarf (Platzkomplexität)** von M ist eine Funktion f: N → N,
    - wobei f(n) die maximale Anzahl von Bandzellen von M ist, die M verwendet (ein Kopf der TM zeigt auf eine Bandzelle)
  - Falls f(n) der Platzverbrauch einer TM M ist, nennt man M auch eine **f(n)-Platz-Turing-Maschine**
    - z.B. Linear-Platz-TM für f(n) = c n für eine Konstante c
    - z.B. Polynom-Platz-TM für f(n) = c n^k für Konstanten c und k

- Zumeist beschreibt n die Eingabelänge.

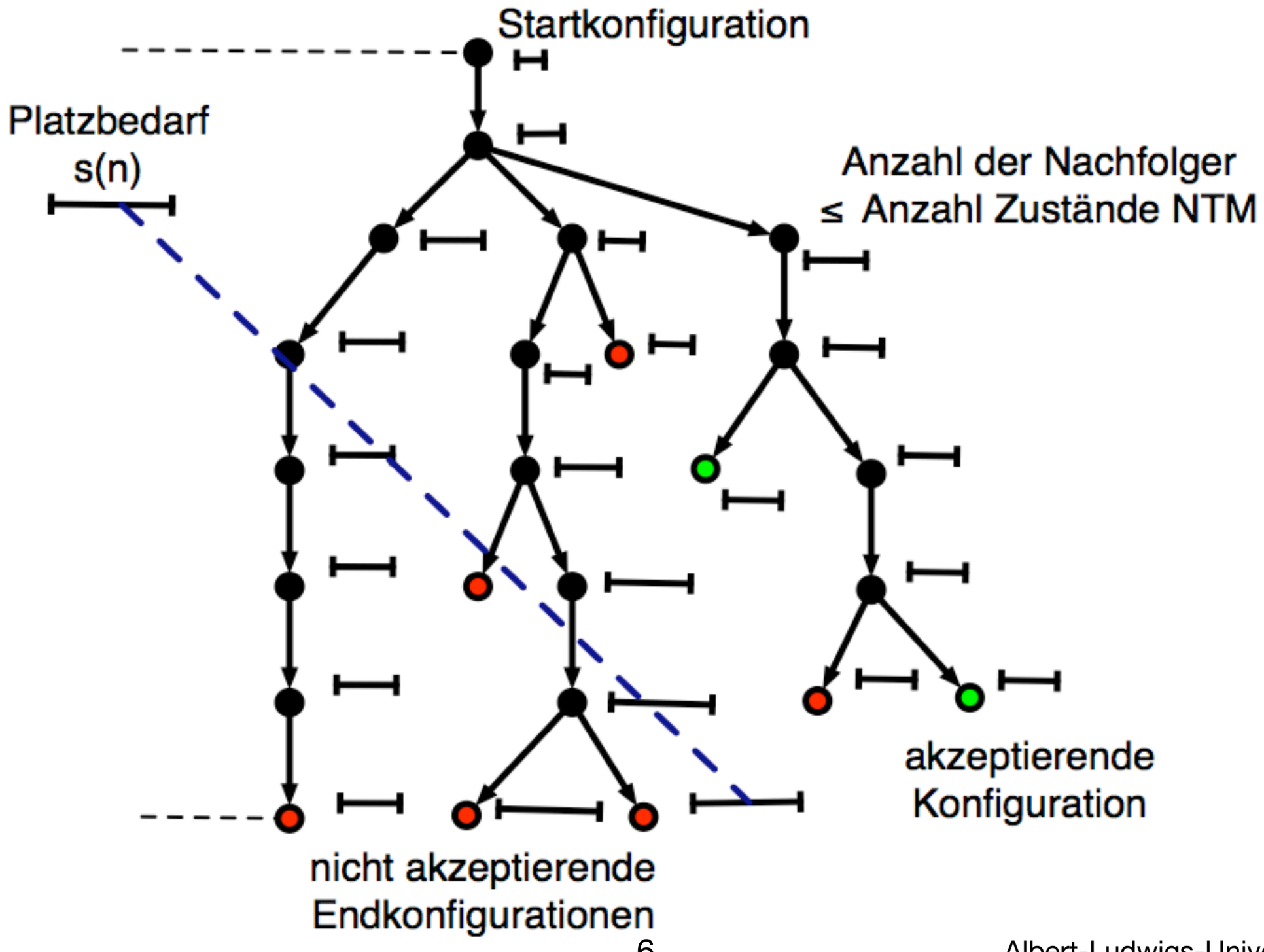
Konfiguration einer DTM/NTM

- **Momentaufnahme einer TM**
  - Bei Bandinschrift uv
    - dabei beginnt u am linken Ende des Bandes und
    - hinter v stehen nur Blanks,
  - Zustand q,
  - Kopf auf erstem Zeichen von v
- **Konfiguration C = uqv**

- **Bei Mehrband-TMs entsprechende Beschreibung aller Bänder**



Nichtdeterministische Platzkomplexitätsklassen



Platzkomplexitätsklassen

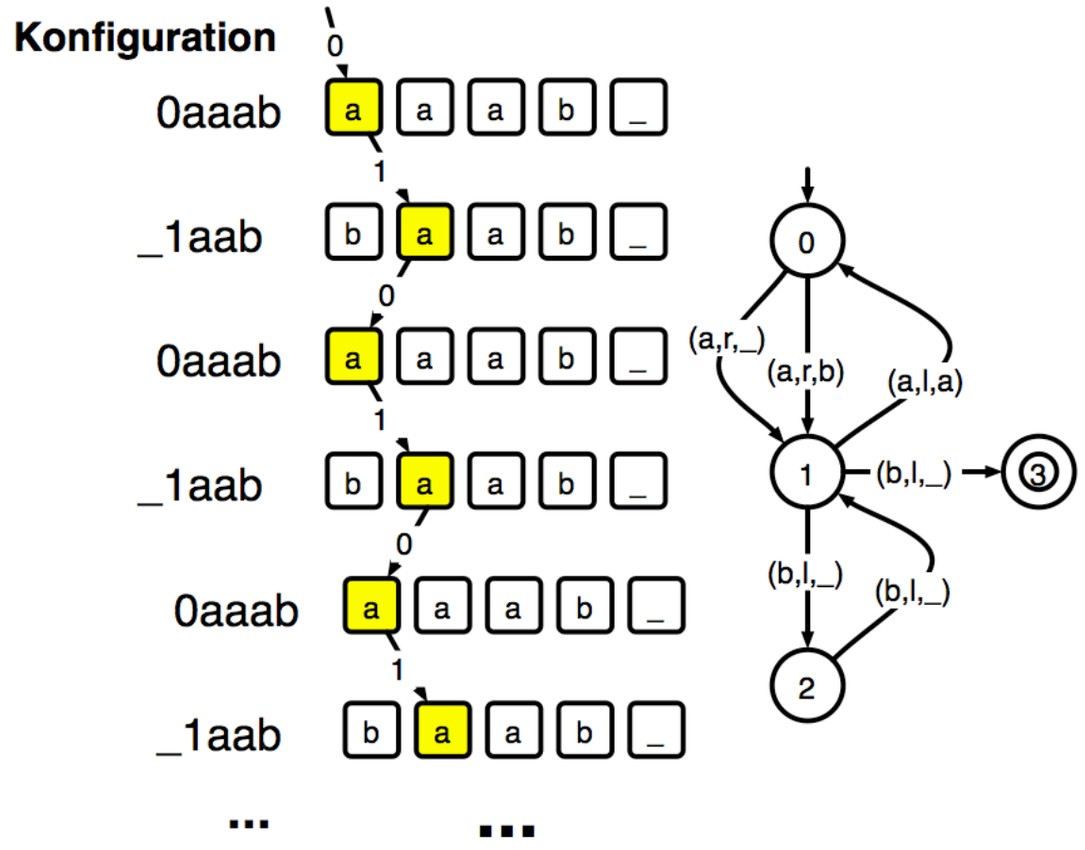
- **Definition**
  - Sei f:$\mathbb{N} \to \mathbb{N}$ eine Funktion. Die Platzkomplexitätsklasse **SPACE(f(n))** und **NSPACE(f(n))** sind wie folgt definiert
  - **SPACE(f(n))** = { L | L ist eine Sprache, die durch eine O(f(n))-Platz-**DTM** entscheiden wird }
  - **NSPACE(f(n))** = { L | L ist eine Sprache, die durch eine O(f(n))-Platz-**NTM** entscheiden wird }
  - Wird die Anzahl der Bänder auf k beschränkt, schreiben wir
    - **SPACE**$_{\text{k-Band}}$(f(n)) oder einfach **SPACE$_k$**(f(n)),
    - **NSPACE**$_{\text{k-Band}}$(f(n)) oder einfach **NSPACE$_k$**(f(n)).

Beispiel: SAT ist in $SPACE_2(N)$

- **Betrachte folgende 2-Band-DTM:**
  - "Gegeben eine boolesche Formel F mit m Variablen x_1, ..., x_m
  - Für alle Belegungen von z_1, ..., $z_m \in \{0,1\}^m$
    - Setze Belegung z_1, ..., z_m in F ein
      - Ist $F(z_1, ..., z_m)$ wahr, dann akzeptiere und halte
  - Sonst verwerfe"
- **Laufzeitanalyse:**
  - Auswerten der Funktion: O(n)
  - Anzahl verschiedener Belegungen ($m \leq n$): 2n
  - Insgesamt: $O(n\ 2^n)$
- **Platzbedarf:**
  - Auswerten der Funktion: O(n)
  - Speichern der Belegung: O(n)
  - Insgesamt: O(n)

Maximale Berechnungszeit einer s(n)-Platz-DTM/NTM

- **Lemma**
  - Jede s(n)-Platz-DTM hat eine Laufzeit von $2^{O(s(n))}$.
- **Beweis:**
  - Es gibt höchstens $2^{O(s(n))}$ verschiedene Konfigurationen der DTM
  - Wiederholt sich eine Konfiguration, so wiederholt sich diese immer wieder und die DTM hält niemals
- **Lemma**
  - Jede s(n)-Platz-NTM hat eine Laufzeit von $2^{O(s(n))}$.
- **Beweis**
  - Analog zur DTM:
  - Wenn sich auf einem Berechnungspfad eine Konfiguration wiederholt, dann wird sie sich immer wieder und wieder und wieder wiederholen.

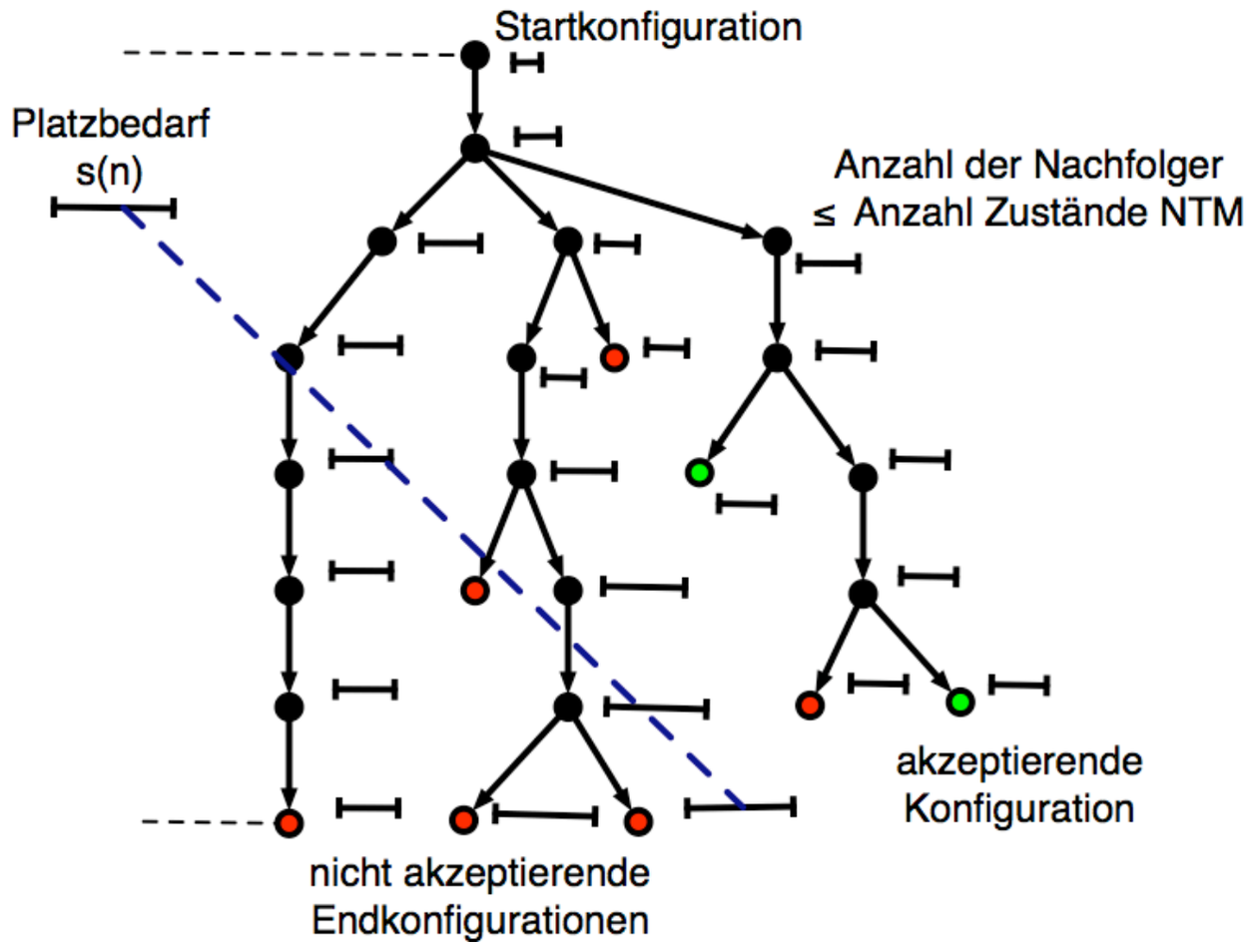


Komplexitätstheorie

Der Satz von Savitch

Nicht der Satz von Savitch

- **Schwacher-Satz: Für s(n)≥n**
  - $NSPACE_1(s(n)) \subseteq SPACE_3(2^{O(s(n))})$, d.h.
  - Jede Berechnung einer s(n)-Platz 1-Band-NTM kann von einer 3-Band DTM in Platz $2^{O(s(n))}$ durchgeführt werden.
- **Beweis:**
  - Simuliere alle Berechnungspfade der NTM durch
  - Dadurch wird die Berechnung deterministisch
- **Warum ist die Platzschranke so schlecht?**
  - Der Baum kann exponentiell tief sein: $2^{O(s(n))}$
  - Dann braucht man allein zum Abspeichern, welchen Ast man gerade berechnet, ein exponentiell langes Wort



Der Satz von Savitch

- **Theorem: Für jede Funktion s(n) ≥ n**
  - **$NSPACE_1(s(n)) \subseteq SPACE_3(s^2(n))$, d.h.**
    - Jede Berechnung einer t-Zeit 1-Band-NTM kann von einer 3-Band DTM in Platz **$s^2(n)$** durchgeführt werden.
- **Beweis:**
  - Betrachte NTM für $L \in NSPACE_1(s(n))$, die mit einer **eindeutigen Konfiguration C_{akz}** akzeptiert, d.h.
    - Es gibt nur einen akzeptierenden Zustand
    - Am Ende der Berechnung wird das Band gelöscht
  - Betrachte das Prädikat **`Erreicht-Konf`(C,C',S,T):**
    - Dieses Prädikat ist wahr, wenn die S-Platz-NTM M ausgehend von der Konfiguration C die Konfiguration C' innerhalb von T Schritten erreicht.
  - **Lemma: Erreicht-Konf(C,C',S,T) kann von einer 2-Band-(S log T)-Platz-DTM entschieden werden**
    - noch zu beweisen
  - Nun ist $T \leq 2^{O(s(n))}$ und damit ist $\log T = O(s(n)) \leq c\ s(n)$ für eine Konstante $c>0$.
    - Sei C_{start} die Startkonfiguration
    - Das Prädikat `Erreicht-Konf`(C_{start}, C_{akz}, s(n), $2^{c\ s(n)}$) entscheidet L.
  - Dann kann eine 3-Band-DTM L in Platz $c\ s(n)\ s(n) = c\ s(n)^2 = O(s^2(n))$ die Sprache L entscheiden

Beweis des Lemmas

- **Lemma: Das Prädikat Erreicht-Konf(C,C',S,T) kann eine DTM mit 2 Bändern mit Platzbedarf s(n) log T entscheiden.**
  - Diese Prädikat Erreicht-Konf(C,C',S,T) ist wahr, wenn die S-Platz-NTM M ausgehend von der Konfiguration C die Konfiguration C' innerhalb von T Schritten erreicht.
- **Beweis**
  - Betrachte folgende DTM M' auf Eingabe (C,C',S,T)
    - Falls T=0 dann
      - Akzeptiere falls C=C' und verwerfe falls C≠C'
    - Falls T=1 dann
      - Akzeptiere falls C' eine erlaubte Nachfolgekonfiguration von C ist oder C=C', ansonsten verwerfe
    - Falls T>1 dann
      - Für alle Konfigurationen Z der Länge S
        - Berechne rekursiv r_1 = Erreicht-Konf(C,Z,S, $\lfloor T/2 \rfloor$)
        - Berechne rekursiv r_2 = Erreicht-Konf(Z,C',S, $\lceil T/2 \rceil$)
        - Falls r_1 und r_2 gilt, dann halte und akzeptiere
      - Verwerfe

- **Analyse Platzverbrauch**
- **Platzverbrauch = Eingabelänge = s(n)**
- **Platzverbrauch: zusätzlich s(n)+1 in jeder Rekursionstiefe**
- **Anzahl Rekursionstiefen: log T**

Der Satz von Savitch (Nachschlag)

- **Theorem: Für jede Funktion s(n) ≥ n**
  - $NSPACE_1(s(n)) \subseteq SPACE_3(s^2(n))$, d.h.
- **Beweis:**
  - Zusammenfassung
    - Sei C_{start} die Startkonfiguration
    - Sei C_{akz} die eindeutige akzeptierende Endkonfiguration
    - Das Prädikat Erreicht-Konf(C_{start}, C_{akz}, s(n), $2^{c\ s(n)}$) entscheidet L in Platz $O(s^2(n))$

Komplexitätstheorie

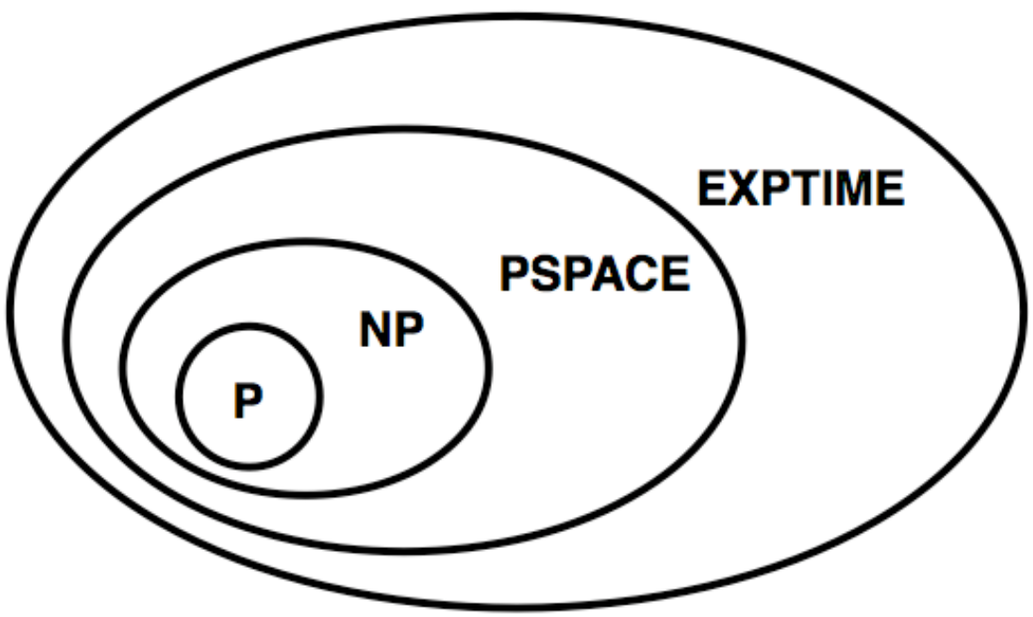
PSPACE

Die Frage P versus NP versus PSPACE

- **P =** **Klasse aller Probleme, die effizient *entschieden* werden können**
- **NP =** **Klasse aller Probleme, die effizient *verifiziert* werden können**
- **PSPACE =** **Klasse aller Probleme, die auf polynomiellen Platz entschieden werden können**
- **EXPTIME =** $\bigcup_k \mathbf{TIME}(2^{n^k})$
- **Man weiß nur, dass P ≠ EXPTIME und**

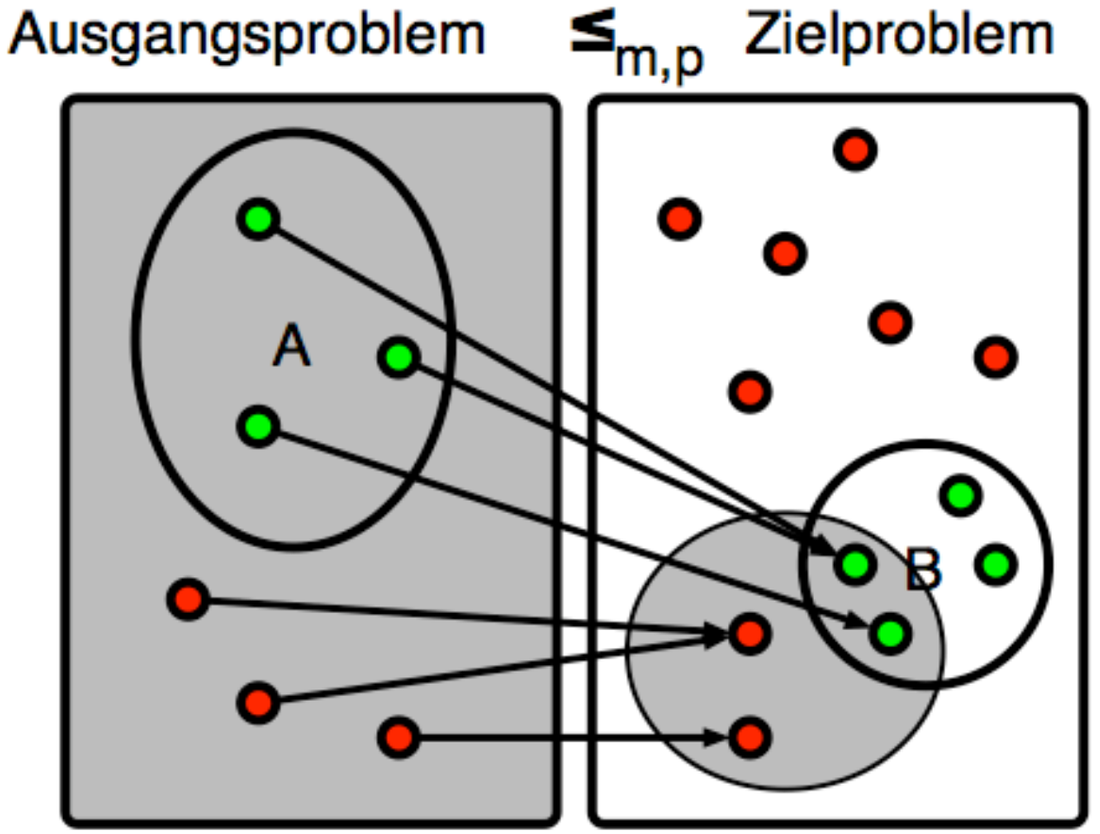
$$\mathbf{P} \subseteq \mathbf{NP} \subseteq \mathbf{PSPACE} \subseteq \mathbf{EXPTIME}$$

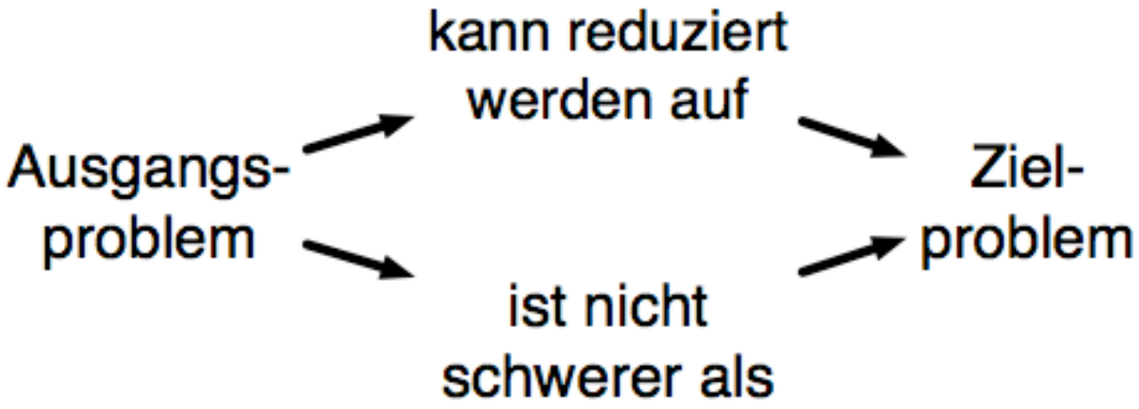
- **Allgemein wird aber vermutet, dass alle Inklusionen echt sind, d.h.**



Die Polynom-Zeit-Abbildungsreduktion

- **Definition (Abbildungsreduktion, Polynomial Time Mapping Reduction, Many-one)**
  - Eine Sprache A kann durch Abbildung auf eine Sprache B in Polynom-Zeit reduziert werden: $A \leq_{m,p} B$,
    - falls es eine in Polynom-Zeit-berechenbare Funktion $f: \Sigma^* \rightarrow \Sigma^*$ gibt,
    - so dass für alle w: $w \in A \Leftrightarrow f(w) \in B$
  - Die Funktion f heißt die Reduktion von A auf B.
- **Theorem**
  - Falls $A \leq_{m,p} B$ und B ist in PSPACE, dann ist A auch in PSPACE.

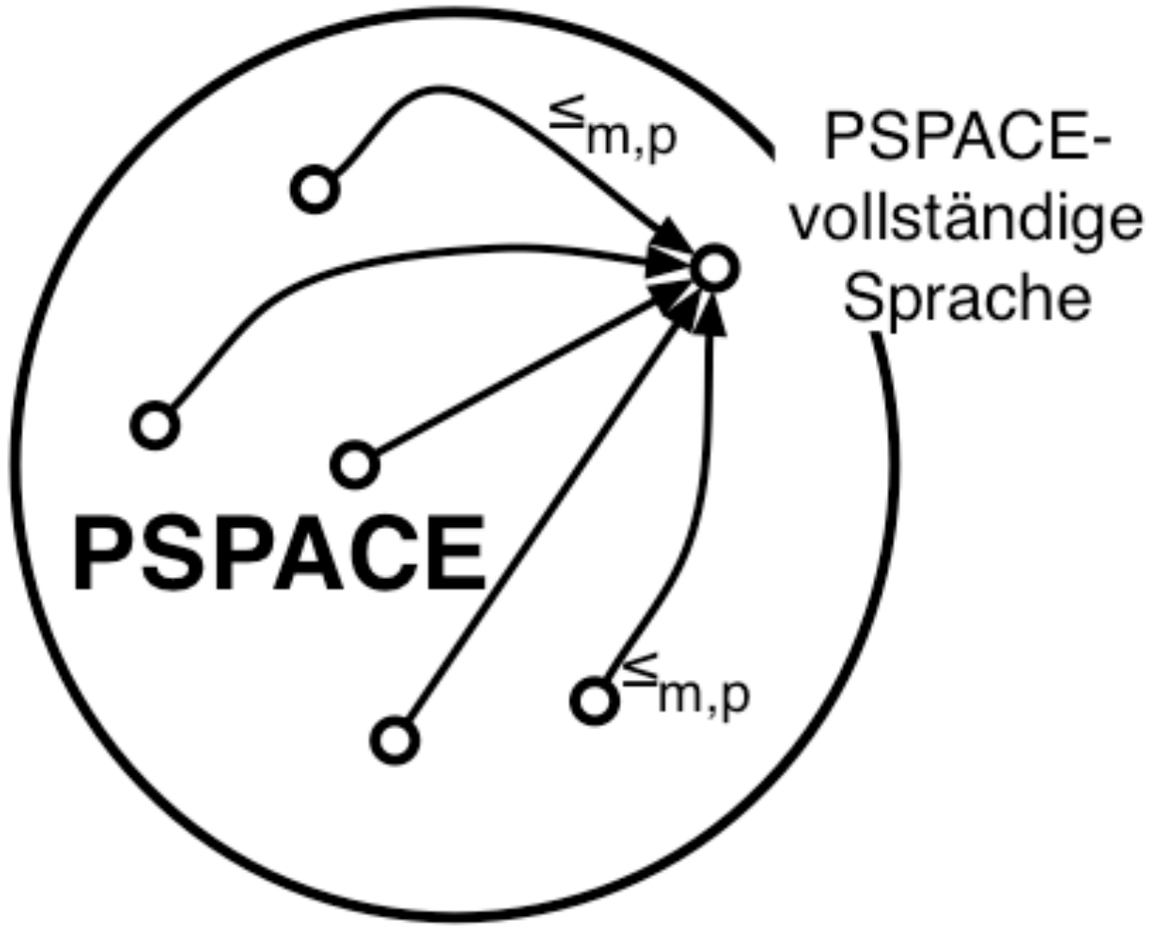




PSPACE-Vollständigkeit

- **Definition:**
  - Eine Sprache S ist **PSPACE-schwierig**, falls
    - für alle L $\in$ PSPACE: L $\leq_{m,p}$ S
  - Eine Sprache S ist **PSPACE-vollständig** wenn:
    - S $\in$ PSPACE
    - S ist PSPACE-schwierig

- **Theorem**
  - QBF ist PSPACE-vollständig



Komplexitätstheorie

QBF

Quantifizierte Boolesche Formeln

- **Eine Boolesche Funktion ist definiert durch**
  - Eine Konstante 0 oder 1
  - Eine Variable, z.B. x,y,z
  - Die Negation einer Booleschen Funktion, z.B. ¬ F(x,y,z)
  - Die Disjunktion zweier Booleschen Funktionen, z.B. F(x,y,z) ∨ G(x,y,z)
  - Die Konjunktion zweier Booleschen Funktionen, z.B. F(x,y,z) ∧ G(x,y,z)
- **Eine quantifizierte Boolesche Formel (QBF) besteht aus**
  - Einer Folge von Quantoren ∃x, ∀y mit daran gebundenen Variablen
  - Einer Booleschen 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: ∃x F(x) ⇔ F(0) ∨ F(1)
    - wobei F eine weitere QBF sein kann
  - Im Falle eines Allquantors: ∀x F(x) ⇔ F(0) ∧ F(1)

Beispiele:

- **Fakt:**
  - $\exists x\ F(x) \Leftrightarrow F(0) \vee F(1), \qquad \forall x\ F(x) \Leftrightarrow F(0) \wedge F(1)$
- **Sind die folgenden Aussagen wahr?**
  - $\exists x\ \forall y\ (x \wedge y) \vee (\neg\ x \wedge \neg\ y)$
  - $\forall y\ \exists x\ (x \wedge y) \vee (\neg\ x \wedge \neg\ y)$

Boolesche Funktionen

- **Definition QBF (Quantified Boolean Formula Problem)**
  - Das Quantifizierte Boolesche Erfüllbarkeitsproblem der Booleschen Funktion ist definiert als:
  - QBF = { φ | φ ist eine wahre quantifizierte Boolesche Formel}
  - Gegeben: Boolesche quantifizierte Formel Q
  - Gesucht: Ist Q wahr?

QBF ist NP-schwierig

- **Spezialfall: SAT**
  - Jedes Boolesche Erfüllbarkeitsproblem ist als QBF darstellbar
- **Theorem**
  - SAT $\leq_{m,p}$ QBF
- **Beweis:**
  - Reduktionsfunktion:
    - gegeben Funktion ϕ
    - Füge für alle Variablen der Funktion Existenzquantoren hinzu
    - Ausgabe: $\exists x_1 \exists x_2 ... \exists x_m \ \phi(x_1,..,x_m)$
  - Korrektheit
    - folgt aus der Äquivalenz der Formeln
  - Laufzeit: Linear
- **Korollar:**
  - QBF ist NP-schwierig

QBF ist in PSPACE

- **Theorem**
  - QBF ist in PSPACE
- **Beweis:**
  - Konstruiere TM
    - gegeben QBF:
      $Q_1x_1\ Q_2x_2 \ldots Q_mx_m\ \phi(x_1,..,x_m)$
      - für $Q_i \in \{\forall,\exists\}$
    - Setze x_1=0: Berechne
      $a= Q_2x_2 \ldots Q_mx_m\ \phi(x_1,..,x_m)$
    - Setze x_1=1: Berechne
      $b= Q_2x_2 \ldots Q_mx_m\ \phi(x_1,..,x_m)$
    - Falls $Q_1 = \forall$
      - Falls a und b gib 1 aus, sonst 0.
    - Falls $Q_1 = \exists$
      - Falls a oder b gib 1 aus, sonst 0.”
  - Platzbedarf:
    - O(1) in jeder Rekursionstiefe
    - Anzahl Rekursionstiefen: m≤n
    - Gesamtspeicher: O(n)

QBF ist PSPACE-schwierig

- **Theorem**
  - Für alle L ∈ PSPACE gilt $L \leq_{m,p}$ QBF
- **Beweis**
  - Betrachte 1-Band s(n)-Platz-TM mit $s(n) = O(n^k)$
  - Dann gibt es eine Boolesche Funktion polynomieller Größe, die wahr ist, falls **`Erreicht-Konf`**(C,C',S,1) für gegebene Eingabelänge und die in Polynom-Zeit beschreibbar ist.
  - **Lemma**
    - **`Erreicht-Konf`**(C,C',S,T) kann von einer quantifizierten Booleschen Funktion der Länge O(S log T) beschrieben werden.
    - Diese Funktion lässt sich von einer DTM in Zeit O(S log T) konstruieren
  - Konstruiere QBF zu **`Erreicht-Konf`**(Anfangskonfiguration, Endkonfiguratoon,s(n),$2^{cs(n)}$)

Wie man `Erreicht-Konf` nicht darstellt

- **Betrachte nun folgende QBF für**
  - Erreicht-Konf(C,C',S, 0) = ("C=C'")
  - Erreicht-Konf(C,C',S, 1) = ("C=C'") $\vee$ ("C geht über in C'")
  - Erreicht-Konf(C,C',S, 2T) = $\exists$ Z: Erreicht-Konf(C,Z,S,T) $\wedge$ Erreicht-Konf(Z,C'S,T)
- **Größe G(T) der Formel für Erreicht-Konf:**
  - G(0) = c S
  - G(1) = c' S
  - G(2T) = 2 G(T) + c"
    - für geeignete Konstanten c,c',c' '$\geq$ 1
  - Beobachtung: G(T) = Ω(S T)
    - Siehe nächste Folie ...
- **Problem:**
  - Rekursiv definierte Formel wächst linear in T
  - T = $2^{c\ s(n)}$ notwendig die Berechnung einer s(n)-Platz-DTM
  - Daher ergibt das keine Polynom-Zeit-Reduktion

Die zugehörige Rekursion

- **Betrachte Rekursion**
  - g(0) = S
  - g(1) = S
  - g(2T) = 2 g(T)
- **Beobachtung: g(t) ≤ G(t)**
  - Beweis durch vollständige Induktion
- **Behauptung: $g(2^k) = S\ 2^k$, für k≥0**
- **Beweis:**
  - Aussage ist korrekt für k=0
  - Angenommen die Aussage ist korrekt für k
  - Dann ist $g(2^{k+1}) = 2\ g(2^k) = 2\ S\ 2^k = S\ 2^{k+1}$
- **Daraus folgt die Beobachtung: G(T) = Ω(S T)**

Wie man `Erreicht-Konf` darstellt

- **Lemma**
  - Erreicht-Konf(C,C’,S,T) kann von einer quantifizierten Booleschen Funktion der Länge O(S log T) beschrieben werden.
  - Diese Funktion lässt sich von einer DTM in Zeit O(S log T) konstruieren Lösung:
- **Beweis: Betrachte nun folgende QBF für**
  - Erreicht-Konf(C,C’,S, 0) = (“C=C’”)
  - Erreicht-Konf(C,C’,S, 1) = (“C=C’”) $\vee$ (“C geht über in C’”)
  - Erreicht-Konf(C,C’,S, 2T) =
    - $\exists$ Z: $\forall$ A,B:
      (“(A,B)=(C,Z)” $\vee$ “(A,B)=(Z,C’)”) $\Rightarrow$
      Erreicht-Konf(A,B,S,T))
  - Behauptung 1: Die QBF ist korrekt
    - Voraussetzung die Rekursion:
    - Erreicht-Konf(C,C’,S, 2T) =
      $\exists$ Z: Erreicht-Konf(C,Z,S,T) $\wedge$
      Erreicht-Konf(Z,C’,S,T)
    - ist korrekt
  - Behauptung 2: Die Länge der QBF ist in O(S log T)
  - Wahl $T = 2^{c\,S}$ reicht für die Berechnung einer S-Platz-DTM
  - Konstruktion läßt sich in von einer Polynom-Zeit-DTM berechnen, da $S \log T = O(S^2)$

Sind die Formeln äquivalent?

∃Z: ∀ A,B:

((A,B)=(C,Z) ∨ (A,B)=(Z,C'))

⇒ `Erreicht-Konf`(A,B,S,T)

Def.: P(A,B)= `Erreicht-Konf`(A,B,S,T)

∀ A,B:

((A,B)=(C,Z) ∨ (A,B)=(Z,C')) ⇒ P(A,B)

$$\bigwedge_{A,B}((A,B)=C,Z) \vee (A,B)=(Z,C')) \Rightarrow P(A,B)$$

3 Fälle:

- (A,B)=(C,Z): Ergebnis: P(C,Z)
- (A,B)=(Z,C'): Ergebnis: P(Z,C')
- Weder noch: Ergebnis: Wahr

Ergibt:

P(C,Z) ∧ P(Z,C')

∃Z: `Erreicht-Konf`(C,Z,S,T) ∧

`Erreicht-Konf`(Z,C',S,T)

und betrachte nur Term hinter ∃ Z

P(C,Z) ∧ P(Z,C')

Die Länge der QBF ist O(S log T)

- **Rekursion:**
  - **`Erreicht-Konf`**(C,C',S, 0) = ("C=C'")
  - **`Erreicht-Konf`**(C,C',S, 1) = ("C=C'") v ("C geht über in C'")
  - **`Erreicht-Konf`**(C,C',S, 2T) = ∃ Z: ∀ A,B: ("(A,B)=(C,Z)" v "(A,B)=(Z,C')") ⇒ **`Erreicht-Konf`**(A,B,S,T))
- **Sei g(T) die Größe:**
  - g(0) = c S
  - g(1) = c S
  - g(2T) = g(T) + c S
    - für geeignete Konstante c≥1
- **Behauptung: $g(T) = g(2^{\log T}) \leq c\,(1+\log T)\,S$, für T≥0**
  - Korrekt für T=1
  - Induktionsannahme: Behauptung korrekt für T
  - Induktionsschluss:
    $g(2T) = g(2^{1+\log T})$
    $= g(T) + c\,S$
    $= c\,(1+\log T)\,S + c\,S$
    $= c\,(2+\log T)\,S = c\,(1+\log 2T)\,S$

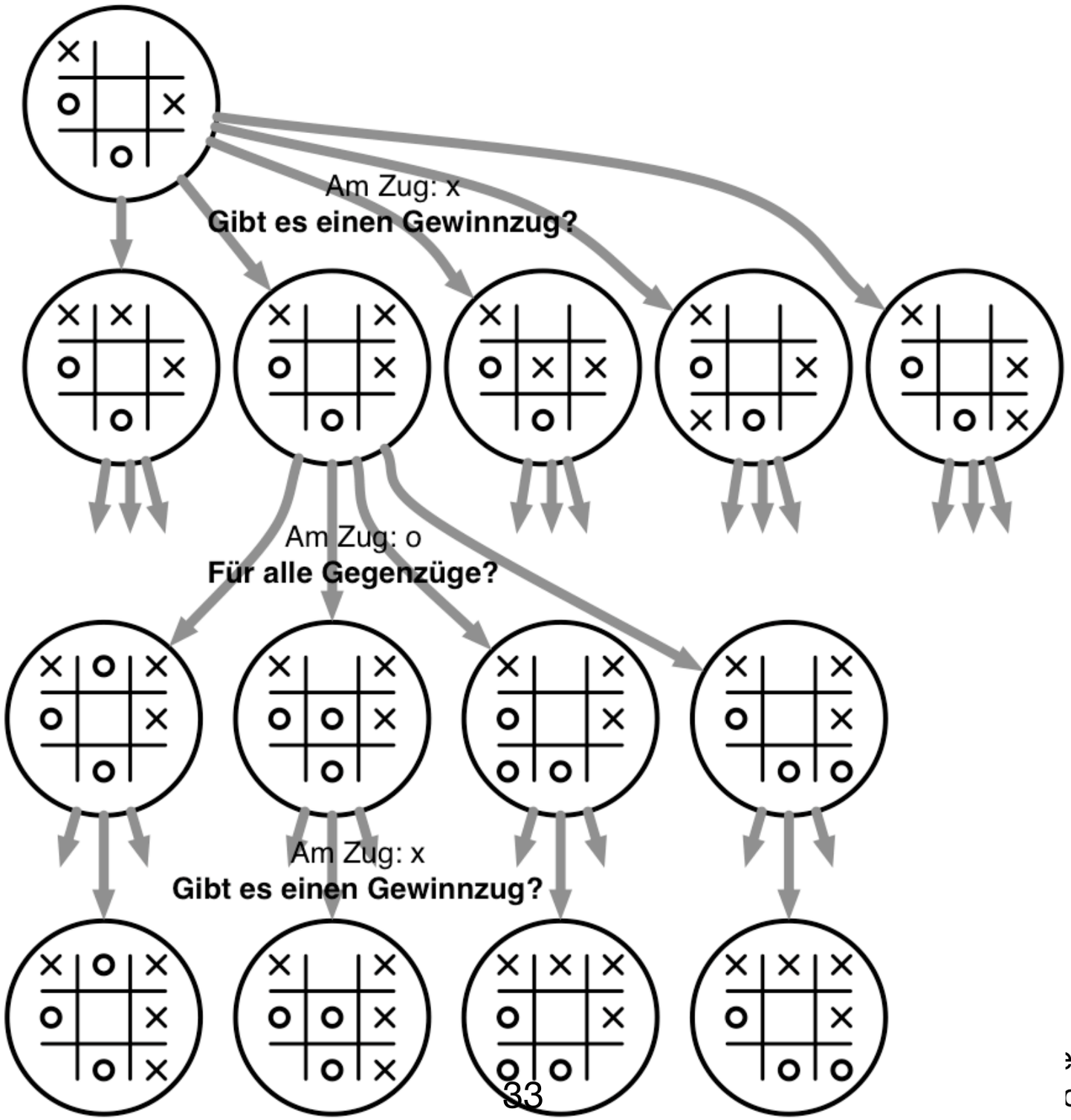
Komplexitätstheorie

Die Komplexität von Spielen

PSPACE und Spiele

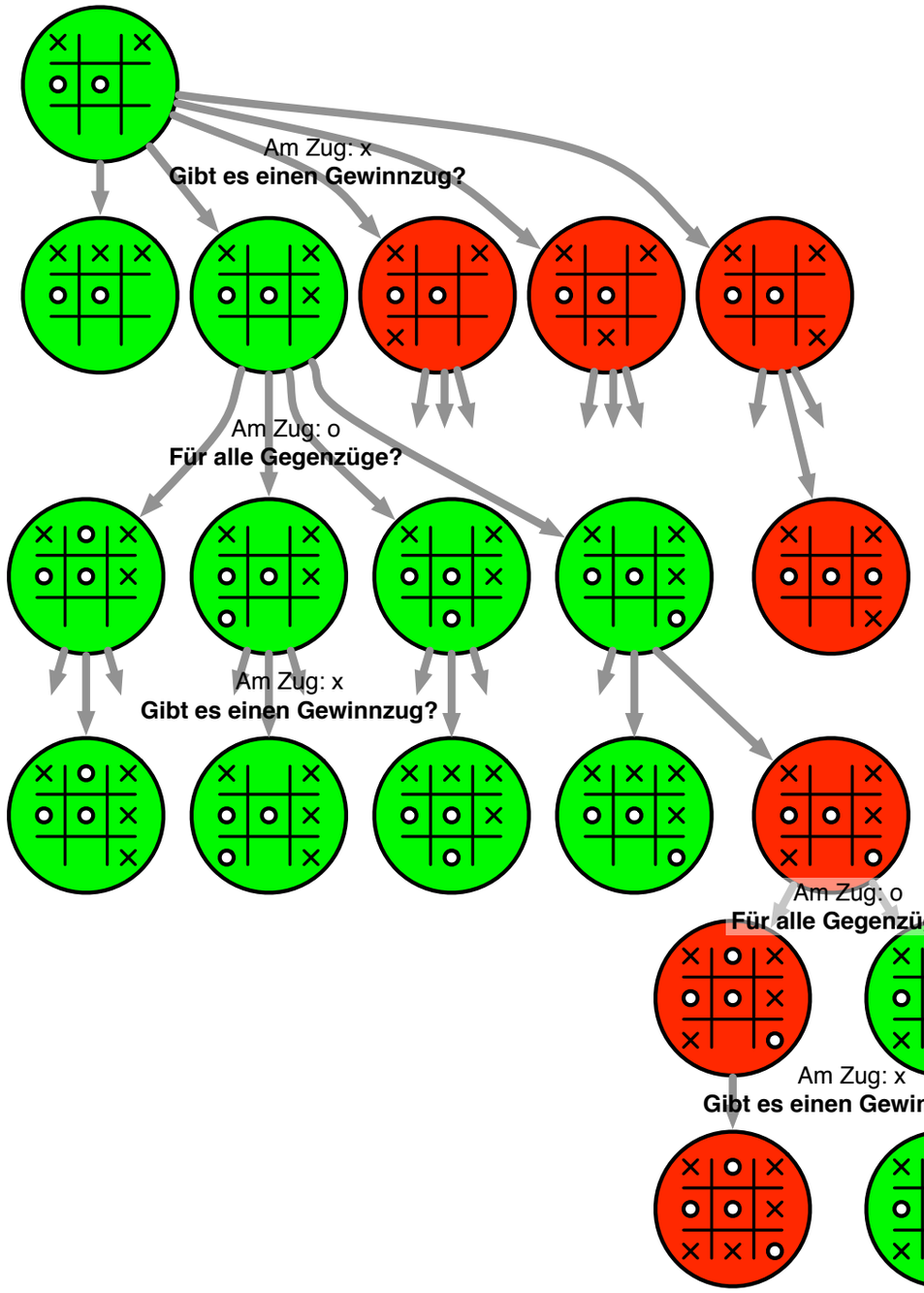
- **Tic-Tac-Toe kann als quantifizierte Boolesche Funktion beschrieben werden:**
  - Gibt es einen Zug für mich,
  - So dass für alle gültige Züge des Gegners,
  - es einen Gewinnzug für mich gibt oder einen gültigen Zug für mich gibt,
  - so dass für alle gültige Züge des Gegners kein Gewinnzug für ihn ist und
  - es einen Gewinnzug für mich gibt oder einen gültigen Zug für mich gibt,
  - so dass für alle gültige Züge des Gegners kein Gewinnzug für ihn ist und
  - es einen Gewinnzug für mich gibt oder einen gültigen Zug für mich gibt,
  - so dass für alle gültige Züge des Gegners kein Gewinnzug für ihn ist und
  - es einen Gewinnzug für mich gibt.

Tic-Tac-Toe als Spielbaum



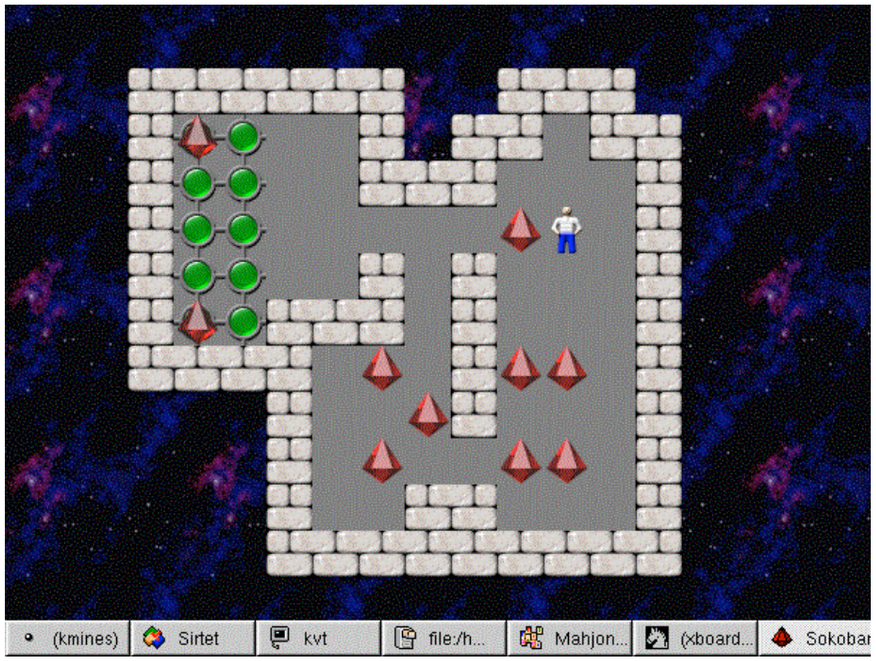
Tic-Tac-Toe als Spielbaum

- **Eine QBF beantwortet die Frage, ob Spieler x gewinnen kann**
- **Aber welcher Zug ist der Gewinnzug?**
- **Lösung:**
  - Probiere alle Möglichkeiten aus
  - Konstruiere die entsprechende QBF für Spieler o
  - Wähle die Möglichkeit, in der o verliert
- **Diese Art der Reduktion kennen wir als Turing-Reduktion**
- **PSPACE ist abgeschlossen gegenüber Polynom-Zeit-Turing-Reduktionen**



Mehr Spiele

- **Schach auf allgemeiner Brettgröße mit Fortschrittsregel ist PSPACE-vollständig**
  - Fortschrittsregel:
    - innerhalb von 50 Zügen muss ein Bauer bewegt werden oder eine Figur geschlagen werden
- **Sokoban:**
  - PSPACE-vollständig
- **Schach in verallgemeinerter Form ohne Fortschrittsregel ist**
  - EXPTIME-vollständig
- **Dame: für allgemeine Brettgrößen**
  - EXPTIME-vollständig



Informatik III
Winter 2007/08

Rechnernetze und Telematik
Albert-Ludwigs-Universität Freiburg
Christian Schindelhauer

4.4 Space

Ende