Komplexitätstheorie

WiSe 2011/12 in Trier

Henning Fernau
Universität Trier
fernau@uni-trier.de

Komplexitätstheorie Rückschau

- Organisatorisches / Einführung
  Motivation / Erinnerung / Fragestellungen

- Diskussion verschiedener Komplexitätsklassen:
  Zeitkomplexität
  Platzkomplexität

- zugehörige Reduktionsbegriffe

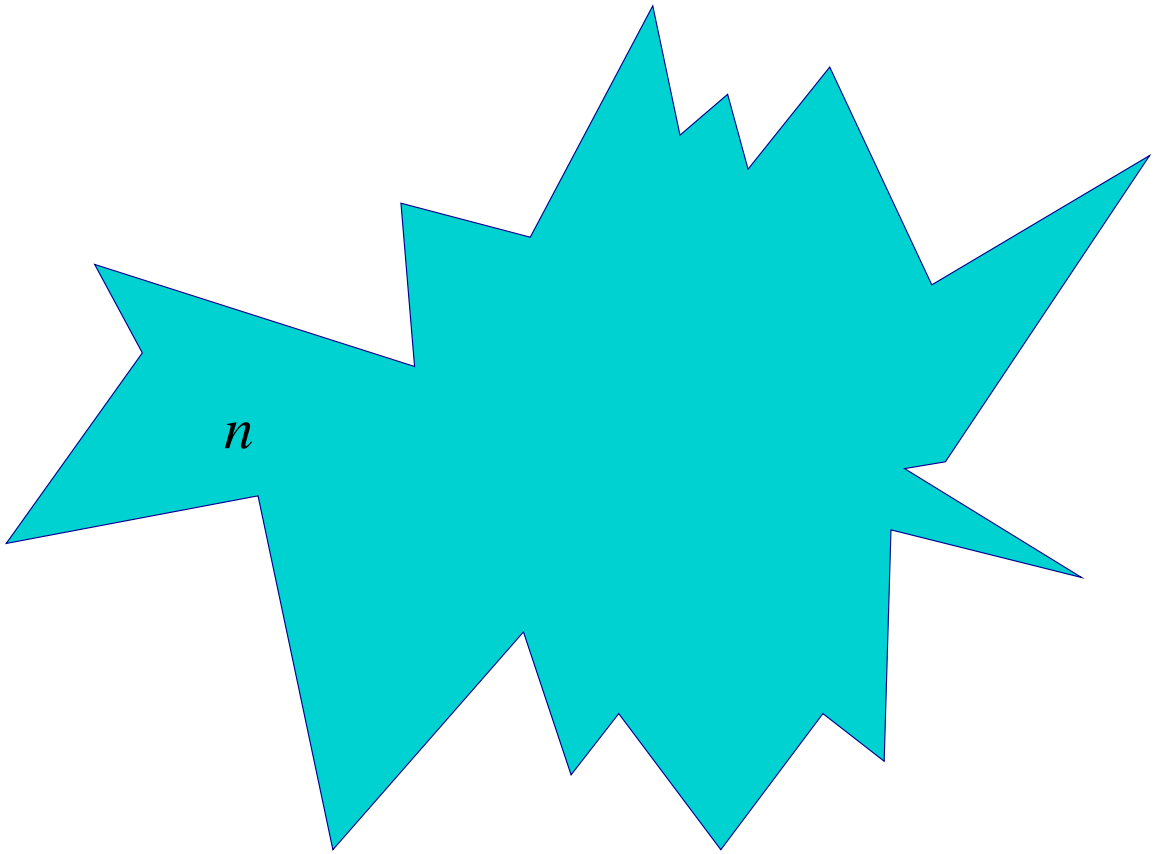
- vollständige Probleme

- Anpassung von Klassenbegriffen und Reduktionen

Was macht gute Komplexitätstheorie aus ?

- Eine schöne Klasse $\mathcal{K}$ “guter Probleme”:
  Bsp.: $\mathbf{NL}$, $\mathbf{P}$.
- Eine entsprechend passende Reduktion, unter der $\mathcal{K}$ abgeschlossen ist:
  Bsp.: LOGSPACE-Reduktion f; ist P “gut”, so auch $\mathrm{f}(\mathrm{P})$.
- Reduktionsbegriff ist kompositionsstabil
- Eine größere, böse Klasse $\mathcal{K}'$, z.B. $\mathbf{NP}$.
- Reduktionsbegriff ist schwächer als $\mathcal{K}'$, $\mathcal{K}'$ ist aber abgeschlossen gegenüber den betrachteten Reduktionen..
- Die Echtheit der (bekannten) Inklusion $\mathcal{K} \subseteq \mathcal{K}'$ ist unbekannt.

The Curse of Combinatorics (Folklore zur NP-Härte)



Auswege:

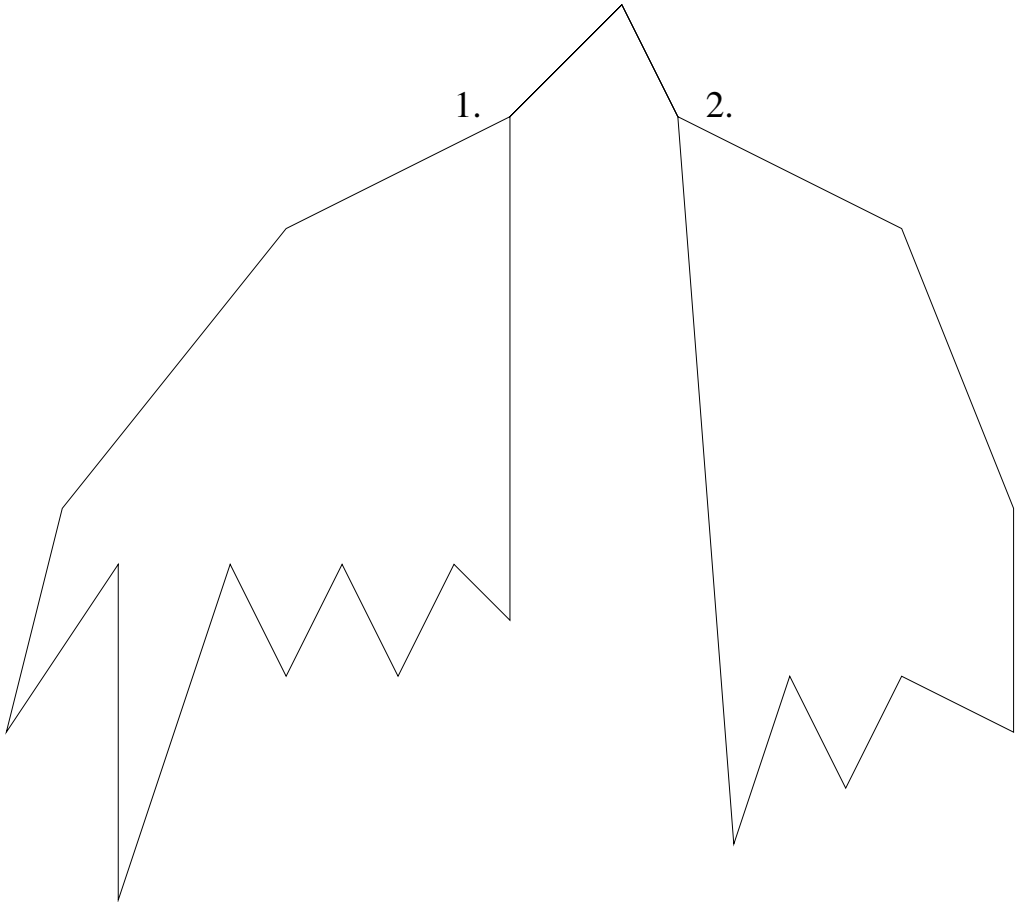
- Suchbäume (branch & bound):
  Exponentialzeit; Laufzeitgarantien?

- Näherungsalgorithmen $\rightsquigarrow$ eigene, regelmäßige Spezialvorlesung:
  Polynomzeit; Güteschranken; nicht optimal

- Heuristiken, u.a. Reduktionsregeln

- Randomisierung

Suchbäume:



Wie “schlimm” ist Exponentialzeit?

Vergleichen wir exponentielle mit polynomiellen Laufzeiten für $n = 50$ auf einem Rechner, der 10^8 Operationen je Sekunde bearbeitet.

Poynomiell		Exponentiell	
Komplexität	Laufzeit	Komplexität	Laufzeit
n^2	25 μs	1.2^n	91 μs
n^3	1 ms	1.5^n	6 s
n^5	3 s	2^n	130 Tage
n^{100}	$9.13 \cdot 10^{156}$ Jahre	3^n	$228 \cdot 10^6$ Jahre

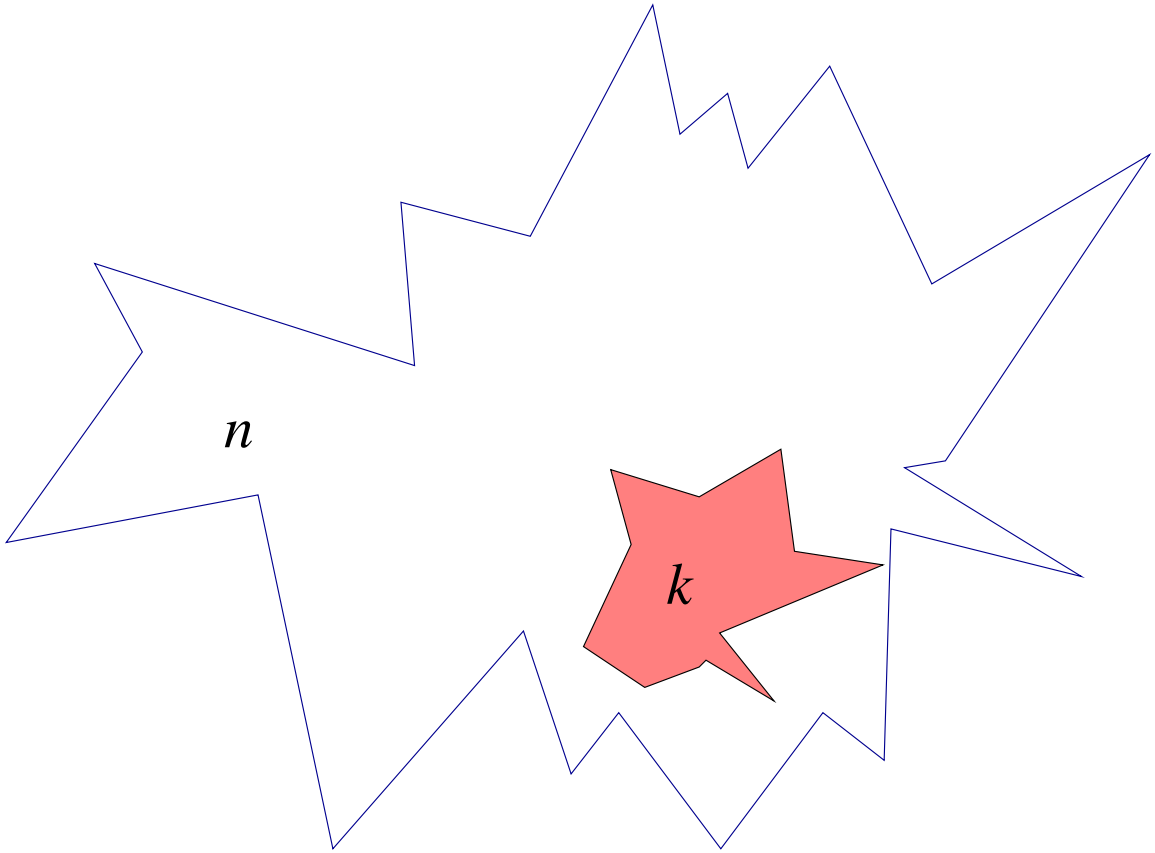
Sehr wichtig: Komplexitätsabschätzung

Thm. 3-HITTING SET ist lösbar in Zeit $\mathcal{O}(2.17\ldots^k + kn)$.

$k = \ldots$	10	15	20	25	30
3^k	60 ms	15 s	3487 s $\approx$ 1 h	847289 s $\approx$ 10 d	57192 h $\approx$ 6.5 y
2.18^k	3 ms	0.12 s	6 s	290 s $\approx$ 4 min	4 h

(Annahme: 10^6 Suchbaumknoten pro Sek.)

The Curse of Combinatorics Eine zweidimensionale Sicht



Ziel:

- exakte Algorithmen mit
- Laufzeitgarantien

Methoden: (u.a.)

- “Problemkerne” (Datenreduktion)
- Suchbäume

Parameterisierte Algorithmik

Die Klasse FPT (fixed parameter tractable) enthält Sprachen $L \subseteq \Sigma^* \times \mathbb{N}$, für die es einen Algorithmus gibt, der die Frage "$(x, k) \in L$?" in Zeit

$$f(k) \cdot p(|x|)$$

entscheidet (f bel., p Polynom).

Satz. Gleichwertig hiermit ist: Es gibt einen Problemkern (x', k') zu Instanz (x, k) mit $k', |x'| \in \mathcal{O}(g(k))$ für g beliebig.
Die zugehörige Problemkernreduktion ist in Polynomzeit berechenbar.

Mehr ?! $\rightsquigarrow$ Spezialvorlesung Parameterisierte Algorithmen

Algorithm 1 A brute force $\mathbf{FPT}$ algorithm from kernelization

Input(s): kernelization function K, a brute-force solving algorithm $\mathcal{A}$ for $\mathcal{P}$, instance (I, k) of $\mathcal{P}$

Output(s): solve (I, k) in $\mathbf{FPT}$-time

Compute kernel $(\mathrm{I}', \mathrm{k}') = \mathrm{K}(\mathrm{I}, \mathrm{k})$

Solve $(\mathrm{I}', \mathrm{k}')$ by brute force, i.e., return $\mathcal{A}(\mathrm{I}', \mathrm{k}')$.

Beispiel: Knotenüberdeckungsproblem gemäß S. Buss

R1: Ist v ein Knoten vom Grad größer k in der Graph-Instanz (G, k), so lösche v und erniedrige den Parameter um Eins; d.h., die resultierende Instanz ist $(G - v, k - 1)$.

R2: Lösche isolierte Knoten (ohne Parameteränderung).

Lemma 1 *Ist* (G, k) *eine Knotenüberdeckungs-Instanz mit isolierten Knoten* I. (G, k) *ist eine* YES*-Instanz gdw.* $(G - I, k)$ *ist eine* YES*-Instanz.*

Wie wird aus Datenreduktionsregel(n) eine Kernreduktion ?

Algorithm 2 A kernelization algorithm for VC, called Buss-kernel

Input(s): A VC instance (G, k)

Output(s): an instance (G', k') with $k' \leq k$, $|E(G')|, |V(G')| \leq (k')^2$, such that (G, k) is a YES-instance of VC iff (G', k') is a YES-instance

if possible **then**
  Apply Rule R1 or Rule R2; producing instance (G', k').
  return Buss-kernel(G', k').
else if $|E(G)| > k^2 \vee k < 0$ **then**
  return $((\{x, y\}, \{\{x, y\}\}), 0)$ {encoding NO}
else
  return (G, k)

Suchbaumalgorithmus Laufzeit: $\mathcal{O}(2^k \cdot p(n))$.

Algorithm 3 A simple search tree algorithm, called VCMH

Input(s): a graph $G = (V, E)$, a positive integer k

Output(s): YES if there is a vertex cover $C \subseteq V$, $|C| \leq k$, (and it will implicitly produce such a small cover then) or NO if no vertex cover of size at most k exists.

if $k \leq 0$ and $E \neq \emptyset$ **then**
  return NO
else if $k \geq 0$ and $E = \emptyset$ **then**
  return YES
else
  Choose edge $e = \{x, y\} \in E$
  if VCMH$(G - x, k - 1)$ **then**
    return YES
  else
    return VCMH$(G - y, k - 1)$

Reduktionen

Es seien (P, k) und (P', k') parameterisierte Probleme (formal gegeben als Sprachen über den Alphabeten Σ bzw. Σ').
Eine **FPT**-*(many-one)-Reduktion* von (P, k) auf (P', k') ist eine Abbildung $R : (\Sigma^* \times \mathbb{N}) \to ((\Sigma')^* \times \mathbb{N})$ mit:
(1) $\forall (x, k) \in \Sigma^* \times \mathbb{N} : ((x, k) \in L(P) \iff R(x, k) \in L(P'))$
(2) R ist in Zeit $f(k) \cdot p(|x|)$ berechenbar für bel. f und Polynom p.
(3) Es gibt Funktion $g : \mathbb{N} \to \mathbb{N}$, sodass $\forall k \in \mathbb{N} : k' \leq g(k)$ für alle x, sodass $R(x, k) = (x', k')$.
Schreibweise: $(P, k) \leq_{\mathbf{FPT}} (P', k')$.

FPT und parameterisierte Reduktionen

Satz 2 $\mathbf{FPT}$ *ist abgeschlossen unter* $\mathbf{FPT}$*-Reduktionen, d.h., gilt* $(P, k) \leq^{FPT} (P', k')$ *und* $(P', k') \in \mathbf{FPT}$*, so auch* $(P, k) \in \mathbf{FPT}$*.*

Beweis: Betrachte Reduktion R, die in Zeit $f(k) \cdot p(|x|)$ eine Instanz (x, k) von (P, k) auf eine Instanz (x', k') von (P', k') transformiert, wobei $k' \leq g(k)$. Die Instanz (x', k') kann in Zeit $h(k') \cdot q(|x'|)$ gelöst werden. Die Gesamtlaufzeit dieses Algorithmus ist:

$$f(k) \cdot p(|x|) + h(k') \cdot q(|x'|) \leq f(k) \cdot p(|x|) + h(g(k)) \cdot q(f(k) \cdot p(|x|)) \leq f'(k) \cdot p'(|x|)$$

FPT und parameterisierte Reduktionen

Lemma 3 *Die Relation* $\leq^{\mathrm{FPT}}$ *ist eine Quasiordnung, also reflexiv und transitiv.*

Es ist also sinnvoll, den Begriff der **FPT**-*Äquivalenz* $\equiv^{\mathrm{FPT}}$ einzuführen.

Lemma 4 *Ist* (P, k) *irgendein nicht-triviales Problem aus* **FPT** *und ist* $(\mathrm{P}', \mathrm{k}')$ *ein nicht-triviales parameterisertes Problem, so gilt:*
$(\mathrm{P}', \mathrm{k}') \in \mathbf{FPT}$ *gdw.* $(\mathrm{P}, \mathrm{k}) \equiv^{\mathrm{FPT}} (\mathrm{P}', \mathrm{k}')$.

Der eingeführte Begriff der **FPT**-Reduktion scheint also sinnvoll zu sein.

Beispiele für parameterisierte Reduktionen I

Lemma 5 *Das Problem, Cliquen der Mindestgröße* k *in Graphen zu finden, ist* **FPT***-äquivalent zu dem Problem, unabhängige Mengen der Mindestgröße* k *in Graphen zu finden.*

Das *Graph-Komplement* $\bar{G}$ von $G = (V, E)$ ist definiert durch:

$$\bar{G} = (V, \bar{E}) \quad \text{mit} \quad vw \in \bar{E} \iff (v \neq w \wedge vw \notin E)$$

Die Reduktion $(G, k) \mapsto (\bar{G}, k)$ leistet das Gewünschte (beide Richtungen).

Hinweis: Die Abbildung, die einer Instanz ihr parameterisiertes Dual zuordnet, ist i.d.R. keine parameterisierte Reduktion. Das gilt insbesondere für den bekannten Zusammenhang zwischen VERTEX COVER und INDEPENDENT SET.

Beispiele für parameterisierte Reduktionen II

Lemma 6 *Das Problem, dominierende Mengen der Höchstgröße* k *in Graphen zu finden, ist* $\mathbf{FPT}$*-äquivalent zu dem Problem, eine Knotenüberdeckung der Höchstgröße* k *in einem Hypergraphen zu finden.*

Beweis: Statt eine dominierende Menge $D \subseteq V$ in einem Graphen $G = (V, E)$ zu suchen, kann man auch nach einer Knotenüberdeckung in dem Hypergraphen $H = (V, \{N_G[v] \mid v \in V\})$ fahnden.
Statt nach einer Knotenüberdeckung $C \subseteq V$ im Hypergraphen $H = (V, E)$ zu suchen, kann man auch nach einer dominierenden Menge im Graphen $G = (V \cup E, E_1 \cup E_2)$ fahnden mit $E_1 = \{ve \mid v \in V, e \in E, v \in e\}$ und $E_2 = \{uv \mid u, v \in V, u \neq v\}$.
Beobachte:
(1) Eine dominierende Menge D mit $D \cap E \neq \emptyset$ kann man ersetzen durch $D' = (D \cap V) \cup \{v_e \mid \exists e \in D\}$ mit $|D'| \leq |D|$, wobei v_e ein willkürlich aber festes $v \in e$ meint.
(2) Eine dominierende Menge $D \subseteq V$ in G ist auch eine Knotenüberdeckung in H.

Das Halteproblem auf Turing-Maschinen

ist einer der natürlichen Kandidaten, um “harte Probleme” zu definieren.
In der “klassischen Berechenbarkeit” ist das allgemeine Halteproblem das typische Beispiel für ein unentscheidbares Problem.
Schränkt man die Schrittzahl, die eine Turing-Maschine machen darf, geeignet ein, so gelangt man in der “klassischen Komplexitätstheorie” zu typischen NP-harten Problemen.
Man kann hierbei sowohl Ein- als auch Mehrband-Turing-Maschinen betrachten.
Die einfachste Variante ist vielleicht:
KURZE NTM-AKZEPTANZ

Eingabe: Einband-NTM M, Eingabe x
Parameter: $k \in \mathbb{N}$
Frage: Gibt es Berechnung von M auf x, die in $\leq k$ Schritten akzeptiert?

Das Halteproblem auf Turing-Maschinen: Glaubensfragen

Aufgabe: Wie kann man eine nichtdeterministische TM durch eine deterministische simulieren ?
Am einfachsten geht das durch systematisches Durchprobieren aller möglichen Verzweigungen.
$\rightsquigarrow$ Für das Problem NTM-Akzeptanz führt das auf eine (deterministische) Laufzeit von n^k für ein von M abhängiges n.
Jedenfalls "auf diese Weise" liegt das Problem nicht in $\mathbf{FPT}$.
Eine ähnliche Intuition, die einen dazu verleitet anzunehmen, dass KURZE NTM-AKZEPTANZ nicht in Polynomzeit (auf deterministischen Maschinen) gelöst werden kann, führt einen auch auf den Gedanken, das "geht nicht" in $\mathbf{FPT}$-Zeit.

$\rightsquigarrow$ *W[1]*: Klasse von parameterisierten Problemen (P, k) mit
$(P, k) \leq^{FPT}$ KURZE NTM-AKZEPTANZ.

Das Halteproblem auf Turing-Maschinen: Glaubensfragen

Wenn wir nicht glauben, dass KURZE NTM-AKZEPTANZ in $\mathbf{FPT}$ liegt, heißt das:

(1) Wir vermuten, dass es Probleme gibt, die in W[1] liegen, aber nicht in $\mathbf{FPT}$.
Ein Kandidat dafür ist KURZE NTM-AKZEPTANZ.
(2) Da $\mathbf{FPT}$ unter $\mathbf{FPT}$-Reduktionen abgeschlossen ist, liegt es nahe, Folgendes zu definieren:
(P, k) *W[1]-hart*: gdw. KURZE NTM-AKZEPTANZ $\leq^{\mathrm{FPT}} (P, k)$.
(P, k) *W[1]-vollständig*: gdw. KURZE NTM-AKZEPTANZ $\equiv^{\mathrm{FPT}} (P, k)$.

Vergleichen Sie mit den Ihnen bekannten “klassischen Komplexitätsbegriffen” !
Entsprechende Sprechweisen verwenden wir für die anderen noch einzuführenden parameterisierten Komplexitätsklassen.

Ein Beispiel für ein W[1]-Problem

Klar: $(P, k) \in \mathbf{FPT} \implies (P, k) \in W[1]$.
Wir erwähnten schon: UNABHÄNGIGE MENGEN ist "wohl nicht" in **FPT**.

Lemma 7 UNABHÄNGIGE MENGEN *liegt in W[1].*

Unsere Reduktion speichert die Adjazenzrelation von G in der Übergangstabelle von M_G.
Dabei genehmigt sich die Reduktion für jeden Knoten ein Extra-Eingabezeichen.
M_G rät in den ersten k Schritten k Knoten und schreibt sie auf sein Arbeitsband.
In den folgenden $\mathcal{O}(k^2)$ Schritten [bitte ausführen!] überprüft M_G (deterministisch!), ob die geratene Knotenmenge eine unabhängige Menge bildet.
Es gilt: G enthält unabhängige Menge der Größe k gdw. M_G akzeptiert das leere Wort in $f(k) \in \mathcal{O}(k^2)$ Schritten.
Mitteilung: UNABHÄNGIGE MENGEN ist W[1]-vollständig.
Folgerung: CLIQUE ist W[1]-vollständig.

Die A-Hierarchie: Das parameterisierte Analogon zur Polynomzeithierarchie

Es sei Φ eine Menge von Formeln.

MODEL-CHECKING(Φ)

Eingabe: Struktur $\mathcal{S}$, Formel $\phi \in \Phi$
Parameter: Größe von ϕ
Frage: Gilt $\phi(\mathcal{S}) \neq \emptyset$?

Sei $A[t]$ die Klasse aller Probleme, die sich auf MODEL-CHECKING(Σ_t) **FPT**-reduzieren lassen.

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

Die Abbildung $(G, k) \mapsto (G, is_k)$ ist eine **FPT**-Reduktion vom natürlich parameterisierten Cliquenproblem auf MODEL-CHECKING(Σ_1).

Zusammenfassung

Wir haben eine neue “schöne Klasse”: $\mathbf{FPT}$
Wir haben einen geeigneten Reduktionsbegriff.
Wir haben “böse Klassen”: W[1] und die ganze A-Hierarchie.
Man kann zeigen: W[1] und $\mathrm{A}[\mathrm{t}]$ sind gegen $\mathbf{FPT}$-Reduktionen abgeschlossen.

Wir glauben daran, dass $\mathbf{FPT} \subset W[1]$ gilt (ohne einen Beweis zu besitzen).

$\rightsquigarrow$ Eine sinnvolle Ergänzung der klassischen Komplexitätstheorie wurde gefunden: die *Parameterisierte Komplexitätstheorie*.
Einen sehr guten Zugang liefert das Buch von Flum und Grohe.