Komplexitätstheorie

WiSe 2008/09 in Trier

Henning Fernau
Universität Trier
fernau@uni-trier.de

Komplexitätstheorie Gesamtübersicht

- Organisatorisches / Einführung
  Motivation / Erinnerung / Fragestellungen

- Diskussion verschiedener Komplexitätsklassen:
  Zeitkomplexität
  Platzkomplexität

- zugehörige Reduktionsbegriffe

- vollständige Probleme

- Anpassung von Klassenbegriffen und Reduktionen

PSPACE

nur kurz: zwei typische Probleme aus Logik / Graphentheorie

Eine *quantifizierte Formel* ist ein Wort der Form

$$Q_1 v_1 Q_2 v_2 \dots Q_k v_k \; w$$

wobei
—$Q_i \in \{\exists, \forall\}$,
—alle v_i sind logische Variablen,
—w ist wohlgeformte Formel.

$Q_1 v_1 Q_2 v_2 \dots Q_k v_k \; w$ heißt *voll quantifiziert*, wenn jede der Variablen aus w in einem der Q_i auftritt.

Beispiel für voll quantifizierte Formel:

$$\exists x \forall y \exists z((x \vee y \vee z) \wedge (x \vee \bar{y} \vee \bar{z}) \wedge (\bar{x} \vee y \vee z))$$

Induktiv: *Wert der quantifizierten Formel* q unter Interpretation / Belegung ϕ:

- q ohne Quantor:
  q ist normale wohlgeformte Formel, $\mathtt{Wert}_\phi(q)$ wie üblich.

- $q = \exists x\, q'$:
  $\mathtt{Wert}_\phi(q) = 1$ genau dann, wenn ein ϕ' existiert mit $z \neq x \Rightarrow \phi'(z) = \phi(z)$ und $\mathtt{Wert}_{\phi'}(q') = 1$.

- $q = \forall x\, q'$:
  $\texttt{Wert}_\phi(q) = 1$ genau dann, wenn für alle ϕ' mit $z \neq x \Rightarrow \phi'(z) = \phi(z)$ gilt $\texttt{Wert}_{\phi'}(q') = 1$.

Voll quantifizierte Formeln q:
$\texttt{Wert}_\phi(q)$ unabhängig von ϕ, d.h. $\texttt{Wert}(q)$ sinnvoll definiert!

O.B.d.A.: keine Variable wird mehrfach quantifiziert
(nur rechteste Quantifizierzung von Variable x hat Einfluss auf Wert!)

Problem 3-QBF (quantifizierte Boole'sche Formel) ist definiert als

- 3-QBF: Gegeben sei eine voll quantifizierte Formel

$$Q_1 v_1 Q_2 v_2 \dots Q_k v_k \, w$$

  wobei w ein 3-CNF-Ausdruck sei.
  Frage: Gilt $\texttt{Wert}(q) = 1$?

Sind $x_1, \dots, x_n$ Variablen aus 3-CNF-Ausdruck w, so gilt

$$w \in \texttt{3-SAT} \iff \exists x_1 \dots \exists x_n w \in \texttt{3-QBF}$$

$\Rightarrow$ 3-QBF offensichtlich NP-hart

3-QBF sogar PSPACE-vollständig (hier jedoch nur: polynomialer Platzbedarf und Grundidee des Vollständigkeitsbeweises)

Satz 1 *(1)* $\texttt{3-QBF}$ *liegt in* $\texttt{PSPACE}$

(2) $\texttt{3-QBF}$ *ist* $\texttt{NPSPACE}$*-hart bzgl.* $\leq_{\log}$

Anm.: Aus Satz folgt zudem

$$\texttt{NPSPACE} \subseteq \{A \mid A \leq_{\log} \texttt{3-QBF}\} \subseteq \texttt{PSPACE}$$

d.h. zweiter Beweis der Beziehung $\texttt{PSPACE} = \texttt{NPSPACE}$.

3-QBF **liegt in** PSPACE

Sei $q = Q_1 x_1 \ldots w_n x_n w$ eine voll quantifizierte Formel mit 3-CNF-Ausdruck w.
Ziel: Bestimmung von $\mathtt{Wert}(q)$!

Dazu muss $\mathtt{Wert}_\phi(w)$ für verschiedene Belegungen ϕ bestimmt werden.

Die folgende rekursive Prozedur **WERT**(ℓ), $1 \leq \ell \leq n+1$ bestimmt $\mathtt{Wert}(q_\ell)$,
wobei sich q_ℓ aus q durch Fixierung der Booleschen Variablen $x_1, \ldots, x_{\ell-1}$ ergibt.
Diese Fixierung wird in einem Booleschen Feld $\phi = (\phi(x_1), \ldots, \phi(x_{\ell-1}))$ gespeichert.
$\rightsquigarrow \mathtt{Wert}(q) = \mathtt{Wert}(q_1) =$ **WERT**(1).

WERT(ℓ):
(1) Falls $\ell = n+1$, so **WERT** $\leftarrow$ **WERT**$_\phi(w)$.
(2) Falls $Q_\ell = \exists$: Setze $\phi(x_\ell) \leftarrow 0$. Bestimme **WERT**$(\ell+1)$.
(2a) Falls Ergebnis 1, so **WERT** $\leftarrow 1$.
(2b) Ansonsten setze $\phi(x_\ell) \leftarrow 1$. **WERT** $\leftarrow$ **WERT**$(\ell+1)$.
(3) Falls $Q_\ell = \forall$: Setze $\phi(x_\ell) \leftarrow 0$. Bestimme **WERT**$(\ell+1)$.
(3a) Falls Ergebnis 0, so **WERT** $\leftarrow 0$.
(3b) Ansonsten setze $\phi(x_\ell) \leftarrow 1$. **WERT** $\leftarrow$ **WERT**$(\ell+1)$.

Grundidee der Vollständigkeit

Betrachte $L \in \mathbf{NPSPACE}$ und M TM, die L mit pol. Platzbedarf akzeptiert.

$\rightsquigarrow \exists k\colon x \in L \iff I_M(x) \rightarrow_M^{\leq 2^{|x|^k}} K_e$ für eine Endkonfig. K_e.

Def. für zwei Konfig. K_1, K_2: $F_i(K_1, K_2)$ gdw. $K_1 \rightarrow_M^{\leq 2^i} K_2$; damit:
$x \in L \iff \exists$ Endkonfig. K_e: $F_{|x|^k}(I_M(x), K_e)$.

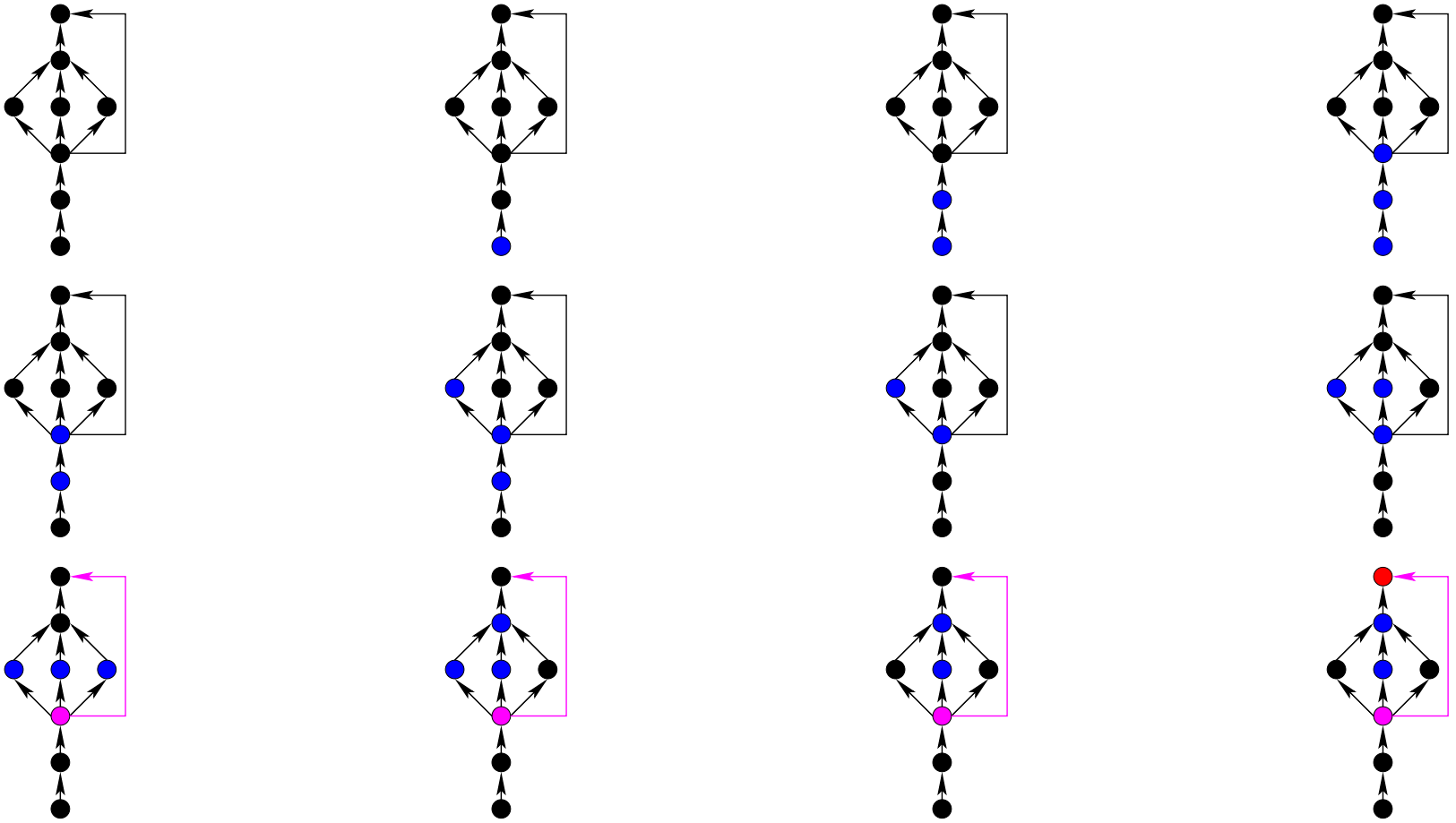
Für $F_{|x|^k}(I_M(x), K_e)$ wird induktiv eine äquivalente Formel konstruiert, die nur noch elementare Bestandteile hat:
$F_0(K_1, K_2) \iff K_1 = K_2 \vee K_1 \rightarrow_M K_2$.
$F_{i+1}(K_1, K_2) \iff \exists K_3 \forall K_4 \forall K_5 : ((K_1 = K_4 \wedge K_3 = K_5) \vee (K_3 = K_4 \wedge K_2 = K_5)) \Rightarrow F_i(K_4, K_5)$
Diese Formel ist äquivalent zu $\exists K : F_i(K_1, K) \wedge F_i(K, K_2)$.
(Diese Fassung kann man aber nicht direkt nehmen, da die QBF dann exponent. lang wäre.)
Der so induktiv aufgebaute Ausdruck hat eine Länge polynomiell in $|x|$.
Es muss “nur” noch eine Normalform für **3**-QBF erzeugt werden. . .
Details $\rightsquigarrow$ `http://www.thi.informatik.uni-frankfurt.de/~gramlich/thi2/V26.pdf`

Weiteres PSPACE-vollständiges Problem aus bekanntem Pebble-Spiel:

(1) In einem gerichteten Graphen darf jeder Knoten ohne Vorgänger markiert werden,
(2) sind alle Vorgänger eines Knotens markiert, so darf der Knoten selbst markiert werden,
(3) sind alle Vorgänger eines Knotens markiert, so darf die Markierung von einen Vorgänger entfernt und auf den Knoten selbst übertragen werden,
(4) jederzeit dürfen die Markierungen von Knoten auch wieder entfernt werden.

- PEBBLE: Gegeben seien ein gerichteter Graph G und eine Zahl k.
  Frage: Kann mit den obigen Regeln ein Knoten ohne Nachfolger markiert werden, wenn *gleichzeitig nie mehr als* k *Knoten* markiert sein dürfen?

Beispiel zu `PEBBLE`: vier Steine sind wegen rechtem Ast nötig



Satz 2 PEBBLE *ist* NPSPACE*-vollständig (bzgl.* $\leq_{\log}$*).*

Beweis von $\texttt{PEBBLE} \in \texttt{NPSPACE}$ trivial:
—Wähle stets nichtdeterministisch eine der Regeln (1) bis (4) aus.
—Wende diese Regel nichtdeterministisch auf die Menge der bereits markierten Knoten angewandt (falls möglich).
—Achte darauf, dass nach Anwendung der Regel nicht mehr als k Knoten markiert sind.
—Kann ein Knoten ohne Nachfolger markiert werden, so wird die Eingabe akzeptiert.

Hilfsspeicher:
Bitvektor für markierte Knoten und Zähler für deren Anzahl.

Auf den Beweis von $\texttt{3-QBF} \leq_{\log} \texttt{PEBBLE}$ wird hier verzichtet.

QBF als Spiel

Jede vollständig quantifizierte Formel ψ kann als Spiel zwischen A und E angesehen werden:
Ist $\psi = \exists x \phi(x)$, so ist E am Zug: dieser legt $x = 0$ bzw. $x = 1$ fest, und das Spiel wird mit der Formel $\phi(0)$ bzw. $\phi(1)$ fortgesetzt.
Ist $\psi = \forall x \phi(x)$, ist A am Zug und handelt analog.
Das Spiel wird fortgesetzt, bis eine quantorenfreie Formel vorliegt.
E hat gewonnen, wenn diese ja auch variablenfreie finale Formel wahr ist.

Satz 3 *Die Sprache* FORMULA-GAME *aller vollständig quantifizierten Formeln* ψ*, bei denen* E *eine Gewinnstrategie besitzt, ist* **PSPACE***-vollständig.*

Beispiel: Bei $\exists x \forall y \exists z((x \vee y) \wedge (y \vee z) \wedge (y \vee \neg z))$ hat A die Gewinnstrategie, $y = 0$ zu wählen.

Viele Spielprobleme sind NPSPACE-vollständig; ein Beispiel

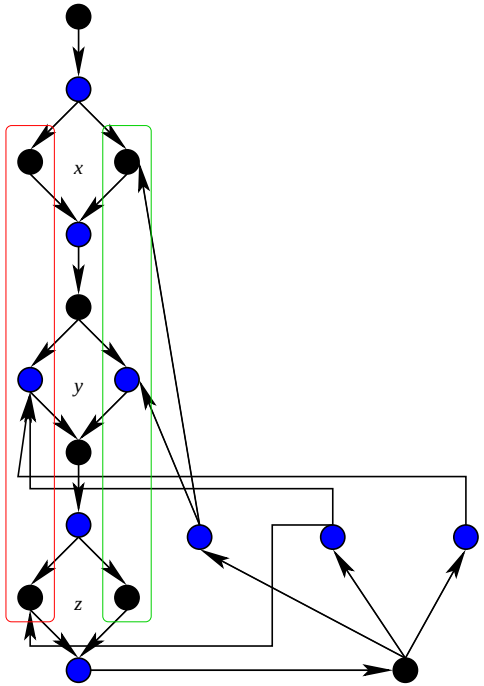
Der Spielgraph des *Geographie-Spiel*s ist ein gerichteter Graph $G = (V, E)$ mit Startknoten s. Die beiden Spieler markieren abwechselnd Knoten aus G nach den folgenden Regeln:

1. Im ersten Zug markiert Spieler 1 s.
2. Im t-ten Zug, $t > 1$, markiert der jeweilige Spieler, der am Zug ist, einen direkten Nachfolgeknoten des Knotens, welcher im $(t-1)$-ten Zug markiert wurde.
3. Ein Knoten darf nicht mehrmals markiert werden.
4. Es gewinnt der Spieler, welcher den letzten Zug ausführt.

$\texttt{GEOGRAPHIE} := \{c(G) \mid c(G)$ ist eine Kodierung eines Spielgraph des Geographie-Spiels, auf dem Spieler 1 eine Gewinnstrategie besitzt. $\}$

Satz 4 GEOGRAPHIE *ist* **PSPACE***-vollständig.*

Idee für die Härte:
Der Graph kodiert die Formel:
$\exists x \forall y \exists z((\neg x \vee \neg y) \wedge (y \vee z) \wedge (y))$.



Allgemeiner Aufbau:
Wir gehen aus von einer Instanz von FORMULA-GAME
"Diamanten" für die Variablenbelegungsauswahl.
Dabei seien (o.E.) die erste und letzte Variable existenzquantifiziert.
E hat Gewinnstrategie in FORMULA-GAME gdw. Spieler 1 hat Gewinnstrategie in abgeleiteter GEOGRAPHIE-Instanz.
Im Bild sind die möglichen Knotenwahlen von Spieler 1 bzw. Spieler 2 mit schwarz bzw. blau bezeichnet; einzige Ausnahme: ein möglicher letzter Zug von Spieler 1, mit dem er beweist, dass die Formel 1 liefert.
Im Beispiel links hat Spieler 2 eine Gewinnstrategie, indem von ihm
1. $y = 0$ ausgewählt wird (linker Weg bei Variablenauswahl),
2. die letzte Klausel (y) gewählt wird.
Nach der Klauselwahl kann Spieler 1 nicht mehr ziehen und hat verloren.

Was ist eigentlich Nichtdeterminismus ?

Eine mögliche Beschreibung wäre fußend auf folgender Festlegung:
“Eine Konfiguration *führt zur Akzeptierung*, wenn sie eine finale akzeptierende Konfiguration ist oder (rekursiv), wenn irgendeine ihrer Nachfolgekonfigurationen zur Akzeptierung führt.”

Akzeptanz wird also als implizite ODER-Verknüpfung der Akzeptanzeigenschaft der Nachfolgekonfigurationen angesehen.
$\leadsto$ Schwierigkeit der Komplementierung bei nichtdeterministisch definierten Komplexitätsklassen

Nach dem deMorganschen Gesetz wäre dazu nämlich eine UND-Verknüpfung der Akzeptanzeigenschaft der Nachfolgekonfigurationen vonnöten.

Idee: Führe solche Verknüpfungen in das Modell mit ein.

Alternierung

Eine *alternierende Turingmaschine* (ATM) hat zunächst dieselbe Beschreibung wie eine nichtdeterministische Turingmaschine, nur dass zusätzlich ihre Zustandsmenge Z zerlegt ist in $Z_\wedge$ und $Z_\vee$.
Eine Konfiguration *führt zur Akzeptierung*,
(1) wenn sie eine finale akzeptierende Konfiguration ist oder (rekursiv),
(2) wenn sie einen Zustand aus $Z_\vee$ enthält und irgendeine ihrer Nachfolgekonfigurationen zur Akzeptierung führt, oder
(3) wenn sie einen Zustand aus $Z_\wedge$ enthält und alle ihre Nachfolgekonfigurationen zur Akzeptierung führen.
Ein Wort x wird von einer ATM M akzeptiert, wenn $I_M(x)$ zur Akzeptierung führt.
$\rightsquigarrow$ Komplexitätsklassen wie $\mathbf{AL}$ und $\mathbf{AP}$.

Alternierung: Vermittler von Raum und Zeit

Satz 5 $\mathbf{AL} = \mathbf{P}$.

Satz 6 $\mathbf{AP} = \mathbf{PSPACE}$.

Idee: Statt einer direkten Simulation kann man auch ein z.B. $\mathbf{P}$-vollständiges Problem wie `MONOTONE CIRCUIT VALUE` nehmen und zeigen, wie man dieses in $\mathbf{AL}$ lösen kann bzw. wie man dieses benutzen kann, um eine beliebige (generische) $\mathbf{AL}$-Maschine zu simulieren (Rechenteppich-Trick wie beim Satz von Cook).

Grundprobleme bei Sprach- und Grammatikfamilien & ihre Komplexitäten

	Typ 3	Typ 2	Typ 1	Typ 0
$\in$	NC^1	SAC^1	**PSPACE**	unentsch.
$\neq \emptyset$	**NL**	**P**	unentsch.	unentsch.
$\equiv$	**PSPACE**	unentsch.	unentsch.	unentsch.

ACHTUNG:
Lesen Sie diese Tabelle (u.ä. aus der Literatur) “geeignet”, d.h., unter Zugrundelegung “vernünftiger” Sprachbeschreibungsmechanismen.
Schaltkreisfamilien

Strukturen

Eine *Struktur* $\mathcal{S}$ besteht aus einer (endlichen) Grundmenge (Universum) U und einer (endlichen) Menge ρ von Relationensymbolen (auch *relationales Alphabet* genannt); jedes $R \in \rho$ besitzt eine Stelligkeit $\alpha(R)$.

—Graphen lassen sich als Strukturen begreifen mit einem einzigen zweistelligen Relationensymbol, welches die Kantenrelation darstellt.

—Hypergraphen: interpretiert als bipartite Graphen (Inzidenzrelation).

—Schaltkreise: Neben der Kantenrelation gibt es einige einstellige "Knotenbeschriftungen" (AND, OR, ...).

—Zeichenketten der Länge n über Alphabet Σ: Grundmenge $U = \{1, \ldots, n\}$; zweistellig: $\leq$; einstellig: P_a für jedes $a \in \Sigma$.
$P_a(i)$ bedeutet: an der i-ten Position in der Zeichenkette steht ein a.

Strukturen und Logik

Es sei $\mathcal{S} = (U, \rho)$ eine Struktur.

Atomare Formeln (über $\mathcal{S}$) haben die Form $x = y$ bzw. $Rx_1 \dots x_n$, mit $R \in \rho$, $\alpha(R) = n$. x, y, bzw. x_i sind Variablen.

Formeln *erster Ordnung* sind aus atomaren Formeln mit Hilfe der Verknüpfungen $\neg$, $\vee$, $\wedge$ (andere seien möglich) sowie der Quantoren $\exists, \forall$ aufgebaut.

Variablen können durch Quantoren gebunden werden, sonst heißen sie frei.

Eine Formel ϕ mit freien Variablen $x_1, \dots, x_k$ lässt sich induktiv als k-stellige Relation $\phi(\mathcal{S})$ über U begreifen.

—Ist $\phi(x_1, \dots, x_k) = Rx_{i_1} \dots x_{i_r}$ mit $i_j \in \{1, \dots, k\}$, so

$\phi(\mathcal{S}) = \{(a_1, \dots, a_k) \in U^k \mid (a_{i_1}, \dots, a_{i_r}) \in R\}$.

—Ist $\phi(x_1, \dots, x_k) = \psi(x_{i_1}, \dots, x_{i_p}) \wedge \chi(x_{j_1}, \dots, x_{j_q})$, so

$\phi(\mathcal{S}) = \{(a_1, \dots, a_k) \in U^k \mid (a_{i_1}, \dots, a_{i_p}) \in \psi(\mathcal{S}) \wedge (a_{j_1}, \dots, a_{j_q}) \in \chi(\mathcal{S})\}$.

—Ist $\phi(x_1, \dots, x_k) = \exists x_{k+1} \psi(x_{i_1}, \dots, x_{i_p})$ mit $i_1, \dots, i_p \in \{1, \dots, k+1\}$, so

$\phi(\mathcal{S}) = \{(a_1, \dots, a_k) \in U^k \mid \exists a_{k+1} \in U : (a_{i_1}, \dots, a_{i_p}) \in \psi(\mathcal{S})\}$.

Formelarten und Modelle I

Literale sind atomare oder negierte atomare Formeln.
Eine Formel ist in *Negationsnormalform*, wennn Negationssymbole nur vor atomaren Formeln vorkommen.
Eine Formel ist in *pränexer Normalform*, wenn sie von der Form $Q_1 x_1, \ldots, Q_\ell x_\ell \psi$ ist, wobei $Q_i \in \{\forall, \exists\}$ und ψ quantorenfrei.
$\Sigma_0 = \Pi_0$ bezeichnen die quantorenfreien Formeln.
Für $t \geq 0$ ist Σ_{t+1} die Klasse der Formeln $\exists x_1 \ldots \exists x_k \phi$ mit $\phi \in \Pi_t$, sowie Π_{t+1} die Klasse der Formeln $\forall x_1 \ldots \forall x_k \phi$ mit $\phi \in \Sigma_t$.

Beispiel für Σ_1-Formel über der Struktur "Graph" $G = (V, E)$:
$is_k = \exists x_1 \ldots \exists x_k \left(\bigwedge_{1 \leq i < j \leq k} (x_i \neq x_j \wedge \neg E x_i x_j) \right)$.

Gilt $\phi(\mathcal{S}) \neq \emptyset$, so heißt $\mathcal{S}$ auch *Modell* für ϕ.
Jeder Graph mit einer unabhängigen Menge der Größe k ist Modell für is_k.
Model-Checking: Ggb. $\mathcal{S}$ und Formel ϕ: Gilt $\phi(\mathcal{S}) \neq \emptyset$?

Formelarten und Modelle II

FO: Menge der Formeln erster Ordnung, also FO$= \bigcup_t \Sigma_t = \bigcup_t \Pi_t$.
FO^k: Fragment von FO, bei dem alle Formeln höchstens k Variablen enthalten.
Ist F Formelmenge, so bezeichnet $MC(F)$ das Model-Checking Problem bzgl. F.

Satz 7 $MC(\mathrm{FO}^2)$ *ist* $\mathbf{P}$*-vollständig.*

Satz 8 *Für jedes* $t \geq 1$ *ist* $MC(\Sigma_t)$ *vollständig für die* t*-te Stufe* $\Sigma_t\mathbf{P}$ *der Polynomialzeit-Hierarchie.*

Satz 9 $MC(\mathrm{FO})$ *ist* $\mathbf{PSPACE}$*-vollständig.*