5. Model-Checking für FO und FO+MOD auf Klassen von beschränkter lokaler Baumweite

Definition 5.1 (beschränkte lokale Baumweite)
Sei σ eine Signatur.

(a) Sei $f : \mathbb{N} \to \mathbb{N}$.

- Eine σ-Struktur $\mathcal{A}$ hat eine durch f beschränkte lokale Baumweite, falls f.a. $r \in \mathbb{N}$ und alle $a \in A$ gilt:

$$bw\left(N_r^{\mathcal{A}}(a)\right) \leq f(r)$$

D.h. Für jede Zahl r ist $f(r)$ eine obere Schranke für die Baumweite der r-Nachbarschaften von Elementen im Universum von $\mathcal{A}$.

- Eine Klasse C von σ-Strukturen hat eine durch f beschränkte lokale Baumweite, falls jedes $\mathcal{A} \in C$ eine durch f beschränkte lokale Baumweite hat.

(b) Eine Klasse C von σ-Strukturen hat beschränkte lokale Baumweite, falls es eine Funktion $f : \mathbb{N} \to \mathbb{N}$ gibt, s.d. C eine durch f beschränkte lokale Baumweite hat.

Beispiel 5.2

Sei σ eine Signatur.

(a) Sei $w \in \mathbb{N}$.
Die Klasse $C_{bw \leq w}$ aller σ-Strukturen $\mathcal{A}$ mit $bw(\mathcal{A}) \leq w$ hat beschränkte lokale Baumweite; die lokale Baumweite ist beschränkt durch die Funktion $f: \mathbb{N} \to \mathbb{N}$ mit $f(r) := w$ f.a. $r \in \mathbb{N}$.

(b) Sei $d \in \mathbb{N}$.
Die Klasse $C_{Grad \leq d}$ aller σ-Strukturen $\mathcal{A}$ vom Grad $\leq d$ hat beschränkte lokale Baumweite; die lokale Baumweite ist beschränkt durch die Funktion $f: \mathbb{N} \to \mathbb{N}$ mit $f(r) := v_d(r) \underset{\text{Lemma 0.2}}{\overset{\text{Def. in}}{=}} 1 + d \cdot \sum_{i=0}^{r-1} (d-1)^i$

Beweis: Gemäß Lemma 0.2 gilt für jede σ-Struktur $\mathcal{A}$ vom Grad $\leq d$, jedes $r \in \mathbb{N}$ und jedes $a \in A$, dass $|N_r^{\mathcal{A}}(a)| \leq v_d(r)$ ist.
Gemäß Beispiel 4.4 ist
$$bw(W_r^{\mathcal{A}}(a)) \leq |N_r^{\mathcal{A}}(a)| - 1.$$
Insbes. gilt also: $bw(W_r^{\mathcal{A}}(a)) \leq v_d(r)$. □

(c) (Hier ohne Beweis) Die Klasse C_{planar} aller σ-Strukturen $\mathcal{A}$, deren Gaifman-Graph planar ist, hat beschränkte lokale Baumweite.

(d) Aus (b) (bzw. analog aus (c)) folgt direkt, dass die Klasse $\{G_{n\times n} : n \in \mathbb{N}_{\geq 1}\}$ aller quadratischen Gitter beschränkte lokale Baumweite besitzt.

Ziel für den Rest dieses Kapitels ist, das folgende algorithmische Metatheorem zu beweisen.

Theorem 5.3

Sei σ eine Signatur, sei φ ein FO+MOD[σ]-Satz und sei C eine Klasse von σ-Strukturen mit lokal beschränkter Baumweite.
Dann ist das Problem

$\text{EVAL}_{\varphi,C}$	
Eingabe:	$\mathcal{A} \in C$
Frage:	Gilt $\mathcal{A} \models \varphi$?

in Pseudo-Linearzeit lösbar, d.h. für jedes $k \in \mathbb{N}_{\geq 1}$ gibt es einen Algorithmus, der das Problem $\text{EVAL}_{\varphi,C}$ in Zeit $O(\|\mathcal{A}\|^{1+\frac{1}{k}})$ löst.

Bemerkung:
Die Aussage von Theorem 5.3 für FO statt FO+MOD wurde von Frick und Grohe (2001) bewiesen; die Verallgemeinerung zu FO+MOD erfolgte in einer Arbeit von Kuske und Schweikardt (ICALP 2018).

Der Beweis von Theorem 5.3 verwendet das im folgenden vorgestellte Konzept der Nachbarschaftsüberdeckungen.

Definition 5.4 (Nachbarschaftsüberdeckung)
Sei σ eine Signatur, sei $\mathcal{A}$ eine σ-Struktur, sei $r, s \in \mathbb{N}$.
Eine (r,s)-Nachbarschaftsüberdeckung (kurz: Nbü) von $\mathcal{A}$ ist eine Familie $\mathcal{W}$ von Teilmengen von A, s.d. gilt:

(1) F.a. $a \in A$ ex. $N \in \mathcal{W}$ s.d. $N_r^{\mathcal{A}}(a) \subseteq N$,

und

(2) f.a. $N \in \mathcal{W}$ ex. $b \in A$ s.d. $N \subseteq N_s^{\mathcal{A}}(b)$.

Die Größe $\|\mathcal{W}\|$ von $\mathcal{W}$ ist definiert als

$$\|\mathcal{W}\| := \sum_{N \in \mathcal{W}} |N|.$$

Lemma 5.5 (Lemma von Peleg, 1993)
Sei $k \in \mathbb{N}_{\geq 1}$.
Es gibt einen Algorithmus, der bei Eingabe eines ungerichteten Graphen G und einer Zahl $r \in \mathbb{N}_{\geq 1}$ eine $(r, 2kr)$-Nbü $\mathcal{W}$ von G der Größe

$$\|\mathcal{W}\| \leq |V(G)|^{1+1/k} \quad \text{in Zeit } O\Big(\sum_{N \in \mathcal{W}} \|G[N]\|\Big)$$

berechnet.
Hierbei ist die Größe $\|H\|$ eines ungerichteten Graphen H definiert als $\|H\| := |V(H)| + |E(H)|$.

Beweis: Sei $k \in \mathbb{N}_{\geq 1}$ fest

Bei Eingabe von G und r geht der Algorithmus wie folgt vor.

1. $W_0 := \emptyset$, $V_0 := V(G)$, $i := 0$, $n := |V(G)|$
2. while $V_i \neq \emptyset$ do
3. $\quad i := i+1$
4. $\quad$ Wähle ein beliebiges $b \in V_{i-1}$
5. $\quad N_{i,0} := \{b\}$, $j := 0$
6. $\quad$ repeat
7. $\quad\quad j := j+1$
8. $\quad\quad L_{i,j} := N_r^G(N_{i,j-1}) \cap V_{i-1}$
9. $\quad\quad N_{i,j} := N_r^G(L_{i,j})$
10. $\quad$ until $|N_{i,j}| \leq n^{1/k} \cdot |N_{i,j-1}|$
11. $\quad j_i := j$
12. $\quad N_i := N_{i,j_i}$
13. $\quad W_i := W_{i-1} \cup \{N_i\}$
14. $\quad V_i := V_{i-1} \setminus L_{i,j_i}$
15. endwhile
16. Ausgabe: $W := W_i$; $i_{max} := i$

Wir analysieren im Folgenden einen Lauf des Algorithmus bei Eingabe von G und r. Es sei $n := |V(G)|$.

Behauptung 1: Für jede Zahl i gilt: Wenn es einen i-ten Durchlauf durch die while-Schleife gibt, dann gibt es innerhalb des i-ten Durchlaufs durch die while-Schleife höchstens k Durchläufe durch die repeat-Schleife, und am Ende des i-ten Durchlaufs durch die while-Schleife ist $j_i \in \{1,\dots,k\}$, $|N_i| \leq i$, $|V_i| \leq n-i$ und $V_i \subsetneq V_{i-1}$ und $W_i \supseteq W_{i-1}$.

Beweis: Gemäß Zeile 5 des Algorithmus ist $|N_{i,0}| = 1$.

Gemäß Zeilen 6-10 gilt für jede Zahl $j \geq 1$, für die es einen j-ten und einen $(j+1)$-ten Durchlauf durch die repeat-Schleife gibt, dass $|N_{i,j}| > n^{1/k} \cdot |N_{i,j-1}|$.

Per Induktion ergibt sich, dass für jede solche Zahl j gilt:

$$|N_{i,j}| > n^{1/k} \cdot |N_{i,j-1}| > n^{1/k} \cdot n^{1/k} \cdot |N_{i,j-2}| > \dots > \left(n^{1/k}\right)^j \cdot \underbrace{|N_{i,0}|}_{=1} = n^{j/k}$$

Außerdem ist $N_{i,j} \subseteq V(G)$ und somit $|N_{i,j}| \leq n$.

Also gilt: $n \geq |N_{i,j}| > n^{j/k}$.

Daher muss $j < k$ sein. D.h.: Für jede Zahl $j \geq 1$, für die es einen j-ten und einen $(j+1)$-ten Durchlauf durch die repeat-Schleife gibt, ist $j \leq k-1$. Es gibt also höchstens k Durchläufe durch die repeat-Schleife insges. ist am Ende des i-ten Durchlaufs durch die while-Schleife $j_i \in \{1,\dots,k\}$.

Außerdem gilt gemäß Zeile 1: $|W_0| = 0$ und $|V_0| = n$; und per Induktion nach i gilt gemäß Zeile 13: $|W_i| \leq |W_{i-1}| + 1 \overset{\text{Ind.ann.}}{\leq} (i-1)+1 = i$.

Gemäß Zeile 14 gilt: $|V_i| = |V_{i-1}| - |L_{i,j_i} \cap V_{i-1}| \overset{\text{Ind.ann.}}{\leq} n-(i-1) - |L_{i,j_i} \cap V_{i-1}|$

Um den Beweis von Beh 1 abzuschließen reicht es also zu zeigen, dass $|L_{i,j_i} \cap V_{i-1}| \geq 1$ ist. Sei $b \in V_{i-1}$ dasjenige Element, das zu Beginn des i-ten Durchlaufs durch die while-Schleife in Zeile 4 gewählt wird.

Per Induktion nach j zeigen wir, dass f.a. $j \in \{1,\dots,j_i\}$ gilt: $b \in L_{i,j}$:

$j=1$: $L_{i,1} \overset{\text{Zeile 8}}{=} N_r^G(N_{i,0}) \cap V_{i-1} \overset{\text{Zeile 5}}{=} N_r^G(b) \cap V_{i-1} \ni b$. ✓

$j-1 \to j$: $L_{i,j} \underset{\text{Zeile 8}}{=} N_r^{\subseteq}(N_{i,j-1}) \cap V_{i-1}$ und

$N_{i,j-1} \underset{\text{Zeile 9}}{=} N_r^{\subseteq}(L_{i,j-1})$

Gemäß Ind.annahme ist $b \in L_{i,j-1}$ — also gilt auch: $b \in N_{i,j-1}$ und $b \in N_r^{\subseteq}(N_{i,j-1})$. Da außerdem $b \in V_{i-1}$ ist, gilt: $b \in L_{i,j}$. ✓

Insbes. haben wir gezeigt, dass $b \in L_{i,j_i} \cap V_{i-1}$.

□ Beh 1.

Beachte: Aus Beh 1 folgt insbes., dass der Algorithmus nach max. n Durchläufen durch die while-Schleife terminiert.

Im Folgenden seien W und i_{max} die Ausgabe des Algorithmus.

Behauptung 2: $\forall a \in V(G)$ $\exists N \in W$ s.d. $N_r^{\subseteq}(a) \subseteq N$

Beweis: Betrachte ein beliebiges $a \in V(G)$. Laut Beh. 1 und den Zeilen 1 und 2 des Algorithmus ist

$$V(G) = V_0 \supsetneq V_1 \supsetneq \cdots \supsetneq V_{i_{max}} = \emptyset$$

Somit existiert ein $i \in \{1, \dots, i_{max}\}$ s.d. $a \in V_{i-1} \setminus V_i$.

Wir betrachten den i-ten Durchlauf durch die while-Schleife.

Gemäß Zeile 14 ist $a \in L_{i,j_i}$. Gemäß Zeilen 3 und 11-13 ist

$$N_i = N_{i,j_i} = N_r^{\subseteq}(L_{i,j_i}) \in W_i$$

Insbes ist also $N_r^{\subseteq}(a) \subseteq N_i$ (da $a \in L_{i,j_i}$).

Und wegen $W_i \subseteq W$ ist $N_i \in W$.

□ Beh 2.

Behauptung 3: $\forall N \in \mathcal{W}$ $\exists b \in V(G)$ s.d. $N \subseteq N^{\uparrow G}_{2kr}(b)$

Beweis: Betrachte ein beliebiges $N \in \mathcal{W}$.
Gemäß Aufbau des Algorithmus gibt es ein $i \in \{1, \dots, i_{max}\}$
s.d. $N = N_i$
Sei b das zu Beginn des i-ten Durchlaufs durch die while-Schleife in Zeile 4 gewählte Element aus V_{i-1}.
Aus Beh 1 wissen wir, dass $j_i \in \{1, \dots, k\}$ ist.
Per Induktion nach j zeigen wir, dass $\forall j \in \{1, \dots, j_i\}$ gilt:

$$N_{i,j} \subseteq N^{\uparrow G}_{2jr}(b)$$

– daraus folgt dann direkt Beh 3, da $j_i \leq k$ und $N_i = N_{i,j_i}$ ist.

$\underline{j=1}$: $N_{i,1} = N^{\uparrow G}_r(L_{i,1})$ für $L_{i,1} = N^{\uparrow G}_r(b) \cap V_{i-1}$. Somit ist

$N_{i,1} = N^{\uparrow G}_r\left(N^{\uparrow G}_r(b) \cap V_{i-1}\right) \subseteq N^{\uparrow G}_r\left(N^{\uparrow G}_r(b)\right) = N^{\uparrow G}_{2r}(b)$. ✓

$\underline{j-1 \to j}$: Sei $j \in \{2, \dots, j_i\}$. Gemäß Induktionsannahme ist
$N_{i,j-1} \subseteq N^{\uparrow G}_{2(j-1)r}(b)$. Zu zeigen: $N_{i,j} \subseteq N^{\uparrow G}_{2jr}(b)$.

Gemäß Zeilen 8 und 9 des Algorithmus ist

$$\begin{aligned}
N_{i,j} &= N^{\uparrow G}_r\left(N^{\uparrow G}_r(N_{i,j-1}) \cap V_{i-1}\right) \\
&\subseteq N^{\uparrow G}_r\left(N^{\uparrow G}_r(N_{i,j-1})\right) \\
&= N^{\uparrow G}_{2r}(N_{i,j-1}) \\
&\underset{\text{Ind.ann.}}{\subseteq} N^{\uparrow G}_{2r}\left(N^{\uparrow G}_{2(j-1)r}(b)\right) \\
&= N^{\uparrow G}_{2jr}(b) \qquad \checkmark
\end{aligned}$$

□ Beh 3

Behauptung 4: F.a. $i_1, i_2 \in \{1, \dots, i_{max}\}$ mit $i_1 \neq i_2$ gilt:

$$N_{i_1, j_{i_1}-1} \cap N_{i_2, j_{i_2}-1} = \emptyset$$

Beweis: Sei oBdA $i_1 < i_2$.
Wir gehen in 2 Schritten vor:

Schritt 1: Zeige, dass $N_{i_1, j_{i_1}-1} \cap N_r^G(V_{i_1}) = \emptyset$

Schritt 2: Zeige, dass $N_{i_2, j_{i_2}-1} \subseteq N_r^G(V_{i_2-1})$

Beh 4 folgt dann direkt, da wegen $i_1 \leq i_2 - 1$ gilt: $V_{i_2-1} \subseteq V_{i_1}$.
Es reicht also, Schritt 1 und Schritt 2 zu beweisen.

Zu Schritt 1:
Gemäß Zeilen 8 und 14 des Algo. für $i := i_1$ gilt:

$$L_{i_1, j_{i_1}} = N_r^G(N_{i_1, j_{i_1}-1}) \cap V_{i_1-1} \quad \text{und} \quad V_{i_1} = V_{i_1-1} \setminus L_{i_1, j_{i_1}}$$

Somit ist $N_r^G(N_{i_1, j_{i_1}-1}) \cap V_{i_1} = \emptyset$,

d.h.: $N_{i_1, j_{i_1}-1} \cap N_r^G(V_{i_1}) = \emptyset$. ✓(Schritt 1)

Zu Schritt 2:
Sei b das Element aus V_{i_2-1}, das zu Beginn des i_2-ten Durchlaufs durch die while-Schleife in Zeile 4 gewählt wurde.
Dann ist insbes. $N_{i_2,0} \underset{\text{Zeile 5}}{=} \{b\} \subseteq N_r^G(V_{i_2-1})$.
Und gemäß den Zeilen 6-10 des Algorithmus gilt f.a. $j \in \{1, \dots, j_{i_2}\}$:

$$L_{i_2, j} \subseteq V_{i_2-1} \quad \text{und} \quad N_{i_2 j} = N_r^G(L_{i_2 j}) \subseteq N_r^G(V_{i_2-1}).$$

Insbes. für $j := j_{i_2}-1$ folgt: $N_{i_2, j_{i_2}-1} \subseteq N_r^G(V_{i_2-1})$ ✓(Schritt 2)

Insgesamt schließt dies den Beweis von Beh. 4 ab

□ Beh 4

Behauptung 5 Es gilt: $\|\mathcal{N}\| \leq n^{1+1/k}$

Beweis:
Gemäß Aufbau des Algorithmus ist $\mathcal{N} = \{N_{i,0}, \dots, N_{i,i_{max}}\}$
Gemäß Zeilen 10-12 des Algorithmus gilt für jedes $i \in \{1, \dots, i_{max}\}$:

$$|N_{i,j}| \leq n^{1/k} \cdot |N_{i,j-1}|.$$

Somit gilt:

$$\|\mathcal{N}\| = \sum_{i=1}^{i_{max}} |N_{i,j}| \leq \sum_{i=1}^{i_{max}} n^{1/k} \cdot |N_{i,j-1}| = n^{1/k} \cdot \sum_{i=1}^{i_{max}} |N_{i,j-1}|$$

Aus Beh. 4 folgt, dass $\sum_{i=1}^{i_{max}} |N_{i,j-1}| = \left| \bigcup_{i=1}^{i_{max}} N_{i,j-1} \right|$.

Außerdem ist $\bigcup_{i=1}^{i_{max}} N_{i,j-1} \subseteq V(G)$, also $\left| \bigcup_{i=1}^{i_{max}} N_{i,j-1} \right| \leq n$.

Insgesamt erhalten wir:

$$\|\mathcal{N}\| \leq n^{1/k} \cdot n = n^{1+1/k}$$

□ Beh 5

Beachte:
Aus Beh 1 erhalten wir, dass der Algorithmus bei jeder Eingabe (G, r) terminiert.
Aus Beh 2 und Beh 3 folgt, dass die Ausgabe $\mathcal{N}$ des Algorithmus eine $(r, 2kr)$-Nbh von G ist.
Beh 5 besagt, dass $\mathcal{N}$ die Größe $\|\mathcal{N}\| \leq |V(G)|^{1+1/k}$ hat.
Um den Beweis von Lemma 5.5 abzuschließen müssen wir noch zeigen, dass der Algorithmus Laufzeit $O\left(\sum_{N \in \mathcal{N}} \|G[N]\|\right)$ hat.

Dazu genügt es, Folgendes zu zeigen:

Behauptung 6 Für jedes $i \in \{1, \dots, i_{max}\}$ gilt:
Der i-te Durchlauf durch die while-Schleife hat Laufzeit $O(\|G[N_i]\|)$.

Beweis: Übung!

Insgesamt schließt dies den Beweis von Lemma 5.5 ab

□ Lem 5.5

Durch Kombination von Lemma 5.5 und Satz B.10 erhalten wir:

Korollar 5.6
Sei σ eine Signatur, sei C eine Klasse von σ-Strukturen mit beschränkter lokaler Baumweite und sei $k, r \in \mathbb{N}_{\geq 1}$.

Dann gibt es einen Algorithmus, der bei Eingabe einer σ-Struktur $A \in C$ eine $(r, 2kr)$-NBü N von A der Größe $\|N\| \leq |A|^{1+1/k}$ in Zeit $O(|A|^{1+\frac{1}{k}})$ berechnet.

Beweis: Sei $f: \mathbb{N} \to \mathbb{N}$ so dass C eine durch f beschränkte lokale Baumweite hat. Sei $w := f(2kr)$.
Sei $c := c(w)$ gemäß Satz 4.10 gewählt, so dass für jeden ungerichteten Graphen G' der Baumweite $\leq w$ gilt:
$\|G'\| \overset{\text{def.}}{=} |V(G')| + |E(G')| \overset{\text{Satz 4.10}}{\leq} c \cdot |V(G')|$.

Unser Algorithmus berechnet zunächst den Gaifman-Graphen G von A und nutzt dann den Algorithmus aus dem Lemma von Peleg (Lemma 5.5), um eine $(r, 2kr)$-NBü

$\mathcal{W}$ von G zu berechnen. Gemäß Def. 5.4 ist dies auch eine $(r, 2kr)$-NSü von A.
Gemäß Lemma 5.5 hat $\mathcal{W}$ die Größe $\|\mathcal{W}\| \leq |A|^{1+1/k}$ und wird in Zeit $O\left(\sum_{N \in \mathcal{W}} \|G[N]\|\right)$ berechnet

Da $\mathcal{W}$ eine $(r, 2kr)$-NÜb von G ist, gilt für jedes $N \in \mathcal{W}$:
ex $b \in A$ s.d. $N \subseteq N^G_{2kr}(b) = N^A_{2kr}(b)$. Somit ist

$$bw(G[N]) = bw(A[N]) \leq bw\left(N^A_{2kr}(b)\right) \leq f(2kr) = k$$

($A \in \mathcal{C}$ und $\mathcal{C}$ hat durch f beschränkte lokale Baumweite)

Gemäß Satz 4.10 ist daher $\|G[N]\| \leq c \cdot |N|$.
Die Laufzeit zur Berechnung von $\mathcal{W}$ ist somit

$$O\left(\sum_{N \in \mathcal{W}} \|G[N]\|\right) = O\left(\sum_{N \in \mathcal{W}} c \cdot |N|\right) = O\left(c \cdot \sum_{N \in \mathcal{W}} |N|\right)$$

$$= O(c \cdot \|\mathcal{W}\|) = O\left(c \cdot |A|^{1+1/k}\right) = O\left(|A|^{1+1/k}\right).$$

Um den Beweis abzuschließen muss nur noch gezeigt werden, dass bei Eingabe von $A \in \mathcal{C}$ der Gaifman-Graph G von A in Zeit $O\left(|A|^{1+1/k}\right)$ berechnet werden kann.
Details dazu: Übung!

□ Korollar 5.6

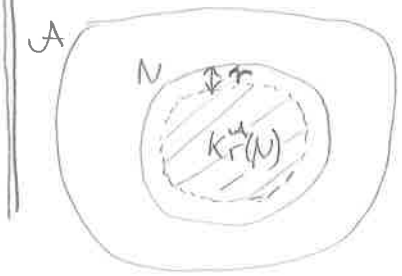
Ein weiteres Konzept, das zum Beweis von Theorem 5.3 benutzt wird, ist der folgende Begriff.

Definition 5.7

Sei $\mathcal{A}$ eine σ-Struktur, sei $N \subseteq A$ und sei $r \in \mathbb{N}$.
Der r-Kern von N in $\mathcal{A}$ ist die Menge

$$K_r^{\mathcal{A}}(N) := \{a \in N : N_r^{\mathcal{A}}(a) \subseteq N\}$$

Skizze:



Beobachtung 5.8

Für alle $r \in \mathbb{N}$ gilt: $K_{r+1}^{\mathcal{A}}(N) = K_1^{\mathcal{A}}\left(K_r^{\mathcal{A}}(N)\right)$.

Beweis: Sei $r \in \mathbb{N}$ und sei $M := K_r^{\mathcal{A}}(N)$.
Zu zeigen: $K_{r+1}^{\mathcal{A}}(N) = K_1^{\mathcal{A}}(M)$.

"$\subseteq$": Sei $a \in K_{r+1}^{\mathcal{A}}(N)$. Zu zeigen: $a \in K_1^{\mathcal{A}}(M)$.

Wegen $a \in K_{r+1}^{\mathcal{A}}(N)$ gilt: $a \in N$ und $N_{r+1}^{\mathcal{A}}(a) \subseteq N$.
Wegen $N_r^{\mathcal{A}}(a) \subseteq N_{r+1}^{\mathcal{A}}(a)$ gilt also auch: $N_r^{\mathcal{A}}(a) \subseteq N$.
Somit ist $a \in K_r^{\mathcal{A}}(N) = M$.
Wir müssen noch zeigen, dass $N_1^{\mathcal{A}}(a) \subseteq M$ ist.
Betrachte ein beliebiges $b \in N_1^{\mathcal{A}}(a)$. Wegen $N_1^{\mathcal{A}}(a) \subseteq N_{r+1}^{\mathcal{A}}(a) \subseteq N$ ist $b \in N$. Außerdem ist $N_r^{\mathcal{A}}(b) \subseteq N_{r+1}^{\mathcal{A}}(a) \subseteq N$.
Somit ist $b \in K_r^{\mathcal{A}}(N) = M$. Es gilt also: $N_1^{\mathcal{A}}(a) \subseteq M$ ✓

"$\supseteq$": Sei $a \in K_1^{\mathcal{A}}(M)$. Zu zeigen: $a \in K_{r+1}^{\mathcal{A}}(N)$.

Wegen $a \in K_1^{\mathcal{A}}(M)$ gilt: $a \in M$ und $N_1^{\mathcal{A}}(a) \subseteq M = K_r^{\mathcal{A}}(N) \subseteq N$.
Also gilt insbes: $a \in N$. Wir müssen noch zeigen, dass $N_{r+1}^{\mathcal{A}}(a) \subseteq N$ ist.
Betrachte ein beliebiges $b \in N_{r+1}^{\mathcal{A}}(a)$. Dann gibt es ein $c \in N_1^{\mathcal{A}}(a)$ s.d. $b \in N_r^{\mathcal{A}}(c)$. Wegen $N_1^{\mathcal{A}}(a) \subseteq M$ ist $c \in M = K_r^{\mathcal{A}}(N)$.
Also ist $N_r^{\mathcal{A}}(c) \subseteq N$. Somit ist $b \in N$.
Es gilt also: $N_{r+1}^{\mathcal{A}}(a) \subseteq N$. ✓

□

Lemma 5.9

Es gibt einen Algorithmus, der bei Eingabe eines ungerichteten Graphen G, einer Menge $N \subseteq V(G)$ und einer Zahl $r \in \mathbb{N}$ in Zeit $O(r \cdot \|G[N]\|)$ den r-Kern $K_r^G(N)$ berechnet.

Beweis: Wir geben hier den Algorithmus für $r = 1$ an; unter Verwendung von Beobachtung 5.8 kann daraus leicht ein Algorithmus für allgemeines $r \in \mathbb{N}$ konstruiert werden (Details: Übung).

Die Eingabe G soll als Adjazenzliste repräsentiert sein, und N soll als verkettete Liste der Elemente $a_1, a_2, \ldots, a_m$ gegeben sein, wobei $m := |N|$ und $N = \{a_1, \ldots, a_m\}$ gilt.

Unser Ziel ist, $K_1^G(N)$ in Zeit $O(\|G[N]\|)$ zu berechnen.

Wir nutzen hierfür den folgenden Algorithmus:

```
0) K := ∅
1) for i := 1 to m do
2)     ok := true
3)     für jeden Nachbarn b von a_i do
4)         teste, ob b ∈ N ist
5)         falls nein, setze ok := false und brich diese Schleife ab
6)     end
7)     falls ok = true, so füge a_i in K ein
8) end
9) stopp mit Ausgabe K
```

Man sieht leicht, dass die von diesem Algorithmus erzeugte Menge K genau die Menge $K_1^G(N)$ ist.

(*) Im Folgenden zeigen wir, wie man in Zeit $O(\|G[N]\|)$ eine Datenstruktur aufbauen kann, die Folgendes leistet: Bei Eingabe eines beliebigen $b \in V(G)$ kann man in Zeit $O(1)$ testen, ob $b \in N$ ist.

Unter Verwendung dieser Datenstruktur kann der Test in Zeile 4 jeweils in Zeit $O(1)$ erfolgen.
Insgesamt hat die Ausführung der Zeilen 0-9) eine Laufzeit $O\left(\sum_{i=1}^{m}\left(1+\sum_{\substack{b:\\ (a_i,b)\in E(G)\\ \text{und } b\in N}} 2\right)\right) = O(|N|+|E(G[N])|) = O(\|G[N]\|)$.

Wir müssen also nur noch (*) zeigen. Hierfür nutzen wir folgende, unter dem Stichwort "Lazy Array Initialization" bekannte Technik:

Sei $n := |V(G)|$ und $V(G) = \{1,\dots,n\}$.
Zur Erinnerung: $m = |N|$ und $N = \{a_1,\dots,a_m\}$.

Nutze zwei Arrays $A[1,\dots,n]$ und $B[1,\dots,m]$, von denen wir nicht wissen, welche initialen Einträge sie haben.
Der Aufbau unserer Datenstruktur erfolgt so:

```
for i := 1 to m do
    A[a_i] := i und B[i] := a_i
```

Das geht in Zeit $O(m) = O(|N|) = O(\|G[N]\|)$.
Bei Eingabe eines $b \in V(G)$ erfolgt der Test, ob $b \in N$ ist, so:

```
i := A[b]
if i ∉ {1,...,m} then Stopp mit Ausgabe "b ∉ N"
else if B[i] = b then Stopp mit Ausgabe "b ∈ N"
     else Stopp mit Ausgabe "b ∉ N".
```

Man kann sich leicht davon überzeugen (Details: Übung!), dass dies die korrekte Ausgabe liefert.

Insgesamt beendet dies den Beweis von Lemma 5.9. □

Wir können nun endlich Theorem 5.3 beweisen.

Beweis von Theorem 5.3:

Sei C eine Klasse von σ-Strukturen und sei $f: \mathbb{N} \to \mathbb{N}$ so, dass C eine durch f beschränkte lokale Baumweite hat. Sei $k \in \mathbb{N}_{\geq 1}$.
Sei φ ein FO+MOD[σ]-Satz.
Unser Ziel ist, einen Algorithmus mit Laufzeit $O(|A|^{1+\frac{1}{k}})$ zu finden, der bei Eingabe von $\mathcal{A} \in C$ entscheidet, ob $\mathcal{A} \models \varphi$ gilt.

Schritt 1: Nutze den Satz von Gaifman, um φ in einen äquivalenten FO+MOD[σ]-Satz γ in GNF zu transformieren. Dann ist γ eine Boolesche Kombination

- von basis-lokalen Sätzen χ der Form

$$\exists x_1 \dots \exists x_m \Big(\bigwedge_{1 \leq i < j \leq m} \operatorname{dist}(x_i, x_j) > 2r \wedge \bigwedge_{i=1}^{m} \lambda(x_i) \Big)$$

  mit $m \in \mathbb{N}_{\geq 1}$, $r \in \mathbb{N}$, $\lambda(x)$ r-lokale FO+MOD[σ]-Formel,
- und von lokalen FO+MOD-Zählsätzen ϑ der Form

$$\exists^{i \bmod m} x \; \lambda(x)$$

  mit $m \in \mathbb{N}$, $m \geq 2$, $i \in \{0, \dots, m-1\}$, $r \in \mathbb{N}$, $\lambda(x)$ r-lokale FO+MOD[σ]-Formel.

Man sieht leicht (Details: Übung), dass es genügt, für jeden basis-lokalen Satz der Form χ und für jeden lokalen FO+MOD-Zählsatz der Form ϑ einen Algorithmus mit Laufzeit $O(|A|^{1+\frac{1}{k}})$ zu finden, der bei Eingabe von $\mathcal{A} \in C$ entscheidet, ob $\mathcal{A}$ den Satz erfüllt.

Schritt 2: Betrachte einen lokalen FO+MOD-Zählsatz $\vartheta := \exists^{i \bmod m} x \; \lambda(x)$, wobei $\lambda(x)$ eine r-lokale FO+MOD[σ]-Formel ist.

Bei Eingabe von $\mathcal{A} \in C$ gehen wir wie folgt vor, um zu entscheiden, ob $\mathcal{A} \models \vartheta$ gilt:

1) Berechne eine $(r, 2kr)$-Nbh $\mathcal{N}$ von $\mathcal{A}$ der Größe $\|\mathcal{N}\| \leq |A|^{1+\frac{1}{k}}$ in Zeit $O(|A|^{1+\frac{1}{k}})$
(Nutze hierfür den Algorithmus aus Korollar 5.6)

2) for all $N \in \mathcal{N}$ do

3) berechne $K_r^{\mathcal{A}}(N)$
(gemäß Lemma 5.9 geht das in Zeit $O(r \cdot \|G_{\mathcal{A}}[N]\|)$)

4) berechne $P_N := \{a \in K_r^{\mathcal{A}}(N) : \mathcal{A}[N] \models \lambda[a]\}$
(Details hierzu siehe unten)

5) endfor

6) berechne $P := \bigcup_{N \in \mathcal{N}} P_N$ und berechne $|P|$

7) falls $|P| \equiv i \bmod m$, so Stopp mit Ausgabe "$\mathcal{A} \models \vartheta$"

8) sonst Stopp mit Ausgabe "$\mathcal{A} \not\models \vartheta$".

Laufzeit für Teile 3):

G_A ist der Gaifman-Graph von A;
$G_A[N]$ ist der durch N induzierte Subgraph von G_A.
Wegen $N \in W$ und da W eine $(r, 2kr)$-Nbh von A ist, gibt es ein $c \in A$ s.d. $N \subseteq N^A_{2kr}(c)$ ist.
Somit ist $G_A[N]$ ein Subgraph des Gaifman-Graphen von $N^A_{2kr}(c)$, und es gilt:

$$bw(G[N]) \leq bw(N^A_{2kr}(c)) \leq f(2kr) =: w$$

↑ da $A \in C$ und C eine durch f beschränkte lokale Baumweite hat

Wir nutzen Satz B.10 und erhalten eine Zahl $c(w) \in \mathbb{N}_{\geq 1}$, s.d. $\|G[N]\| \leq c(w) \cdot |V(G[N])| = c(w) \cdot |N|$ ist.
Somit kann Teile 3) mit Laufzeit $O(r \cdot c(w) \cdot |N|)$ abgearbeitet werden.

Laufzeit für Teile 4):

Analog wie eben erhalten wir, dass
$bw(A[N]) \leq bw(N^A_{2kr}(c)) \leq f(2kr) =: w$ ist.

Zu FO+MOD[σ]-Formel $\lambda(x)$ konstruieren wir eine äquivalente CMSO[σ]-Formel $\lambda'(x)$ und nutzen dann Theorem B.13 (Satz von Courcelle), um eine Zahl $c(\lambda', w) \in \mathbb{N}$ zu erhalten, so dass in Zeit $O(c(\lambda', w) \cdot |N|)$ die Menge

$$[\![\lambda'(x)]\!]^{A[N]} := \{ a \in N : A[N] \models \lambda'[a] \}$$ berechnet

werden kann. Es gilt: $P_N = [\![\lambda'(x)]\!]^{A[N]} \cap K^A_r(N)$.
Somit können wir P_N in Zeit $O(c(\lambda', w) \cdot |N|)$ berechnen.

Laufzeit für die Zeilen 2)-5):

$$\sum_{N \in \mathcal{N}} O\left(r \cdot c(\omega) \cdot |N| + c(\lambda', \omega) \cdot |N|\right)$$

$$= O\left((r \cdot c(\omega) + c(\lambda', \omega)) \cdot \underbrace{\sum_{N \in \mathcal{N}} |N|}_{= \|\mathcal{N}\| \leq |A|^{1+\frac{1}{k}}}\right) = O\left((r \cdot c(\omega) + c(\lambda', \omega)) \cdot |A|^{1+\frac{1}{k}}\right)$$

$$= O\left(|A|^{1+\frac{1}{k}}\right).$$

Laufzeit für Zeile 6):

Sei $n := |A|$ und $A = \{1, \dots, n\}$.

Wir verwenden ein Array $B[1, \dots, n]$ und gehen wie folgt vor:

6.1) for $i := 1$ to n do $B[i] := 0$ endfor
6.2) for $N \in \mathcal{N}$ do
6.3) for each $a \in P_N$ do $B[a] := 1$ endfor
6.4) endfor
6.5) $P := \emptyset$; anz $:= 0$
6.6) for $i := 1$ to n do
6.7) if $B[i] = 1$ then insert i into P and anz $:=$ anz $+ 1$
6.8) endfor
6.9) Stopp mit Ausgabe P und Ausgabe $|P| :=$ anz

Laufzeit
- für Zeilen 6.2)-6.4): $\sum_{N \in \mathcal{N}} \underbrace{|P_N|}_{\leq |N|} \leq \sum_{N \in \mathcal{N}} |N| = \|\mathcal{N}\| \leq |A|^{1+\frac{1}{k}}$
- für Zeilen 6.1) und Zeilen 6.5)-6.9): $O(n) = O(|A|)$

Gesamtlaufzeit für 6): $O\left(|A|^{1+\frac{1}{k}}\right)$

Gesamtlaufzeit für den Algorithmus zum Entscheiden, ob $\mathcal{A} \models \vartheta$:

$O(|A|^{1+\frac{1}{k}})$.

Korrektheit des Algorithmus:

Es genügt zu zeigen, dass für die vom Algorithmus berechnete Menge P gilt: $P = \{a \in A : \mathcal{A} \models \lambda[a]\}$.

"$\subseteq$": Betrachte ein beliebiges $a \in P$. Zu zeigen: $\mathcal{A} \models \lambda[a]$.

Wegen $a \in P$ und Teile 6) gilt: Es gibt ein $N \in \mathcal{N}$ s.d. $a \in P_N$.
Wegen Teile 4) gilt: $a \in K_r^{\mathcal{A}}(N)$ und $\mathcal{A}[N] \models \lambda[a]$.
Gemäß Definition von $K_r^{\mathcal{A}}(N)$ gilt: $N_r^{\mathcal{A}}(a) \subseteq N$.
Da $\lambda(x)$ r-lokal ist, gilt:

$$\mathcal{A} \models \lambda[a] \iff W_r^{\mathcal{A}}(a) \models \lambda[a] \underset{N_r^{\mathcal{A}}(a) \subseteq N}{\iff} \mathcal{A}[N] \models \lambda[a].$$

Wegen $\mathcal{A}[N] \models \lambda[a]$ gilt also auch: $\mathcal{A} \models \lambda[a]$ ✓

"$\supseteq$": Betrachte ein beliebiges $a \in A$ mit $\mathcal{A} \models \lambda[a]$. Zu zeigen: $a \in P$.

Da $\mathcal{N}$ eine $(r, 2kr)$-Nbl von $\mathcal{A}$ ist, gibt es ein $N \in \mathcal{N}$, so dass $N_r^{\mathcal{A}}(a) \subseteq N$ ist. Also ist $a \in K_r^{\mathcal{A}}(N)$.
Da $\lambda(x)$ r-lokal ist, gilt:

$$\mathcal{A} \models \lambda[a] \iff W_r^{\mathcal{A}}(a) \models \lambda[a] \underset{N_r^{\mathcal{A}}(a) \subseteq N}{\iff} \mathcal{A}[N] \models \lambda[a]$$

Wegen $\mathcal{A} \models \lambda[a]$ gilt also auch: $\mathcal{A}[N] \models \lambda[a]$.
Somit ist $a \in P_N$ (gemäß Teile 4), und es gilt: $a \in P$ (gemäß Teile 6)). ✓

Dies beendet Schritt 2.

Schritt 3: Betrachte einen basis-lokalen Satz

$$\chi := \exists x_1 \dots \exists x_m \Big(\bigwedge_{1 \le i < j \le m} \mathrm{dist}(x_i, x_j) > 2r \wedge \bigwedge_{i=1}^{m} \lambda(x_i) \Big),$$

wobei $\lambda(x)$ eine r-lokale FO+MOD[σ]-Formel ist.

Bei Eingabe von $\mathcal{A} \in C$ gehen wir wie folgt vor, um zu entscheiden, ob $\mathcal{A} \models \chi$ gilt:

1) Wir führen die Zeilen 1)–6) des Algorithmus von Schritt 2 aus, um in Zeit $O(|A|^{1+\frac{1}{k}})$ die Menge $P := \{a \in A : \mathcal{A} \models \lambda[a]\}$ zu berechnen.
2) Setze $\hat{r} := 2r$, $Q := P$, $\ell := 0$
3) While $Q \neq \emptyset$ und $\ell < m$ do
4) $\quad \ell := \ell + 1$
5) $\quad$ Wähle ein beliebiges $a_\ell \in Q$
6) $\quad Q := Q \setminus N^{\mathcal{A}}_{\hat{r}}(a_\ell)$
7) endwhile
8) if $\ell = m$ then Stopp mit Ausgabe "$\mathcal{A} \models \chi$"
9) else if $\ell = 0$ then Stopp mit Ausgabe "$\mathcal{A} \not\models \chi$"
10) $\quad$ else
11) $\quad\quad$ Sei $B := N^{\mathcal{A}}_{2\hat{r}}(\{a_1, \dots, a_\ell\})$ und
$\quad\quad$ sei $\mathcal{B}$ die Expansion von $\mathcal{A}[B]$ mit einer zusätzlichen 1-stelligen Relation $Y^{\mathcal{B}} := P$
12) $\quad\quad$ Entscheide, ob gilt:
$$\mathcal{B} \models \underbrace{\exists x_1 \dots \exists x_m \bigwedge_{1 \le i < j \le m} \mathrm{dist}(x_i, x_j) > 2r \wedge \bigwedge_{i=1}^{m} Y(x_i)}_{=: \psi}$$
13) $\quad\quad$ Falls "ja", so Stopp mit Ausgabe "$\mathcal{A} \models \chi$"
14) $\quad\quad$ sonst stopp mit Ausgabe "$\mathcal{A} \not\models \chi$".
15) $\quad$ endelse

Korrektheit des Algorithmus:

Man sieht leicht, dass Folgendes gilt:

- Wenn der Algorithmus bei Zeile 7) angekommen ist, dann ist $\ell \in \{0,\dots,m\}$ und $a_1,\dots,a_\ell$ sind Elemente vom paarweisen Abstand $> 2r = \hat{r}$ und es gilt: $\mathcal{A} \models \lambda[a_i]$ f.a. $i \in [\ell]$.

- Daraus folgt: Wenn $\ell = m$ ist, so gilt $\mathcal{A} \models \chi$. D.h: wenn der Algo. in Zeile 8) mit Ausgabe "$\mathcal{A} \models \chi$" anhält, dann gibt er die korrekte Ausgabe.

- Andererseits gilt: $\ell = 0 \iff P = \emptyset$; und wenn $P = \emptyset$ ist, so gilt: $\mathcal{A} \not\models \chi$. D.h: Wenn der Algo in Zeile 9) mit Ausgabe "$\mathcal{A} \not\models \chi$" anhält, dann gibt er die korrekte Ausgabe

- Falls der Algo. in Zeile 10) ankommt, dann ist $1 \le \ell < m$, und gemäß Zeilen 3)-7) gilt: $P \subseteq N^{\mathcal{A}}_{\hat{r}}(\{a_1,\dots,a_\ell\})$ (Warum? ↗ Übung!)

  Also gilt: $N^{\mathcal{A}}_{\hat{r}}(P) \subseteq N^{\mathcal{A}}_{2\hat{r}}(\{a_1,\dots,a_\ell\}) \overset{\text{Zeile 11)}}{=} B$.

  somit gilt:

$$B \models \exists x_1 \dots \exists x_m \bigwedge_{1 \le i < j \le m} \underbrace{\text{dist}(x_i,x_j) > 2r}_{=\hat{r}} \wedge \bigwedge_{i=1}^{m} \underbrace{Y(x_i)}_{\text{"}x_i \in P\text{"}} \iff$$

  es gibt m Elemente in P mit paarweisem Abstand (in $\mathcal{A}$) $> 2r$
  $\iff \mathcal{A} \models \chi$ (Warum? ↗ Übung!)

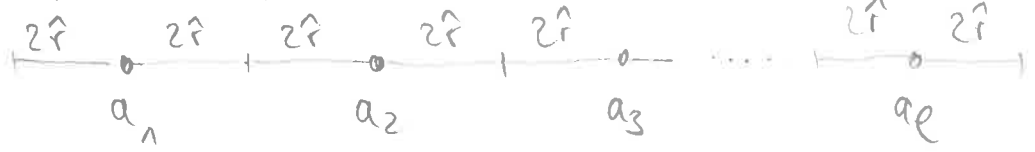
  Daher gibt der Algorithmus die korrekte Ausgabe, wenn er in Zeile 13) oder Zeile 14) anhält.

Laufzeitanalyse:

- Man sieht leicht, dass die Zeilen 1)–11) und 13)–15) in Zeit $O(|A|^{1+\frac{1}{k}})$ durchgeführt werden

- Zeile 12) wird wie folgt ausgeführt:

12.1) Zerlege die Struktur $\mathcal{B}$ in ihre Zusammenhangskomponenten. Wegen $B = N^{\mathcal{A}}_{2\hat{r}}(\{a_1, \dots, a_\ell\})$ gibt es $1 \le \hat{\ell} \le \ell$ Zusammenhangskomponenten, und für jedes $j \in \{1, \dots, \hat{\ell}\}$ gibt es ein $i_j \in \{1, \dots, \ell\}$ so dass die j-te Zusammenhangskomponente von $\mathcal{B}$ in $N^{\mathcal{A}}_{4\hat{r}\ell}(a_{i_j})$ enthalten ist (warum? → Übung! Skizze für den Spezialfall, dass es nur 1 Zusammenhangskomponente gibt:



Also hat jede Zusammenhangskomponente von $\mathcal{B}$ die Baumweite $\le f(4\hat{r}\ell)$, und insgesamt hat $\mathcal{B}$ Baumweite $\le f(4\hat{r}\ell) =: w'$.

12.2) Der FO-Satz φ ist auch ein CMSO-Satz. Wir nutzen Theorem 3.13 (Satz von Courcelle), um eine Zahl $c(\varphi, w') \in \mathbb{N}$ zu erhalten, so dass in Zeit $O(c(\varphi, w') \cdot \|\mathcal{B}\|)$ entschieden werden kann, ob $\mathcal{B} \models \varphi$ gilt. Da $\mathcal{B}$ aus $\mathcal{A}$ entsteht, sieht man leicht, dass $\|\mathcal{B}\| = O(\|\mathcal{A}\|)$ und $O(c(\varphi, w') \cdot \|\mathcal{B}\|) = O(|A|^{1+\frac{1}{k}})$ ist

□ Theorem 5.3