TECHNISCHE UNIVERSITÄT DRESDEN

Fakultät Informatik, Institut für Künstliche Intelligenz, Professur Computational Logic

THEORETISCHE INFORMATIK UND LOGIK

11. Vorlesung: NL und PSpace

Hannes Straß

TU Dresden, 16. Mai 2022

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:

- **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 Zeiger und Zwischenergebnisse).
- Linearer Speicher genügt zur Erkennung kontextsensitiver Sprachen (durch linear beschränkte Automaten, LBA).
- Jedes NP-vollständige Problem ist in polynomiellem 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
- Zeiger aufs (Nur-Lese-)Eingabeband

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

Wesentliche Programmierfeatures:

- Initialisiere Zeiger oder Zähler auf festen Wert;
- inkrementiere/dekrementiere Zeiger oder Zähler;
- vergleiche Speicherinhalte von zwei Zeigern oder zwei Zählern (und führe je nach Ergebnis anderen Code aus).

Optionales Ausgabeband: Jede Zelle ist einmalig beschreibbar und nicht (wieder) lesbar.

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 Zeiger auf eine zufällige Eingabeposition“

Beispiel: Erreichbarkeit

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

Gegeben: Ein 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 verwenden einen Zeiger 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 verwirf;
  - andernfalls: Inkrementiere z und setze p auf einen Nachfolger des aktuellen Knotens $*p$ (nichtdeterministisch). □

NL-Vollständigkeit

Man kann NL-Schwere ähnlich wie für NP definieren:

- An Stelle polynomieller 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: Der Satz von Savitch besagt, dass NPSpace = PSpace.
Daraus folgt auch NPSpace = coNPSpace.

Für logarithmischen Speicher ergibt Savitchs Ergebnis aber lediglich:[1]

$$\mathsf{NL} \subseteq \mathsf{DSpace}\left(\log^2 n\right)$$

$\rightsquigarrow$ 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.)

[1] Notation: $\log^2 n = (\log n)^2 \neq \log\left(n^2\right) = 2 \log n$.

Quiz: Erreichbarkeit

Quiz: Gegeben sei der ungerichtete Graph $G = (V, E)$ mit ...

PSpace

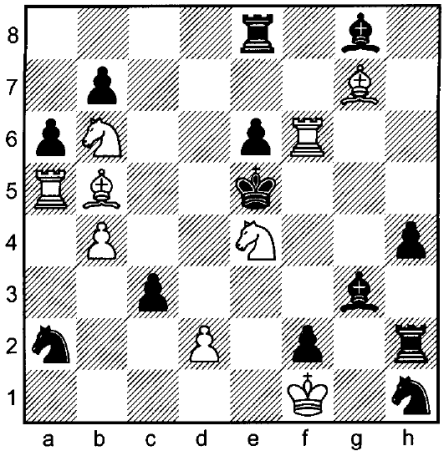
Noch schwerere Probleme?

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 ∩ coNP
- Erreichbarkeit in Graphen: in NP (Zertifikat ist Pfad); sogar in P (z.B. Breitensuche)

Gibt es überhaupt noch schwerere entscheidbare Probleme?

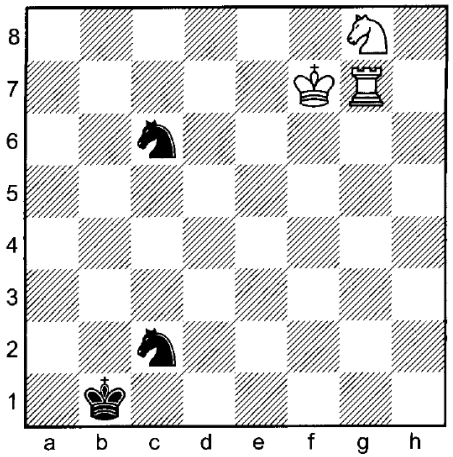
Beispiel: Schachrätsel



Matt in drei Zügen; Weiß ist am Zug

(Samuel Loyd, 1903)

Beispiel: Schachrätsel



Matt in 262 Zügen; Weiß ist am Zug

(Lewis Stiller, 1995)

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 binäre 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?

$\rightsquigarrow$ (Implizite) existenzielle und universelle Quantoren über Wahrheitswerten.

Ein Problem in PSpace

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

Eine Quantifizierte Boolesche 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 $\vee$, und fassen gleiche Quantoren zusammen.

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 $W(Q) = 1$?

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

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

Quiz: **TrueQBF**

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 $\bot$.

Quiz: Welche der folgenden Quantifizierten Booleschen Formeln sind wahr? . . .

TrueQBF in polynomiellem Speicher

Satz: **TrueQBF** ist in PSpace.

Beweis: Durch Angabe eines (Pseudo-)Algorithmus:

```
01 function 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 ist möglich in PSpace.
- Rekursionen in Zeilen 05 und 07 können der Reihe nach abgearbeitet werden, wobei Speicher wiederverwendet wird.
- Jeder Rekursionsschritt benötigt polynomiellen Speicher.
- Maximale Rekursionstiefe ist die Anzahl der Atome (also linear in der Eingabe). □

PSpace-Schwere

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

Satz: **TrueQBF** ist PSpace-schwer.

Beweisidee: Nächste Vorlesung.

QBF als Spiel

Man kann **TrueQBF** als Spiel auffassen:

- Das „Spielbrett“ ist eine QBF.
- Zwei Personen, 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;
- andernfalls gewinnt Anton.

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

Beispiel: Sipsers Geography

Ein Kinderspiel:

- Zwei Personen benennen abwechselnd Städte.
- Jede Stadt muss mit dem letzten Buchstaben der zuvor genannten beginnen.
- Wiederholungen sind verboten.
- Die erste Person, die keine Stadt mehr nennen kann, verliert.

Ein Mathematikerspiel:

- Zwei Personen markieren Knoten in einem gerichteten Graphen.
- Jeder Knoten muss ein Nachfolger des vorigen sein.
- Wiederholungen sind verboten.
- Die erste Person, die keinen Knoten markieren kann, verliert.

Entscheidungsproblem **Geography**:
Gegeben: Ein gerichteter Graph und ein Startknoten.
Frage: Hat die beginnende Person eine Gewinnstrategie für dieses Spiel?

Geography ist PSpace-vollständig

Satz: Geography ist PSpace-vollständig.

Beweis: Nächste Vorlesung.

Und was ist mit Schach?

Schach selbst ist endlich:

- Es gibt nur 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.
$\rightsquigarrow$ Eine Partie 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 verallgemeinern Aussagenlogik.

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

Was erwartet uns als nächstes?

- Alternierung
- Noch mehr Logik