Technische Universität Dresden

Theoretische Informatik und Logik

12. Vorlesung: PSpace-Vollständigkeit

Markus Krötzsch

Professur Wissensbasierte Systeme

TU Dresden, 4. Juni 2026

NLogSpace

NL: Probleme, die eine nichtdeterministische TM mit logarithmischem Speicher lösen kann

- Deterministisch lösbar in polynomieller Zeit
- In der Regel deutlich besser parallelisierbar als typische Probleme in P
- Aber: seriell nicht unbedingt schneller lösbar als andere Probleme in P

Typisches Beispiel: Erreichbarkeit in gerichteten Graphen

PSpace

PSpace-vollständige Probleme
= Probleme, die mindestens so schwer sind, wie alle anderen Probleme in PSpace
= die schwersten Probleme in PSpace.

Vermutung:
Kein PSpace-vollständiges Problem ist in NP,
d.h. PSpace ist echt schwerer als NP und coNP

PSpace-vollständige Probleme

- **TrueQBF:** Wahrheit von quantifizierten Booleschen Formeln (und das Formelspiel)
- **Geography:** Spiel auf einem Graphen

PSpace-Vollständigkeit

Satz: TrueQBF ist in PSpace.

Beweis: Durch Angabe eines (Pseudo-)Algorithmus

```
01 TrueQBF(F) :
02   if F „hat keine Quantoren“ :
03     return „Aussagenlogische Auswertung von F“
04   else if F = ∃p.G :
05     return (TrueQBF(G[p/⊤]) OR TrueQBF(G[p/⊥]))
06   else if F = ∀p.G :
07     return (TrueQBF(G[p/⊤]) AND TrueQBF(G[p/⊥]))
```

- Evaluation in Zeile `03` möglich in PSpace
- Rekursion in Zeilen `05` und `07` können der Reihe nach abgearbeitet werden, wobei Speicher wiederverwendet wird
- Jeder Rekursionsschritt benötigt polynomiellen Speicher
- Maximale Rekursionstiefe = Zahl der Atome (linear) □

PSpace-Schwere

Ein Problem **Q** ist PSpace-schwer wenn für jedes Problem **P** in PSpace ein polynomielle Reduktion **P** $\leq_p$ **Q** existiert. **Q** ist PSpace-vollständig wenn es PSpace-schwer ist und in PSpace liegt.

Satz: TrueQBF ist PSpace-schwer.

Beweisidee:

- Wie beim Satz von Cook/Levin stellen wir den Lauf einer Turingmaschine mit aussagenlogischen Atomen dar
- Speicher ist wie bei NP polynomiell (einfach kodierbar)
- Zeit ist problematisch: in PSpace kann ein TM exponentiell lang laufen $\leadsto$ wir können nicht einfach für jeden Zeitschritt eine neue Version der Konfigurationsvariablen anlegen und jeden Übergang mit Formeln axiomatisieren

Beweisidee (Fortsetzung)

Einsichten:

- Man kann eine komplette TM-Konfiguration in polynomiell vielen Atomen kodieren (wie bei Cook-Levin)
- Ebenso kann man zwei oder drei Konfigurationen kodieren – aber nicht exponentiell viele (für jeden Schritt)
- Direkte Übergänge, Start- und Endkonfiguration kann man mit Formeln axiomatisieren

Akzeptanz ausdrücken

- Gewünschte Aussage: „Ausgehend von der Startkonfiguration kann eine Endkonfiguration in endlich vielen Übergängen erreicht werden“
- = Erreichbarkeitsproblem in einem exponentiell großen Graphen

Beweisidee (Fortsetzung)

Erreichbarkeitsproblem in einem exponentiell großen Graphen

Lösung: „Middle-First-Suche“

- Um zu prüfen, ob man von s nach t gelangen kann
- Prüfe zunächst, ob $s = t$ oder s direkter Vorgänger von t
- Rate einen Punkt m in der „Mitte“ eines Pfades von s nach t
- Prüfe (rekursiv) ob man von s nach m gelangen kann
- Prüfe (rekursiv) ob man von m nach t gelangen kann

$\leadsto$ Anzahl der zu ratenden Mittelpunkte ist logarithmisch in Länge des Pfades
$\leadsto$ Speicher kann in jedem Schritt wiederverwendet werden

Diesen Algorithmus kann man polynomiell in QBF kodieren

Alternierende Quantoren

TrueQBF$_{\text{alt}}$ ist das Problem **TrueQBF**, beschränkt auf QBF der Form $\exists p_1.\forall p_2.\exists p_3.\ldots.\forall p_{\ell-1}.\exists p_\ell.F$, wobei F in Klauselform ist.

Diese Version entspricht eher der Idee eines Spiels, bei dem Emilia und Anton abwechselnd Belegungen festlegen. Emilia erhält den ersten und den letzten Zug.

Satz: TrueQBF$_{\text{alt}}$ ist PSpace-vollständig.

Beweis: TrueQBF$_{\text{alt}}$ $\in$ PSpace folgt aus **TrueQBF** $\in$ PSpace.

Für die Schwere zeigen wir **TrueQBF** $\leq_p$ **TrueQBF$_{\text{alt}}$**:

- man fügt einfach nach Bedarf zusätzliche Quantoren ein
- es müssen nicht alle quantifizierten Atome in F vorkommen □

Rückblick: Geography

Ein Kinderspiel:

- Zwei Spieler benennen abwechselnd Städte
- Jede Stadt muss mit dem letzten Buchstaben der zuvor genannten beginnen
- Wiederholungen sind verboten
- Der erste Spieler, der keine Stadt mehr nennen kann, verliert

Ein Mathematikerspiel:

- Zwei Spieler markieren Knoten in einem gerichteten Graph
- Jeder Knoten muss ein Nachfolger des vorigen sein
- Wiederholungen sind verboten
- Der erste Spieler, der keinen Knoten markieren kann, verliert

Entscheidungsproblem **Geography**:

Gegeben: Ein gerichteter Graph und ein Startknoten
Frage: Hat Emilia eine Gewinnstrategie für dieses Spiel?

Geography ist PSPACE-vollständig (1)

Satz: Geography ist PSPACE-vollständig.

Beweis: (1) **Geography** ist in PSPACE

Bei Startknoten s und Graph G erhalten wir die Antwort durch Aufruf von `eCanWin`(G, s, $\{s\}$), definiert wie folgt:

```
eCanWin(G, n, Visited) :
  result = false
  for all Nachfolger* m von n in G mit m ∉ Visited :
    result = result OR aCannotWin(G,m,Visited ∪ {m})
  return result

aCannotWin(G, n, Visited) :
  result = true
  for all Nachfolger* m von n in G mit m ∉ Visited :
    result = result AND eCanWin(G,m,Visited ∪ {m})
  return result
```

* gemeint sind jeweils direkte Nachfolger

Geography ist PSPACE-vollständig (2)

Satz: Geography ist PSPACE-vollständig.

Beweis: (2) **Geography** ist PSPACE-schwer

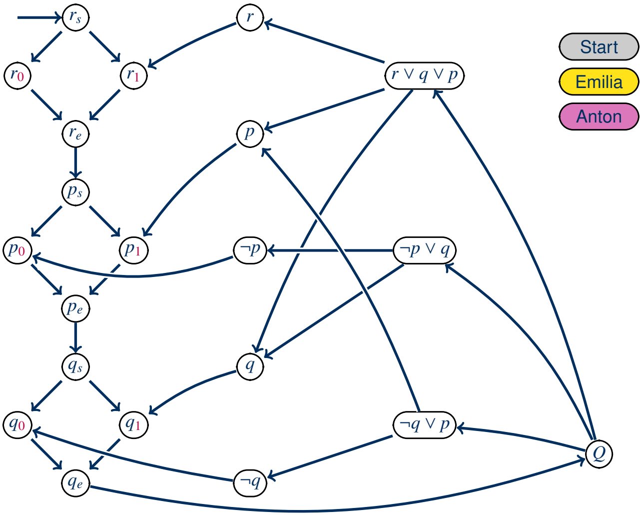
Durch Reduktion **TrueQBF$_{\text{alt}}$** $\leq_p$ **Geography**:
Wir betrachten eine QBF $\exists p_1.\forall p_2 \ldots \exists p_\ell.F$ und konstruieren daraus einen Graphen für **Geography**

Grundidee: (siehe auch M. Sipser: Introduction to the Theory of Computation; 2013; Thm. 8.14)

- Der Graph beginnt mit einer Abfolge von Rauten-Strukturen, welche die Entscheidungen der Spieler darstellen: für jedes Atom p gibt es dort zwei Knoten, p_1 und p_0, von denen genau einer besucht wird
- Es folgt eine Baumstruktur: ein Knoten für F, darunter ein Knoten für jede Klausel und darunter jeweils ein Knoten für jedes Literal
- Von jedem Literal p bzw. $\neg p$ gibt es eine Kante zurück zum Knoten p_1 bzw. p_0

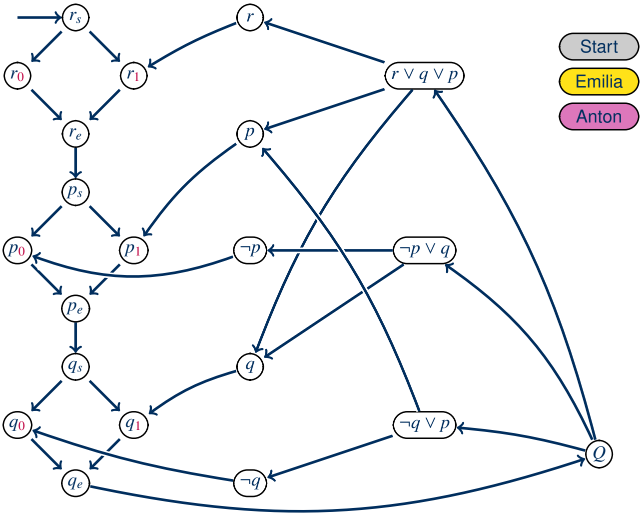
Geography ist PSPACE-vollständig: Beispiel

Wir betrachten die Formel $\exists r.\forall p.\exists q.(r \vee q \vee p) \wedge (\neg p \vee q) \wedge (\neg q \vee p)$



Geography ist PSPACE-vollständig: Beispiel

Wir betrachten die Formel $\exists r.\forall p.\exists q.(r \vee q \vee p) \wedge (\neg p \vee q) \wedge (\neg q \vee p)$



Geography ist PSPACE-vollständig (3)

Satz: Geography ist PSPACE-vollständig.

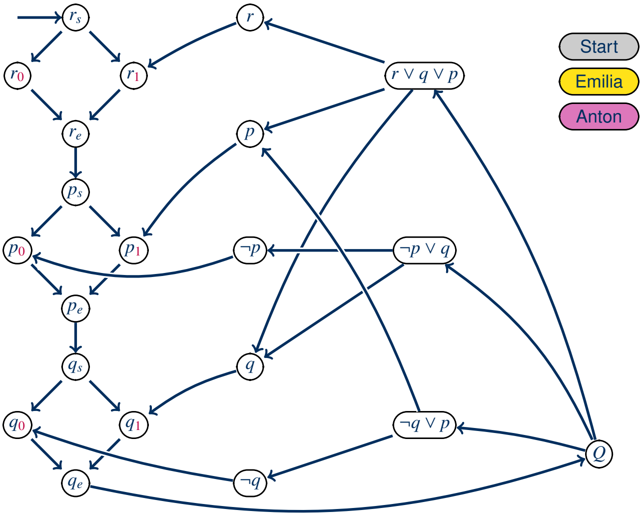
Beweis: (2) **Geography** ist PSPACE-schwer

Beweisidee:

- Zuerst bestimmen Anton und Emilia abwechselnd Wahrheitswerte, indem sie die Rauten jeweils links oder rechts durchlaufen
- Danach darf Anton zur Auswertung eine Klausel auswählen (die er für nicht erfüllt hält)
- Anschließend wählt Emilia ein Literal dieser Klausel (welches ihrer Meinung nach erfüllt ist)
- Emilia gewinnt, wenn das Literal wirklich erfüllt war (denn dann gibt es keinen Weg zu einem noch nicht besuchten Knoten)
- Anton gewinnt, wenn das Literal nicht erfüllt war (denn dann kann er den Weg noch genau einen Schritt fortsetzen)

Geography ist PSPACE-vollständig: Beispiel

Wir betrachten die Formel $\exists r.\forall p.\exists q.(r \lor q \lor p) \land (\neg p \lor q) \land (\neg q \lor p)$



Weitere Komplexitätsklassen

Komplexität jenseits von PSpace?

Wir haben uns auf drei Klassen konzentriert:

$$\text{P} \subseteq \text{NP} \subseteq \text{PSpace}$$

Ersetzt man „polynomiell“ durch „exponentiell“, so erhält man jenseits von PSpace ein ganz ähnliches Bild:

$$\text{ExpTime} \subseteq \text{NExpTime} \subseteq \text{ExpSpace}$$

Man kann beliebig hohe Türme von Exponenten konstruieren:

$$\text{2ExpTime} \subseteq \text{N2ExpTime} \subseteq \text{2ExpSpace}$$

$$\text{3ExpTime} \subseteq \text{N3ExpTime} \subseteq \text{3ExpSpace}$$

$$\vdots$$

Vieles, was wir gelernt haben, gilt hier wie zuvor – aber diese Klassen sind von immer geringerer praktischer Relevanz

Gibt es so schwere Probleme?

Für k-ExpP-Klassen lassen sich künstliche Probleme leicht finden: „Gegeben eine det. Turingmaschine $\mathcal{M}$, eine Eingabe w, und ein 'Füllwort' $\mathtt{a}^m$ der Länge m, wird $\mathcal{M}$ das Wort w in Zeit $\underbrace{2^{\cdot^{\cdot^{\cdot^{2^m}}}}}_{k\text{-mal } 2}$ akzeptieren?" (Warum ist das k-Exp-vollständig? Zeigen Sie Inklusion und Schwere.)

Jenseits von ExpSpace gibt es nur wenige relevante Probleme.

Beispiel: Der W3C-Standard Web Ontology Language (OWL, Version 2) definiert die logische Beschreibungsspache OWL 2 DL. Logisches Schließen in dieser Sprache ist N2ExpTime-vollständig.

Es gibt aber auch entscheidbare Probleme, die durch keine vielfach exponentiell beschränkte TM entschieden werden: Dies sind Probleme nicht-elementarer Komplexität

Beispiel: Die Äquivalenz von regulären Ausdrücken mit einem zusätzlichen Komplementierungsoperator ist entscheidbar, aber nicht elementar.

Viele schwere praktische Probleme sind einfach unentscheidbar
(z.B. Syntax von C++)

Komplexitäten unterhalb von P?

Wir haben die folgenden Komplexitäten betrachtet

$$\mathsf{L} \subseteq \mathsf{NL} \subseteq \mathsf{P}$$

Hier gibt es bereits **technische Probleme:**

- TMs mit logarithmischem Speicher benötigen getrennte Ein- und Ausgabebänder
- Logarithmen sind nicht abgeschlossen unter polynomiellen Operationen; z.B. sagt uns Savitch nicht ob $\mathsf{L} \stackrel{?}{=} \mathsf{NL}$

Will man **noch kleinere Komplexitätsklassen** betrachten, dann muss man zu anderen Berechnungsmodellen greifen:

- Schaltkreise statt TMs ($\leadsto$ circuit complexity)
- Logarithmisch zeitbeschränkte TMs mit beschränkter Parallelverarbeitung ($\leadsto$ alternating, random access TM)
- Automatenmodelle ($\leadsto$ DFA, PDA, ...)

Manche der Klassen sind dann sehr stark von Details der Problemdefinition abhängig (wenig robust)

Gibt es so leichte Probleme?

Auch bei den subpolynomiellen Klassen ergeben sich in der Regel typische Probleme aus der Definition.

Es gibt eine Reihe interessanter, praktisch relevanter L-Probleme, z.B.

- Erreichbarkeit in ungerichteten Graphen (Omar Reingold, 2005)
- Zwei-Färbbarkeit von Graphen

Es gibt auch noch einfachere praktisch relevante Probleme, z.B.

- logische und arithmetische Operationen
- Beantwortung von fest vorgegebenen SQL-Anfragen auf beliebigen Datenbanken
- Wortprobleme bei endlichen Automaten

In diesen Fällen wird es immer schwerer, von „Schwere“ und „Vollständigkeit“ zu sprechen

Komplexität und Spiele

NP ist eine typische Klasse für Solitaire-Spiele:

- Sudoku, Minesweeper, Tetris, ...

PSpace ist eine typische Klasse für Spiele, bei denen zwei Spieler abwechselnd eine polynomielle Zahl an Zügen durchführen:

- Geography, Reversi, Tic-Tac-Toe, aber auch: Sokoban, ...

ExpTime findet sich bei Spielen, bei denen man Züge rückgängig machen kann (polynomielles Spielbrett – exponentiell viele Züge):

- Schach, Dame, Go, Stern-Halma, ...

In jedem Fall muss man (nicht-endliche) Verallgemeinerungen der Spiele betrachten, um die „wahre“ Komplexität zu sehen

(Menschen spielen nicht, indem sie innerlich eine endliche Datenbank aller möglichen Stellungen konsultieren)

Spiele sollten komplex sein, um lange zu motivieren

Und sonst so?

Es gibt viele weitere Themen in der Komplexitätstheorie

$\leadsto$ siehe Vorlesung „Complexity Theory“ (Wintersemester)

Zusammenfassung und Ausblick

TrueQBF und **Geography** sind PSPACE-vollständig

Wir kennen die praktisch wichtigsten Komplexitätsklassen und jeweils typische Probleme

Spiele liefern interessante Beispiele komplexer Probleme

Was erwartet uns als nächstes?

- Einführung in die Prädikatenlogik
- Entscheidbare logische Probleme
- Unentscheidbare logische Probleme