Theoretische Informatik 1

Strukturelle Komplexitätstheorie

David Kappel

Institut für Grundlagen der Informationsverarbeitung
Technische Universität Graz

22.05.2015

Übersicht

Strukturelle Komplexitätstheorie

- Die **Komplexitätstheorie** betrachtet nur Probleme, die in endlicher Zeit berechnet werden können.
- Aus der Struktur der Kategorisierung der Probleme in eine Vielzahl von verschiedenen, teilweise überlappenden Mengen, erhofft man sich Erkenntnisse über die prinzipielle Natur der Komplexität, um daraus wiederum effizientere Algorithmen zu gewinnen oder Beweise über prinzipielle Schranken der Effizienz.
- Die Kategorisierung von Problemen erfolgt durch sog. Komplexitätsklassen, das sind Mengen von Problemen (Sprachen), die jeweils gemeinsame Komplexitätseigenschaften (Zeitbedarf, Platzbedarf) haben.

Das Halteproblem

- Bisher: nur Probleme die prinzipiell lösbar sind.
- Aber: Es gibt Probleme für die *bewiesen* werden kann, dass sie *unlösbar* sind.
- Das Halteproblem ist eines der fundamentalsten dieser Ergebnisse.
- Seine Unlösbarkeit zeigt, dass die automatisierte Verifikation von Software im allgemeinen unlösbar ist.

Vorüberlegung

Turingmaschine $M = (Q, \Sigma, \Gamma, \delta, q_0, \square, F)$

- Q, Zustände: endliche Menge
- Σ, Eingabealphabet: endliche Menge von Zeichen
- Γ, Bandalphabet: endliche Menge von Zeichen
- $F \subset Q$, Endzuständen: endliche Menge von Zuständen
- $\delta : Q \times \Gamma \to Q \times \Gamma \times \{\leftarrow, -, \rightarrow\}$ Übergangsfunktion: endliche Menge von Regeln

$\to$ M kann durch eine endliche Zeichenkette $\langle M \rangle$ über einem endlichen Alphabet dargestellt werden.

Vorüberlegung: Universelle Turingmaschine

- Jede Turingmaschine kann als Zeichenkette $\langle M \rangle$ dargestellt werden.
- Als *universelle Turingmaschine* bezeichnet man eine Maschine, die $\langle M, w \rangle$ als Eingabe nimmt, und dann M auf Eingabe w ausführt.
- Hierzu muss M in einer geeigneten Darstellung repräsentiert werden, etwa durch Kodierung der Konfiguration κ von M in einem bestimmten Alphabet Σ_M.

Definition: Halteproblem

Definition (Halteproblem)

$\mathit{HALT} = \{\langle M, w\rangle \mid M \textit{ ist eine Turingmaschine und akzeptiert } w\}$

Obwohl HALT ein wohldefiniertes Entscheidungsproblem ist kann bewiesen werden, dass keine TM existieren *kann*, die es entscheidet. Daher gilt:

Satz

HALT ist nicht entscheidbar.

Beweis: HALT ist nicht entscheidbar[1]

Beweis Druch Widerspruch: Wir nehmen an, dass HALT entschieden werden kann und stoßen auf einen Widerspruch.

- Angenommen HALT sei entscheidbar.
- Dann existiert eine TM H, die die Eingabe $\langle M, w \rangle$ genau dann akzeptiert, wenn M die Eingabe w akzeptiert.
- Weiters hält und verwirft H, wenn M die Eingabe w nicht akzeptiert.

[1] siehe *Sipser* S.181f

Beweis: HALT ist nicht entscheidbar

- Nun konstruieren wir eine zweite Turingmaschine D mit Eingabe $\langle M \rangle$, die H als Unterprogramm benutzt.
- Diese TM ruft H auf um zu entscheiden wie M entscheidet, wenn sie mit ihrer eigenen Beschreibung $\langle M \rangle$ als Eingabe aufgerufen wird.
- D ruft also H mit Eingabe $\langle M, \langle M \rangle \rangle$ auf.
- Sobald H entscheidet, *invertiert* D diese Entscheidung.

Algorithmus $D(\langle M \rangle)$

Auf Eingabe $\langle M \rangle$, wobei M eine Turingmaschine ist:

1. Rufe H auf Eingabe $\langle M, \langle M \rangle \rangle$ auf.
2. Wenn H akzeptiert, gib '*nein*' aus, sonst gib '*ja*' aus.

Beweis: HALT ist nicht entscheidbar

Also:

$$D(\langle M \rangle) = \begin{cases} \textit{'ja'} & \text{wenn } H(\langle M, M \rangle) = \textit{'nein'} \\ \textit{'nein'} & \text{wenn } H(\langle M, M \rangle) = \textit{'ja'} \end{cases}$$

mit:

$$H(\langle M, w \rangle) = \begin{cases} \textit{'ja'} & \text{wenn } w \text{ von } M \text{ akzeptiert wird} \\ \textit{'nein'} & \text{sonst} \end{cases}$$

Beweis: HALT ist nicht entscheidbar

Was passiert nun, wenn D auf sich selbst, also $\langle D \rangle$ aufgerufen wird?

$$D(\langle D \rangle) = \begin{cases} \text{'ja'} & \text{wenn } H(\langle D, D \rangle) = \text{'nein'} \\ \text{'nein'} & \text{wenn } H(\langle D, D \rangle) = \text{'ja'} \end{cases}$$

Da D und H Entscheider sind folgt daraus:

$$D(\langle D \rangle) = \begin{cases} \text{'ja'} & \text{wenn } D(\langle D \rangle) = \text{'nein'} \\ \text{'nein'} & \text{wenn } D(\langle D \rangle) = \text{'ja'} \end{cases}$$

was offensichtlich ein Widerspruch ist. Daher kann weder D noch H existieren.

Turingmaschinen sind abzählbar unendlich

- Die Menge an Turingmaschine ist offensichtlich unendlich
- Daher könnte man vermuten, dass jedes beliebige Problem von einer TM lösbar ist
- Unendliche Mengen sind aber nicht unbedingt gleich mächtig
- Jede Turingmaschine M lässt sich als Zeichenkette $\langle M \rangle$ über dem Alphabet Σ_M darstellen, daher $\langle M \rangle \in \Sigma_M^*$
- Die Menge an möglichen Turingmaschinen ist abzählbar unendlich
- Daher lässt sich jeder Turingmaschine eine Zahl aus den natürlichen Zahlen $\mathbb{N}$ zuordnen

Formale Sprachen sind überabzählbar unendlich

- Eine formale Sprache *A* ist ein Element der Menge $\mathcal{P}(\Sigma^*)$
- Die Potenzmenge ($\mathcal{P}$) einer abzählbar unendlichen Menge ist überabzählbar unendlich
- Daher lässt sich *nicht* jeder formale Sprachen eine Zahl aus den natürlichen Zahlen $\mathbb{N}$ zuordnen
- Gleichzeitig lässt sich *nicht* jeder formale Sprachen eine Turingmaschine zuordnen
- Es gibt daher beliebig viele formale Sprachen die nicht von einer Turingmaschine entschieden werden können!

Häufig gebrauchte Platzkomplexitätsklassen

Deterministische Platzkomplexitätsklassen:

- $\mathrm{L} = \mathrm{DSPACE}(\log n)$
- $\mathrm{PSPACE} = \bigcup_{k \in \mathbb{N}} \mathrm{DSPACE}(n^k)$
- $\mathrm{EXPSPACE} = \bigcup_{k \in \mathbb{N}} \mathrm{DSPACE}(2^{n^k})$

Nichtdeterministische Komplexitätsklassen:

- $\mathrm{NL} = \mathrm{NSPACE}(\log n)$
- $\mathrm{NPSPACE} = \bigcup_{k \in \mathbb{N}} \mathrm{NSPACE}(n^k) = \mathrm{PSPACE}$ *(Savitch!)*
- $\mathrm{NEXPSPACE} = \bigcup_{k \in \mathbb{N}} \mathrm{NSPACE}(2^{n^k}) = \mathrm{EXPSPACE}$ *(Savitch!)*

Häufig gebrauchte Zeitkomplexitätsklassen

Deterministische Zeitkomplexitätsklassen:

- $\mathrm{P} = \bigcup_{k \in \mathbb{N}} \mathrm{DTIME}(n^k)$
- $\mathrm{EXP} = \bigcup_{k \in \mathbb{N}} \mathrm{DTIME}(2^{n^k})$

Nichtdeterministische Zeitkomplexitätsklassen:

- $\mathrm{NP} = \bigcup_{k \in \mathbb{N}} \mathrm{NTIME}(n^k)$
- $\mathrm{NEXP} = \bigcup_{k \in \mathbb{N}} \mathrm{NTIME}(2^{n^k})$

Zeit- und Platz-Hierarchiesätze

- Man erwartet: je mehr Ressourcen (Zeit/Platz) einer Turingmaschine zur Verfügung gestellt werden, umso mehr Sprachen können entschieden werden.
- Hierarchiesätze formalisieren diese Intuition.

Satz (Hierarchiesatz)

Für jede (zeit/platz-konstruierbare[1]) Funktion $f : \mathbb{N} \to \mathbb{N}$ *existiert eine Sprache A, die in* $\mathcal{O}(f(n))$ *Zeit/Platz entscheidbar ist aber nicht in* $o(f(n))$ *Zeit/Platz.*

[1] Für die genauen Anforderungen an *f* sowie den genauen Beweis des Satzes siehe *Sipser*, Kapitel 9.1.

Beweisidee

- Wir zeigen, dass die TM *tatsächlich* an Berechnungsstärke gewinnt, wenn man ihr mehr Zeit/Platz zur Verfügung stellt.
- Zumindest für die Komplexitätsklassen die hier betrachtet wurden P, EXP, ... kann dies bewiesen werden.
- Wir zeigen, dass in *jeder* Komplexitätsklasse Sprachen A existieren, die in $\mathcal{O}(f(n))$ aber nicht in $o(f(n))$ Zeit/Platz entschieden werden können.
- Der Beweis kann als eine Erweiterung des Halteproblems verstanden werden.

Beweisidee (für Platz)

wir beschreiben A durch Angabe eines Algorithmus D der A entscheidet.

- D muss in $\mathcal{O}(f(n))$ Platz laufen.

Um zu garantieren, dass D nicht in $o(f(n))$ Platz laufen kann benutzen wir:

- Sei M eine TM, die eine Sprache in $o(f(n))$ Platz entscheidet.
- D stellt sicher, dass A sich zumindest für eine Eingabe von M's Sprache unterscheidet.
- Diese Eingabe ist genau die Beschreibung von M selbst, d.h. $\langle M \rangle$.

Beweisidee (für Platz)

- D nimmt die Beschreibung einer Maschine M als Eingabe.
- D simuliert M mit dem Input $\langle M \rangle$ auf Platz $f(n)$.
- Akzeptiert M so verwirft D, und umgekehrt.
- Hält M nicht, so wird die Eingabe einfach von D verworfen (hier möglich, da nur Entscheider-TM).
- Kann realisiert werden durch anlegen eines ‘Maßbandes’.

Der Algorithmus D

D auf Eingabe $w = \langle M \rangle$

- Sei n die Länge der Eingabe w
- Berechne $f(n)$ und markiere das Band an dieser Stelle. Wird diese Stelle später überschritten verwerfe.
- Simuliere M auf w und zähle die Anzahl an Zeitschritten. Wird $2^{f(n)}$ überschritten verwerfe.
- Wenn M verwirft, akzeptiere. Sonst verwerfe.

Beweisidee

- Laut annahme läuft D in $\mathcal{O}(f(n))$ Platz.
- Wir zeigen, dass A nicht in $o(f(n))$ entschieden werden kann.

Beweis durch Widerspruch:

- Angenommen eine Maschine M' existiert, die A in $o(f(n))$ Platz entscheidet.
- Wir führen D auf Eingabe $\langle M' \rangle$ aus.
- Da, M' in $o(f(n))$ läuft, kann sie von D simuliert werden.
- Daher entscheidet D genau das Gegenteil von M'.
- Daher kann M' nicht existieren und es folgt $A \notin \text{SPACE}(o(f(n)))$.

Konsequenzen der Hierarchiesätze

- $P \subsetneq EXP$
- $NP \subsetneq NEXP$
- $L \subsetneq PSPACE \subsetneq EXPSPACE$

Einfache Inklusionen der grundlegendsten Klassen

DTM ist Spezialfall der NTM

$P \subseteq NP, EXP \subseteq NEXP$

$L \subseteq NL, PSPACE \subseteq NPSPACE$

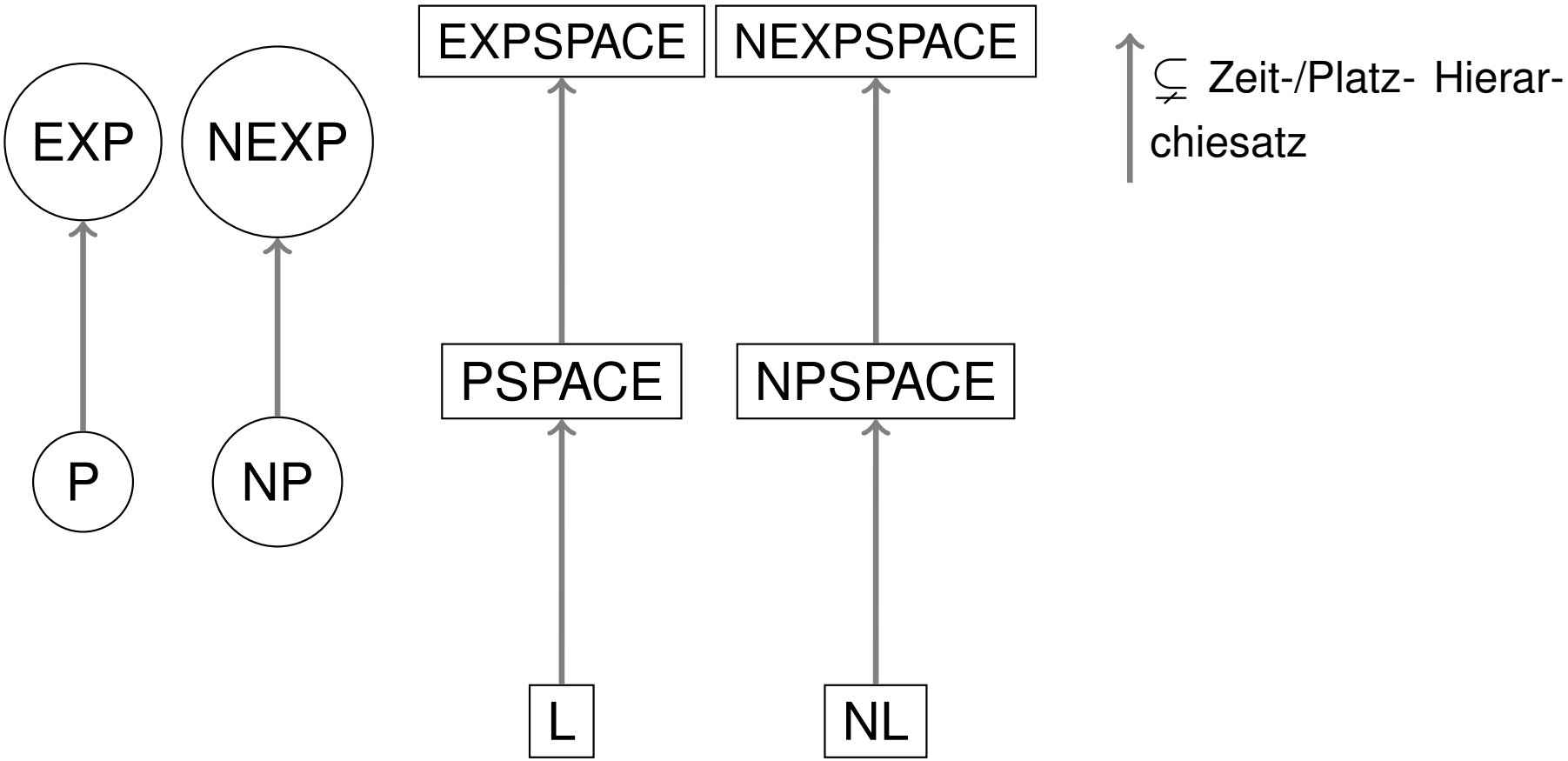
Hierarchiesätze:

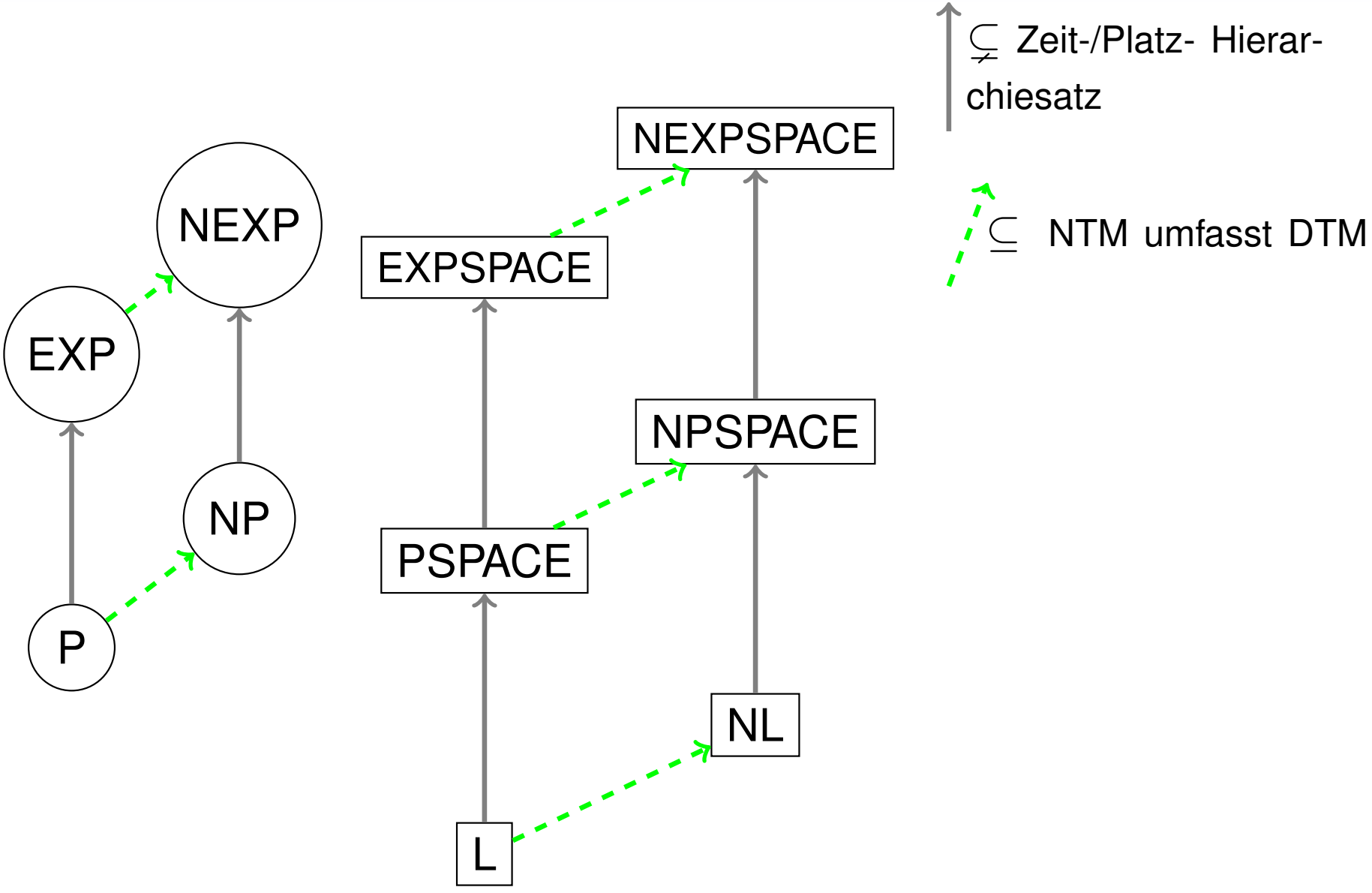
$P \subsetneq EXP$, $NP \subsetneq NEXP$

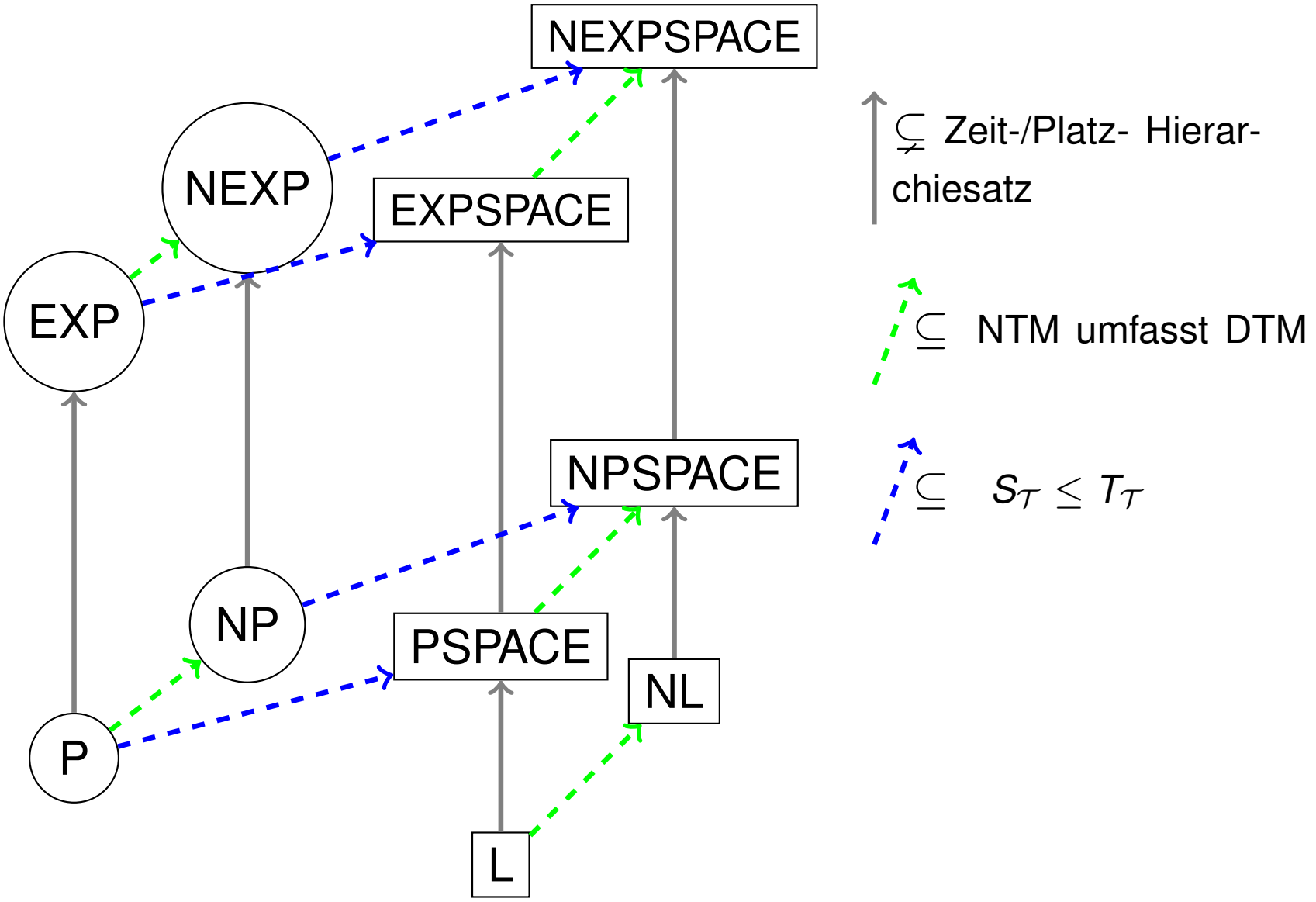
$L \subsetneq PSPACE \subsetneq EXPSPACE$, $NL \subsetneq NPSPACE \subsetneq NEXPSPACE$

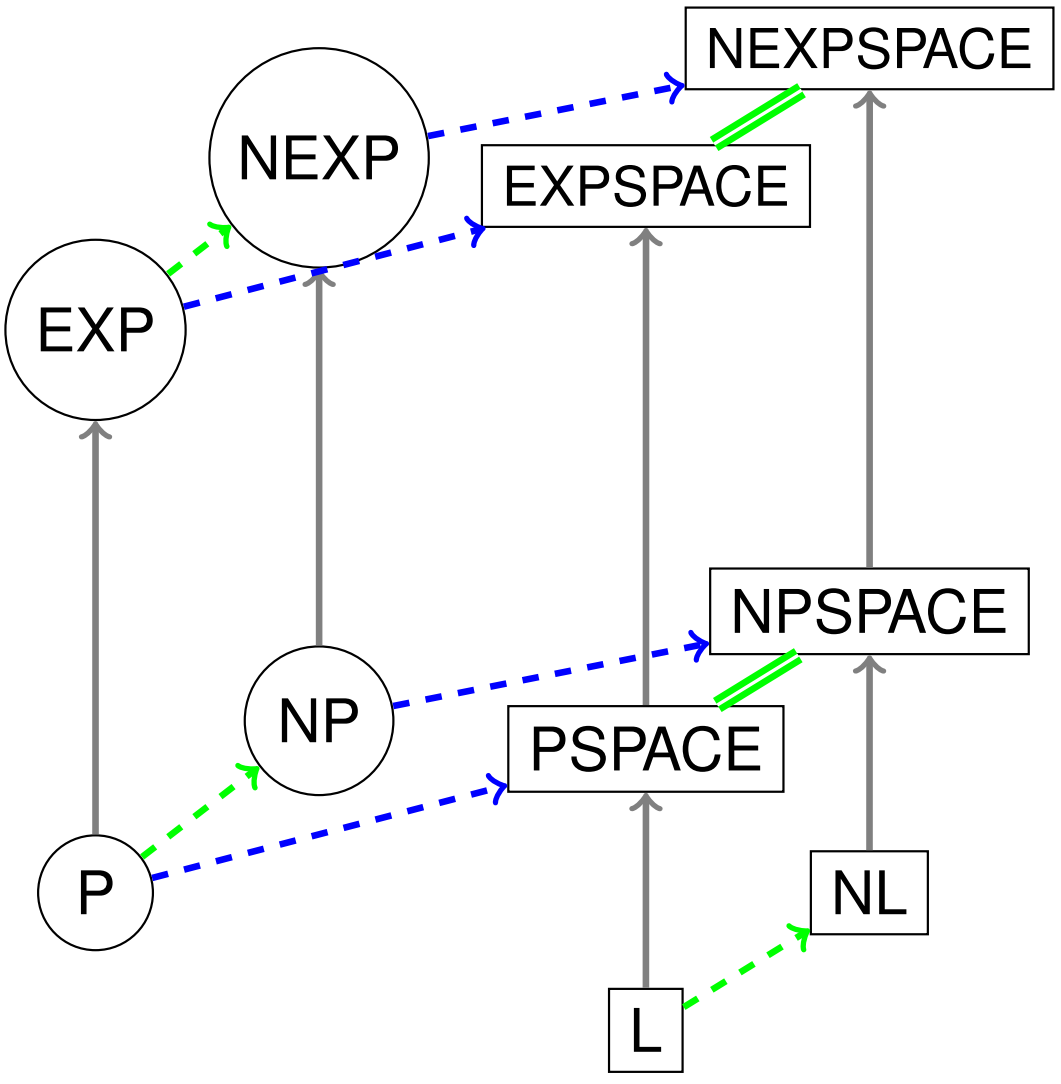
Platzkomplexität $\leq$ Zeitkomplexität

$P \subseteq PSPACE, NP \subseteq NPSPACE$


EXP
NEXP
P
NP
EXPSPACE
NEXPSPACE
PSPACE
NPSPACE
L
NL
$\subsetneq$ Zeit-/Platz- Hierarchiesatz


$\subsetneq$ Zeit-/Platz- Hierarchiesatz
$\subseteq$ NTM umfasst DTM
NEXPSPACE
NEXP
EXPSPACE
EXP
NPSPACE
NP
PSPACE
P
NL
L


NEXPSPACE
NEXP
EXPSPACE
EXP
NPSPACE
NP
PSPACE
NL
P
L
$\subsetneq$ Zeit-/Platz- Hierarchiesatz
$\subseteq$ NTM umfasst DTM
$\subseteq$ $S_{\mathcal{T}} \leq T_{\mathcal{T}}$

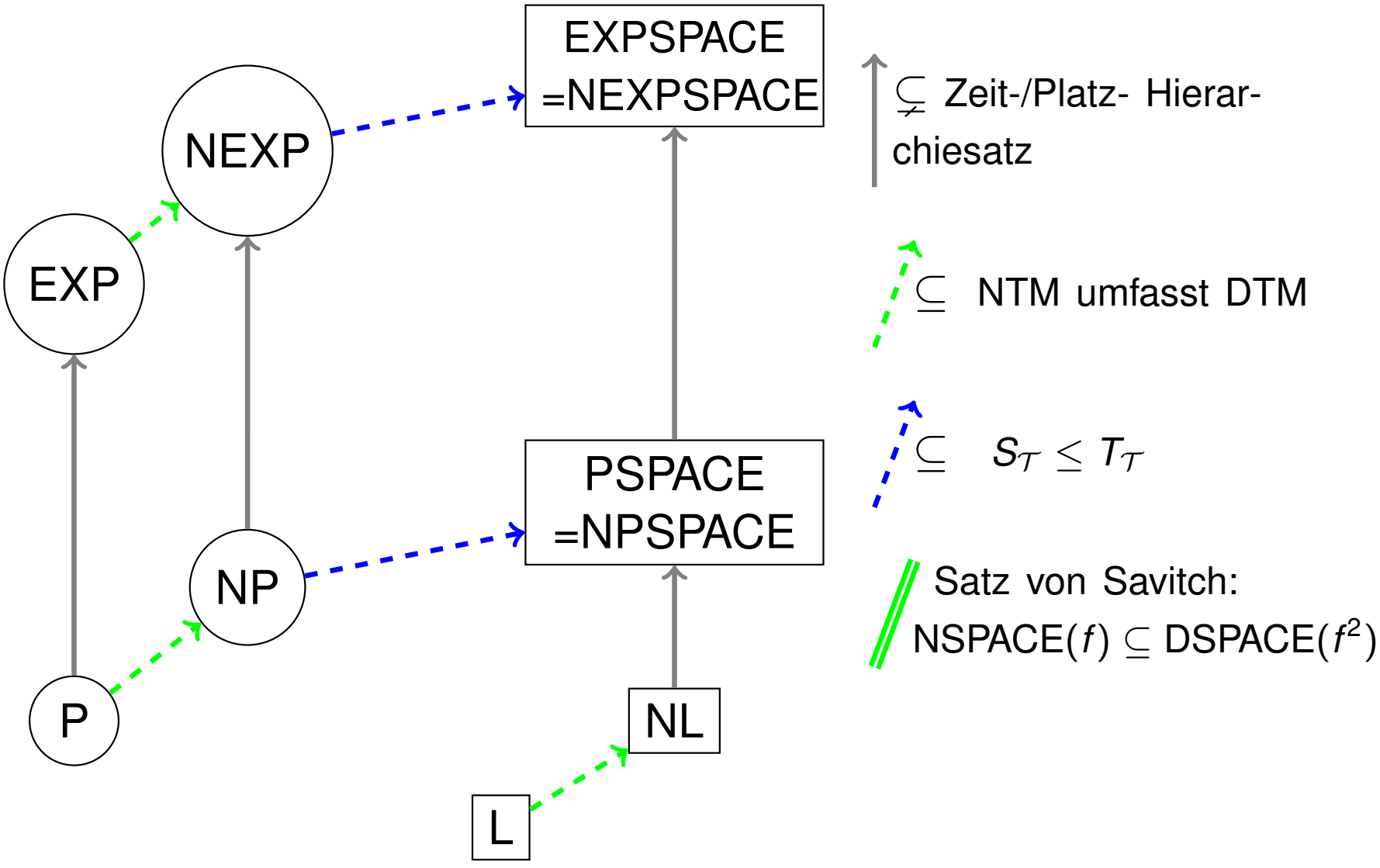


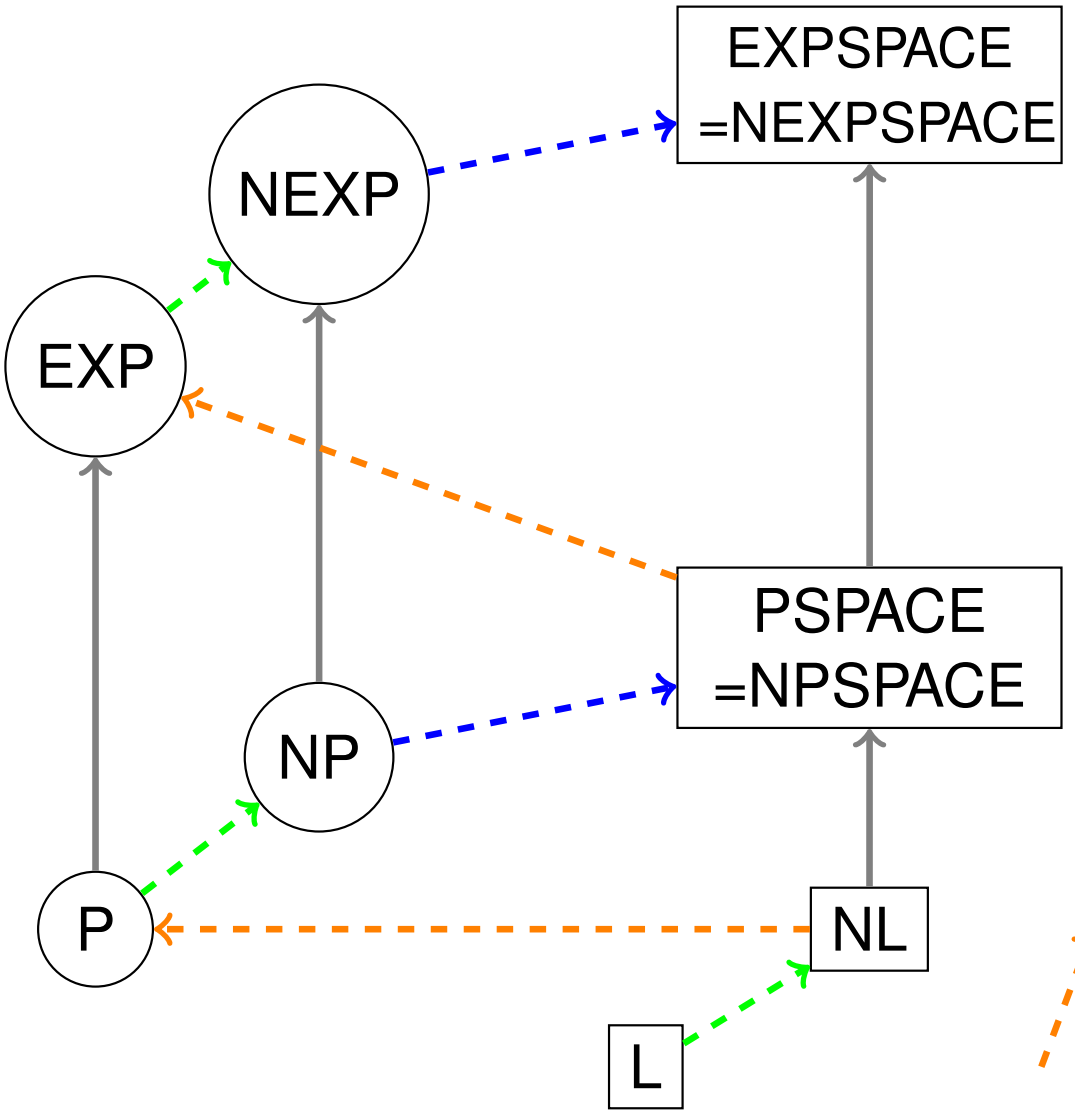
$\subsetneq$ Zeit-/Platz- Hierarchiesatz

$\subseteq$ NTM umfasst DTM

$\subseteq$ $S_{\mathcal{T}} \leq T_{\mathcal{T}}$

Satz von Savitch:
$\mathrm{NSPACE}(f) \subseteq \mathrm{DSPACE}(f^2)$


EXPSPACE
=NEXPSPACE
NEXP
EXP
PSPACE
=NPSPACE
NP
P
NL
L
$\subsetneq$ Zeit-/Platz- Hierarchiesatz
$\subseteq$ NTM umfasst DTM
$\subseteq$ $S_{\mathcal{T}} \leq T_{\mathcal{T}}$
Satz von Savitch:
$\mathrm{NSPACE}(f) \subseteq \mathrm{DSPACE}(f^2)$



$\subsetneq$ Zeit-/Platz- Hierarchiesatz

$\subseteq$ NTM umfasst DTM

$\subseteq$ $S_{\mathcal{T}} \leq T_{\mathcal{T}}$

Satz von Savitch:
$\mathrm{NSPACE}(f) \subseteq \mathrm{DSPACE}(f^2)$

Konfigurationenanzahl:
$\mathrm{NSPACE}(s(n)) \subseteq \mathrm{DTIME}(2^{\mathcal{O}(s(n))})$

Zusammenfassung

- Das Halteproblem ist nicht entscheidbar
- Klassenstruktur: Beziehung zwischen Zeit/Platz-Komplexitätsklassen
- Hierarchiesätze: echte Mengeninklusion innerhalb der Klassenhierarchien bekannt
- Klassenstruktur: von L bis EXPSPACE