TECHNISCHE UNIVERSITÄT DRESDEN

Fakultät Informatik, Institut für Künstliche Intelligenz, Professur Computational Logic

THEORETISCHE INFORMATIK UND LOGIK

21. Vorlesung: Endliche Interpretationen und Datenbanken

Sebastian Rudolph

TU Dresden, 3. Juli 2025

Rückblick: Kompaktheit

Die Existenz von vollständigen und korrekten logischen Schließverfahren wie Resolution ist eng verwandt mit einer grundsätzlichen Eigenschaft der Prädikatenlogik:

Satz (Endlichkeitssatz, Kompaktheitssatz):
Eine unendliche Menge $\mathcal{T}$ prädikatenlogischer Sätze ist genau dann erfüllbar, wenn jede endliche Teilmenge von $\mathcal{T}$ erfüllbar ist.

Rückblick: Kompaktheit

Die Existenz von vollständigen und korrekten logischen Schließverfahren wie Resolution ist eng verwandt mit einer grundsätzlichen Eigenschaft der Prädikatenlogik:

Satz (Endlichkeitssatz, Kompaktheitssatz):
Eine unendliche Menge $\mathcal{T}$ prädikatenlogischer Sätze ist genau dann erfüllbar, wenn jede endliche Teilmenge von $\mathcal{T}$ erfüllbar ist.

Beweis: Wir zeigen: $\mathcal{T}$ ist genau dann unerfüllbar, wenn es eine endliche Teilmenge von $\mathcal{T}$ gibt, die unerfüllbar ist.

Laut Resolutionssatz (Vollständigkeit) kann die Unerfüllbarkeit von $\mathcal{T}$ nach endlich vielen Schritten durch Ableitung der leeren Klausel nachgewiesen werden.

Dabei können nur endlich viele Klauseln aus der Klauselform von $\mathcal{T}$ verwendet worden sein.

Laut Resolutionssatz (Korrektheit) folgt die Existenz einer unerfüllbaren endlichen Teilmenge von $\mathcal{T}$. □

Die Grenzen der Prädikatenlogik

Kompaktheit zeigt uns auch erste Grenzen der Prädikatenlogik auf.

Eine logische Formel F mit zwei freien Variablen x und y drückt genau dann den transitiven Abschluss einer binären Relation r aus, wenn in jeder Interpretation $\mathcal{I} = \langle \Delta^{\mathcal{I}}, \cdot^{\mathcal{I}} \rangle$ und für alle $\delta_1, \delta_2 \in \Delta^{\mathcal{I}}$ gilt:

$$\mathcal{I}, \{x \mapsto \delta_1, y \mapsto \delta_2\} \models F \quad \text{gdw.} \quad \langle \delta_1, \delta_2 \rangle \in \left(r^{\mathcal{I}}\right)^+$$

Die Grenzen der Prädikatenlogik

Kompaktheit zeigt uns auch erste Grenzen der Prädikatenlogik auf.

Eine logische Formel F mit zwei freien Variablen x und y drückt genau dann den transitiven Abschluss einer binären Relation r aus, wenn in jeder Interpretation $\mathcal{I} = \langle \Delta^{\mathcal{I}}, \cdot^{\mathcal{I}} \rangle$ und für alle $\delta_1, \delta_2 \in \Delta^{\mathcal{I}}$ gilt:

$$\mathcal{I}, \{x \mapsto \delta_1, y \mapsto \delta_2\} \models F \quad \text{gdw.} \quad \langle \delta_1, \delta_2 \rangle \in \left(r^{\mathcal{I}}\right)^+$$

Satz: Es gibt keine prädikatenlogische Formel, die den transitiven Abschluss einer binären Relation ausdrückt.

Die Grenzen der Prädikatenlogik

Kompaktheit zeigt uns auch erste Grenzen der Prädikatenlogik auf.

Eine logische Formel F mit zwei freien Variablen x und y drückt genau dann den transitiven Abschluss einer binären Relation r aus, wenn in jeder Interpretation $\mathcal{I} = \langle \Delta^{\mathcal{I}}, \cdot^{\mathcal{I}} \rangle$ und für alle $\delta_1, \delta_2 \in \Delta^{\mathcal{I}}$ gilt:

$$\mathcal{I}, \{x \mapsto \delta_1, y \mapsto \delta_2\} \models F \quad \text{gdw.} \quad \langle \delta_1, \delta_2 \rangle \in \left(r^{\mathcal{I}}\right)^+$$

Satz: Es gibt keine prädikatenlogische Formel, die den transitiven Abschluss einer binären Relation ausdrückt.

Beweis: Angenommen, es gäbe so eine Formel F.

Dann ist die folgende unendliche Theorie unerfüllbar:

$$\{ \quad F\{x \mapsto a, y \mapsto b\}, \quad \neg r(a,b), \quad \neg\exists x_1.(r(a,x_1) \wedge r(x_1,b)),$$
$$\neg\exists x_1, x_2.(r(a,x_1) \wedge r(x_1,x_2) \wedge r(x_2,b)), \quad \ldots \qquad \}$$

Aber jede endliche Teilmenge der Theorie ist erfüllbar. Die Existenz der Formel F würde also dem Kompaktheitssatz widersprechen. □

Endliche Interpretationen

Löwenheim und Skolem

Eine Interpretation $\mathcal{I} = \langle \Delta^{\mathcal{I}}, \cdot^{\mathcal{I}} \rangle$ heißt genau dann endlich (abzählbar), wenn ihre Domäne $\Delta^{\mathcal{I}}$ endlich (abzählbar) ist.

Satz (Löwenheim[1] und Skolem):
Jede erfüllbare prädikatenlogische Formel hat ein abzählbares Modell.

[1] Leopold Löwenheim[2] (1878–1957): deutscher Logiker; 1915 erster Beweis des obigen Satzes; 1916 Soldat im 1. Weltkrieg; 1934 von den Nazis zwangspensioniert und als Privatlehrer tätig.

Löwenheim und Skolem

Eine Interpretation $\mathcal{I} = \langle \Delta^{\mathcal{I}}, \cdot^{\mathcal{I}} \rangle$ heißt genau dann endlich (abzählbar), wenn ihre Domäne $\Delta^{\mathcal{I}}$ endlich (abzählbar) ist.

Satz (Löwenheim[1] und Skolem):
Jede erfüllbare prädikatenlogische Formel hat ein abzählbares Modell.

[1] Leopold Löwenheim[2] (1878–1957): deutscher Logiker; 1915 erster Beweis des obigen Satzes; 1916 Soldat im 1. Weltkrieg; 1934 von den Nazis zwangspensioniert und als Privatlehrer tätig.
[2] „Note: One of the best predictors of success in mathematical logic is having an umlaut in your name.“ – S. Aaronson, Quantum Computing since Democritus. Cambridge, 2013.

Löwenheim und Skolem

Eine Interpretation $\mathcal{I} = \langle \Delta^{\mathcal{I}}, \cdot^{\mathcal{I}} \rangle$ heißt genau dann endlich (abzählbar), wenn ihre Domäne $\Delta^{\mathcal{I}}$ endlich (abzählbar) ist.

Satz (Löwenheim[1] und Skolem):
Jede erfüllbare prädikatenlogische Formel hat ein abzählbares Modell.

Beweis:

- Jede Formel F kann in erfüllbarkeitsäquivalente Skolemform F' überführt werden.
- Ist F' erfüllbar, so hat F' ein Herbrand-Modell. (VL 19)
- Jede Herbrand-Interpretation hat immer eine abzählbare Domäne (das Herbrand-Universum).
- Man kann aus den Beweisen der Erfüllbarkeitsäquivalenz erkennen: Jedes Modell von F' ist auch ein Modell für F.

□

[1] Leopold Löwenheim[2] (1878–1957): deutscher Logiker; 1915 erster Beweis des obigen Satzes; 1916 Soldat im 1. Weltkrieg; 1934 von den Nazis zwangspensioniert und als Privatlehrer tätig.
[2] „Note: One of the best predictors of success in mathematical logic is having an umlaut in your name.“ – S. Aaronson, Quantum Computing since Democritus. Cambridge, 2013.

Historische Anmerkungen

- Der eben gezeigte „Satz von Löwenheim und Skolem“ ist eigentlich der nach unten gerichtete Satz von Löwenheim und Skolem (engl. Downward Löwenheim-Skolem Theorem).
- Der nach oben gerichtete Satz von Löwenheim und Skolem (engl. Upward Löwenheim-Skolem Theorem) besagt, dass jede erfüllbare Formel auch Modelle beliebig großer Kardinalität besitzt.

Historische Anmerkungen

- Der eben gezeigte „Satz von Löwenheim und Skolem“ ist eigentlich der nach unten gerichtete Satz von Löwenheim und Skolem (engl. Downward Löwenheim-Skolem Theorem).
- Der nach oben gerichtete Satz von Löwenheim und Skolem (engl. Upward Löwenheim-Skolem Theorem) besagt, dass jede erfüllbare Formel auch Modelle beliebig großer Kardinalität besitzt.
- Der Legende nach war Skolem die Verwendung seines Namens in diesem Kontext unangenehm:

  „I follow custom in calling Corollary 6.1.4 the upward Löwenheim-Skolem theorem. But in fact Skolem didn’t even believe it, because he didn’t believe in the existence of uncountable sets.“
  – W. Hodges, Model Theory, Cambridge 1993

Endlichkeit von Modellen

Löwenheim/Skolem: Jede erfüllbare Formel hat ein abzählbar großes Modell.

Kann man dies noch verstärken? Hat jede erfüllbare Formel vielleicht sogar ein endliches Modell?

Endlichkeit von Modellen

Löwenheim/Skolem: Jede erfüllbare Formel hat ein abzählbar großes Modell.

Kann man dies noch verstärken? Hat jede erfüllbare Formel vielleicht sogar ein endliches Modell?

Nein. Prädikatenlogik kann unendliche Modelle erzwingen:

Beispiel:

$$\forall x.(\text{Mensch}(x) \rightarrow \exists y.(\text{hatMutter}(x,y) \wedge \text{Mensch}(y)))$$
$$\forall x,y.(\text{hatMutter}(x,y) \rightarrow \text{hatVorfahr}(x,y))$$
$$\forall x,y,z.((\text{hatVorfahr}(x,y) \wedge \text{hatVorfahr}(y,z)) \rightarrow \text{hatVorfahr}(x,z))$$
$$\forall x.\neg\text{hatVorfahr}(x,x)$$

Diese Theorie ist erfüllbar, aber hat nur unendliche Modelle. (Kontrollfrage: Warum?)

Logik über endlichen Interpretationen

Sind unendliche Interpretationen in der Praxis überhaupt wünschenswert?
Geht es auch endlich?

Logik über endlichen Interpretationen

Sind unendliche Interpretationen in der Praxis überhaupt wünschenswert?
Geht es auch endlich?

Prädikatenlogik mit endlichen Interpretationen verwendet die gleiche Syntax und Semantik wie Prädikatenlogik allgemein, aber mit der zusätzlichen Bedingung, dass die Domäne von Interpretationen endlich sein muss.

Logik über endlichen Interpretationen

Sind unendliche Interpretationen in der Praxis überhaupt wünschenswert?
Geht es auch endlich?

Prädikatenlogik mit endlichen Interpretationen verwendet die gleiche Syntax und Semantik wie Prädikatenlogik allgemein, aber mit der zusätzlichen Bedingung, dass die Domäne von Interpretationen endlich sein muss.

Anmerkung: Wir nutzen auch den Begriff Prädikatenlogik mit endlichen Modellen, da die Endlichkeit jeder Interpretation die Endlichkeit jedes Modells impliziert.
Monotonie (Rückblick): Weniger Modelle führen zu mehr Konsequenzen.

Beispiel:

$$\forall x.(\text{Mensch}(x) \rightarrow \exists y.(\text{hatVorfahr}(x,y) \wedge \text{Mensch}(y)))$$

$$\forall x,y,z.((\text{hatVorfahr}(x,y) \wedge \text{hatVorfahr}(y,z)) \rightarrow \text{hatVorfahr}(x,z))$$

Diese Theorie ist in der Prädikatenlogik mit endlichen Interpretationen erfüllbar, aber jedes endliche Modell muss einen hatVorfahr-Zyklus enthalten.
Daher folgt aus der Theorie $\exists x.\text{hatVorfahr}(x,x)$, obwohl dies in der allgemeinen Prädikatenlogik keine Konsequenz wäre.

Erfüllbarkeit wird semi-entscheidbar

Die Bezeichnung der Elemente einer Interpretationsdomäne ist irrelevant – für die Wahrheit von Sätzen kommt es nur darauf an, wie Konstanten und Prädikatensymbole interpretiert werden.

Erfüllbarkeit wird semi-entscheidbar

Die Bezeichnung der Elemente einer Interpretationsdomäne ist irrelevant – für die Wahrheit von Sätzen kommt es nur darauf an, wie Konstanten und Prädikatensymbole interpretiert werden.

Erfüllbarkeitstest

Gegeben: Ein Satz F.

- Betrachte systematisch alle endlichen Interpretationen der Symbole in F (z.B. geordnet nach aufsteigender Größe der Domäne).
- Prüfe für jede Interpretation $\mathcal{I}$, ob $\mathcal{I} \models F$ gilt:
  - Falls ja, dann gib aus „erfüllbar“;
  - falls nein, dann fahre mit der nächsten Interpretation fort.

Es ist leicht zu sehen, dass dieser Algorithmus die Erfüllbarkeit in endlichen Interpretationen semi-entscheidet.

Endlich = einfach?

Endlich = einfach?

Trotzdem bleibt logisches Schließen schwer:

Satz von Trakhtenbrot: Logisches Schließen (Erfüllbarkeit, Allgemeingültigkeit, logische Konsequenz) in der Prädikatenlogik mit endlichen Modellen ist unentscheidbar.

(Ohne Beweis; mehr dazu in der Vorlesung Database Theory.)

Endlich = einfach?

Trotzdem bleibt logisches Schließen schwer:

Satz von Trakhtenbrot: Logisches Schließen (Erfüllbarkeit, Allgemeingültigkeit, logische Konsequenz) in der Prädikatenlogik mit endlichen Modellen ist unentscheidbar.

(Ohne Beweis; mehr dazu in der Vorlesung Database Theory.)

Korollar: Es gibt kein korrektes und vollständiges Beweissystem für Prädikatenlogik mit endlichen Modellen.

Beweis: Angenommen, es gäbe ein solches System.

Endlich = einfach?

Trotzdem bleibt logisches Schließen schwer:

Satz von Trakhtenbrot: Logisches Schließen (Erfüllbarkeit, Allgemeingültigkeit, logische Konsequenz) in der Prädikatenlogik mit endlichen Modellen ist unentscheidbar.

(Ohne Beweis; mehr dazu in der Vorlesung Database Theory.)

Korollar: Es gibt kein korrektes und vollständiges Beweissystem für Prädikatenlogik mit endlichen Modellen.

Beweis: Angenommen, es gäbe ein solches System.

- Dann wäre logische Konsequenz und speziell auch Unerfüllbarkeit semi-entscheidbar.

Endlich = einfach?

Trotzdem bleibt logisches Schließen schwer:

Satz von Trakhtenbrot: Logisches Schließen (Erfüllbarkeit, Allgemeingültigkeit, logische Konsequenz) in der Prädikatenlogik mit endlichen Modellen ist unentscheidbar.

(Ohne Beweis; mehr dazu in der Vorlesung Database Theory.)

Korollar: Es gibt kein korrektes und vollständiges Beweissystem für Prädikatenlogik mit endlichen Modellen.

Beweis: Angenommen, es gäbe ein solches System.

- Dann wäre logische Konsequenz und speziell auch Unerfüllbarkeit semi-entscheidbar.
- Wir wissen, dass Erfüllbarkeit ebenfalls semi-entscheidbar ist.

Endlich = einfach?

Trotzdem bleibt logisches Schließen schwer:

Satz von Trakhtenbrot: Logisches Schließen (Erfüllbarkeit, Allgemeingültigkeit, logische Konsequenz) in der Prädikatenlogik mit endlichen Modellen ist unentscheidbar.

(Ohne Beweis; mehr dazu in der Vorlesung Database Theory.)

Korollar: Es gibt kein korrektes und vollständiges Beweissystem für Prädikatenlogik mit endlichen Modellen.

Beweis: Angenommen, es gäbe ein solches System.

- Dann wäre logische Konsequenz und speziell auch Unerfüllbarkeit semi-entscheidbar.
- Wir wissen, dass Erfüllbarkeit ebenfalls semi-entscheidbar ist.
- Zusammen ergäbe sich also ein Entscheidungsverfahren für logisches Schließen.

Endlich = einfach?

Trotzdem bleibt logisches Schließen schwer:

Satz von Trakhtenbrot: Logisches Schließen (Erfüllbarkeit, Allgemeingültigkeit, logische Konsequenz) in der Prädikatenlogik mit endlichen Modellen ist unentscheidbar.

(Ohne Beweis; mehr dazu in der Vorlesung Database Theory.)

Korollar: Es gibt kein korrektes und vollständiges Beweissystem für Prädikatenlogik mit endlichen Modellen.

Beweis: Angenommen, es gäbe ein solches System.

- Dann wäre logische Konsequenz und speziell auch Unerfüllbarkeit semi-entscheidbar.
- Wir wissen, dass Erfüllbarkeit ebenfalls semi-entscheidbar ist.
- Zusammen ergäbe sich also ein Entscheidungsverfahren für logisches Schließen.
- Widerspruch zum Satz von Trakhtenbrot. □

Endliche Interpretationen in der Praxis

Wozu endliche Interpretationen?

In gewisser Weise ist Schließen mit endlichen Interpretationen also schwerer als mit unendlichen, weil man statt logischer Konsequenz nunmehr nur Nicht-Konsequenz semi-entscheiden kann.

Trotzdem sind endliche Interpretationen in der Informatik praktisch relevant:

Eine endliche Interpretation $\mathcal{I}$ ist (im Wesentlichen) das gleiche wie eine relationale Datenbankinstanz.

Intuition:

- Prädikatensymbole p bezeichnen Tabellen.
- Relationen $p^{\mathcal{I}}$ entsprechen den in der Datenbank gespeicherten Tabelleninhalten.

Benannte Parameter

Relationale Datenbanken verwenden Namen für die Parameter (Spalten) in Relationen, anstatt sie mittels Reihenfolge zu adressieren:

linien:

Linie	Typ
85	Bus
3	Tram
F1	Fähre
...	...

haltestellen:

SID	Name	Rollstuhl
17	Hauptbahnhof	true
42	Helmholtzstr.	true
57	Stadtgutstr.	true
123	Gustav-Freytag-Str.	false
...	...	...

verbindung:

Von	Zu	Linie
57	42	85
17	789	3
...	...	...

Die einfache Arität der Prädikatenlogik wird durch ein Schema mit Namen (und oft auch Datentypen) ersetzt:

- linien[Linie:string, Typ:string]
- haltestellen[SID:int, Halt:string, Rollstuhl:bool]
- verbindung[Von:int, Zu:int, Linie:string]

Benannte Parameter

Relationale Datenbanken verwenden Namen für die Parameter (Spalten) in Relationen, anstatt sie mittels Reihenfolge zu adressieren:

linien:

Linie	Typ
85	Bus
3	Tram
F1	Fähre
...	...

haltestellen:

SID	Name	Rollstuhl
17	Hauptbahnhof	true
42	Helmholtzstr.	true
57	Stadtgutstr.	true
123	Gustav-Freytag-Str.	false
...	...	...

verbindung:

Von	Zu	Linie
57	42	85
17	789	3
...	...	...

Die einfache Arität der Prädikatenlogik wird durch ein Schema mit Namen (und oft auch Datentypen) ersetzt:

- linien[Linie:string, Typ:string]
- haltestellen[SID:int, Halt:string, Rollstuhl:bool]
- verbindung[Von:int, Zu:int, Linie:string]

Mögliche Anfrage (SQL):

```
SELECT verbindung.Von, verbindung.Zu, linien.Typ
FROM linien JOIN verbindung ON linien.Linie=verbindung.Linie
WHERE ...
```

Formeln = Anfragen

Benannt oder nicht – sofern die Parameter eine definierte Reihenfolge haben, können wir sie mit prädikatenlogischen Atomen addressieren.

Beispiel: Die Formel

$$Q = \exists z_{\text{Linie}}.(\text{verbindung}(x_{\text{Von}}, x_{\text{Zu}}, z_{\text{Linie}}) \wedge \text{linien}(z_{\text{Linie}}, x_{\text{Typ}}))$$

hat drei freie Variablen.
Für eine gegebene Datenbankinstanz (endliche Interpretation) $\mathcal{I}$ bedeutet

$$\mathcal{I}, \{x_{\text{Von}} \mapsto \delta_1, x_{\text{Zu}} \mapsto \delta_2, x_{\text{Typ}} \mapsto \delta_3\} \models Q$$

dass es in der Datenbank eine Verbindung von δ_1 nach δ_2 vom Typ δ_3 gibt.

Formeln = Anfragen

Benannt oder nicht – sofern die Parameter eine definierte Reihenfolge haben, können wir sie mit prädikatenlogischen Atomen addressieren.

Beispiel: Die Formel

$$Q = \exists z_{\text{Linie}}.(\text{verbindung}(x_{\text{Von}}, x_{\text{Zu}}, z_{\text{Linie}}) \wedge \text{linien}(z_{\text{Linie}}, x_{\text{Typ}}))$$

hat drei freie Variablen.
Für eine gegebene Datenbankinstanz (endliche Interpretation) $\mathcal{I}$ bedeutet

$$\mathcal{I}, \{x_{\text{Von}} \mapsto \delta_1, x_{\text{Zu}} \mapsto \delta_2, x_{\text{Typ}} \mapsto \delta_3\} \models Q$$

dass es in der Datenbank eine Verbindung von δ_1 nach δ_2 vom Typ δ_3 gibt.

Das Beispiel illustriert:

Formeln (evtl. mit freien Variablen) $\hat{=}$ Datenbankanfragen

Erfüllende Zuweisungen $\hat{=}$ Anfrage-Ergebnisse

Logik und Datenbanken

Relationale Datenbankinstanzen $\hat{=}$ Endliche Interpretationen

- Tabellen(namen) entsprechen Prädikatensymbolen
- Kleinere syntaktische Unterschiede (benannte vs. geordnete Parameter)

Relationale Datenbankanfragen $\hat{=}$ Prädikatenlogische Formeln

- Zuweisungen $\mathcal{Z}$ zu freien Variablen modellieren mögliche Anfrageergebnisse
- $\mathcal{I}, \mathcal{Z} \models Q$ bedeutet: $\mathcal{Z}$ ist Ergebnis der Anfrage Q auf Datenbankinstanz $\mathcal{I}$

Prädikatenlogik $\approx$ SQL

Was ist eine Datenbank-Anfrage?

- Syntax: Eine Anfrage Q ist ein Wort aus einer Anfragesprache.
- Semantik: Jede Anfrage Q definiert eine Anfragefunktion f_Q, die für jede Datenbankinstanz $\mathcal{I}$ eine Ergebnisrelation $f_Q(\mathcal{I})$ liefert.

Prädikatenlogik ≈ SQL

Was ist eine Datenbank-Anfrage?

- Syntax: Eine Anfrage Q ist ein Wort aus einer Anfragesprache.
- Semantik: Jede Anfrage Q definiert eine Anfragefunktion f_Q, die für jede Datenbankinstanz $\mathcal{I}$ eine Ergebnisrelation $f_Q(\mathcal{I})$ liefert.

Beispiel: Für eine prädikatenlogische Formel Q mit freien Variablen $x_1, \ldots, x_n$ ist f_Q diejenige Funktion, die eine Interpretation $\mathcal{I}$ wie folgt auf die Ergebnisrelation $f_Q(\mathcal{I})$ abbildet:

$$\mathcal{I} \mapsto \{\langle \delta_1, \ldots, \delta_n \rangle \mid \mathcal{I}, \{x_1 \mapsto \delta_1, \ldots, x_n \mapsto \delta_n\} \models Q\}$$

Prädikatenlogik $\approx$ SQL

Was ist eine Datenbank-Anfrage?

- Syntax: Eine Anfrage Q ist ein Wort aus einer Anfragesprache.
- Semantik: Jede Anfrage Q definiert eine Anfragefunktion f_Q, die für jede Datenbankinstanz $\mathcal{I}$ eine Ergebnisrelation $f_Q(\mathcal{I})$ liefert.

Beispiel: Für eine prädikatenlogische Formel Q mit freien Variablen $x_1, \ldots, x_n$ ist f_Q diejenige Funktion, die eine Interpretation $\mathcal{I}$ wie folgt auf die Ergebnisrelation $f_Q(\mathcal{I})$ abbildet:

$$\mathcal{I} \mapsto \{\langle \delta_1, \ldots, \delta_n \rangle \mid \mathcal{I}, \{x_1 \mapsto \delta_1, \ldots, x_n \mapsto \delta_n\} \models Q\}$$

Mit solch einer allgemeinen Definition kann man sehr unterschiedliche Anfragesprachen (über ihre Anfragefunktionen) miteinander vergleichen.

Satz: Die Menge der durch prädikatenlogische Formeln Q darstellbaren Anfragefunktionen f_Q ist genau die Menge der Anfragefunktionen, die durch einfache SQL-Anfragen darstellbar sind.

„Einfache SQL-Anfragen“: Relationale Algebra, der Kern von SQL; SELECT, JOIN, UNION, MINUS, aber keine komplexeren Features wie WITH RECURSIVE etc. Außerdem keine Datentypen, da wir diese in der Prädikatenlogik nicht eingeführt haben.

Relationale Algebren

Datenbankanfragen werden oft in relationaler Algebra dargestellt, bei der man Relationen mit Operationen zu einem Anfrageergebnis kombiniert.

Beispiel: Die Anfrage

$$Q = \exists z_{\text{Linie}}.(\text{verbindung}(x_{\text{Von}}, x_{\text{Zu}}, z_{\text{Linie}}) \wedge \text{linien}(z_{\text{Linie}}, x_{\text{Typ}}))$$

entspricht einer (natürlichen) Join-Operation ($\wedge$) mit anschließender Projektion ($\exists$):

$$\pi_{\text{Von,Zu,Typ}}(\text{verbindung} \bowtie \text{linien})$$

Anmerkung: SQL hat noch einen leicht anderen Stil. Variablen stehen dort für ganze Tabellenzeilen und man verwendet Notation der Form „linien.Typ“, um auf deren Einträge zuzugreifen („Tuple-Relational Calculus“).
$\rightsquigarrow$ Das ändert an der Ausdrucksstärke nichts.

Anfragebeantwortung als Model Checking

Erkenntnis: Die wesentliche Berechnungsaufgabe bei der Beantwortung von Datenbankabfragen ist das folgende Entscheidungsproblem:

Das Auswertungsproblem (Model Checking) der Prädikatenlogik lautet wie folgt:

Gegeben:

- Eine Formel Q mit freien Variablen $x_1, \ldots, x_n$;
- eine endliche Interpretation $\mathcal{I}$;
- Elemente $\delta_1, \ldots, \delta_n \in \Delta^{\mathcal{I}}$.

Frage: Gilt $\mathcal{I}, \{x_1 \mapsto \delta_1, \ldots, x_n \mapsto \delta_n\} \models Q$?

Anfragebeantwortung als Model Checking

Erkenntnis: Die wesentliche Berechnungsaufgabe bei der Beantwortung von Datenbankabfragen ist das folgende Entscheidungsproblem:

Das Auswertungsproblem (Model Checking) der Prädikatenlogik lautet wie folgt:

Gegeben:

- Eine Formel Q mit freien Variablen $x_1, \ldots, x_n$;
- eine endliche Interpretation $\mathcal{I}$;
- Elemente $\delta_1, \ldots, \delta_n \in \Delta^{\mathcal{I}}$.

Frage: Gilt $\mathcal{I}, \{x_1 \mapsto \delta_1, \ldots, x_n \mapsto \delta_n\} \models Q$?

Naive Methode der Anfragebeantwortung:

- Betrachte alle $\left(\Delta^{\mathcal{I}}\right)^n$ möglichen Ergebnisse;
- entscheide jeweils das Auswertungsproblem.

Praktisch relevante Frage:

Wie schwer ist das Auswertungsproblem?

Ein Algorithmus für das Auswertungsproblem

Wir nehmen an, dass die Formel F nur $\neg$, $\wedge$ und $\exists$ enthält. (Durch Umformung möglich.)

```
    function eval(F, I, Z) {
01    switch(F) {
02      case p(c_1, ..., c_n) : return <c_1^{I,Z}, ..., c_n^{I,Z}> ∈ p^I;
03      case ¬G : return not eval(G, I, Z);
04      case G_1 ∧ G_2 : return eval(G_1, I, Z) and eval(G_2, I, Z);
05      case ∃x.G :
06        for c ∈ Δ^I {
07          if eval(G{x ↦ c}, I, Z) then return true;}
08        return false; }
    }
```

Anmerkung: Wenn Konstanten c in der Anfrage vorkommen, dann nimmt man in der Regel an, dass $c^{\mathcal{I}} = c$ ist.

Anmerkung 2: In der Praxis stimmt das nicht ganz. Insbesondere bei Verwendung von Datentypen haben DB-Systeme normalerweise eingebaute Interpretationsfunktionen. Zum Beispiel würden die Konstanten `"42"` und `"+42"` die selbe Ganzzahl bezeichnen.

Quiz: Auswertungsproblem

Das Auswertungsproblem (Model Checking) der Prädikatenlogik lautet wie folgt:

Gegeben:

- Eine Formel Q mit freien Variablen $x_1, \ldots, x_n$;
- eine endliche Interpretation $\mathcal{I}$;
- Elemente $\delta_1, \ldots, \delta_n \in \Delta^{\mathcal{I}}$.

Frage: Gilt $\mathcal{I}, \{x_1 \mapsto \delta_1, \ldots, x_n \mapsto \delta_n\} \models Q$?

Quiz: Wir betrachten die untenstehenden Instanzen des Auswertungsproblems. ...

Zeitkomplexität

Sei $|F| = m$ die Größe von F und $|\mathcal{I}| = n$ die Gesamtgröße der Datenbank.

- Maximale Rekursionstiefe?

Zeitkomplexität

Sei $|F| = m$ die Größe von F und $|\mathcal{I}| = n$ die Gesamtgröße der Datenbank.

- Maximale Rekursionstiefe?
  $\rightsquigarrow$ beschränkt durch Zahl der Teilformeln: $\leq m$
- Maximale Zahl der Iterationen (und rekursiven Aufrufe) in **for**-Schleife?

Zeitkomplexität

Sei $|F| = m$ die Größe von F und $|\mathcal{I}| = n$ die Gesamtgröße der Datenbank.

- Maximale Rekursionstiefe?
  $\rightsquigarrow$ beschränkt durch Zahl der Teilformeln: $\leq m$
- Maximale Zahl der Iterationen (und rekursiven Aufrufe) in **for**-Schleife?
  $\rightsquigarrow$ $|\Delta^{\mathcal{I}}| \leq n$ pro rekursivem Aufruf
  $\rightsquigarrow$ insgesamt $\leq n^m$ Iterationen

Zeitkomplexität

Sei $|F| = m$ die Größe von F und $|\mathcal{I}| = n$ die Gesamtgröße der Datenbank.

- Maximale Rekursionstiefe?
  $\rightsquigarrow$ beschränkt durch Zahl der Teilformeln: $\leq m$
- Maximale Zahl der Iterationen (und rekursiven Aufrufe) in **for**-Schleife?
  $\rightsquigarrow$ $|\Delta^{\mathcal{I}}| \leq n$ pro rekursivem Aufruf
  $\rightsquigarrow$ insgesamt $\leq n^m$ Iterationen
- Rekursive Aufrufe in anderen Fällen?
  $\rightsquigarrow$ 1 oder 2

Zeitkomplexität

Sei $|F| = m$ die Größe von F und $|\mathcal{I}| = n$ die Gesamtgröße der Datenbank.

- Maximale Rekursionstiefe?
  $\rightsquigarrow$ beschränkt durch Zahl der Teilformeln: $\leq m$
- Maximale Zahl der Iterationen (und rekursiven Aufrufe) in **for**-Schleife?
  $\rightsquigarrow$ $|\Delta^{\mathcal{I}}| \leq n$ pro rekursivem Aufruf
  $\rightsquigarrow$ insgesamt $\leq n^m$ Iterationen
- Rekursive Aufrufe in anderen Fällen?
  $\rightsquigarrow$ 1 oder 2
- $\langle c_1^{\mathcal{I},\mathcal{Z}}, \ldots, c_n^{\mathcal{I},\mathcal{Z}} \rangle \in p^{\mathcal{I}}$ ist entscheidbar in linearer Zeit bezüglich n.

Zeitkomplexität

Sei $|F| = m$ die Größe von F und $|\mathcal{I}| = n$ die Gesamtgröße der Datenbank.

- Maximale Rekursionstiefe?
  $\rightsquigarrow$ beschränkt durch Zahl der Teilformeln: $\leq m$
- Maximale Zahl der Iterationen (und rekursiven Aufrufe) in **for**-Schleife?
  $\rightsquigarrow$ $|\Delta^{\mathcal{I}}| \leq n$ pro rekursivem Aufruf
  $\rightsquigarrow$ insgesamt $\leq n^m$ Iterationen
- Rekursive Aufrufe in anderen Fällen?
  $\rightsquigarrow$ 1 oder 2
- $\langle c_1^{\mathcal{I},\mathcal{Z}}, \ldots, c_n^{\mathcal{I},\mathcal{Z}} \rangle \in p^{\mathcal{I}}$ ist entscheidbar in linearer Zeit bezüglich n.

Gesamtlaufzeit in $\max(n, 2)^m \cdot n \leq (n + 2)^{m+1}$:

- Komplexität des Algorithmus: in ExpTime
- Komplexität bezüglich der Größe der Datenbank (m konstant): in P

Speicherkomplexität

Wir erhalten eine bessere Komplexitätsabschätzung, wenn wir den Speicherbedarf betrachten:

Sei $|F| = m$ die Größe von F und $|\mathcal{I}| = n$ die Gesamtgröße der Datenbank.

- Speichere pro (rekursivem) Aufruf einen Zeiger auf eine Teilformel von F: $\log m$
- Speichere für jede Variable in F (maximal m) die aktuelle Zuweisung (als Zeiger): $m \cdot \log n$
- $\langle c_1^{\mathcal{I},\mathcal{Z}}, \ldots, c_n^{\mathcal{I},\mathcal{Z}} \rangle \in p^{\mathcal{I}}$ ist entscheidbar in logarithmischem Speicher bezüglich n.

Speicherkomplexität

Wir erhalten eine bessere Komplexitätsabschätzung, wenn wir den Speicherbedarf betrachten:

Sei $|F| = m$ die Größe von F und $|\mathcal{I}| = n$ die Gesamtgröße der Datenbank.

- Speichere pro (rekursivem) Aufruf einen Zeiger auf eine Teilformel von F: $\log m$
- Speichere für jede Variable in F (maximal m) die aktuelle Zuweisung (als Zeiger): $m \cdot \log n$
- $\langle c_1^{\mathcal{I},\mathcal{Z}}, \ldots, c_n^{\mathcal{I},\mathcal{Z}} \rangle \in p^{\mathcal{I}}$ ist entscheidbar in logarithmischem Speicher bezüglich n.

Speicher in $m \log m + m \log n + \log n = m \log m + (m+1) \log n$

- Komplexität des Algorithmus: in PSpace
- Komplexität bezüglich der Größe der Datenbank (m konstant): in LogSpace

Zur Erinnerung: PSpace $\subseteq$ ExpTime und LogSpace $\subseteq$ P, d.h. die obigen Schranken sind besser.

Komplexität des Auswertungsproblems

Satz: Das Auswertungsproblem der Prädikatenlogik ist PSpace-vollständig.

Komplexität des Auswertungsproblems

Satz: Das Auswertungsproblem der Prädikatenlogik ist PSpace-vollständig.

Beweis: Durch Reduktion vom Auswertungsproblem quantifizierter Boolescher Formeln (**TrueQBF**).

Sei $Q_1 p_1 . Q_2 p_2 . \cdots Q_n p_n . F[p_1, \ldots, p_n]$ eine QBF (mit $Q_i \in \{\forall, \exists\}$).

Komplexität des Auswertungsproblems

Satz: Das Auswertungsproblem der Prädikatenlogik ist PSpace-vollständig.

Beweis: Durch Reduktion vom Auswertungsproblem quantifizierter Boolescher Formeln (**TrueQBF**).

Sei $Q_1 p_1 . Q_2 p_2 . \cdots Q_n p_n . F[p_1, \ldots, p_n]$ eine QBF (mit $Q_i \in \{\forall, \exists\}$).

- Wir nutzen die Datenbankinstanz $\mathcal{I}$ mit $\Delta^{\mathcal{I}} = \{0, 1\}$; außerdem
- ein einstelliges Prädikat true mit $\text{true}^{\mathcal{I}} = \{1\}$ (eine Tabelle mit einer Spalte: true(1)).
- Aus der gegebenen QBF erstellen wir die folgende prädikatenlogische Formel ohne freie Variablen:

$$Q_1 x_1 . Q_2 x_2 . \cdots Q_n x_n . F[p_1/\text{true}(x_1), \ldots, p_n/\text{true}(x_n)]$$

  wobei $F[p_1/\text{true}(x_1), \ldots, p_n/\text{true}(x_n)]$ die Formel ist, die aus F entsteht, wenn man jedes aussagenlogische Atom p_i durch das prädikatenlogische Atom $\text{true}(x_i)$ ersetzt.

Die Korrektheit dieser Reduktion ist leicht zu zeigen. □

Wie schwer sind Datenbankabfragen?

Korollar: Die Beantwortung von SQL-Anfragen ist PSpace-schwer, sogar wenn die Datenbank nur eine einzige Tabelle mit einer einzigen Zeile enthält.

Die Komplexität steckt vor allem in der Struktur der Anfrage.

Ist die Anfrage fest vorgegeben oder in ihrer Größe beschränkt, dann wird das praktische Verhalten oft von der Datenbankgröße dominiert: Bezüglich dieser Größe ist das Problem aber in LogSpace.

Man kann sogar noch niedrigere Komplexitätsschranken bezüglich der Datenbankgröße angeben (siehe Vorlesung Database Theory).

$\rightsquigarrow$ SQL-Anfragebeantwortung ist praktisch implementierbar, aber nur solange die Anfragen nicht zu komplex werden.

Die Grenzen der Prädikatenlogik

Reprise: Formeln als Anfragen

linien:

Linie	Typ
85	Bus
3	Tram
F1	Fähre
...	...

haltestellen:

SID	Name	Rollstuhl
17	Hauptbahnhof	true
42	Helmholtzstr.	true
57	Stadtgutstr.	true
123	Gustav-Freytag-Str.	false
...	...	...

verbindung:

Von	Zu	Linie
57	42	85
17	789	3
...	...	...

Die einfache Arität der Prädikatenlogik wird durch ein Schema mit Namen (und oft auch Datentypen) ersetzt:

- linien[Linie:string, Typ:string]
- haltestellen[SID:int, Halt:string, Rollstuhl:bool]
- verbindung[Von:int, Zu:int, Linie:string]

Relationale Algebra: Parameter (Spalten) durch Namen adressiert
Prädikatenlogik: Parameter durch Reihenfolge adressiert

Die Anfrage $\exists z_{\text{Linie}}.(\text{verbindung}(x_{\text{Von}}, x_{\text{Zu}}, z_{\text{Linie}}) \wedge \text{linien}(z_{\text{Linie}}, x_{\text{Typ}}))$ entspricht einer (natürlichen) Join-Operation ($\wedge$) mit anschließender Projektion ($\exists$):
$\pi_{\text{Von,Zu,Typ}}(\text{verbindung} \bowtie \text{linien})$.

Prädikatenlogik als Anfragesprache

Beispielanfragen:

“Haltestellen, die Helmholtzstr. sind:”

$$Q_0[x_0] = (x_0 \approx 42)$$

Prädikatenlogik als Anfragesprache

Beispielanfragen:

“Haltestellen, die Helmholtzstr. sind:”

$$Q_0[x_0] = (x_0 \approx 42)$$

“Haltestellen direkt neben Helmholtzstr.:”

$$Q_1[x_1] = \exists x_0, z_{\text{Linie}}.(\text{verbindung}(x_0, x_1, z_{\text{Linie}}) \wedge Q_0[x_0])$$

Prädikatenlogik als Anfragesprache

Beispielanfragen:

"Haltestellen, die Helmholtzstr. sind:"

$$Q_0[x_0] = (x_0 \approx 42)$$

"Haltestellen direkt neben Helmholtzstr.:"

$$Q_1[x_1] = \exists x_0, z_{\text{Linie}}.(\text{verbindung}(x_0, x_1, z_{\text{Linie}}) \wedge Q_0[x_0])$$

"Haltestellen, die zwei Halte weit von Helmholtzstr. entfernt sind:"

$$Q_2[x_2] = \exists x_1, z_{\text{Linie}}.(\text{verbindung}(x_1, x_2, z_{\text{Linie}}) \wedge Q_1[x_1])$$

Prädikatenlogik als Anfragesprache

Beispielanfragen:

"Haltestellen, die Helmholtzstr. sind:"

$$Q_0[x_0] = (x_0 \approx 42)$$

"Haltestellen direkt neben Helmholtzstr.:"

$$Q_1[x_1] = \exists x_0, z_{\text{Linie}}.(\text{verbindung}(x_0, x_1, z_{\text{Linie}}) \wedge Q_0[x_0])$$

"Haltestellen, die zwei Halte weit von Helmholtzstr. entfernt sind:"

$$Q_2[x_2] = \exists x_1, z_{\text{Linie}}.(\text{verbindung}(x_1, x_2, z_{\text{Linie}}) \wedge Q_1[x_1])$$

Und so weiter ...

"Haltestellen, die von Helmholtzstr. aus mit einem Kurzstreckenticket erreichbar sind:"

$$Q_0[x] \vee Q_1[x] \vee Q_2[x] \vee Q_3[x] \vee Q_4[x]$$

Die Grenzen der Prädikatenlogik

Wie finden wir alle Haltestellen, die man von Helmholtzstr. aus erreichen kann?

Die Grenzen der Prädikatenlogik

Wie finden wir alle Haltestellen, die man von Helmholtzstr. aus erreichen kann?

Es stellt sich heraus, dass das unmöglich ist.

Intuition: Prädikatenlogik kann nur “lokale” Eigenschaften überprüfen.

Die Grenzen der Prädikatenlogik

Wie finden wir alle Haltestellen, die man von Helmholtzstr. aus erreichen kann?

Es stellt sich heraus, dass das unmöglich ist.

Intuition: Prädikatenlogik kann nur “lokale” Eigenschaften überprüfen.

Das lässt sich mathematisch ausdrücken:

Vage Behauptung (frei nach Gaifmans Locality Theorem): Für jeden Satz F gibt es eine Zahl d, so dass für beliebige Interpretationen $\mathcal{I}$ und $\mathcal{J}$ gilt:

- Wenn man nur zusammenhängende endliche Teile von $\mathcal{I}$ und $\mathcal{J}$ betrachtet, die höchstens Pfade der Länge d enthalten („Durchmesser $\leq d$“),
- und wenn sich $\mathcal{I}$ und $\mathcal{J}$ bezüglich dieser d-Umgebungen nicht unterscheiden,
- dann kann auch F die Interpretationen nicht unterscheiden: $\mathcal{I} \models F$ gdw. $\mathcal{J} \models F$.

Der Durchmesser d, der angibt wie weit F höchstens „schauen“ kann, hängt exponentiell von der Schachtelungstiefe der Quantoren ab.

Rekursive Anfragen

Nichtlokale Eigenschaften wie z.B. die Erreichbarkeit in Graphen sind praktisch relevant (speziell in Graphdatenbanken).

Wie kann man solche Anfragen logisch ausdrücken?

Rekursive Anfragen

Nichtlokale Eigenschaften wie z.B. die Erreichbarkeit in Graphen sind praktisch relevant (speziell in Graphdatenbanken).

Wie kann man solche Anfragen logisch ausdrücken?

Idee: Um beliebig weit zu „schauen“, muss man Rekursion einführen.

Beispiel: Eine Haltestelle ist von Helmholtzstr. aus erreichbar, wenn

(1) sie die Haltestelle Helmholtzstr. ist, oder

(2) sie neben einer Haltestelle liegt, die von Helmholtzstr. aus erreichbar ist.

Zusammenfassung und Ausblick

Beschränkt man Prädikatenlogik auf endliche Interpretationen, so gibt es kein vollständiges und korrektes Verfahren zum logischen Schließen – dafür wird Erfüllbarkeit semi-entscheidbar.

Das Auswertungsproblem auf endlichen Interpretationen entspricht der Anfragebeantwortung in Datenbanken.
(Komplexität: PSpace-vollständig, aber sub-polynomiell bezüglich der Datenbankgröße.)

Prädikatenlogik hat Grenzen:

- bei der Modellierung (logisches Schließen), z.B. transitiver Abschluss;
- bei logischen Abfragen (Model Checking), z.B. Erreichbarkeit.

Was erwartet uns als nächstes?

- Datalog und Logik höherer Ordnung
- Gödel