TECHNISCHE UNIVERSITÄT DRESDEN

THEORETISCHE INFORMATIK UND LOGIK

11. Vorlesung: NL und PSpace

Markus Krötzsch

Professur Wissensbasierte Systeme

TU Dresden, 16. Mai 2024

NP-vollständige Probleme

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

Alles oder nichts:
Entweder sind alle NP-vollständigen Probleme in P
oder kein einziges NP-vollständiges Problem ist in P

NP-vollständige Probleme

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

Alles oder nichts:
Entweder sind alle NP-vollständigen Probleme in P
oder kein einziges NP-vollständiges Problem ist in P

Ladner: „Alle glauben P ≠ NP. Dann gibt es aber auch beliebig viele Probleme in NP, die nicht NP-vollständig sind und dennoch nicht in P liegen.“

Anders gesagt: Neben den „schwersten“ Problemen in NP gibt es dann auch noch viele „mittelschwere“, welche dennoch nicht in P liegen. Bisher wissen wir nicht, welche das sind.

Leichte NP-vollständige Probleme

Pseudopolynomielle Probleme sind polynomiell in der Größe von Eingabe und gegebenen Zahlenbeträgen.

Das macht sie in der Praxis oft eher einfach.

Beispiel: Das Rucksackproblem ist nur dann NP-vollständig, wenn die Gewichte der Gegenstände über-polynomiell wachsen dürfen. Ein Problem mit so schweren Gegenständen ist aber nur dann interessant, wenn auch der Rucksack eine sehr große Kapazität hat. Alternativ könnte man mit sehr hoher Genauigkeit wiegen.

NL

Die Macht des Speichers

Selbst innerhalb kleiner Speichergrenzen ist sehr viel machbar:

Die Macht des Speichers

Selbst innerhalb kleiner Speichergrenzen ist sehr viel machbar:

- **SAT** ist mit linearem Speicher lösbar:
  Wir iterieren durch alle Wahrheitswertbelegungen (jeweils linear groß) und testen jeweils, ob die Formel erfüllt ist (logarithmischer Speicher für ein paar Pointer und Zwischenergebnisse)

Die Macht des Speichers

Selbst innerhalb kleiner Speichergrenzen ist sehr viel machbar:

- **SAT** ist mit linearem Speicher lösbar:
  Wir iterieren durch alle Wahrheitswertbelegungen (jeweils linear groß) und testen jeweils, ob die Formel erfüllt ist (logarithmischer Speicher für ein paar Pointer und Zwischenergebnisse)
- Linearer Speicher genügt zur Erkennung kontextsensitiver Sprachen (durch linear beschränkte Automaten, LBA)

Die Macht des Speichers

Selbst innerhalb kleiner Speichergrenzen ist sehr viel machbar:

- **SAT** ist mit linearem Speicher lösbar:
  Wir iterieren durch alle Wahrheitswertbelegungen (jeweils linear groß) und testen jeweils, ob die Formel erfüllt ist (logarithmischer Speicher für ein paar Pointer und Zwischenergebnisse)
- Linearer Speicher genügt zur Erkennung kontextsensitiver Sprachen (durch linear beschränkte Automaten, LBA)
- Jedes NP-vollständige Problem ist in polynomiellen Speicher lösbar:
  Wir iterieren durch alle polynomiellen Zertifikate und simulieren einen polynomiellen Verifikator auf ihnen

$\rightsquigarrow$ sehr kleine Speichergrenzen sind sinnvoll

Erinnerung: L

LogSpace (L): Sprachen die man mit sehr wenig Arbeitsspeicher erkennen kann

Wesentliche Datentypen:

- Zähler, Maximalwert polynomiell beschränkt
- Pointer auf (read-only) Eingabeband

Jeweils fest deklariert, d.h. ihre Anzahl hängt nicht von der Eingabe ab

Wesentliche Programmierfeatures:

- Initialisiere Pointer oder Zähler auf festen Wert
- Inkrementiere/dekrementiere Pointer oder Zähler
- Vergleiche Speicherinhalte von zwei Pointern oder zwei Zählern (und führe je nach Ergebnis anderen Code aus)

Optionaler Ausgabestrom: Write-only, write once

NLogSpace

Nichtdeterministische TM mit logarithmischem Speicher:

$$\text{NL} = \text{NLogSpace} = \text{NSpace}(\log n)$$

Alternativ:
„Probleme, deren Lösung in L verifiziert werden kann“

- Gleiche Programmierfeatures wie in L
- Aber nichtdeterministische Operationen möglich,
  z.B. setze Pointer auf eine zufällige Eingabeposition

Beispiel: Erreichbarkeit

Das Problem der (**s-t-**)**Erreichbarkeit** in gerichteten Graphen lautet wie folgt:

Gegeben: gerichteter Graph G mit Knoten s und t
Frage: Gibt es in G einen gerichteten Pfad von s nach t?

Beispiel: Erreichbarkeit

Das Problem der (**s-t-**)**Erreichbarkeit** in gerichteten Graphen lautet wie folgt:

Gegeben: gerichteter Graph G mit Knoten s und t
Frage: Gibt es in G einen gerichteten Pfad von s nach t?

Satz: Erreichbarkeit in gerichteten Graphen liegt in NL.

Beispiel: Erreichbarkeit

Das Problem der (**s-t-**)**Erreichbarkeit** in gerichteten Graphen lautet wie folgt:

Gegeben: gerichteter Graph G mit Knoten s und t
Frage: Gibt es in G einen gerichteten Pfad von s nach t?

Satz: Erreichbarkeit in gerichteten Graphen liegt in NL.

Beweis (Algorithmus):

- Wir verweden einen Pointer p auf einen Knoten (in der Eingabe) und einen Zähler z
- Initialisiere $*p = s$ und $z = 1$
- Schleife:
  - Falls $*p = t$ dann akzeptiere
  - Falls $z =$ Anzahl der Knoten in G dann verwerfe
  - Andernfalls: inkrementiere z und setze p auf einen Nachfolger des aktuellen Knotens $*p$ (nichtdet.)

□

NL-Vollständigkeit

Man kann NL-Schwere ähnlich wie für NP definieren:

- Statt polynomiellen Reduktionen verwendet man Logspace-Reduktionen
- NL-schwer: jedes Problem in NL ist darauf logspace-reduzierbar
- NL-vollständig: in NL und NL-schwer

Intuition: NL-vollständige Probleme sind die schwersten in NL

NL-Vollständigkeit

Man kann NL-Schwere ähnlich wie für NP definieren:

- Statt polynomiellen Reduktionen verwendet man Logspace-Reduktionen
- NL-schwer: jedes Problem in NL ist darauf logspace-reduzierbar
- NL-vollständig: in NL und NL-schwer

Intuition: NL-vollständige Probleme sind die schwersten in NL

Beispiel: Erreichbarkeit in gerichteten Graphen ist NL-vollständig.

NL-Vollständigkeit

Man kann NL-Schwere ähnlich wie für NP definieren:

- Statt polynomiellen Reduktionen verwendet man Logspace-Reduktionen
- NL-schwer: jedes Problem in NL ist darauf logspace-reduzierbar
- NL-vollständig: in NL und NL-schwer

Intuition: NL-vollständige Probleme sind die schwersten in NL

Beispiel: Erreichbarkeit in gerichteten Graphen ist NL-vollständig.

Beispiel: Erreichbarkeit in ungerichteten Graphen ist in NL aber (vermutlich) nicht NL-schwer: Das Problem liegt in L (Omer Reingold, 2005).

L, NL und coNL

Rückblick: Savitch sagte uns, dass NPSpace = PSpace, also auch NPSpace = coNPSpace

L, NL und coNL

Rückblick: Savitch sagte uns, dass NPSpace = PSpace, also auch NPSpace = coNPSpace

Für logarithmischen Speicher ergibt Savitchs Ergebnis aber lediglich:
NL $\subseteq$ DSpace$(\log^2 n) \rightsquigarrow$ daraus folgt nicht NL $\subseteq$ L!

L, NL und coNL

Rückblick: Savitch sagte uns, dass NPSpace = PSpace, also auch NPSpace = coNPSpace

Für logarithmischen Speicher ergibt Savitchs Ergebnis aber lediglich:
NL $\subseteq$ DSpace$(\log^2 n) \rightsquigarrow$ daraus folgt nicht NL $\subseteq$ L!

Man weiß dennoch:

Satz (Immerman 1987, Szelepcsényi 1987): NL = coNL.

L, NL und coNL

Rückblick: Savitch sagte uns, dass NPSpace = PSpace, also auch NPSpace = coNPSpace

Für logarithmischen Speicher ergibt Savitchs Ergebnis aber lediglich:
NL $\subseteq$ DSpace$(\log^2 n) \rightsquigarrow$ daraus folgt nicht NL $\subseteq$ L!

Man weiß dennoch:

Satz (Immerman 1987, Szelepcsényi 1987): NL = coNL.

Beispiel: Nichterreichbarkeit in gerichteten Graphen kann in NL entschieden werden. Betrachtet man den NL-Algorithmus für Erreichbarkeit, dann ist das zunächst überraschend ...

(Eng verwandtes Resultat: kontextsensitive Sprachen sind unter Komplement abgeschlossen)

PSpace

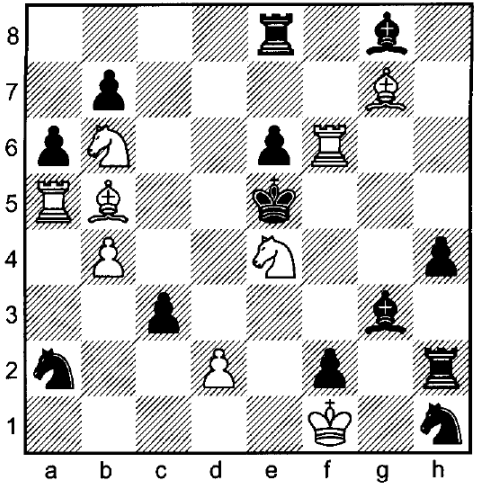
Noch schwerere Probleme?

Beobachtung: Bisher waren alle entscheidbaren schweren Probleme der Vorlesung auch in NP, d.h. ihre Lösung war leicht verifizierbar:

- Erfüllbarkeit, Hamiltonpfad, Clique, Rucksack: NP-vollständige Probleme mit polynomiellen Verifikatoren
- Faktorisierung: in NP $\cap$ coNP
- Erreichbarkeit in Graphen: in NP (Zertifikat ist Pfad); sogar in P (z.B. Breitensuche)

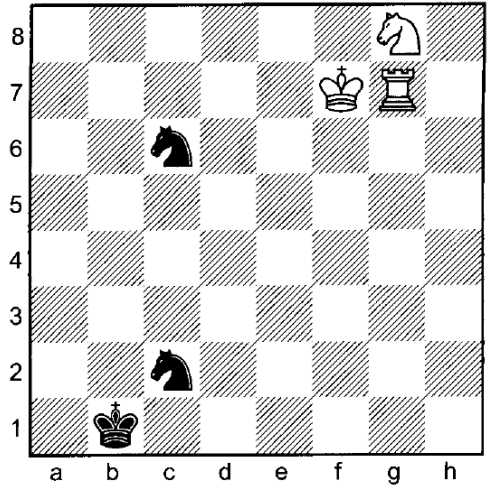
Gibt es überhaupt noch schwerere entscheidbare Probleme?

Beispiel: Schachrätsel



Matt in drei Zügen; Weiß ist am Zug

Beispiel: Schachrätsel



Matt in 262 Zügen; Weiß ist am Zug

Rückblick Aussagenlogik

- Aussagenlogische Formeln basieren auf Atomen (Propositionen, Variablen)
- Atome werden mit Junktoren verknüpft: $\neg$, $\wedge$, $\vee$, $\rightarrow$
  (wir setzen immer Klammern zwischen verschiedene binären Junktoren)
- Wir erlauben außerdem die nullstelligen Operatoren $\top$ (wahr) und $\bot$ (falsch)
- Belegungen ordnen Atomen Wahrheitswerte 1 oder 0 zu

Rückblick Aussagenlogik

Rückblick Aussagenlogik

- Aussagenlogische Formeln basieren auf Atomen (Propositionen, Variablen)
- Atome werden mit Junktoren verknüpft: $\neg$, $\wedge$, $\vee$, $\rightarrow$
  (wir setzen immer Klammern zwischen verschiedene binären Junktoren)
- Wir erlauben außerdem die nullstelligen Operatoren $\top$ (wahr) und $\bot$ (falsch)
- Belegungen ordnen Atomen Wahrheitswerte 1 oder 0 zu

- **SAT**: Gegeben eine aussagenlogische Formel φ, existiert eine Belegung der Atome in φ, für die φ wahr wird?

Rückblick Aussagenlogik

Rückblick Aussagenlogik

- Aussagenlogische Formeln basieren auf Atomen (Propositionen, Variablen)
- Atome werden mit Junktoren verknüpft: $\neg$, $\wedge$, $\vee$, $\rightarrow$
  (wir setzen immer Klammern zwischen verschiedene binären Junktoren)
- Wir erlauben außerdem die nullstelligen Operatoren $\top$ (wahr) und $\bot$ (falsch)
- Belegungen ordnen Atomen Wahrheitswerte 1 oder 0 zu

- **SAT**: Gegeben eine aussagenlogische Formel φ, existiert eine Belegung der Atome in φ, für die φ wahr wird?
- **Tautologie**: Gegeben eine aussagenlogische Formel φ, wird φ für alle Belegungen der Atome in φ wahr?

Rückblick Aussagenlogik

Rückblick Aussagenlogik

- Aussagenlogische Formeln basieren auf Atomen (Propositionen, Variablen)
- Atome werden mit Junktoren verknüpft: $\neg$, $\wedge$, $\vee$, $\rightarrow$
  (wir setzen immer Klammern zwischen verschiedene binären Junktoren)
- Wir erlauben außerdem die nullstelligen Operatoren $\top$ (wahr) und $\bot$ (falsch)
- Belegungen ordnen Atomen Wahrheitswerte 1 oder 0 zu

- **SAT**: Gegeben eine aussagenlogische Formel φ, existiert eine Belegung der Atome in φ, für die φ wahr wird?
- **Tautologie**: Gegeben eine aussagenlogische Formel φ, wird φ für alle Belegungen der Atome in φ wahr?

$\rightsquigarrow$ Existentielle und universelle Quantoren über Wahrheitswerte

Ein Problem in PSpace

Ein Beispiel für ein erstes typisches PSpace-Problem ergibt sich, wenn man **SAT** und **Tautologie** verallgemeinert:

Eine Quantifizierte Boolsche Formel (QBF) ist eine logische Formel der folgenden Form:

$$Q_1 p_1 . Q_2 p_2 . \cdots Q_\ell p_\ell . F[p_1, \ldots, p_\ell]$$

mit $i \geq 0$, $Q_i \in \{\exists, \forall\}$ Quantoren, p_i aussagenlogischen Atomen (Variablen) und F einer aussagenlogischen Formel mit Atomen $p_1, \ldots, p_\ell$.

Ein Problem in PSpace

Ein Beispiel für ein erstes typisches PSpace-Problem ergibt sich, wenn man **SAT** und **Tautologie** verallgemeinert:

Eine Quantifizierte Boolsche Formel (QBF) ist eine logische Formel der folgenden Form:

$$Q_1 p_1 . Q_2 p_2 . \cdots Q_\ell p_\ell . F[p_1, \ldots, p_\ell]$$

mit $i \geq 0$, $Q_i \in \{\exists, \forall\}$ Quantoren, p_i aussagenlogischen Atomen (Variablen) und F einer aussagenlogischen Formel mit Atomen $p_1, \ldots, p_\ell$.

Beispiele:

- $\forall p . \exists q . (p \to q) \wedge (q \to p)$
- $\forall p_1, p_2, p_3 . \exists q . (p_1 \vee p_2 \vee p_3) \to ((p_1 \vee q) \wedge (\neg q \vee p_2 \vee p_3))$

Anmerkung: wir sparen uns die äußerste Klammer sowie Klammern in Ketten von $\wedge$ und $\vee$

Semantik von QBF

Jeder QBF-Formel Q wird ein eindeutiger Wahrheitswert $W(Q)$ zugeordnet:

- QBF-Formeln ohne Atome (d.h. nur mit $\top$ und $\bot$) werden wie aussagenlogische Formeln evaluiert
- $W(\exists p.F[p]) = 1$ falls $W(F[p/\top]) = 1$ oder $W(F[p/\bot]) = 1$
- $W(\forall p.F[p]) = 1$ falls $W(F[p/\top]) = 1$ und $W(F[p/\bot]) = 1$

  Dabei heißt $\varphi[p/\top]$: „φ mit p ersetzt durch $\top$“; analog für $\bot$.

Semantik von QBF

Jeder QBF-Formel Q wird ein eindeutiger Wahrheitswert $W(Q)$ zugeordnet:

- QBF-Formeln ohne Atome (d.h. nur mit $\top$ und $\bot$) werden wie aussagenlogische Formeln evaluiert
- $W(\exists p.F[p]) = 1$ falls $W(F[p/\top]) = 1$ oder $W(F[p/\bot]) = 1$
- $W(\forall p.F[p]) = 1$ falls $W(F[p/\top]) = 1$ und $W(F[p/\bot]) = 1$

  Dabei heißt $\varphi[p/\top]$: „φ mit p ersetzt durch $\top$“; analog für $\bot$.

Beispiel:

$W(\forall p.\exists q.(p \to q) \wedge (q \to p)) = 1$

gdw. $W(\exists q.(\top \to q) \wedge (q \to \top)) = 1$ und

$W(\exists q.(\bot \to q) \wedge (q \to \bot)) = 1$

gdw. $W((\top \to \top) \wedge (\top \to \top)) = 1$ oder $W((\top \to \bot) \wedge (\bot \to \top)) = 1$ und

$W((\bot \to \top) \wedge (\top \to \bot)) = 1$ oder $W((\bot \to \bot) \wedge (\bot \to \bot)) = 1$

Wahre QBF erkennen

Durch die Quantoren steht der Wahrheitswert jeder QBF fest, d.h. er hängt nicht von Belegungen ab.

Das Problem **TrueQBF** ist wie folgt

Gegeben: eine QBF Q
Frage: Ist $W(Q) = 1$?

Wahre QBF erkennen

Durch die Quantoren steht der Wahrheitswert jeder QBF fest, d.h. er hängt nicht von Belegungen ab.

Das Problem **TrueQBF** ist wie folgt

Gegeben: eine QBF Q
Frage: Ist $W(Q) = 1$?

Beispiel: SAT lässt sich auf **TrueQBF** reduzieren, indem man jedes Atom der gegebenen aussagenlogischen Formel existentiell quantifiziert.

Wahre QBF erkennen

Durch die Quantoren steht der Wahrheitswert jeder QBF fest, d.h. er hängt nicht von Belegungen ab.

Das Problem **TrueQBF** ist wie folgt

Gegeben: eine QBF Q
Frage: Ist $W(Q) = 1$?

Beispiel: SAT lässt sich auf **TrueQBF** reduzieren, indem man jedes Atom der gegebenen aussagenlogischen Formel existentiell quantifiziert.

Beispiel: Tautologie lässt sich auf **TrueQBF** reduzieren, indem man jedes Atom der gegebenen aussagenlogischen Formel universell quantifiziert.

TrueQBF in polynomiellem Speicher

Satz: TrueQBF ist in PSpace.

TrueQBF in polynomiellem Speicher

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 $\mathbf{P} \leq_p \mathbf{Q}$ existiert. **Q** ist PSpace-vollständig wenn es PSpace-schwer ist und in PSpace liegt.

PSpace-Schwere

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

Satz: TrueQBF ist PSpace-schwer.

PSpace-Schwere

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

Satz: TrueQBF ist PSpace-schwer.

Beweisidee: siehe nächste Vorlesung

QBF als Spiel

Man kann **TrueQBF** als Spiel auffassen:

- Das „Spielbrett“ ist eine QBF
- Zwei Spieler, Anton und Emilia, wählen der Reihe nach Wahrheitswerte
- Steht $\forall p$ vorn, so darf Anton einen Wert für p wählen und den Quantor löschen
- Steht $\exists p$ vorn, so darf Emilia einen Wert für p wählen und den Quantor löschen
- Emilia gewinnt, wenn die Formel nach Entfernen aller Quantoren wahr wird

QBF als Spiel

Man kann **TrueQBF** als Spiel auffassen:

- Das „Spielbrett“ ist eine QBF
- Zwei Spieler, Anton und Emilia, wählen der Reihe nach Wahrheitswerte
- Steht $\forall p$ vorn, so darf Anton einen Wert für p wählen und den Quantor löschen
- Steht $\exists p$ vorn, so darf Emilia einen Wert für p wählen und den Quantor löschen
- Emilia gewinnt, wenn die Formel nach Entfernen aller Quantoren wahr wird

Beobachtung: Emilia hat eine Gewinnstrategie im Formelspiel genau dann wenn die gegebene QBF wahr ist.

Beispiel: Sipsers 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

Beispiel: Sipsers 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

Beispiel: Sipsers 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

Satz: Geography ist PSpace-vollständig

Geography ist PSpace-vollständig

Satz: Geography ist PSpace-vollständig

Beweis: nächste Vorlesung

Und was ist mit Schach?

Und was ist mit Schach?

Schach selbst ist endlich:

- endlich viele mögliche Stellungen
- in jeder hat Weiß eine Gewinnstrategie oder nicht

$\rightsquigarrow$ Problem in $O(1)$

Und was ist mit Schach?

Schach selbst ist endlich:

- endlich viele mögliche Stellungen
- in jeder hat Weiß eine Gewinnstrategie oder nicht

$\rightsquigarrow$ Problem in $O(1)$

Verallgemeinertes Schach:

- Beliebig großes Spielbrett
- Beliebig viele Figuren

$\rightsquigarrow$ ExpTime-vollständig (d.h. vermutlich nicht in PSpace)

Und was ist mit Schach?

Schach selbst ist endlich:

- endlich viele mögliche Stellungen
- in jeder hat Weiß eine Gewinnstrategie oder nicht

$\rightsquigarrow$ Problem in $O(1)$

Verallgemeinertes Schach:

- Beliebig großes Spielbrett
- Beliebig viele Figuren

$\rightsquigarrow$ ExpTime-vollständig (d.h. vermutlich nicht in PSpace)

Intuition: Schach ist schwerer als typische PSpace-Spiele, da man Züge rückgängig machen kann

$\rightsquigarrow$ Spiel kann mehr als polynomiell viele Züge dauern

Zusammenfassung und Ausblick

Erreichbarkeit in gerichteten Graphen ist das typische NL-vollständige Problem

Es gibt schwere Probleme, die keine leicht zu prüfende Lösung haben

Quantifizierte Boolesche Formeln verallgmeinern Aussagenlogik

PSpace ist die Klasse der interessanten Zwei-Spieler-Spiele, die nicht zu lange dauern

Was erwartet uns als nächstes?

- Alternierung
- noch mehr Logik