Logische Methoden in der Informatik

Stephan Kreutzer

Technische Universität Berlin

Sommersemester 2024

Inhalt des Moduls

Worum geht es?

Reguläre Sprachen und die monadische Logik zweiter Stufe.

1. Formale Sprachen und Automaten
2. Die Monadische Logik zweiter Stufe
3. Der Satz von Büchi und Elgot
4. Baumautomaten

Model-Checking und Auswerten von Formeln.

1. Komplexität von Auswertungsproblemen
2. Baumweite und der Satz von Courcelle
3. Lokalität der Prädikatenlogik

Logik und Komplexität.

1. Existentielle Logik zweiter Stufe und NP
2. Der Satz von Fagin

Wiederholung regulärer Sprachen

Formale Sprachen

Notation.
Σ: Alphabet.
ϵ: leeres Wort.

Formale Sprachen. Sei Σ ein Alphabet. Ein *Wort* ist eine endliche Folge $w = a_1 \dots a_n$ von Buchstaben $a_i \in \Sigma$.
Wir schreiben ϵ für das leere Wort.

Grammatiken. Sprachen im Kontext von Programmier- oder Skriptsprachen werden oft über Grammatiken definiert.

Beispiel. Sei $\sigma := \{E\}$. E: 2-stelliges Relationssymbol

Folgende Grammatik generiert $\mathrm{FO}[\sigma]$.

$$F \to A \mid \neg F \mid (F \vee F) \mid (F \wedge F) \mid (F \to F) \mid \exists V\ F \mid \forall V\ F$$
$$A \to V = V \mid \mathrm{E}(V, V)$$
$$V \to \mathrm{x}D$$
$$D \to 0 \dots 9 \mid 0\ D \mid 1\ D \mid \dots \mid 9\ D$$

F: Formeln
A: Atomare Formeln
V: Variablen
D: Zahlen

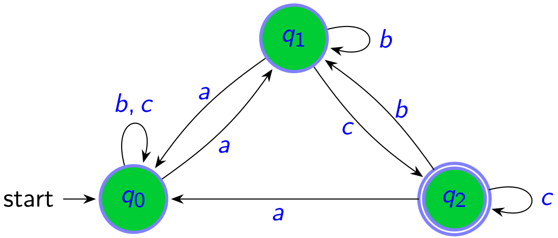
Reguläre Wortsprachen

Notation.
Σ: Alphabet.

Reguläre Sprachen. Eine Sprache ist *regulär*, wenn sie durch eine Typ-3 Grammatik generiert oder durch einen endlichen Automaten erkannt wird.

Beispiel. Betrachten wir folgenden Automaten $\mathcal{A}$.

$\mathcal{L}(\mathcal{A}) := \{w : w$ endet auf c und hat ungerade viele $a\}$.



	q_0	q_1	q_1	q_0	q_1	q_1	q_2	q_0	q_1	q_2
$w :=$	a	b	a	a	b	c	a	a	c	

Definition.
Ein *endlicher nicht-deterministischer Automat* ist ein Tupel $\mathcal{A} := (Q, \Sigma, q_0, \Delta, F)$, wobei

- Q eine endliche Menge von Zuständen,
- Σ ein endliches Alphabet,
- q_0 der *Anfangszustand*,
- $F \subseteq Q$ die Menge der *Endzustände* und
- $\Delta \subseteq Q \times \Sigma \times Q$ die Übergangsrelation ist.

Abschlusseigenschaften regulärer Sprachen

Abschlusseigenschaften

Die Klasse der regulären Sprachen ist unter vielen wichtigen Operationen abgeschlossen, unter anderem unter Schnitt, Komplementbildung und Vereinigung.

Lemma. Sei Σ ein endliches Alphabet.

1. Sind $\mathcal{L}_1, \mathcal{L}_2 \subseteq \Sigma^*$ reguläre Sprachen, so sind auch $\mathcal{L}_1 \cup \mathcal{L}_2$ und $\mathcal{L}_1 \cap \mathcal{L}_2$ regulär.
2. Ist $\mathcal{L} \subseteq \Sigma^*$ regulär, so ist auch $\Sigma^* \setminus \mathcal{L}$ regulär.

Abschluss unter Projektion

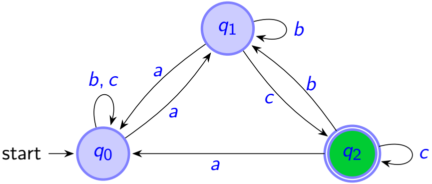
Projektion. Eine weitere wichtige Operation ist die *Projektion*.

Reguläre Wortsprachen

Reguläre Sprachen. Eine Sprache ist *regulär*, wenn sie durch eine Typ-3 Grammatik generiert oder durch einen endlichen Automaten erkannt wird.

Beispiel. Betrachten wir folgenden Automaten $\mathcal{A}$.

$\mathcal{L}(\mathcal{A}) := \{w : w$ endet auf c und hat ungerade viele $a\}$.

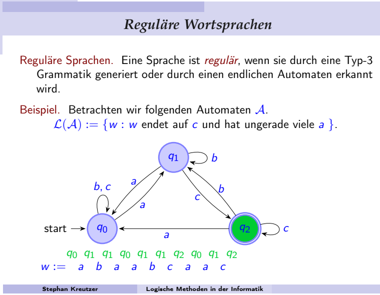


$$\begin{array}{cccccccccc} q_0 & q_1 & q_1 & q_0 & q_1 & q_1 & q_2 & q_0 & q_1 & q_2 \\ w := & a & b & a & a & b & c & a & a & c \end{array}$$

Notation.
Σ: Alphabet.

Definition.
Ein *endlicher nicht-deterministischer Automat* ist ein Tupel $\mathcal{A} := (Q, \Sigma, q_0, \Delta, F)$, wobei
- Q eine endliche Menge von Zuständen,
- Σ ein endliches Alphabet,
- q_0 der *Anfangszustand*,
- $F \subseteq Q$ die Menge der *Endzustände* und
- $\Delta \subseteq Q \times \Sigma \times Q$ die Übergangsrelation ist.

Abschluss unter Projektion



Projektion. Eine weitere wichtige Operation ist die *Projektion*.

Beispiel. $\Sigma := \{a, b, c\}$.

$L := \{w : w$ endet auf c und enthält ungerade viele $a\}$.

$w = abbabcabc \in L$.

Akzeptierender Lauf $\rho = q_0 q_1 q_1 q_1 q_0 q_1 q_2 q_0 q_1 q_2$.

Erweitertes Alphabet. Betrachten wir nun $\Gamma := \Sigma \times \{q_0, q_1, q_2\}$.

Wir können nun w zusammen mit dem Lauf ρ als ein Wort

$(a, q_1)(b, q_1)(b, q_1)(a, q_0)(b, q_1)(c, q_2)(a, q_0)(b, q_1)(c, q_2) \in \Gamma^*$

auffassen.

Sei $L' := \{w' = (a_1, s_1) \dots (a_n, s_n) \in \Gamma^* :$
$q_0 s_1 \dots s_n$ ist akzeptierender Lauf von $\mathcal{A}$ auf $w\}$.

L ist Projektion von L' auf Σ.

Abschluss unter Projektion

Definition. Seien Σ, Σ' endliche Alphabete und $\Gamma := \Sigma \times \Sigma'$.

Sei $\mathcal{L} \subseteq \Gamma^*$ eine Sprache.

Die *Projektion* von $\mathcal{L}$ auf Σ ist definiert als die Sprache $\mathcal{L}_\pi \subseteq \Sigma^*$ mit

$$\mathcal{L}_\pi := \{a_1 \dots a_n \in \Sigma^* : \quad \text{es ex. Folge } a'_1 \dots a'_n \in \Sigma'^* \text{ mit } (a_1, a'_1) \dots (a_n, a'_n) \in \mathcal{L}\}.$$

Beispiel 1.

$L := \{ \, w : w \text{ endet auf } c \text{ und hat ungerade viele } a \, \}.$

$$L' := \{w' = (a_1, s_1) \dots (a_n, s_n) \in \Gamma^* : q_0 s_1 \dots s_n \text{ ist akzeptierender Lauf von } \mathcal{A} \text{ auf } w \, \}.$$

Hier ist $\Sigma := \{a, b, c\}$ und $\Sigma' := \{q_0, q_1, q_2\}$ und $L = L'_\pi$.

Abschluss unter Projektion

Lemma. Seien Σ, Σ' endliche Alphabete und sei $\Gamma := \Sigma \times \Sigma'$. Sei $\mathcal{L} \subseteq \Gamma^*$ eine Sprache. Wenn $\mathcal{L}$ regulär ist, dann ist auch die Projektion $\mathcal{L}_\pi$ von $\mathcal{L}$ auf Σ regulär.

Beweis. Sei $\mathcal{A} := (Q, \Gamma, q_0, \Delta, F)$ ein NFA mit $\mathcal{L}(\mathcal{A}) = \mathcal{L}$.
Wir definieren einen NFA $\mathcal{B} := (Q, \Sigma, q_0, \Delta', F)$ mit den gleichen Zuständen wie $\mathcal{A}$ und mit der Übergangsrelation

$$\Delta' := \{(q, a, q') : \text{es existiert } a' \in \Sigma' \text{ mit } (q, (a, a'), q') \in \Delta\}.$$

Sei nun $w = a_1 \cdots a_n \in \Sigma^*$. Der Automat $\mathcal{B}$ akzeptiert das Wort w genau dann, wenn es eine Folge $a'_1, \dots, a'_n$ mit $a'_j \in \Sigma'$ gibt, so dass $\mathcal{A}$ das Wort $w' := (a_1, a'_1) \cdots (a_n, a'_n)$ akzeptiert. Also ist $w \in \mathcal{L}_\pi$ genau dann, wenn $w' \in \mathcal{L}$.

Definierbare Wortsprachen

Definierbare Wortsprachen

Definierbare Sprachen. Wir wollen Sprachen durch logische Formeln definieren.

Sei Σ ein Alphabet, sagen wir $\Sigma = \{a, b, c\}$.

Beispiel. $w = abcccbcababcbabacb$

Wir wollen Eigenschaften von Wörtern formalisieren, wie z.B.:

- Nach jedem a folgt sofort ein b
- Wenn das Wort mit a beginnt, endet es auch auf a
- Das Wort hat gerade Länge.
- Nach jedem a folgt irgendwann ein b.

Problem. Die Logik spricht über *Strukturen*, nicht Wörter.

Wir müssen also zunächst Wörter durch Strukturen kodieren.

Definierbare Wortsprachen

Wortstrukturen. Sei Σ ein Alphabet.

Definiere Signatur $\sigma = \sigma(\Sigma) := \{\leq, s, (P_a)_{a\in\Sigma}\}$, wobei $\leq, s$ zweistellige und die P_a jeweils einstellige Relationssymbole sind.

Zu $w = a_1 \dots a_n \in \Sigma^*$ definieren wir die *Wortstruktur*

$$\mathcal{W} = \mathcal{W}_w = (W, \leq^{\mathcal{W}}, s^{\mathcal{W}}, (P_a^{\mathcal{W}})_{a\in\Sigma}),$$

wobei:

- $W = \{1, \dots, n\}$ ist die Menge der Positionen in w.
- $\leq^{\mathcal{W}}$ ist die natürliche Ordnung auf $\{1, \dots, n\}$.
- $s^{\mathcal{W}} := \{(i, i+1) : 1 \leq i < n\}$ ist die Nachfolgerrelation.
- Für $a \in \Sigma$ gilt $P_a^{\mathcal{W}} := \{i : a_i = a\}$.

Intuition.
- *Universum* entspr. Positionen der Buchstaben.
- $\leq^{\mathcal{W}}, s^{\mathcal{W}}$ bestimmen Reihenfolge der Buchstaben
- P_a bestimmt die Positionen, an denen der Buchstabe a steht.

Definierbare Wortsprachen

Definition. Sei Σ ein Alphabet.

Signatur $\sigma := \{\leq, s, (P_a)_{a\in\Sigma}\}$

Zu $w = a_1 \dots a_n \in \Sigma^*$ ist

$$\mathcal{W} = (W, \leq^{\mathcal{W}}, s^{\mathcal{W}}, (P_a^{\mathcal{W}})_{a\in\Sigma}),$$

mit

- $W = \{1, \dots, n\}$ Positionen in w.
- $\leq^{\mathcal{W}}$ Ordnung auf $\{1, \dots, n\}$.
- $s^{\mathcal{W}} := \{(i, i+1) : 1 \leq i < n\}$.
- Für $a \in \sigma$ gilt $P_a^{\mathcal{W}} := \{i : a_i = a\}$.

Beispiel. Das Wort $w = ababc$ über $\Sigma := \{a, b, c\}$ entspricht der Struktur

$$\mathcal{W}_w := (\{1,2,3,4,5\}, \leq^{\mathcal{W}_w}, s^{\mathcal{W}_w}, P_a^{\mathcal{W}_w}, P_b^{\mathcal{W}_w}, P_c^{\mathcal{W}_w}),$$

wobei

- $P_a^{\mathcal{W}_w} := \{1,3\}$
- $P_b^{\mathcal{W}_w} := \{2,4\}$
- $P_c^{\mathcal{W}_w} := \{5\}$

$$\begin{array}{cccccc} & P_a & P_b & P_a & P_b & P_c \\ w := & \mathrm{a} & \mathrm{b} & \mathrm{a} & \mathrm{b} & \mathrm{c} \end{array}$$

Formel $\varphi := \forall x\big(P_a(x) \rightarrow \exists y(P_b(y) \wedge s(x,y))\big)$.

Definierbare Wortsprachen

Definition. Sei Σ ein Alphabet.

Eine Sprache $\mathcal{L} \subseteq \Sigma^*$ ist FO-*definierbar*, wenn es einen Satz $\varphi \in \text{FO}[\sigma(\Sigma)]$ gibt, so dass für alle $w \in \Sigma^+$ gilt:

$$w \in \mathcal{L} \quad \Longleftrightarrow \quad \mathcal{W}_w \models \varphi.$$

Beispiel. Sei $\Sigma = \{a, b, c\}$.

Die Sprache

$$\mathcal{L} = \{w \in \Sigma^* : \text{auf jedes } a \text{ folgt irgendwann ein } c\ \}$$

ist FO-definierbar, z.B. durch

$$\varphi := \forall x \left(P_a(x) \rightarrow \exists y(P_c(y) \land x \leq y)\right).$$

Charakterisierungen formaler Sprachen

Arten, formale Sprachen zu definieren.

- *Grammatiken*. Eignen sich z.B. zur Spezifikation von Programmiersprachen.
- *Automaten*. Liefern einfache und effiziente Algorithmen um das Wortproblem oder Leerheit einer Sprache zu entscheiden.
- *Logik*. Eine deklarative Art, Sprachen mit bestimmten Eigenschaften zu definieren.

Frage. Welche Art von Sprachen lassen sich in FO definieren?

Vereinfachung der Notation

Vereinfachung der Notation: $<, >, \leq, \geq$

Sei $\mathcal{W} = \mathcal{W}_w = (W, \leq^{\mathcal{W}}, s^{\mathcal{W}}, (P_a^{\mathcal{W}})_{a\in\Sigma})$ eine Wortstruktur.

Ordnungssymbole.

Die Formel $\varphi_<(x,y) := x \leq y \wedge \neg x = y$ definiert die natürliche strikte Ordnung $<^{\mathcal{W}}$ auf W:

Für $i, j \in W$ gilt $\mathcal{W} \models \varphi_<[i,j]$ gdw. $i < j$.

Erinnerung. $\mathcal{W} \models \varphi_<[i,j]$ bedeutet $(\mathcal{W}, \beta) \models \varphi_<$ für die Belegung $\beta = [x/i, y/j]$

Wir erlauben daher ab jetzt auch die Verwendung von Atomen wie $x < y$ in Formeln über Wortstrukturen.

Ebenso sind $\geq$ und $>$ definierbar und können in Formeln verwendet werden.

Vereinfachung der Notation: $<, >, \leq, \geq$

Erinnerung. Für $\varphi(x_1, \dots, x_k)$ ist
$$\varphi(\mathcal{A}) := \{(a_1, \dots, a_k) \in A^k : (\mathcal{A}, [a_1, \dots, a_k]) \models \varphi\}$$

Sei $\mathcal{W} = \mathcal{W}_w = (W, \leq^{\mathcal{W}}, s^{\mathcal{W}}, (P_a^{\mathcal{W}})_{a \in \Sigma})$ eine Wortstruktur.

Ordnungssymbole.

Die Formel $\varphi_<(x, y) := x \leq y \wedge \neg x = y$ definiert die natürliche strikte Ordnung $<^{\mathcal{W}}$ auf W:

$$\text{Für } i, j \in W \text{ gilt } \mathcal{W} \models \varphi_<[i, j] \text{ gdw. } i < j.$$

Erinnerung. $\mathcal{W} \models \varphi_<[i, j]$ bedeutet $(\mathcal{W}, \beta) \models \varphi_<$ für die Belegung $\beta = [x/i, y/j]$

Wir erl

$x < y$

Ebenso

werden

Was bedeutet eigentlich

„$\varphi_<(x, y) := x \leq y \wedge \neg x = y$ *definiert* $<^{\mathcal{W}}$ *auf* $\mathcal{W}$“?

Definition. Sei $\varphi(x, y) \in \text{FO}[\sigma]$ eine Formel. Sei $\mathcal{A} = (A, \sigma^{\mathcal{A}})$ eine σ-Struktur und $R \subseteq A \times A$ eine zweistellige Relation über A.

φ *definiert* R *in* $\mathcal{A}$ genau dann, wenn $R = \varphi(\mathcal{A})$.

Notation: Die Nachfolgerfunktion

Sei $\mathcal{W} = \mathcal{W}_w = (W, \leq^{\mathcal{W}}, s^{\mathcal{W}}, (P_a^{\mathcal{W}})_{a \in \Sigma})$ eine Wortstruktur.

Notation. Wir schreiben die Nachfolgerrelation *funktional* als $+1$. Wir schreiben zum Beispiel

$y = x + 1$ als Abkürzung für $s(x, y)$
$P_b(x + 1)$ als Abkürzung für $\exists y \big(s(x, y) \wedge P_b(y) \big)$

Beispiel. $\forall x (P_a(x) \rightarrow P_b(x + 1))$ ist eine Abkürzung für $\forall x (P_a(x) \rightarrow \exists y (s(x, y) \wedge P_b(y)))$.

Achtung. Wenn x das maximale Element in W ist, ist $x + 1$ nicht definiert und die Unterformel $P_b(x + 1)$ falsch.

Erster und letzter Buchstabe

Sei $\mathcal{W} = \mathcal{W}_w = (W, \leq^{\mathcal{W}}, s^{\mathcal{W}}, (P_a^{\mathcal{W}})_{a\in\Sigma})$ eine Wortstruktur.

Notation. Abschließend führen wir noch Notation für den ersten und den letzten Buchstaben eines Wortes ein.

- Die Formel $max(x) := \neg\exists y x < y$ gilt für eine Belegung $x \mapsto i$ mit $i \in W$ gdw. i die größte Zahl in W ist.
- Die Formel $min(x) := \neg\exists y y < x$ gilt für eine Belegung $x \mapsto i$ gdw. i die kleinste Zahl in W ist.

Wir werden daher ab jetzt auch „Konstantensymbole“ min und max für die erste und die letzte Position des Worts verwenden.

Beispiele definierbarer Wortsprachen

FO-definierbare Sprachen

Definition. Σ: Alphabet.
Eine Sprache $\mathcal{L} \subseteq \Sigma^*$ ist *FO-definierbar*, wenn es ein $\varphi \in \text{FO}[\sigma(\Sigma)]$ gibt, so dass für alle $w \in \Sigma^+$:
$w \in \mathcal{L} \iff \mathcal{W}_w \models \varphi$.

Beispiel. Sei $\Sigma := \{a, b, c\}$.
Sei $\mathcal{L} := \{w \in \Sigma^+ : w$ beginnt und endet mit c und dazwischen wechseln sich a und b immer ab, beginnend mit $a\}$.

Behauptung. Die Sprache $\mathcal{L}$ ist in FO definierbar.

- „w beginnt und endet mit c"

  $\varphi_1 := P_c(min) \wedge P_c(max)$

- „dazwischen wechseln sich a und b immer ab"

  $$\begin{aligned} \varphi_2 := &\forall x(min < x \wedge x < max \to (P_a(x) \vee P_b(x)))\wedge && \text{„zwischen min und max nur } a \text{ oder } b\text{"} \\ &\forall x(P_a(x) \to (P_b(x+1) \vee x+1 = max))\wedge && \text{„nach jedem } a \text{ ein } b \text{ oder letzter Buchst."} \\ &\forall x(P_b(x) \to (P_a(x+1) \vee x+1 = max)) && \text{„nach jedem } b \text{ ein } a \text{ oder letzter Buchst."} \end{aligned}$$

Also definiert $\varphi_1 \wedge \varphi_2 \wedge P_a(min+1)$ die Sprache $\mathcal{L}$.

FO-*definierbare Sprachen*

Beispiel. Sei $\Sigma := \{a, b, c\}$.

Sei $\mathcal{L} := \{w \in \Sigma^+ : w \text{ enthält eine gerade Anzahl von } a\}$.

Behauptung. Die Sprache $\mathcal{L}$ ist nicht in FO definierbar.

Beweisidee. Im Modul *Logik* haben wir bewiesen, dass ein FO-Satz φ mit Quantorenrang $\leq m$ zwei endliche lineare Ordnungen der Länge $> 2^m$ nicht unterscheiden kann.

Angenommen, $\mathcal{L}$ wäre definierbar durch $\varphi \in \text{FO}[\sigma(\Sigma)]$.

Wir wollen zeigen, dass dann auch Ordnungen gerader Länge definierbar sind.

Definition. Σ: Alphabet.
Eine Sprache $\mathcal{L} \subseteq \Sigma^*$ ist FO-*definierbar*, wenn es ein $\varphi \in \text{FO}[\sigma(\Sigma)]$ gibt, so dass für alle $w \in \Sigma^+$:
$w \in \mathcal{L} \iff \mathcal{W}_w \models \varphi$.

FO-*definierbare Sprachen*

Beispiel. S

Sei $\mathcal{L}$:

Behauptur

Beweisidee

mit Qu

$> 2^m$ r

Angen

Wir wo

definier

EF-Spiele auf linearen Ordnungen

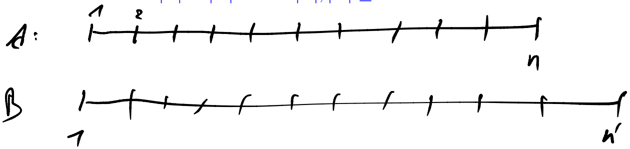
Theorem. Seien $\mathcal{A}$ und $\mathcal{B}$ endliche $\{\leq\}$-Strukturen mit Universum A bzw. B, wobei $\leq^{\mathcal{A}}, \leq^{\mathcal{B}}$ lineare Ordnungen auf dem jeweiligen Universum sind.

Dann gilt für alle $m \in \mathbb{N}$:

$$\text{Die Duplikatorin gewinnt das Spiel } \mathfrak{G}_m(\mathcal{A}, \mathcal{B})$$

$$\iff$$

$$|A| = |B| \quad \text{oder} \quad |A|, |B| \geq 2^m.$$



FO-definierbare Sprachen

$\mathcal{L} := \{w \in \Sigma^+ : w$ enthält eine gerade Anzahl von $a\}$

Behauptung 1. Wenn $\mathcal{L}$ FO-definierbar ist, dann ist auch $\mathcal{L}' := \mathcal{L} \cap \{a\}^*$ FO-definierbar.

Beweis. Sei $\varphi_{\mathcal{L}} \in$ FO ein Satz, der $\mathcal{L}$ definiert.

Dann definiert $\mathcal{L}_a := \varphi_{\mathcal{L}} \wedge \forall x\, P_a(x)$ die Sprache $\mathcal{L}'$. $\square$

Folgerung. Wenn wir also zeigen, dass $\mathcal{L}'$ nicht definierbar ist, dann ist auch $\mathcal{L}$ nicht definierbar.

FO-*definierbare Sprachen*

$\mathcal{L}' := \{w \in \{a\}^+ : w$ hat gerade Länge $\}$

Behauptung 2. $\mathcal{L}'$ ist nicht FO-definierbar.

Beweis. Ang. $\mathcal{L}'$ wäre durch $\varphi' \in$ FO definierbar.

Sei φ_{even} der Satz, der aus φ' entsteht, indem

- alle Atome der Form $P_b(x)$ oder $P_c(x)$ durch *false* ($\exists x \neg x = x$),
- alle Atome der Form $P_a(x)$ durch *true* (z.B. $\exists x\, x = x$) und
- alle Atome der Form $s(x, y)$ durch $\varphi_s(x, y)$ ersetzt werden.

$\varphi_s(x, y)$: Formel, die $s(x, y)$ aus $\leq$ definiert.

Dann gilt für alle $w \in \{a\}^+$: $\quad \mathcal{W}_w \models \varphi'$ gdw. $\mathcal{W}_w \models \varphi_{even}$.

Nach dem Koinzidenzlemma gilt nun für alle

$$\mathcal{W}_w := \left(W, \leq^{\mathcal{W}_w}, s^{\mathcal{W}_w}, P_a^{\mathcal{W}_w} = W, P_b^{\mathcal{W}_w} = \emptyset, P_c^{\mathcal{W}_w} = \emptyset\right):$$

$$\mathcal{W}_w \models \varphi_{even} \text{ gdw. } (W, \leq^{\mathcal{W}_w}) \models \varphi_{even}.$$

Also definiert φ_{even} die Klasse der linearen Ordnungen gerader Länge innerhalb der Klasse der endlichen linearen Ordnungen. Widerspruch! □

FO-definierbare Sprachen

Folgerung. Aus dem vorherigen Ergebnis folgt, dass

$$\mathcal{L}' := \{w \in \{a\}^* : w \text{ hat gerade Länge}\}$$

nicht FO-definierbar ist.

Beispiel. Sei nun $\Sigma = \{(a,0),(a,1)\} = \{a\} \times \{0,1\}$ und

$$\mathcal{L}_2 := \{w \in \Sigma^+ : \begin{array}{l} w \text{ hat gerade Länge, beginnt mit } (a,1) \text{ und} \\ (a,0) \text{ und } (a,1) \text{ wechseln sich ab } \end{array}\}.$$

Behauptung. $\mathcal{L}_2$ ist FO-definierbar durch

$$\varphi_2 := \quad P_{(a,1)}(min) \wedge P_{(a,0)}(max) \wedge \forall x \big(x < max \rightarrow (P_{(a,0)}(x) \leftrightarrow P_{(a,1)}(x+1))\big)$$

In $w \in \mathcal{L}_2$ steht an allen geraden Positionen der Buchstabe $(a,0)$ und an allen ungeraden $(a,1)$.

Definierbarkeit in FO

Satz. Es gibt reguläre Sprachen, die nicht in FO definierbar sind.

Definierbarkeit in FO. Die Beispiele zeigen:

- Die Wörter gerader Länge sind nicht FO-definierbar.
- Aber wenn wir die Buchstaben wie im zweiten Beispiel „annotieren“, dann können wir mit Hilfe der Annotationen Wörter gerader Länge definieren.

Idee. Wir erweitern die Prädikatenlogik um Quantoren, die uns erlauben, diese „Annotationen“ zu „raten“.

Etwas genauer: Wir erweitern FO um Quantoren über Mengen von Positionen in Wörtern.

Damit können wir dann auch Sprachen wie in den Beispielen vorhin definieren.

Die Monadische Logik zweiter Stufe

Monadische Logik zweiter Stufe

Definition. Wir fixieren eine abzählbar unendliche Mengen

- Var_1 von *Variablen erster Stufe*, sowie
- mVar_2 von *monadischen Variablen zweiter Stufe*.

Notation.
$x, y, \ldots$. für Var_1
$X, Y, \ldots$. für mVar_2

Die *Monadische Logik zweiter Stufe (MSO)* ist die Erweiterung der Prädikatenlogik um Quantoren

$\exists X \varphi$ und $\forall X \varphi$ wobei $X \in \mathrm{mVar}_2$ und
atomaren Formeln $X(x)$ für $X \in \mathrm{mVar}_2, x \in \mathrm{Var}_1$.

Notation. Wir schreiben oft $x \in X$ für $X(x)$.

Semantik. Sei σ eine Signature und $\mathcal{A} = (A, \sigma)$ eine σ-Struktur.

$\mathcal{A} \models \exists X \varphi$, wenn es ein $U \subseteq A$ gibt mit $(\mathcal{A}, [X/U]) \models \varphi$.

$\mathcal{A} \models \forall X \varphi$, wenn für jede Menge $U \subseteq A$ gilt: $(\mathcal{A}, [X/U]) \models \varphi$.

Definition: Die Monadische Logik zweiter Stufe

Sei σ eine Signatur. $\text{MSO}[\sigma]$ ist induktiv definiert durch:

- $t = t', R(t_1, \dots, t_k) \in \text{MSO}[\sigma]$ für Terme $t, t', t_1, \dots, t_k$ und k-stellige Relationssymbole $R \in \sigma$.
- $X = Y \in \text{MSO}[\sigma]$ für alle $X, Y \in \text{mVar}_2$.
- $X(x) \in \text{MSO}[\sigma]$ für alle $x \in \text{Var}_1$ und $X \in \text{mVar}_2$.
- Wenn $\varphi \in \text{MSO}[\sigma]$, dann $\neg\varphi \in \text{MSO}[\sigma]$.
- Wenn $\varphi, \psi \in \text{MSO}[\sigma]$, dann $(\varphi \vee \psi), (\varphi \wedge \psi), (\varphi \rightarrow \psi), (\varphi \leftrightarrow \psi) \in \text{MSO}[\sigma]$.
- Wenn $\varphi \in \text{MSO}[\sigma]$ und $x \in \text{Var}_1$, dann $\exists x\varphi \in \text{MSO}[\sigma]$ und $\forall x\varphi \in \text{MSO}[\sigma]$.
- Wenn $\varphi \in \text{MSO}[\sigma]$ und $X \in \text{mVar}_2$, dann $\exists X\varphi \in \text{MSO}[\sigma]$ und $\forall X\varphi \in \text{MSO}[\sigma]$.

Wir definieren $\text{MSO} := \bigcup\{\text{MSO}[\sigma] : \sigma \text{ ist eine Signatur }\}$.

Semantik von MSO

MSO erweitert FO auf natürliche Weise um Mengenquantoren.

Semantik. Sei σ eine Signature und $\mathcal{A} = (A, \sigma)$ eine σ-Struktur.

$\mathcal{A} \models \exists X \varphi$, wenn es ein $U \subseteq A$ gibt mit $(\mathcal{A}, [X/U]) \models \varphi$.

$\mathcal{A} \models \forall X \varphi$, wenn für jede Menge $U \subseteq A$ gilt: $(\mathcal{A}, [X/U]) \models \varphi$.

Definition. (Belegungen und Interpretationen).
Sei σ eine Signatur und sei $\mathcal{A}$ eine σ-Struktur.

1. Eine *Belegung* in $\mathcal{A}$ ist eine Funktion $\beta : \mathrm{def}(\beta) \to A \cup \mathcal{P}(A)$ mit $\mathrm{def}(\beta) \subseteq \mathrm{Var}_1 \cup \mathrm{mVar}_2$ und
   - $\beta(x) \in A$ wenn $x \in \mathrm{Var}_1$ und
   - $\beta(X) \in \mathcal{P}(A)$ wenn $X \in \mathrm{mVar}_2$.
2. Eine *σ-Interpretation* ist ein Paar $(\mathcal{A}, \beta)$, bestehend aus einer σ-Struktur $\mathcal{A}$ und einer Belegung β in $\mathcal{A}$.

Achtung.
Anders als bei FO kommen hier auch Mengenvariablen als freie Variablen und somit in Belegungen vor.

Semantik von MSO

MSO erweitert FO auf natürliche Weise um Mengenquantoren.

Semantik. Sei σ eine Signature und $\mathcal{A} = (A, \sigma)$ eine σ-Struktur.

$\mathcal{A} \models \exists X \varphi$, wenn es ein $U \subseteq A$ gibt mit $(\mathcal{A}, [X/U]) \models \varphi$.

$\mathcal{A} \models \forall X \varphi$, wenn für jede Menge $U \subseteq A$ gilt: $(\mathcal{A}, [X/U]) \models \varphi$.

Definition. (Belegungen und Interpretationen).

Sei σ eine Signatur und sei $\mathcal{A}$ eine σ-Struktur.

1. Eine *Bele*... $\mathcal{P}(A)$ mit def(...
   - $\beta(x) \in A$ wenn $x \in \text{Var}_1$ und
   - $\beta(X) \in \mathcal{P}(A)$ wenn $X \in \text{mVar}_2$.
2. Eine *σ-Interpretation* ist ein Paar $(\mathcal{A}, \beta)$, bestehend aus einer σ-Struktur $\mathcal{A}$ und einer Belegung β in $\mathcal{A}$.

Beispiel.

$\varphi := \exists X\ \exists Y\ \forall x\ (Z(x) \rightarrow (X(x) \vee Y(x)))\ldots$

Achtung.

Anders als bei FO kommen hier auch Mengenvariablen als freie Variablen und somit in Belegungen vor.

Beispiel

Beispiel. Betrachten wir wieder die Sprache $\mathcal{L} \subseteq \{a, b, c\}^+$ mit

$$\mathcal{L} := \{w : w \text{ hat gerade Länge }\}.$$

$\mathcal{L}$ ist MSO-definierbar durch die Formel

$$\begin{aligned}\varphi := \exists X_0 \exists X_1 \big(min \in X_1 \wedge max \in X_0 \big) \ \wedge \\ \forall x \neg \big(x \in X_0 \wedge x \in X_1 \big) \wedge \\ \forall x \big(x < max \rightarrow (x \in X_0 \leftrightarrow x + 1 \in X_1) \big)\end{aligned}$$

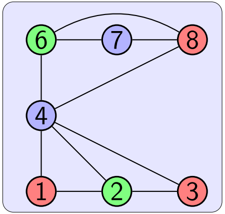
Vergleiche mit der FO-Formel aus dem vorherigen Abschnitt.

$$\begin{aligned}\varphi_2 := \quad & P_{(a,1)}(min) \wedge P_{(a,0)}(max) \wedge \\ & \forall x \big(x < max \rightarrow (P_{(a,0)}(x) \leftrightarrow P_{(a,1)}(x+1)) \big)\end{aligned}$$

Beispiele für MSO-Formeln

Beispiele für MSO-Formeln

Definition. Ein Graph G ist 3-*färbbar*, wenn die Knoten von G so mit drei Farben gefärbt werden können, dass die Endpunkte jeder Kante verschieden gefärbt sind.



$$\begin{aligned}\varphi_{3col} := \exists X_1 \exists X_2 \exists X_3 \quad & \forall x \quad \Big(\bigvee_{i=1}^{3} (X_i(x) \wedge \bigwedge_{j \neq i} \neg X_j(x) \Big) \wedge \\ & \forall x \forall y \ \Big(E(x,y) \rightarrow \bigwedge_{i=1}^{3} \neg (X_i(x) \wedge X_i(y)) \Big)\end{aligned}$$

Beispiel. Die Formel φ_{3col} gilt in G genau dann, wenn G 3-*färbbar* ist.

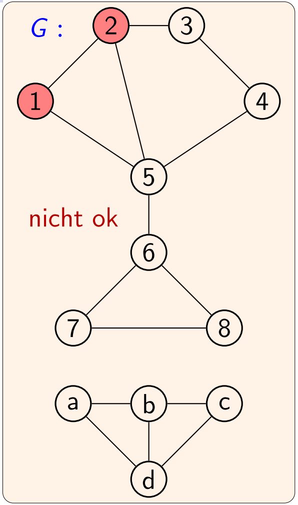
Beispiele für MSO-Formeln: Zusammenhang

Als nächstes wollen wir eine MSO-Formel konstruieren, die genau dann in einem Graph G gilt, wenn dieser zusammenhängend ist.

Zusammenhang in Graphen. Ein Graph G ist zusammenhängend, wenn es zwischen je zwei Knoten einen Pfad gibt.

Eine Menge $X \subseteq V(G)$ von Knoten ist *abgeschlossen unter* $E(G)$, wenn $X \cup N(X) = X$.

Ein Graph G ist zusammenhängend, wenn jede nicht-leere unter $E(G)$ abgeschlossene Menge $X \subseteq V(G)$ alle Knoten aus $V(G)$ enthält.



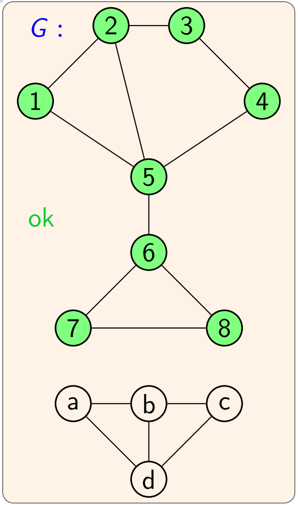
Beispiele für MSO-Formeln: Zusammenhang

Als nächstes wollen wir eine MSO-Formel konstruieren, die genau dann in einem Graph G gilt, wenn dieser zusammenhängend ist.

Zusammenhang in Graphen. Ein Graph G ist zusammenhängend, wenn es zwischen je zwei Knoten einen Pfad gibt.

Eine Menge $X \subseteq V(G)$ von Knoten ist *abgeschlossen unter* $E(G)$, wenn $X \cup N(X) = X$.

Ein Graph G ist zusammenhängend, wenn jede nicht-leere unter $E(G)$ abgeschlossene Menge $X \subseteq V(G)$ alle Knoten aus $V(G)$ enthält.



Beispiele für MSO-Formeln

Beispiel. Wir konstruieren eine Formel φ_{con} die genau dann in G gilt, wenn G *zusammenhängend* ist.

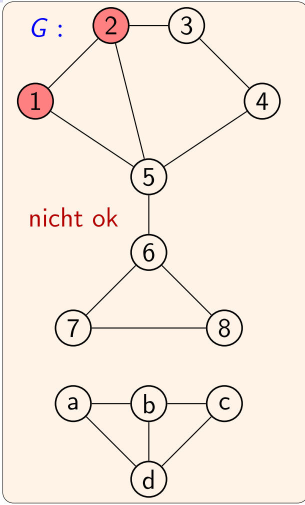
$$\varphi_{cl}(C) := \forall y \forall z \big(C(y) \land E(y,z) \rightarrow C(z)\big)$$

„C ist abgeschlossen unter E“

$C = \emptyset \quad C = \{1, \dots, 8\} \quad C = \{a, \dots, d\} \quad C = V(G)$

$$\varphi_{con}(u,v) := \forall X \big(X(u) \land \varphi_{cl}(X)\big) \rightarrow X(v)$$

„u und v liegen in der selben Komponente“



Beispiele für MSO-Formeln

Beispiel. Wir konstruieren eine Formel φ_{con} die genau dann in G gilt, wenn G *zusammenhängend* ist.

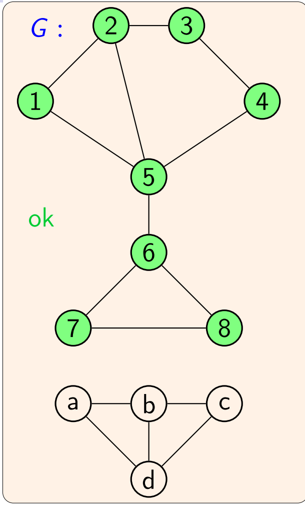
$$\varphi_{cl}(C) := \forall y \forall z \big(C(y) \land E(y,z) \rightarrow C(z)\big)$$

„C ist abgeschlossen unter E“

$C = \emptyset \quad C = \{1, \dots, 8\} \quad C = \{a, \dots, d\} \quad C = V(G)$

$$\varphi_{con}(u,v) := \forall X \big(X(u) \land \varphi_{cl}(X)\big) \rightarrow X(v)$$

„u und v liegen in der selben Komponente“



Beispiele für MSO-Formeln

Beispiel. Wir konstruieren eine Formel φ_{con} die genau dann in G gilt, wenn G *zusammenhängend* ist.

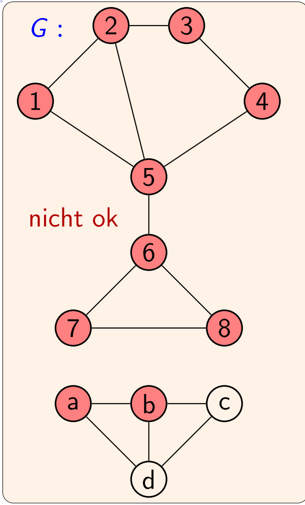
$$\varphi_{cl}(C) := \forall y \forall z \big(C(y) \wedge E(y,z) \rightarrow C(z)\big)$$

„C ist abgeschlossen unter E"

$C = \emptyset \quad C = \{1, \dots, 8\} \quad C = \{a, \dots, d\} \quad C = V(G)$

$$\varphi_{con}(u,v) := \forall X \big(X(u) \wedge \varphi_{cl}(X)\big) \rightarrow X(v)$$

„u und v liegen in der selben Komponente"



Beispiele für MSO-Formeln

Beispiel. Wir konstruieren eine Formel φ_{con} die genau dann in G gilt, wenn G *zusammenhängend* ist.

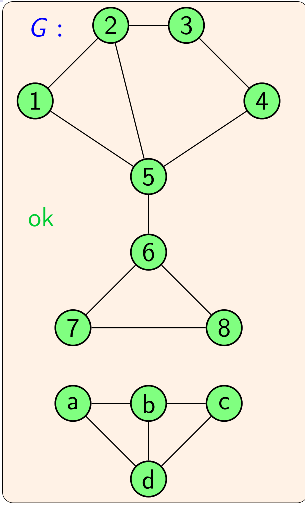
$$\varphi_{cl}(C) := \forall y \forall z \big(C(y) \wedge E(y,z) \rightarrow C(z)\big)$$

„C ist abgeschlossen unter E“

$C = \emptyset \quad C = \{1, \dots, 8\} \quad C = \{a, \dots, d\} \quad C = V(G)$

$$\varphi_{con}(u,v) := \forall X \big(X(u) \wedge \varphi_{cl}(X)\big) \rightarrow X(v)$$

„u und v liegen in der selben Komponente“



Beispiele für MSO-Formeln

Beispiel. Wir konstruieren eine Formel φ_{con} die genau dann in G gilt, wenn G *zusammenhängend* ist.

$$\varphi_{cl}(C) := \forall y \forall z \big(C(y) \land E(y,z) \rightarrow C(z)\big)$$

„C ist abgeschlossen unter E“

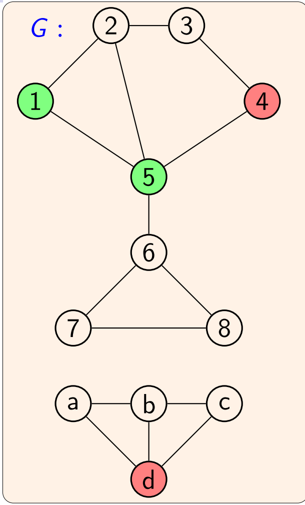
$C = \emptyset \quad C = \{1, \ldots, 8\} \quad C = \{a, \ldots, d\} \quad C = V(G)$

$$\varphi_{con}(u,v) := \forall X \big(X(u) \land \varphi_{cl}(X)\big) \rightarrow X(v)$$

„u und v liegen in der selben Komponente“

$$\varphi_{G-con} := \forall u \forall v \varphi_{con}(u,v)$$

„G ist zusammenhängend“



Syntaktischer Zucker

Teilmengenbeziehungen

Mengenbeziehungen. $\varphi_{\subseteq}(X, Y) := \forall x\ (X(x) \rightarrow Y(x))$

„X ist eine Teilmenge von Y"

Analoge Formeln für $X \subsetneq Y$, $X \supseteq Y$, $X \supsetneq Y$

Notation. Wir erlauben ab jetzt atomate Formeln $X \subseteq Y$ etc.

Mengenoperationen. $\varphi_{\cap}(X, Y, Z) := \forall x \Big(Z(x) \leftrightarrow (X(x) \wedge Y(x)) \Big)$

„$Z = X \cap Y$" Wir erlauben ab jetzt Atome $Z = X \cap Y, \ldots$

Partitionen. $\varphi_{part}(X_1, X_2) := \forall x (X_1(x) \vee X_2(x)) \wedge \neg(X_1(x) \wedge X_2(x))$

„(X_1, X_2) partitionieren das Universum"

$\varphi_{n-part}(X_1, \ldots, X_n) := \forall x \bigvee_{i=1}^{n} X_i(x) \wedge \bigwedge_{1 \leq i < j \leq n} \neg(X_i(x) \wedge X_j(x))$

Beispiel für die neue Notation

Beispiel. Wir konstruieren eine Formel φ_{con} die genau dann in G gilt, wenn G *zusammenhängend* ist.

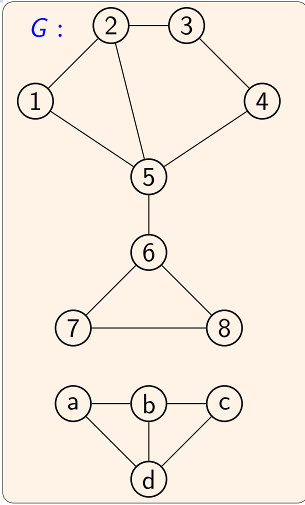
$$\varphi_{cl}(C) := \forall y \forall z \big(C(y) \wedge E(y,z) \rightarrow C(z) \big)$$

„C ist abgeschlossen unter E“

$C = \emptyset \quad C = \{1, \dots, 8\} \quad C = \{a, \dots, d\} \quad C = V(G)$

$$\varphi_{Komponente}(C) := \varphi_{cl}(C) \wedge C \neq \emptyset \wedge$$
$$\neg \exists X \big(\varphi_c l(X) \wedge X \neq \emptyset \wedge X \subsetneq C \big)$$

„C ist die Knotenmenge einer Komponente“



Der Satz von Büchi und Elgot

Der Satz von Büchi und Elgot

Theorem. (Satz von Büchi und Elgot, 1961)

Sei Σ ein endliches Alphabet.

Eine Sprache $\mathcal{L} \subseteq \Sigma^+$ ist genau dann regulär, wenn sie MSO-definierbar ist.

Der Satz von Büchi und Elgot

Theorem. (Büchi und Elgot, 1961) Sei Σ ein endliches Alphabet.

Eine Sprache $\mathcal{L} \subseteq \Sigma^+$ ist genau dann regulär, wenn sie MSO-definierbar ist.

$$\varphi \in \text{MSO} \rightsquigarrow \text{NFA } \mathcal{A}$$

$$\varphi \in \text{MSO} \leftsquigarrow \text{NFA } \mathcal{A}$$

Der Satz von Büchi und Elgot

Wir zeigen als erstes die Rückrichtung.

Lemma. Sei Σ ein endliches Alphabet. Jede reguläre Sprache $\mathcal{L} \subseteq \Sigma^+$ ist MSO-definierbar.

Beweis der Rückrichtung

Lemma. Sei Σ ein endliches Alphabet. Jede reguläre Sprache $\mathcal{L} \subseteq \Sigma^+$ ist MSO-definierbar.

Beweis.

Da $\mathcal{L}$ regulär, existiert ein NFA $\mathcal{A}$ mit $\mathcal{L}(\mathcal{A}) = \mathcal{L}$. Sei $Q := \{q_0, \dots, q_k\}$.

$w = a_1 \dots a_n \in \mathcal{L}$ gdw. es gibt akzeptierenden Lauf $\rho : \{1, \dots, n\} \to Q$ von $\mathcal{A}$ auf w.

$$\begin{array}{lcccccccccc} & q_0 & q_1 & q_1 & q_0 & q_1 & q_1 & q_2 & q_0 & q_1 & q_2 \\ w := & & a & b & a & a & b & c & a & a & c \end{array}$$

$\square$

Definition. *NFA*: Tupel $\mathcal{A} := (Q, \Sigma, q_0, \Delta, F)$, wobei
- Q endl Zustandsmenge,
- Σ ein endl Alphabet,
- q_0 der *Anfangszustand*,
- $F \subseteq Q$ *Endzustände*
- $\Delta \subseteq Q \times \Sigma \times Q$ Übergangsrelation.

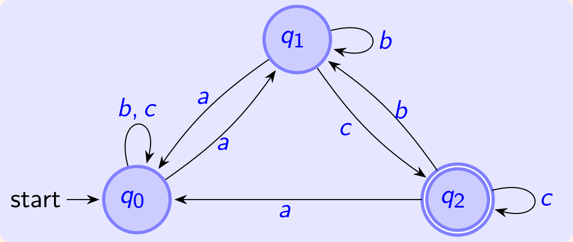
Beweis der Rückrichtung

Len

Bev

Beispiel. Betrachten wir folgenden Automaten $\mathcal{A}$.

$\mathcal{L}(\mathcal{A}) := \{ w : w$ endet auf c und hat ungerade viele a $\}$.



$$\begin{array}{lccccccccccc} & q_0 & q_1 & q_1 & q_0 & q_1 & q_1 & q_2 & q_0 & q_1 & q_2 \\ w := & & a & b & a & a & b & c & a & a & c \end{array}$$

$\square$

Definition. *NFA*: Tupel
$\mathcal{A} := (Q, \Sigma, q_0, \Delta, F)$, wobei

- Q endl Zustandsmenge,
- Σ ein endl Alphabet,
- q_0 der *Anfangszustand*,
- $F \subseteq Q$ *Endzustände*
- $\Delta \subseteq Q \times \Sigma \times Q$ Übergangsrelation.

Beweis der Rückrichtung

$w = a_1 \dots a_n$

Lauf ρ.

$$\begin{array}{lccccccccc} & q_0 & q_1 & q_1 & q_0 & q_1 & q_1 & q_2 & q_0 & q_1 & q_2 \\ w := & & a & b & a & a & b & c & a & a & c \end{array}$$

Idee. Beschreibe den Lauf ρ durch Mengen $Q_0, \dots, Q_k$ mit

$$Q_i := \{j \in \{1, \dots, n\} : \rho(j) = q_i\}$$

Position $j \in Q_i$ wenn $\mathcal{A}$ nach Lesen von a_j im Zustand q_i ist.

Eigenschaften der Mengen Q_i.

1. $Q_0, \dots, Q_k$ bilden eine Partition von $1, \dots, n$.
2. wenn $1 \in Q_i$ dann $(q_0, a_1, q_i) \in \Delta$
3. wenn $p \in Q_i$ und $p + 1 \in Q_{i'}$ dann $(q_i, a_{p+1}, q_{i'}) \in \Delta$
4. wenn $n \in Q_i$ dann $q_i \in F$.

Definition.
NFA: Tupel $\mathcal{A} := (Q, \Sigma, q_0, \Delta, F)$, wobei
- Q endl Zustandsmenge,
- Σ ein endl Alphabet,
- q_0 der *Anfangszustand*,
- $F \subseteq Q$ *Endzustände*
- $\Delta \subseteq Q \times \Sigma \times Q$ Übergangs-relation.

Beweis der Rückrichtung

Eigenschaften der Mengen Q_i.

1. $Q_0, \dots, Q_k$ bilden eine Partition von $1, \dots, n$.
2. wenn $1 \in Q_i$ dann $(q_0, a_1, q_i) \in \Delta$
3. wenn $p \in Q_i$ und $p+1 \in Q_{i'}$ dann $(q_i, a_{p+1}, q_{i'}) \in \Delta$
4. wenn $n \in Q_i$ dann $q_i \in F$.

$$\begin{array}{cccccccccc} & q_0 & q_1 & q_1 & q_0 & q_1 & q_1 & q_2 & q_0 & q_1 & q_2 \\ w := & a & b & a & a & b & c & a & a & c & \end{array}$$

Übersetzung in MSO.

$$\begin{aligned} \varphi_{\mathcal{A}} := & \exists Q_0 \dots \exists Q_k \ \varphi_{k+1-part}(Q_0, \dots, Q_k) \wedge \\ & \bigvee_{(q_0,a,q_i)\in\Delta} \big(P_a(min) \wedge Q_i(min)\big) \wedge \\ & \forall x(x < max \to \bigvee_{(q_i,a,q_j)\in\Delta} \big(Q_i(x) \wedge P_a(x+1) \wedge Q_j(x+1)\big)) \wedge \\ & \bigvee_{q_i\in F} Q_i(max) \end{aligned}$$

Lemma. Für alle $w \in \Sigma^+$ gilt: $\mathcal{W}_w \models \varphi_{\mathcal{A}}$ gdw. $\mathcal{A}$ akzeptiert w.

Beweis der Rückrichtung

Lemma. Sei Σ ein endliches Alphabet. Jede reguläre Sprache $\mathcal{L} \subseteq \Sigma^+$ ist MSO-definierbar.

Beweis.

Da $\mathcal{L}$ regulär, existiert ein NFA $\mathcal{A}$ mit $\mathcal{L}(\mathcal{A}) = \mathcal{L}$.

$w = a_1 \dots a_n \in \mathcal{L}$ gdw. es gibt akzeptierenden Lauf
$\rho : \{1, \dots, n\} \to Q$ von $\mathcal{A}$ auf w.

Nach vorherigem Lemma gilt: $w \in \mathcal{L}(\mathcal{A})$ gdw. $\mathcal{W}_w \models \varphi_{\mathcal{A}}$.

Es folgt: $\mathcal{L}(\varphi) = \mathcal{L}$. $\square$

Definition.
NFA: Tupel $\mathcal{A} := (Q, \Sigma, q_0, \Delta, F)$, wobei
- Q endl Zustandsmenge,
- Σ ein endl Alphabet,
- q_0 der *Anfangszustand*,
- $F \subseteq Q$ *Endzustände*
- $\Delta \subseteq Q \times \Sigma \times Q$ Übergangsrelation.

Der Satz von Büchi und Elgot

Theorem. (Büchi und Elgot, 1961) Sei Σ ein endliches Alphabet.

Eine Sprache $\mathcal{L} \subseteq \Sigma^+$ ist genau dann regulär, wenn sie MSO-definierbar ist.

$$\varphi \in \text{MSO} \rightsquigarrow \text{NFA } \mathcal{A}$$

$$\varphi \in \text{MSO} \leftsquigarrow \text{NFA } \mathcal{A}$$

Der Satz von Büchi und Elgot

Wir haben die Rückrichtung schon bewiesen.

Lemma. *Sei* Σ *ein endliches Alphabet. Jede reguläre Sprache* $\mathcal{L} \subseteq \Sigma^+$ *ist* MSO*-definierbar.*

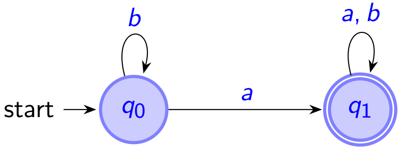
Es bleibt noch die Hinrichtung zu beweisen.

Lemma. Für jedes $\varphi \in \text{MSO}[\sigma(\Sigma)]$ existiert ein NFA $\mathcal{A}$ mit $\mathcal{L}(\varphi) = \mathcal{L}(\mathcal{A})$.

Beweis MSO $\rightarrow$ *NFA*

Lemma. Für jedes $\varphi \in \text{MSO}[\sigma(\Sigma)]$ existiert ein NFA $\mathcal{A}$ mit $\mathcal{L}(\varphi) = \mathcal{L}(\mathcal{A})$.

Beispiel. Sei $\varphi := \exists x P_a(x)$. $(\Sigma := \{a, b\})$



$$\mathcal{L}(\mathcal{A}) = \mathcal{L}(\varphi) = \Sigma^* a \Sigma^*$$

Beispiel

Beispiel. Sei $\varphi := \exists x P_a(x)$. $(\Sigma := \{a, b\})$

$$w := \quad b \quad b \quad b \quad \overset{x}{a} \quad b \quad b$$

Annotierte Wörter. Der Quantor $\exists x$ belegt x mit einer Position i.

Dann wird $P_a(x)$ in $(\mathcal{W}_w, [x/i])$ ausgewertet.

Wir können w mit einer Belegung $[x/i]$ als Wort w' über einem erweiterten Alphabet $\Sigma' := \Sigma \times \{\bar{x}, x\}$ auffassen.

$$w' := \binom{\bar{x}}{b}\binom{\bar{x}}{b}\binom{\bar{x}}{b}\binom{x}{a}\binom{\bar{x}}{b}\binom{\bar{x}}{b}$$

Beispiel

Beispiel. Sei $\varphi := \exists x P_a(x)$. $(\Sigma := \{a, b\})$

Annotierte Wörter. Der Quantor $\exists x$ belegt x mit einer Position i.

Dann wird $P_a(x)$ in $(\mathcal{W}_w, [x/i])$ ausgewertet.

Wir können w mit einer Belegung $[x/i]$ als Wort w' über einem erweiterten Alphabet $\Sigma' := \Sigma \times \{\bar{x}, x\}$ auffassen.

$$w' := \binom{\bar{x}}{b}\binom{\bar{x}}{b}\binom{\bar{x}}{b}\binom{x}{a}\binom{\bar{x}}{b}\binom{\bar{x}}{b}$$

Annotierte Sprache. Der Sprache $\mathcal{L}(\varphi)$ entspricht dann die *annotierte Sprache*

$$\mathcal{L}^a(\varphi) := \left\{ w' = (a_1, v_1) \dots (a_n, v_n) : \begin{array}{l} a_i \in \Sigma, v_i \in \{\bar{x}, x\}, \\ \text{es ex.} i \ \text{ mit } v_i = x \ \text{ und } a_i = a \end{array} \right\}.$$

Beispiel

Beispiel. Sei $\varphi := \exists x P_a(x)$. $(\Sigma := \{a, b\})$

Annotierte Sprache. Der Sprache $\mathcal{L}(\varphi)$ entspricht dann die *annotierte Sprache*

$$\mathcal{L}^a(\varphi) := \left\{ w' = (a_i, v_1) \dots (a_n, i_n) : \begin{array}{l} a_i \in \Sigma, v_i \in \{\bar{x}, x\}, \\ \text{es ex. } i \text{ mit } v_i = x \text{ und } a_i = a \end{array} \right\}.$$

Beobachtung. Es gilt $\mathcal{L}(\varphi) = \left(\mathcal{L}^a(\varphi)\right)_{|\Sigma}$.

$\left(\mathcal{L}^a(\varphi)\right)_{|\Sigma}$: Projektion auf Σ

Übersetzung MSO $\rightarrow$ NFA

Idee. Wir übersetzen $\varphi \in \text{MSO}[\sigma(\Sigma)]$ induktiv über den Formelaufbau, d.h. von innen nach außen.

Eine Unterformel $\psi(x_1, \ldots, x_k)$ definert dabei eine Sprache über dem Alphabet $\Sigma(\psi) := \Sigma \times \{\bar{x}_1, \mathrm{x}_1\} \times \cdots \times \{\bar{x}_k, \mathrm{x}_k\}$.

Quantoren $\exists x$ oder $\exists X$ entsprechen dann der Projektion.

$$\varphi := \exists x P_a(x)$$

$$\Sigma(\varphi) = \Sigma \qquad\qquad \Sigma(P_a(x)) = \Sigma \times \{\bar{x}, \mathrm{x}\}$$

Vereinfachung der Konstruktion

Quantoren und Variablen. In MSO gibt es Elementvariablen x sowie Mengenvariablen X.

Kommt x in Unterformel $\psi(x)$ vor, müssen wir sicherstellen, dass es in der annotierten Sprache für ψ genau einen Buchstaben mit der Annotation x gibt.

Eine *Elementvariable* x ist also nichts anderes als eine *Mengenvariable* X die nur über Mengen der Kardinalität 1 interpretiert wird.

$$\varphi := \exists \mathrm{x} P_a(x)$$

$$\Sigma(\varphi) = \Sigma \qquad\qquad \Sigma(P_a(x)) = \Sigma \times \{\bar{\mathrm{x}}, \mathrm{x}\}$$

Vereinfachung der Konstruktion

Quantoren und Variablen. In MSO gibt es Elementvariablen x sowie Mengenvariablen X.

Kommt x in Unterformel $\psi(x)$ vor, müssen wir sicherstellen, dass es in der annotierten Sprache für ψ genau einen Buchstaben mit der Annotation x gibt.

Eine *Elementvariable* x ist also nichts anderes als eine *Mengenvariable* X die nur über Mengen der Kardinalität 1 interpretiert wird.

MSO_0: Äquivalente Variante von MSO ohne Elementvariablen.

$$\varphi := \exists x P_a(x)$$

$$\Sigma(\varphi) = \Sigma \qquad \Sigma(P_a(x)) = \Sigma \times \{\bar{x}, x\}$$

Die Logik MSO_0

Definition. Sei Σ Alphabet und $\sigma_\Sigma = \sigma(\Sigma)$ Wortsignatur.

Atomare Formeln von MSO_0. Induktiv definiert durch

- $X \subseteq Y \in \mathrm{MSO}_0$ für alle $X, Y \in \mathrm{mVar}_2$.
- $X \subseteq P_a \in \mathrm{MSO}_0$ für alle $X \in \mathrm{mVar}_2$, $P_a \in \sigma_\Sigma$.
- $\mathrm{Sing}(X) \in \mathrm{MSO}_0$ für alle $X \in \mathrm{mVar}_2$.
- $S(X, Y) \in \mathrm{MSO}_0$ für alle $X, Y \in \mathrm{mVar}_2$.
- $X \leq Y \in \mathrm{MSO}_0$ für alle $X, Y \in \mathrm{mVar}_2$.

Die Logik $\mathrm{MSO}_0[\sigma]$. Abschluss der atomaren Formeln unter

- Booleschen Operatoren $\neg, \wedge, \vee$,
- Quantoren $\exists X, \forall X$ für alle $X \in \mathrm{mVar}_2$.

Semantik von MSO_0

Bedeutung der Formeln.

- $X \subseteq Y, X \subseteq P_a$ haben wir schon kennen gelernt

  $X \subseteq Y$ entspricht $\varphi_{\subseteq}(X,Y) := \forall x(X(x) \rightarrow Y(x))$

- $\mathrm{Sing}(X)$ bedeutet „X ist eine Einermenge“

  $$\varphi_{\mathrm{Sing}}(X) := \exists x(X(x) \wedge \forall y(X(y) \rightarrow x = y))$$

- $S(X,Y)$ bedeutet „$X = \{x\}, Y = \{y\}$ und $y = x+1$“

  $$\mathrm{Sing}(X) \wedge \mathrm{Sing}(Y) \wedge \exists x \exists y(X(x) \wedge Y(y) \wedge x+1 = y)$$

- $X \leq Y$ bedeutet „$X = \{x\}, Y = \{y\}$ und $x \leq y$“

  $$\mathrm{Sing}(X) \wedge \mathrm{Sing}(Y) \wedge \exists x \exists y(X(x) \wedge Y(y) \wedge x \leq y)$$

MSO_0

- $X \subseteq Y \in \mathrm{MSO}_0$
- $X \subseteq P_a \in \mathrm{MSO}_0$
- $\mathrm{Sing}(X) \in \mathrm{MSO}_0$
- $S(X,Y) \in \mathrm{MSO}_0$
- $X \leq Y \in \mathrm{MSO}_0$
- Bool. Operatoren $\neg, \wedge, \vee,$
- Quantoren $\exists X, \forall X$

Lemma. Jede MSO-Formel ist über Wortstrukturen äquivalent zu einer MSO_0-Formel.

Semantik von MSO_0

Bedeutung der Formeln.

- $X \subseteq Y, X \subseteq P_a$ haben wir schon kennen gelernt

  $X \subseteq Y$ entspricht $\varphi_{\subseteq}(X, Y) := \forall x(X(x) \rightarrow Y(x))$

- $\mathrm{Sing}(X)$ bedeutet „X ist eine Einermenge“

  $$\varphi_{\mathrm{Sing}}(X) := \exists x(X(x) \wedge \forall y(X(y) \rightarrow x = y))$$

- $S(X, Y)$ bedeutet „$X = \{x\}, Y = \{y\}$ und $y = x + 1$“

Beispiel.

$$\forall x(P_a(x) \rightarrow \exists y(x \leq y \wedge P_b(y)))$$

$$\forall X(\mathrm{Sing}(x) \wedge X \subseteq P_a \rightarrow \exists Y(\mathrm{Sing}(Y) \wedge X \leq Y \wedge Y \subseteq P_b))$$

Lemma. Jede MSO-Formel ist über Wortstrukturen äquivalent zu einer MSO_0-Formel.

MSO_0

- $X \subseteq Y \in \mathrm{MSO}_0$
- $X \subseteq P_a \in \mathrm{MSO}_0$
- $\mathrm{Sing}(X) \in \mathrm{MSO}_0$
- $S(X, Y) \in \mathrm{MSO}_0$
- $X \leq Y \in \mathrm{MSO}_0$
- Bool. Operatoren $\neg, \wedge, \vee$,
- Quantoren $\exists X, \forall X$

Notation

Sei $\varphi(X_1, \dots, X_k) \in \mathrm{MSO}_0[\sigma(\Sigma)]$ und $\Sigma(X_1, \dots, X_k) := \Sigma \times \{\bar{x}_1, x_1\} \times \dots \times \{\bar{x}_k, x_k\}$.

1. $\mathcal{W}_w = (W, \sigma(\Sigma))$ Wortmodell für $w = a_1 \dots a_n$, $\quad U_1, \dots U_k \subseteq W$.

   $\mathcal{W}' := (\mathcal{W}_w, U_1, \dots, U_k)$ entspricht dem Wort

   $$w(\mathcal{W}') := \begin{pmatrix} v_1^1 \\ \dots \\ v_1^k \\ a_1 \end{pmatrix} \dots \begin{pmatrix} v_n^1 \\ \dots \\ v_n^k \\ a_n \end{pmatrix} \text{ mit } v_i^j = \begin{cases} x_i & \text{wenn } i \in U_j \\ \bar{x}_i & \text{sonst} \end{cases}$$

2. Umgekehrt entspricht jedes Wort

   $$w = ((a_1, v_1^1, \dots, v_1^k), \dots, (a_n, v_n^1, \dots, v_n^k)) \in \Sigma(X_1, \dots, X_k)$$

   einem Wortmodel

   $$m(w) := (\mathcal{W}_{a_1 \dots a_n}, U_1(w), \dots, U_l(w)) \text{ mit } i \in U_j(w) \text{ gdw. } v_i^j = x_j$$

   $$1 \leq i \leq n, 1 \leq j \leq k$$

Notation

Definition.

- Sei $\varphi(X_1, \dots, X_k) \in \text{MSO}_0[\sigma(\Sigma)]$.

  $$w\text{Mod}(\varphi) := \{\mathcal{W} : \mathcal{W} = (\mathcal{W}_w, U_1, \dots, U_k) \models \varphi\}$$

  Klasse aller Wortmodelle über $\Sigma(X_1, ..., X_k)$, die φ erfüllen.

- Sei $\mathcal{K}$ eine Klasse von Wortmodellen über $\Sigma(X_1, \dots, X_k)$.

  $$\mathcal{L}(\mathcal{K}) = \{w(\mathcal{W}) : \mathcal{W} \in \mathcal{K}\} \subseteq \Sigma(X_1, \dots, X_k)^+$$

  Sprache, die der Klasse $\mathcal{K}$ entspricht.

- Sei $L \subseteq \Sigma(X_1, \dots, X_k)^+$

  $$\mathcal{M}(L) := \{m(w) : w \in L\}$$

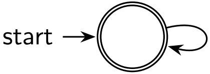
  Klasse von Wortmodellen, die der Sprache L entspricht.

Beweis MSO $\rightarrow$ *NFA*

Lemma. Für alle $\varphi(X_1, ..., X_k) \in \text{MSO}_0[\sigma(\Sigma)]$ ist $\mathcal{L}(\varphi)$ regulär.

Beweis. O.B.d.A. sei $\varphi := Q_{k+1}X_{k+1} \dots Q_{k+l}X_{k+l}\psi(X_1, \dots, X_k, X_{k+1}, \dots, X_{k+l})$ in PNF.

Automat für $X_i \subseteq X_j$. (O.b.d.A. $i < j$)



$(..., \bar{x}_i, ..., \bar{x}_j, ...)$

$(..., \bar{x}_i, ..., \mathrm{x_j}, ...)$

$(..., \mathrm{x_i}, ..., \mathrm{x_j}, ...)$

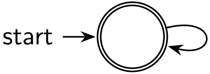
... steht für beliebige Einträge an den übrigen Positionen
b beliebiger Buchstabe $b \neq a$

Beweis MSO → NFA

Lemma. Für alle $\varphi(X_1, ..., X_k) \in \text{MSO}_0[\sigma(\Sigma)]$ ist $\mathcal{L}(\varphi)$ regulär.

Beweis. O.B.d.A. sei $\varphi := Q_{k+1}X_{k+1} \dots Q_{k+l}X_{k+l}\psi(X_1, \dots, X_k, X_{k+1}, \dots, X_{k+l})$ in PNF.

Automat für $X_i \subseteq X_j$. (O.b.d.A. $i < j$)



$(..., \bar{x}_i, ..., \bar{x}_j, ...)$

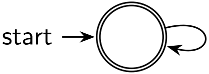
$(..., \bar{x}_i, ..., x_j, ...)$

$(..., x_i, ..., x_j, ...)$

... steht für beliebige Einträge an den übrigen Positionen

b beliebiger Buchstabe $b \neq a$

Beispiel. Sei $\Sigma := \{a, b\}$, $k = 3$, $i = 1$, $j = 3$.



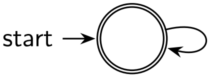
$(a, \bar{x}_1, \bar{x}_2, \bar{x}_3)$
$(b, \bar{x}_1, \bar{x}_2, \bar{x}_3)$
$(a, \bar{x}_1, x_2, \bar{x}_3)$
$(b, \bar{x}_1, x_2, \bar{x}_3)$
$(a, \bar{x}_1, \bar{x}_2, x_3)$
$(b, \bar{x}_1, \bar{x}_2, x_3)$
$(a, \bar{x}_1, x_2, x_3)$
$(b, \bar{x}_1, x_2, x_3)$
$(a, x_1, \bar{x}_2, x_3)$
$(b, x_1, \bar{x}_2, x_3)$
(a, x_1, x_2, x_3)
(b, x_1, x_2, x_3)

Beweis MSO $\rightarrow$ *NFA*

Lemma. Für alle $\varphi(X_1, ..., X_k) \in \mathrm{MSO}_0[\sigma(\Sigma)]$ ist $\mathcal{L}(\varphi)$ regulär.

Beweis. O.B.d.A. sei $\varphi := Q_{k+1}X_{k+1} \dots Q_{k+l}X_{k+l}\psi(X_1, \dots, X_k, X_{k+1}, \dots, X_{k+l})$ in PNF.

Automat für $X_i \subseteq P_a$.



$(b, ..., \bar{x}_i, ...)$

$(a, ..., \bar{x}_i, ...)$

$(a, ..., \mathrm{x}_i, ...)$

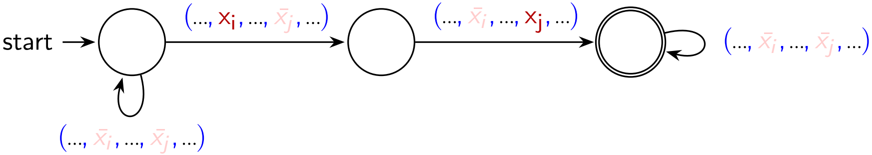
... steht für beliebige Einträge an den übrigen Positionen
b beliebiger Buchstabe $b \neq a$

Beweis MSO $\rightarrow$ *NFA*

Lemma. Für alle $\varphi(X_1, \dots, X_k) \in \mathrm{MSO}_0[\sigma(\Sigma)]$ ist $\mathcal{L}(\varphi)$ regulär.

Beweis. O.B.d.A. sei $\varphi := Q_{k+1}X_{k+1} \dots Q_{k+l}X_{k+l}\psi(X_1, \dots, X_k, X_{k+1}, \dots, X_{k+l})$ in PNF.

Automat für $S(X_i, X_j)$. (O.b.d.A. $i < j$)



Beweis MSO $\to$ NFA

Lemma. Für alle $\varphi(X_1, ..., X_k) \in \text{MSO}_0[\sigma(\Sigma)]$ ist $\mathcal{L}(\varphi)$ regulär.

Beweis. O.B.d.A. sei $\varphi := Q_{k+1}X_{k+1} \dots Q_{k+l}X_{k+l}\psi(X_1, \dots, X_k, X_{k+1}, \dots, X_{k+l})$ in PNF.

Atomare Formeln. Die anderen atomaren Fälle sind ähnlich.

Boolesche Kombinationen. Die Boolschen Operationen $\neg, \vee, \wedge$ entsprechen *Komplement, Vereinigung* und *Schnitt*.

Quantoren $\exists X, \forall X$. $\forall X\psi \equiv \neg\exists X\neg\psi$.

Es bleibt also noch $\varphi(X_1, \dots, X_i) := \exists X_{i+1}\psi$.

$\mathcal{L}(\varphi)$ ist die Projektion von $\mathcal{L}(\psi)$ auf $\Sigma(X_1, \dots, X_i)$.

Da reguläre Sprachen unter Projektion abgeschlossen sind, kann der entsprechende Automat konstruiert werden. $\square$

Beweis MSO $\rightarrow$ *NFA*

Lemma. Für alle $\varphi(X_1, ..., X_k) \in \text{MSO}_0[\sigma(\Sigma)]$ ist $\mathcal{L}(\varphi)$ regulär.

Beweis. O.B.d.A. sei $\varphi := Q_{k+1}X_{k+1} \dots Q_{k+l}X_{k+l}\psi(X_1, \dots, X_k, X_{k+1}, \dots, X_{k+l})$ in PNF.

Atomare Formeln. Die anderen atomaren Fälle sind ähnlich.

Boolesche Kombinationen. Die Boolschen Operationen $\neg, \vee, \wedge$ entsprechen *Komplement*, *Vereinigung* und *Schnitt*.

$\mathcal{L}(\psi) \subseteq \Sigma(X_1, \dots, X_{i+1})$
$\mathcal{L}(\varphi) \subseteq \Sigma(X_1, \dots, X_i)$.

Der Satz von Büchi und Elgot

Theorem. (Büchi und Elgot, 1961) Sei Σ ein endliches Alphabet.

Eine Sprache $\mathcal{L} \subseteq \Sigma^+$ ist genau dann regulär, wenn sie MSO-definierbar ist.

$$\varphi \in \text{MSO} \rightsquigarrow \text{NFA } \mathcal{A}$$

$$\varphi \in \text{MSO} \leftsquigarrow \text{NFA } \mathcal{A}$$

Anmerkungen

Folgerung aus dem Beweis. Die Übersetzung von MSO-Formeln in Automaten und umgekehrt ist *effektiv*.

Das Verfahren liefert uns daher einen einfachen *Auswertungsalgorithmus* für MSO-Formeln über Wortmodellen.

Algorithmus wMC(MSO).

Eingabe. Wort $w \in \Sigma^*$, Formel $\varphi \in \text{MSO}[\sigma(\Sigma)]$.

Ausgabe. *ja*, wenn $\mathcal{W}_w \models \varphi$, *nein* sonst

1. Berechne NFA $\mathcal{A}$ mit $\mathcal{L}(\mathcal{A}) = \mathcal{L}(\varphi)$.
2. Teste ob $\mathcal{A}$ das Wort w akzeptiert.

Laufzeit. $O(\ f(|\varphi|) \cdot |w|)$

Folgerungen aus dem Satz von Büchi und Elgot

Existentielles MSO. Das *existentielle Fragment* $\exists$MSO *von* MSO ist die Menge aller MSO-Formeln in denen Variablen $X \in \mathrm{mVar}_2$ nur existentiell quantifiziert werden dürfen.

Bemerkung.
Keine „versteckten“ Allquantoren.

D.h. keine negierten Existenzquantoren.

Für $x \in \mathrm{Var}_1$ sind Allquantoren erlaubt.

Beispiel.

$$\exists C_1 \exists C_2 \exists C_3 \quad \varphi_{3-part}(C_1, C_2, C_3) \wedge \forall u \forall v \Big(E(u,v) \rightarrow \bigwedge_{i=1}^{3} \neg\big(C_i(u) \wedge C_i(v)\big)\Big) \; .$$

Lemma. Über Wortmodellen ist jede MSO$[\sigma_\Sigma]$-Formel äquivalent zu einer $\exists$MSO-Formel.

Beweis. Sei $\varphi \in$ MSO. Nach dem Satz von Büchi und Elgot existiert ein NFA $\mathcal{A}$ mit $\mathcal{L}(\mathcal{A}) = \mathcal{L}(\varphi)$.
Im Beweis des Satzes wurde eine Formel $\psi \in$ MSO konstruiert mit $\mathcal{L}(\psi) = \mathcal{L}(\varphi)$. Nach Konstruktion ist ψ in $\exists$MSO.

Zusammenfassung

Wir haben gesehen.

- MSO-Formeln können effektiv in äquivalente Automaten übersetzt werden.
- Umgekehrt können Automaten in äquivalente MSO-Formeln übersetzt werden.
- Beide Richtungen zusammen ergeben den Satz von Büchi und Elgot.
- Da die Übersetzung effektiv ist, erhalten wir direkt einen Auswertungsalgorithmus für MSO-Formeln, der für eine feste Formel φ in *Linearzeit* arbeitet.

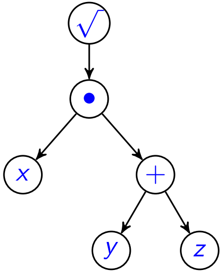
Ausblick. Wir werden diese Analogie zwischen Automaten und MSO als nächstes auf Baumsprachen erweitern.

Baumsprachen und Baumautomaten

Bäume

Ziel. Äquivalenz zwischen MSO und Automaten von endlichen Wörtern auf endliche Bäume erweitern.

Beispiel. Der Term $\sqrt{x \cdot (y + z)}$ hat die Baumstruktur

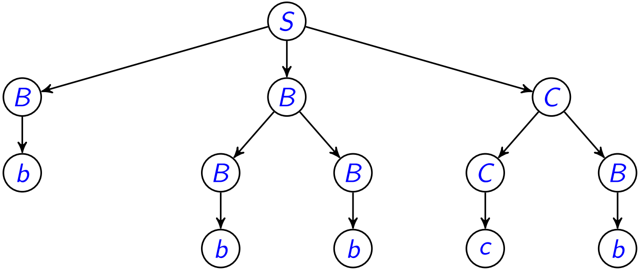


Bäume

Ziel. Äquivalenz zwischen MSO und Automaten von endlichen Wörtern auf endliche Bäume erweitern.

Beispiel. Grammatik $S \to BBC$, $B \to b \mid BB$, $C \to c \mid CB$.

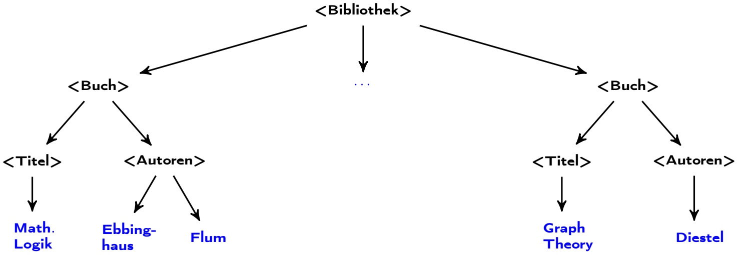
Ein möglicher Ableitungsbaum für das Wort $bbbcb$.



Bäume

Ziel. Äquivalenz zwischen MSO und Automaten von endlichen Wörtern auf endliche Bäume erweitern.

Beispiel (XML-Dokumente).



Bäume

Ziel. Äquivalenz zwischen MSO und Automaten von endlichen Wörtern auf endliche Bäume erweitern.

Bäume mit festem oder beliebigem Rang.

Fester Rang

In Ableitungsbäumen ist die Zahl der Nachfolger eines Knotens durch seine Beschriftung bestimmt.

Freier Rang

In XML-Dokumenten kann ein Element beliebig viele Nachfolger haben.

Rangalphabete

Definition. Ein *Rangalphabet* ist eine nicht leere endliche Menge Γ von Symbolen.

Jedem $a \in \Gamma$ ist eine endliche Menge $\mathrm{rg}(a) \subset \mathbb{N}$ von *Rängen* oder *Stelligkeiten* zugeordnet.

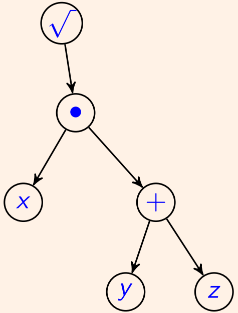
Zu Γ und $i \in \mathbb{N}$ definieren wir $\Gamma_i := \{a \in \Gamma : i \in \mathrm{rg}(a)\}$.

Beispiel.

$$\begin{aligned} \Gamma &= \{x, y, z, \sqrt{}, \cdot, +\}, \\ \Gamma_0 &= \{x, y, z\}, \\ \Gamma_1 &= \{\sqrt{}\} \quad \text{und} \\ \Gamma_2 &= \{\cdot, +\}. \end{aligned}$$

Beispiel

Der Term $\sqrt{x \cdot (y + z)}$ hat die Baumstruktur



Γ-*Bäume*

Definition. Sei Γ ein Rangalphabet.

Ein Γ-*Baum* ist ein Paar (T, β), wobei

- $T = (V(T), E(T))$ ein Baum (mit Wurzel) und
- $\beta : V(T) \to \Gamma$ eine Beschriftungsfunktion ist,

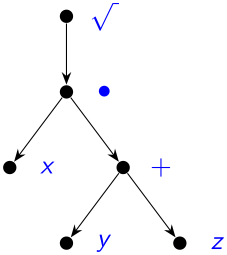
so dass $l \in \mathrm{rg}(\beta(t))$ für alle $t \in V(T)$ mit l Nachfolgern gilt.

Definition. Sei Γ ein Rangalphabet.

Wir schreiben $\mathcal{T}_\Gamma$ für die Klasse aller Γ-Bäume.

Eine *Baumsprache über* Γ ist eine Menge von Γ-Bäumen.

Beispiel.



Baumautomaten

Ziel. Analog zu endlichen Automaten über Wörtern definieren wir Automaten, die Γ-Bäume als Eingabe akzeptieren.

Wie liest man einen Baum. Zwei natürliche Richtungen, in denen ein Baum gelesen werden kann:

- *bottom-up*: von den Blättern zur Wurzel
- *top-down*: von der Wurzel zu den Blättern

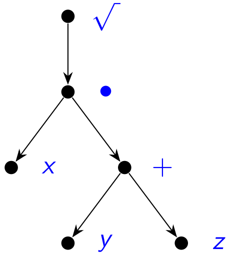
Richtung bei Wörtern. Automaten, die Eingabewörter von rechts nach links lesen, definieren genau die regulären Sprachen.

Richtung bei Bäumen.

Bei Bäumen macht es einen Unterschied, ob wir den Baum von unten nach oben oder von oben nach unten lesen.

Bottom-Up Baumautomaten

Beispiel.



Definition. Ein *nichtdeterministischer Bottom-Up-Baumautomat* $(\mathrm{NBA}_{\uparrow})$ $\mathcal{A}$ ist ein Tupel $\mathcal{A} = (Q, \Gamma, \Delta, F)$, wobei

- Q eine endliche Zustandsmenge ist,
- Γ ein Rangalphabet ist,
- $F \subseteq Q$ eine Menge von Endzuständen ist und
- $\Delta \subseteq \bigcup_{0 \leq i \leq m}(Q^i \times \Gamma_i \times Q)$ ist, wobei m der maximale Rang von Γ ist (und $Q^0 \times \Gamma_0 \times Q \cong \Gamma_0 \times Q$)

$\mathcal{A}$ ist *deterministisch* $(\mathrm{DBA}_{\uparrow})$, wenn Δ funktional ist.

Beispiel

Rangalphabet. $\Gamma := \{0, 1, +, \cdot\} \qquad \Gamma_0 := \{0, 1\} \qquad \Gamma_2 := \{+, \cdot\}$.

Automat. $Q = \{q_0, q_1, q_2\} \qquad F = \{q_0\}$.

$$\begin{tikzcd} & (+, q_{i+j \bmod 3}) & \\ q_i \arrow[ur] & & q_j \arrow[ul] \end{tikzcd} \qquad \begin{tikzcd} & (\cdot, q_{i \cdot j \bmod 3}) & \\ q_i \arrow[ur] & & q_j \arrow[ul] \end{tikzcd} \qquad \begin{tikzcd} q_0 \\ 0 \arrow[u] \end{tikzcd} \qquad \begin{tikzcd} q_1 \\ 1 \arrow[u] \end{tikzcd}$$

$((q_i, q_j), +) \mapsto q_{i+j \bmod 3} \qquad ((q_i, q_j), \cdot) \mapsto q_{i \cdot j \bmod 3} \qquad 0 \mapsto q_0 \qquad 1 \mapsto q_1$

NBA$_\uparrow$. $\mathcal{A} = (Q, \Gamma, \Delta, F)$
- Q endl Zustandsmenge
- Γ Rangalphabet
- $F \subseteq Q$ Endzustände
- $\Delta \subseteq \bigcup_{0 \le i \le m} (Q^i \times \Gamma_i \times Q)$

Lauf eines Baumautomaten

Definition. Sei Γ ein Rangalphabet, $\mathcal{A}$ ein $\mathrm{NBA}_\uparrow$ und $T \in \mathcal{T}_\Gamma$.

Ein *Lauf* von $\mathcal{A}$ auf T ist eine Abbildung $\rho : V(T) \to Q$ s.d.

- $(\beta(t), \rho(t)) \in \Delta$ für alle Blätter $t \in V(T)$ und
- für alle $t \in V(T)$ mit $\beta(t) \in \Gamma_i$ und Nachfolgern $(t_1, \dots, t_i)$ gilt:
$$\Big((\rho(t_1), \dots, \rho(t_i)), \beta(t), \rho(t)\Big) \in \Delta.$$

ρ ist *akzeptierend*, wenn $\rho(w) \in F$ für die Wurzel w von T.

Die *akzeptierte Sprache* $\mathcal{L}(\mathcal{A}) \subseteq \mathcal{T}_\Gamma$ ist die Menge der Bäume $T \in \mathcal{T}_\Gamma$, so dass $\mathcal{A}$ einen akzeptierenden Lauf auf T hat.

Eine Baumsprache $L \subseteq \mathcal{T}_\Gamma$ ist *regulär*, wenn es einen $\mathrm{NBA}_\uparrow$ gibt, der L akzeptiert.

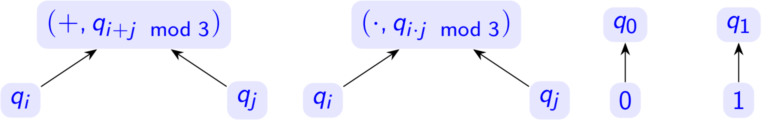
$\mathrm{NBA}_\uparrow$. $\mathcal{A} = (Q, \Gamma, \Delta, F)$
- Q endl Zustandsmenge
- Γ Rangalphabet
- $F \subseteq Q$ Endzustände
- $\Delta \subseteq \bigcup_{0 \le i \le m} (Q^i \times \Gamma_i \times Q)$

Automat $Q := \{q_0, q_1, q_2\}$
$F := \{q_0\}$
$\Delta := \{$
$(q_i, q_j) \mapsto q_{i+j \mod 3}$,
$(q_i, q_j) \mapsto q_{i \cdot j \mod 3}$,
$0 \mapsto q_0$,
$1 \mapsto q_1\}$

Beispiel

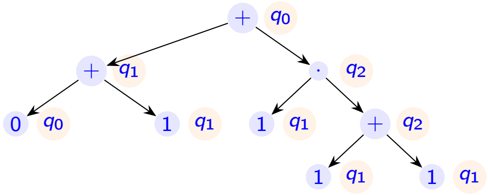
Rangalphabet. $\Gamma := \{0, 1, +, \cdot\}$ $\Gamma_0 := \{0, 1\}$ $\Gamma_2 := \{+, \cdot\}$.

Automat. $Q = \{q_0, q_1, q_2\}$ $F = \{q_0\}$.



$((q_i, q_j), +) \mapsto q_{i+j \mod 3}$ $((q_i, q_j), \cdot) \mapsto q_{i \cdot j \mod 3}$ $0 \mapsto q_0$ $1 \mapsto q_1$

Beispiel.



NBA$_\uparrow$. $\mathcal{A} = (Q, \Gamma, \Delta, F)$

- Q endl Zustandsmenge
- Γ Rangalphabet
- $F \subseteq Q$ Endzustände
- $\Delta \subseteq \bigcup_{0 \le i \le m}(Q^i \times \Gamma_i \times Q)$

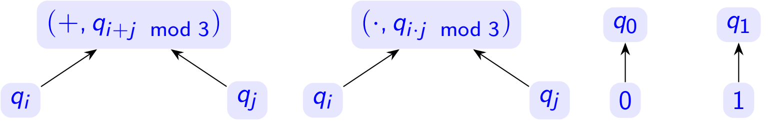
Lauf von $\mathcal{A}$ auf $T \in \mathcal{T}_\Gamma$.
Abbildung $\rho : V(T) \to Q$:

- $(\beta(t), \rho(t)) \in \Delta$
  $t \in V(T)$ Blatt
- $(((\rho(t_1), \dots, \rho(t_i)),$
  $\beta(t), \rho(t))) \in \Delta$
  $t \in V(T)$ mit $\beta(t) \in \Gamma_i$ und Nachfolgern $(t_1, \dots, t_i)$

Beispiel

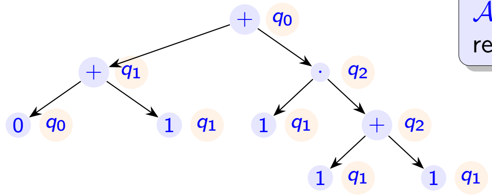
Rangalphabet. $\Gamma := \{0, 1, +, \cdot\}$ $\Gamma_0 := \{0, 1\}$ $\Gamma_2 := \{+, \cdot\}$.

Automat. $Q = \{q_0, q_1, q_2\}$ $F = \{q_0\}$.



$((q_i, q_j), +) \mapsto q_{i+j \mod 3}$ $((q_i, q_j), \cdot) \mapsto q_{i\cdot j \mod 3}$ $0 \mapsto q_0$ $1 \mapsto q_1$

Beispiel.



NBA$_\uparrow$. $\mathcal{A} = (Q, \Gamma, \Delta, F)$
- Q endl Zustandsmenge
- Γ Rangalphabet
- $F \subseteq Q$ Endzustände
- $\Delta \subseteq \bigcup_{0 \leq i \leq m}(Q^i \times \Gamma_i \times Q)$

Lauf von $\mathcal{A}$ auf $T \in \mathcal{T}_\Gamma$.
Abbildung $\rho : V(T) \to Q$:
$\in \Delta$
att
$\rho(t_i)),$
$\rho(t), \rho(t))) \in \Delta$
$t \in V(T)$ mit $\beta(t) \in \Gamma_i$ und Nachfolgern $(t_1, \ldots, t_i)$

Von $\mathcal{A}$ akzeptierte Sprache $\mathcal{L}(\mathcal{A})$.
$\mathcal{A}$ akzeptiert die Sprache aller Bäume, deren Terme zu $0 \mod 3$ auswerten.

Top-Down Baumautomaten

Definition. Sei Γ ein Rangalphabet mit maximalem Rang m.

Ein *nichtdeterministischer Top-Down-Automat* ($\mathrm{NBA}_{\downarrow}$) $\mathcal{A}$ ist ein Tupel $\mathcal{A} = (Q, \Gamma, Q_0, \Delta)$, wobei

- Q eine endliche Zustandsmenge ist,
- $Q_0 \subseteq Q$ eine Menge von Startzuständen ist und
- $\Delta \subseteq \bigcup_{0 \leq i \leq m} (Q \times \Gamma_i \times Q^i)$ eine Transitionsrelation ist.

$\mathcal{A}$ ist *deterministisch* (ein $\mathrm{DBA}_{\downarrow}$), wenn $|Q_0| = 1$ und Δ funktional ist.

Automat $\mathcal{A}$.

$Q := \{q_a, q_f\}, \quad Q_0 := \{q_a\}$

$$\begin{tikzcd}[column sep=small] & (q_a, b) \arrow[dl] \arrow[dr] & \\ q_a & & q_f \end{tikzcd}$$

$(q_a, b, (q_a, q_f))$

$$\begin{tikzcd}[column sep=small] & (q_a, b) \arrow[dl] \arrow[dr] & \\ q_f & & q_a \end{tikzcd}$$

$(q_a, b, (q_f, q_a))$

$$\begin{tikzcd}[column sep=small] & (q_a, a) \arrow[dl] \arrow[dr] & \\ q_f & & q_f \end{tikzcd}$$

$(q_a, a, (q_f, q_f))$

$$\begin{tikzcd}[column sep=small] & (q_f, a/b) \arrow[dl] \arrow[dr] & \\ q_f & & q_f \end{tikzcd}$$

$(q_a, b, (q_f, q_a))$

$(q_f, a/b)$

(q_a, a)

(q_a, a)

$(q_f, a/b)$

Top-Down Baumautomaten

Definition.
$\mathrm{NBA}_{\downarrow}$ $\mathcal{A} = (Q, \Gamma, Q_0, \Delta)$,
- Q Zustandsmenge
- $Q_0 \subseteq Q$ Startzustände
- $\Delta \subseteq \bigcup_{0 \leq i \leq m} (Q \times \Gamma_i \times Q^i)$

Definition. Ein Lauf von $\mathcal{A}$ auf $T = (V(T), \beta)$ mit Wurzel w ist eine Abbildung $\rho : V(T) \to Q$ mit

- $\rho(w) \in Q_0$
- für $t \in V(T)$ mit Nachfolgern $(s_1, \ldots, s_i)$, $i > 0$ ist $\big(\rho(t), \beta(t), (\rho(s_1), \ldots, \rho(s_i))\big) \in \Delta$ und
- für ein Blatt $t \in V(T)$ ist $\big(\rho(t), \beta(t)\big) \in \Delta$.

$\mathcal{A}$ akzeptiert T, wenn es einen Lauf gibt.

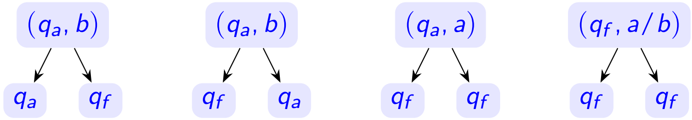
Beispiel Top-Down Automat

Definition.
$\mathrm{NBA}_\downarrow$ $\mathcal{A} = (Q, \Gamma, Q_0, \Delta)$,
- Q Zustandsmenge
- $Q_0 \subseteq Q$ Startzustände
- $\Delta \subseteq \bigcup_{0 \le i \le m} (Q \times \Gamma_i \times Q^i)$

Rangalphabet. $\Gamma := \{a, b\}$ mit $\mathrm{rg}(a) = \{0, 2\} = \mathrm{rg}(b)$.

Sprache. $\mathcal{L}$ = Sprache aller Γ-Bäume die (mind.) ein a enthalten.

Automat $\mathcal{A}$. $Q = \{q_a, q_f\}$ $\quad$ $Q_0 := \{q_a\}$

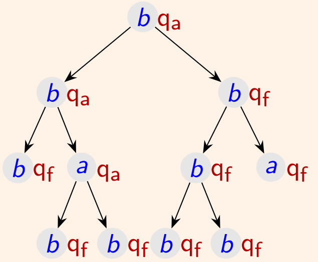


$(q_a, b, (q_a, q_f))$ $\quad$ $(q_a, b, (q_f, q_a))$ $\quad$ $(q_a, a, (q_f, q_f))$ $\quad$ $(q_f, a/b, (q_f, q_f))$

$(q_f, a/b)$ $\quad$ (q_a, a)

$(q_f, a/b)$ $\quad$ (q_a, a)

Baum (T, β).



Beispiel Top-Down Automat

Rangalphabet. $\Gamma := \{a, b\}$ mit $\mathrm{rg}(a) = \{0, 2\} = \mathrm{rg}(b)$.

Spr[Suche nach a]**che aller Γ**[Suche woanders]) ein a enthalten.

Automat $\mathcal{A}$. $Q = \{q_a, q_f\}$ $\qquad Q_0 := \{q_a\}$

$(q_a, b) \to q_a,\ q_f$ $\qquad (q_a, b, (q_a, q_f))$

$(q_a, b) \to q_f,\ q_a$ $\qquad (q_a, b, (q_f, q_a))$

$(q_a, a) \to q_f,\ q_f$ $\qquad (q_a, a, (q_f, q_f))$

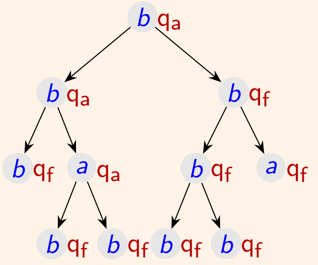
$(q_f, a/b) \to q_f,\ q_f$ $\qquad (q_f, a/b, (q_f, q_f))$

$(q_f, a/b)$ $\qquad (q_f, a/b)$

(q_a, a) $\qquad (q_a, a)$

Definition.
$\mathrm{NBA}_{\downarrow}$ $\mathcal{A} = (Q, \Gamma, Q_0, \Delta)$,
- Q Zustandsmenge
- $Q_0 \subseteq Q$ Startzustände
- $\Delta \subseteq \bigcup_{0 \leq i \leq m} (Q \times \Gamma_i \times Q^i)$

Baum (T, β).



Vergleich Bottom-Up und Top-Down

NBA$_\uparrow$. $\mathcal{A} = (Q, \Gamma, \Delta, F)$
- Q endl Zustandsmenge
- Γ Rangalphabet
- $F \subseteq Q$ Endzustände
- $\Delta \subseteq \bigcup_{0 \le i \le m}(Q^i \times \Gamma_i \times Q)$

NBA$_\downarrow$ $\mathcal{A} = (Q, \Gamma, Q_0, \Delta)$
- Q Zustandsmenge
- $Q_0 \subseteq Q$ Startzustände
- $\Delta \subseteq \bigcup_{0 \le i \le m}(Q \times \Gamma_i \times Q^i)$

Bottom-Up Automat $\mathcal{U}$. $Q := \{q_0, q_1\}$, $F := \{q_1\}$

$$q_i, q_j \to (q_{\max\{q_i,q_j\}}, b) \qquad b \to q_0 \qquad q_1 \qquad q_i, q_j \to (q_1, a)$$

für $i,j \in \{0,1\}$ — für $i,j \in \{0,1\}$

„berechnet" die Lösung

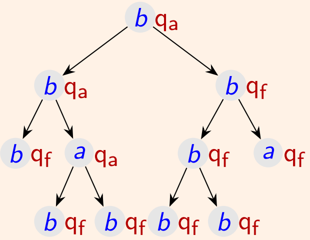
Top-Down Automat $\mathcal{A}$. $Q = \{q_a, q_f\}$ $Q_0 := \{q_a\}$

$$(q_a, b) \to q_a, q_f \qquad (q_a, b) \to q_f, q_a \qquad (q_a, a) \to \dots \qquad (q_f, a/b) \to q_f, q_f$$

$$(q_f, a/b) \qquad (q_a, a)$$

„sucht" die Lösung

Baum (T, β).

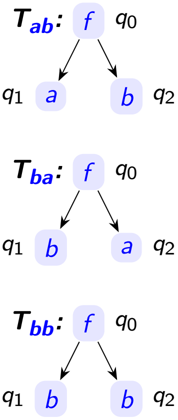


Det. vs. Nicht-Det. Top-Down Automaten

Lemma. Es gibt Baumsprachen, die von *nicht-deterministischen top-down Baumautomaten* erkannt werden, die aber nicht durch einen *deterministischen top-down Baumautomat* erkannt werden können.

Beweis. Betrachte $L := \{T_{ab}, T_{ba}\}$. Sei $\mathcal{A} := (Q, \Gamma, \{q_0\}, \Delta)$ ein $\mathrm{DBA}_{\downarrow}$ der L erkennt.

1. Da $\mathcal{A}$ *deterministisch*, existieren *eindeutige* Zustände $q_1, q_2 \in Q$ mit $(q_0, f, (q_1, q_2)) \in \Delta$.
2. Da $T_{ab} \in \mathcal{L}(\mathcal{A})$ muss $(q_1, a), (q_2, b) \in \Delta$.
3. Da $T_{ba} \in \mathcal{L}(\mathcal{A})$ gilt $(q_1, b), (q_2, a) \in \Delta$.
4. Dann akzeptiert $\mathcal{A}$ auch $T_{bb} \notin L$, Widerspruch zu $\mathcal{L}(\mathcal{A}) = L$.



$\mathrm{NBA}_\uparrow$ *vs.* $\mathrm{NBA}_\downarrow$

Satz.

1. $\mathrm{NBA}_\uparrow, \mathrm{DBA}_\uparrow$ und $\mathrm{NBA}_\downarrow$ erkennen dieselbe Sprachklasse, die *regulären Baumsprachen*.
2. Die durch $\mathrm{DBA}_\downarrow$ erkannte Sprachklasse ist eine echte Teilklasse der regulären Baumsprachen.

$\mathrm{NBA}_\uparrow$ *vs.* $\mathrm{NBA}_\downarrow$

Lemma. $\mathrm{NBA}_\uparrow$ und $\mathrm{DBA}_\uparrow$ erkennen dieselbe Sprachklasse, die *regulären Baumsprachen*.

Beweis. Sei $\mathcal{A} = \{Q, \Gamma, \Delta, F)$ ein $\mathrm{NBA}_\uparrow$.

Konstruiere deterministischen Automat $\mathcal{D} = (Q_d, \Gamma, \Delta_d, F_d)$

- $Q_d := 2^Q = \{Q' : Q' \subseteq Q\}$
- $F_d := \{F' \in Q_d : F \cap F' \neq \emptyset\}$
- $\Delta_D := \big\{(Z_1, \dots, Z_i, a, Z) \ : \ Z_1, \dots, Z_i, Z \in Q_d,$ es ex. $q_1 \in Z_1, \dots, q_i \in Z_i, q \in Z$ mit $(q_1, \dots, q_i, a, q) \in \Delta\big\}$.

Es gilt $\mathcal{L}(\mathcal{A}) = \mathcal{L}(\mathcal{D})$.

$\mathrm{NBA}_\uparrow$ $\mathcal{A}$. $Q := \{q_a, q_0\}$ $\qquad F := \{q_a\}$ $\qquad \Gamma := \{a, b\}$

$$\Delta := \big\{(a, q_a), \quad (a, q_0), \quad (b, q_0), \quad (q_a, q_0, b, q_a), \quad (q_0, q_a, b, q_a), \quad (q_0, q_0, a/b, q_0)\big\}$$

$\mathrm{NBA}_\uparrow$ $\mathcal{D} := (Q_d, \Gamma, \Delta_d, F_d)$

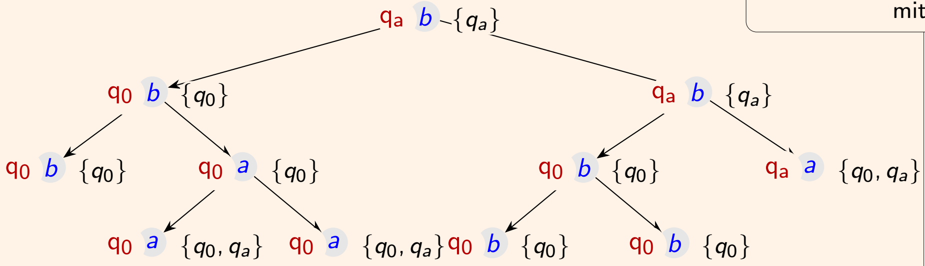
- $Q_d := 2^Q = \{Q' : Q' \subseteq Q\}$
- $F_d := \{F' \in Q_d : F \cap F' \neq \emptyset\}$
- $\Delta_D := \big\{(Z_1, \dots, Z_i, a, Z) :$ $Z_1, \dots, Z_i, Z \in Q_d$, es ex. $q_1 \in Z_1, \dots, q_i \in Z_i, q \in Z$ mit $(q_1, \dots, q_i, a, q) \in \Delta\big\}$.

NBA$_\uparrow$ *vs.* NBA$_\downarrow$

Lemma. NBA$_\uparrow$ und DBA$_\uparrow$ erkennen dieselbe Sprachklasse, die *regulären Baumsprachen*.

Beweis. Sei $\mathcal{A} = (Q, \Gamma, \Delta, F)$ ein NBA$_\uparrow$.

Baum (T, β).



NBA$_\uparrow$ $\mathcal{D} := (Q_d, \Gamma, \Delta_d, F_d)$

- $Q_d := 2^Q = \{Q' : Q' \subseteq Q\}$
- $F_d := \{F' \in Q_d : F \cap F' \neq \emptyset\}$
- $\Delta_D := \{(Z_1, \ldots, Z_i, a, Z) :$ $Z_1, \ldots, Z_i, Z \in Q_d$, es ex. $q_1 \in Z_1, \ldots, q_i \in Z_i, q \in Z$ mit $(q_1, \ldots, q_i, a, q) \in \Delta\}$.

NBA$_\uparrow$ $\mathcal{A}$. $Q := \{q_a, q_0\}$ $\quad F := \{q_a\}$ $\quad \Gamma := \{a, b\}$

$$\Delta := \{(a, q_a), \quad (a, q_0), \quad (b, q_0), \quad (q_a, q_0, b, q_a), \quad (q_0, q_a, b, q_a), \quad (q_0, q_0, a/b, q_0)\}$$

MSO-definierbare Baumsprachen

Definition. Sei Γ ein Rangalphabet mit maximalem Rang m.

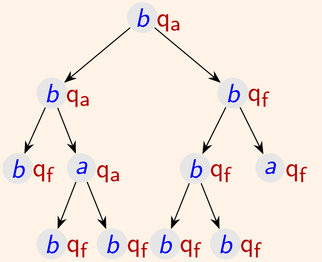
Definiere $\sigma_\Gamma := \{(P_a)_{a\in\Gamma}, S_1, \dots, S_m, \leq\}$, wobei

- P_a, für $a \in \Gamma$, einstellige Relationssymbole und
- S_i, $1 \leq i \leq m$ und $\leq$ zweistellige Relationssymbole sind.

Wir interpretieren einen beschrifteten Baum $(T, \beta) \in \mathcal{T}_\Gamma$ als σ_Γ-Struktur, wobei

- $(s, t) \in S_i$ wenn t der i-te Nachfolger von s ist und
- $\leq$ als Präfixordnung interpretiert wird, d.h. $s \leq t$ wenn der eindeutige Pfad in T von der Wurzel zu t durch s läuft.

Baum (T, β).



MSO-Definierbare Baumsprachen

Rangalphabet. $\Gamma := \{a, b\}$ mit $\mathrm{rg}(a) = \{0, 2\} = \mathrm{rg}(b)$.

Beispiel 1.

$\mathcal{L}$ = Sprache aller Γ-Bäume die (mind.) ein a enthalten.

$\varphi := \exists x P_a(x)$

Beispiel 2.

$\mathcal{L}_2$ = Sprache aller Γ-Bäume in denen jedes mit a beschriftete Blatt einen Vorgänger hat, der mit b beschriftet ist.

Vorgänger: Knoten auf dem Weg von der Wurzel zum Blatt

$S(x, y) := \bigvee_{i=1}^{m} S_i(x, y)$ „y ist ein Nachfolger von x“

$\varphi_l(x) := \neg\exists y S(x, y)$ „x ist ein Blatt“

$\varphi_2 := \forall x\big(\varphi_l(x) \wedge P_a(x) \rightarrow \exists y y \leq x \wedge P_b(y)\big)$

Der Satz von Doner, Thatcher und Wright

Satz (Doner, und unabhängig Thatcher and Wright, 1968). Eine Baumsprache $\mathcal{L}$ ist genau dann regulär, wenn sie MSO-definierbar ist.

Beweis. Der Beweis funktioniert genau wie der Beweis des Satzes von Büchi und Elgot.

1. $\mathrm{NBA}_{\uparrow} \rightarrow \mathrm{MSO}$: Wir definieren einen akzeptierenden Lauf eines Automaten in MSO.
2. $\mathrm{MSO} \rightarrow \mathrm{NBA}_{\uparrow}$: Wir definieren wieder zunächst eine Hilfslogik MSO_0 ohne FO-Variablen.

   Danach übersetzen wir induktiv über den Formelaufbau die MSO_0-Formel in einen Automaten.

Zusammenfassung

Top-Down Baumautomaten. Durchlaufen den Eingabebaum von der Wurzel zu den Blättern. Sie implementieren meistens eine Art von „Suchverfahren".

Bottom-Up Baumautomaten. Durchlaufen den Eingabebaum von den Blättern zur Wurzel. Sie „berechnen" oft die Lösung.

MSO. Mit Hilfe von Formeln können wir die Eigenschaften der Bäume in einer Sprache „beschreiben".

Reguläre Baumsprachen. MSO, $\mathrm{NBA}_{\uparrow}$, $\mathrm{NBA}_{\downarrow}$ und $\mathrm{DBA}_{\uparrow}$ erkennen die Klasse der *regulären Baumsprachen*.

Deterministische Top-Down Automaten erkennen eine echte Teilklasse.

Bisher: Γ-Bäume mit festem Rangalphabet

Definition. Ein *Rangalphabet* ist eine nicht leere endliche Menge Γ von Symbolen. Jedem $a \in \Gamma$ ist eine endliche Menge $\mathrm{rg}(a) \subset \mathbb{N}$ von *Rängen* oder *Stelligkeiten* zugeordnet.

Definition. Sei Γ ein Rangalphabet.

Ein Γ-*Baum* ist ein Paar (T, β), wobei

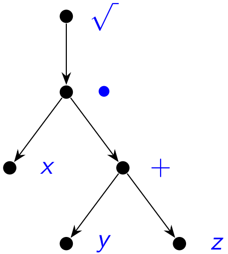
- $T = (V(T), E(T))$ ein Baum (mit Wurzel) und
- $\beta : V(T) \to \Gamma$ eine Beschriftungsfunktion ist,

so dass $l \in \mathrm{rg}(\beta(t))$ für alle $t \in V(T)$ mit l Nachfolgern gilt.

Wir schreiben $\mathcal{T}_\Gamma$ für die Klasse aller Γ-Bäume.

Eine *Baumsprache über* Γ ist eine Menge von Γ-Bäumen.

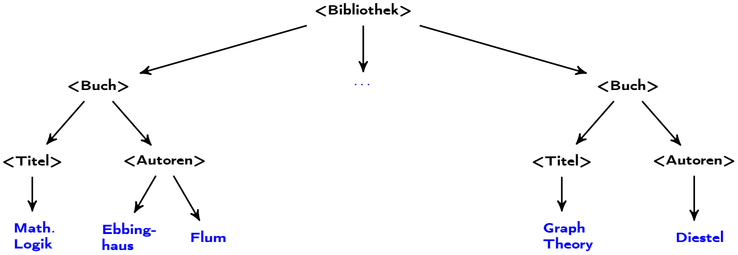
Beispiel.



Wiederholung: Bäume

Ziel. Äquivalenz zwischen MSO und Automaten von endlichen Wörtern auf endliche Bäume erweitern.

Beispiel (XML-Dokumente).

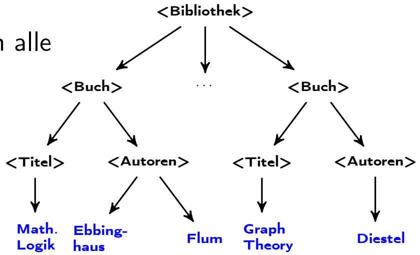


Bäume mit beliebigem Rang

Definition. Ein *(gerichteter) Baum* T ist ein gerichteter azyklischer Graph mit genau einem Element $w \in V(T)$, der *Wurzel* von T, ohne eingehende Kanten in dem es für alle $t \in V(T)$ genau einen Pfad von w zu t gibt.

D.h. (T, w) erhält man aus einem ungerichteten Baum indem alle Kanten von der Wurzel weg orientiert werden.

Automaten. Ohne festes Rangalphabet ist die Definition der Übergangsrelation eines Baumautomaten schwierig.



Sibling Unranked Trees

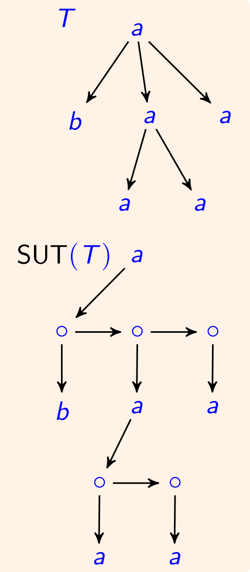
Definition. Sei Σ ein endliches Alphabet und $\circ \notin \Sigma$.

Ein *Sibling Unranked Tree* ist ein (gerichteter) Binärbaum über $\Sigma' := \Sigma \dot{\cup} \{\circ\}$ mit folgenden Eigenschaften:

- Jeder Knoten, der mit $a \in \Sigma$ beschriftet ist, ist entweder ein Blatt oder hat einen $\circ$-Knoten als einzigen Nachfolger.
- Jeder $\circ$-Knoten hat genau einen Σ-Nachfolger und höchstens einen $\circ$-Nachfolger.

Kodierung von Bäumen. Mit SUTs können wir beliebige gerichtete Bäume T durch Bäume T' mit festem Rang kodieren.

$(u, v) \in E(T) \;\hat{=}\;$ Pfad in T' dessen innere Knoten mit $\circ$ beschriftet sind.



Sibling Unranked Trees

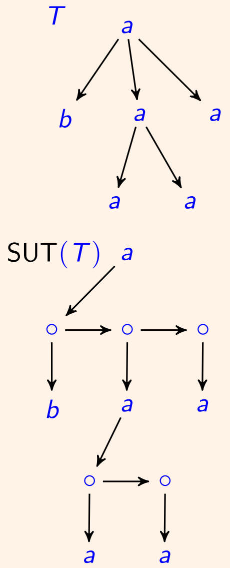
Automaten. Wir können nun einfach die gewohnten Baumautomatenmodelle auf die Kodierung beliebiger Bäume durch SUTs anwenden.

Achtung. Durch die Kodierung als SUTs ordnen wir die Nachfolger eines Knotens zwangsläufig.

Das kann zu Problemen führen, wenn z.B. Automaten Eigenschaften dieser Ordnung testen.

Beispiel. An allen Knoten gilt: die mit b beschrifteten Nachfolgern kommen vor den mit a beschrifteten.

Hier werden Informationen der „Kodierung“ verwendet, die im eigentlichen Baum gar nicht vorhanden sind.



Zusammenfassung

Bäume beliebiger Stelligkeit.

- Bäume mit unbeschränktem Verzweigungsgrad können durch gerichtete Binärbäume kodiert werden.
- Dies erlaubt es, die bekannten Automatenmodelle auf beliebige Wurzelbäume zu erweitern.
- Die Kodierung als SUTs ist nicht eindeutig, was zu Problemen mit der durch Automaten akzeptierten Baumsprache führen kann.

Komplexität logischer Auswertungsprobleme

Auswerten logischer Formeln (Entscheidungsproblem)

Definition.

Sei $\mathcal{C}$ eine Klasse von σ-Strukturen und sei $\mathcal{L}$ eine Logik.

Das *Auswertungsproblem* von $\mathcal{L}$ auf $\mathcal{C}$ ist das Problem

$\mathrm{MC}(\mathcal{L}, \mathcal{C})$.

Eingabe. $\mathcal{A} \in \mathcal{C}$ und ein Satz $\varphi \in \mathcal{L}$

Problem. Entscheide, ob $\mathcal{A} \models \varphi$.

Auswerten logischer Formeln (Berechnungsproblem)

Erweiterung auf Formeln mit freien Variablen.

Sei $\mathcal{C}$ eine Klasse von σ-Strukturen und sei $\mathcal{L}$ eine Logik.

Das *Auswertungsproblem* von $\mathcal{L}$ auf $\mathcal{C}$ ist das Problem

$\mathrm{MC}(\mathcal{L}, \mathcal{C})$.

Eingabe. $\mathcal{A} \in \mathcal{C}$ und $\varphi(x_1, \dots, x_k) \in \mathcal{L}$

Problem. Berechne $\varphi(\mathcal{A}) := \{(\beta(x_1), \dots, \beta(x_k)) \in A^k : (\mathcal{A}, \beta) \models \varphi\}$.

Anwendungsbeispiele

$\mathrm{MC}(\mathcal{L}, \mathcal{C})$

Eingabe. $\mathcal{A} \in \mathcal{C}$ und $\varphi \in \mathcal{L}$

Problem. Entscheide, ob $\mathcal{A} \models \varphi$.

Datenbanken.

Formeln $\varphi(x_1, \ldots, x_k) \in \mathrm{FO}$ entsprechen SQL-Anfragen.

Das Auswerten von Anfragen in Datenbanken ist eines der zentralen Aufgaben von Datenbanksystemen.

Anwendungsbeispiele.

1. Datenbanken
2. Verifikation
3. Algorithmik
4. ...

Anwendungsbeispiele

$\mathrm{MC}(\mathcal{L}, \mathcal{C})$

Eingabe. $\mathcal{A} \in \mathcal{C}$ und $\varphi \in \mathcal{L}$

Problem. Entscheide, ob $\mathcal{A} \models \varphi$.

Verifikation.

Automatische Verifikation, ob ein System eine Spezifikation erfüllt.

Hierzu wird das System durch ein Transitionssystem $\mathcal{T}$ modelliert und die Spezifikation wird durch eine Formel φ einer temporalen Logik formalisiert.

Das algorithmische Problem ist dann zu überprüfen, ob $\mathcal{T} \models \varphi$.

Anwendungsbeispiele.

1. Datenbanken
2. Verifikation
3. Algorithmik
4. ...

Anwendungsbeispiele

$\mathrm{MC}(\mathcal{L}, \mathcal{C})$

Eingabe. $\mathcal{A} \in \mathcal{C}$ und $\varphi \in \mathcal{L}$

Problem. Entscheide, ob $\mathcal{A} \models \varphi$.

Algorithmik. Viele NP-schwere Probleme z.B. über Graphen lassen sich sehr einfach in der Logik MSO formalisieren.

Mit schnellen Algorithmen zum Auswerten einer festen Formel $\varphi \in$ MSO in Graphen wäre MSO eine mächtige und elegante Programmiersprache für Graphprobleme.

Anwendungsbeispiele.

1. Datenbanken
2. Verifikation
3. Algorithmik
4. ...

Komplexität des Auswertungsproblems

Bereits gesehen. In MSO können NP-schwere Graphprobleme formalisiert werden (z.B. 3-col).

Also muss $\mathrm{MC}(\mathrm{MSO}, \mathit{Graph})$ für die Klasse *Graph* aller Graphen NP-schwer sein.

Ziel. Wir werden zeigen, dass $\mathrm{MC}(\mathrm{MSO}, \mathit{Graph})$ und $\mathrm{MC}(\mathrm{FO}, \mathit{Graph})$ Pspace-vollständig sind.

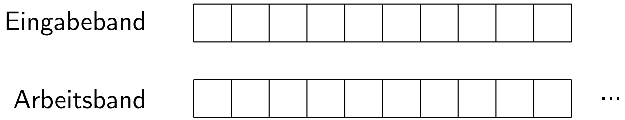
Die Klasse PSPACE

Die Klasse Pspace. Ein Problem ist in *Pspace*, wenn es durch eine deterministische, polynomiell platzbeschränkte Turing-Maschine M gelöst werden kann.

D.h. es gibt ein Polynom $p(n)$ so dass M auf Eingaben der Länge n höchstens $p(n)$ Speicherzellen benutzt.

Die Speicherzellen dürfen aber beliebig oft verwendet werden.

Turing-Maschienen mit separatem Arbeitsband.



Komplexitätsklassen.

$$\text{Logspace} \subseteq \text{P} \subseteq \text{NP} \subseteq \text{NPSpace} = \text{Pspace} \subseteq \text{Exptime}$$

Model-Checking Komplexität von MSO

Satz. $\mathrm{MC}(\mathrm{MSO}, \mathit{Fin})$ ist in Pspace für die Klasse Fin aller endlichen Strukturen lösbar.

Algorithmus $\mathrm{MC}(\mathcal{A}, \beta, \varphi(X_1, \ldots, X_m, y_1, \ldots, y_n))$.

$\mathcal{A} \in \mathit{Fin}$ $\quad\beta$ Belegung der freien Variablen von φ $\quad\varphi \in \mathrm{MSO}$

$$\varphi := \exists X_1 \quad \forall X_2 \quad \exists y_1 \quad \forall y_2 \quad \psi(X_1, X_2, x_1, x_2)$$

for each $U_1 \subseteq A$ **do**
 if $\mathbf{MC}(\mathcal{A}, \beta \cup \{X_1 \mapsto U_1\}, \varphi_2) ==$ *„accept“*
 then *accept*
od
reject

$\exists X / \forall X$:
Schleife über alle $U \subseteq A$
exponentielle Zeit
linearer Platz

$\exists y / \forall y$:
Schleife über alle $a \in A$
polynomielle Zeit
logarithmischer Platz

ψ quantorenfrei
einfaches Auswerten
polynomielle Zeit
logarithmischer Platz

Model-Checking Komplexität von MSO

Satz. $\mathrm{MC}(\mathrm{MSO}, \mathit{Fin})$ ist in Pspace für die Klasse Fin aller endlichen Strukturen lösbar.

Algorithmus $\mathrm{MC}(\mathcal{A}, \beta, \varphi(X_1, \ldots, X_m, y_1, \ldots, y_n))$.

$\mathcal{A} \in \mathit{Fin}$ β Belegung der freien Variablen von φ $\varphi \in \mathrm{MSO}$

$$\varphi := \exists X_1 \quad \forall X_2 \quad \exists y_1 \quad \forall y_2 \quad \psi(X_1, X_2, x_1, x_2)$$

for each $U_2 \subseteq A$ **do**
  if MC$(\mathcal{A}, \beta \cup \{X_2 \mapsto U_2\}, \varphi_3) ==$ *„reject"*
  then *reject*
od
accept

$\exists X / \forall X$:
Schleife über alle $U \subseteq A$
exponentielle Zeit
linearer Platz

$\exists y / \forall y$:
Schleife über alle $a \in A$
polynomielle Zeit
logarithmischer Platz

ψ quantorenfrei
einfaches Auswerten
polynomielle Zeit
logarithmischer Platz

Model-Checking Komplexität von MSO

Satz. $\mathrm{MC}(\textsc{mso}, \mathit{Fin})$ ist in Pspace für die Klasse Fin aller endlichen Strukturen lösbar.

Algorithmus $\mathrm{MC}(\mathcal{A}, \beta, \varphi(X_1, \dots, X_m, y_1, \dots, y_n))$.

$\mathcal{A} \in \mathit{Fin}$ $\quad \beta$ Belegung der freien Variablen von φ $\quad \varphi \in \mathrm{MSO}$

$$\varphi := \exists X_1 \quad \forall X_2 \quad \exists y_1 \quad \forall y_2 \quad \psi(X_1, X_2, x_1, x_2)$$

for each $a_1 \in A$ **do**
  if $\mathbf{MC}(\mathcal{A}, \beta \cup \{y_1 \mapsto a_1\}, \varphi_4) ==$ *„accept“*
  then *accept*
od
reject

$\exists X / \forall X$:
Schleife über alle $U \subseteq A$
exponentielle Zeit
linearer Platz

$\exists y / \forall y$:
Schleife über alle $a \in A$
polynomielle Zeit
logarithmischer Platz

ψ quantorenfrei
einfaches Auswerten
polynomielle Zeit
logarithmischer Platz

Model-Checking Komplexität von MSO

Satz. $\mathrm{MC}(\textsc{mso}, \mathit{Fin})$ ist in Pspace für die Klasse Fin aller endlichen Strukturen lösbar.

Algorithmus $\mathrm{MC}(\mathcal{A}, \beta, \varphi(X_1, \dots, X_m, y_1, \dots, y_n))$.

$\mathcal{A} \in \mathit{Fin}$ β Belegung der freien Variablen von φ $\varphi \in \mathrm{MSO}$

$$\varphi := \exists X_1 \quad \forall X_2 \quad \exists y_1 \quad \forall y_2 \quad \psi(X_1, X_2, x_1, x_2)$$

for each $a_2 \in A$ **do**
  if $\mathbf{MC}(\mathcal{A}, \beta \cup \{y_2 \mapsto a_2\}, \varphi_5) ==$ *„reject“*
  then *reject*
od
accept

$\exists X / \forall X$:
Schleife über alle $U \subseteq A$
exponentielle Zeit
linearer Platz

$\exists y / \forall y$:
Schleife über alle $a \in A$
polynomielle Zeit
logarithmischer Platz

ψ quantorenfrei
einfaches Auswerten
polynomielle Zeit
logarithmischer Platz

Model-Checking Komplexität von MSO

Satz. $\mathrm{MC}(\mathrm{MSO}, \mathit{Fin})$ ist in Pspace für die Klasse Fin aller endlichen Strukturen lösbar.

Algorithmus $\mathrm{MC}(\mathcal{A}, \beta, \varphi(X_1, \ldots, X_m, y_1, \ldots, y_n))$.

$\mathcal{A} \in \mathit{Fin}$ $\quad\beta$ Belegung der freien Variablen von φ $\quad\varphi \in \mathrm{MSO}$

$$\varphi := \exists X_1 \quad \forall X_2 \quad \exists y_1 \quad \forall y_2 \quad \psi(X_1, X_2, x_1, x_2)$$

einfaches Auswerten quantorenfreier Formeln in Linearzeit

$\exists X / \forall X$:
Schleife über alle $U \subseteq A$
exponentielle Zeit
linearer Platz

$\exists y / \forall y$:
Schleife über alle $a \in A$
polynomielle Zeit
logarithmischer Platz

ψ quantorenfrei
einfaches Auswerten
polynomielle Zeit
logarithmischer Platz

Model-Checking Komplexität von MSO

Satz. $\text{MC}(\text{MSO}, \textit{Fin})$ ist in Pspace für die Klasse *Fin* aller endlichen Strukturen lösbar.

Algorithmus $\text{MC}(\mathcal{A}, \beta, \varphi(X_1, \ldots, X_m, y_1, \ldots, y_n))$.

$\mathcal{A} \in \textit{Fin}$ $\quad \beta$ Belegung der freien Variablen von φ $\quad \varphi \in \text{MSO}$

$$\varphi := \exists X_1 \quad \forall X_2 \quad \exists y_1 \quad \forall y_2 \quad \psi(X_1, X_2, x_1, x_2)$$

Komplexität.

$$2^{|A|} \cdot \text{Zahl der Quantoren über Mengenvariablen} \cdot$$
$$|A|^{\text{Zahl der Quantoren über Elementvariablen}} \cdot p(|A|, |\varphi|)$$

Der Algorithmus benötigt polynomiellen Platz und exponentielle Zeit.

$\exists X / \forall X$:
Schleife über alle $U \subseteq A$
exponentielle Zeit
linearer Platz

$\exists y / \forall y$:
Schleife über alle $a \in A$
polynomielle Zeit
logarithmischer Platz

ψ quantorenfrei
einfaches Auswerten
polynomielle Zeit
logarithmischer Platz

Untere Schranke

Quantifizierte Boolesche Formeln.

Eine *quantifizierte Boolesche Formel* ist eine Formel der Form

$$Q_1 X_1 \dots Q_n X_n \psi,$$

wobei $Q_i \in \{\exists, \forall\}$ und ψ eine aussagenlogische Formel ist.

Die Semantik ist auf naheliegende Weise definiert.

($\exists X_i \psi$ wird wahr, wenn ψ für $X_i \mapsto 1$ oder $X_i \mapsto 0$ wahr wird.)

QBF. Problem, zu einer gegebenen quantifizierten Booleschen Formel φ zu entscheiden, ob φ wahr ist.

Satz. **QBF** ist Pspace-vollständig.

PSPACE-Vollständigkeit von $\mathbf{MC}(\mathbf{MSO}, \mathcal{C})$

Beispiel.
$\exists X_1 \forall X_2 (X_1 \vee \neg X_2)$

$\exists t \exists f \Big(t \neq f \wedge$
$\exists x_1 \forall x_2 \big((x_1 = t) \vee (x_2 = f)\big)\Big)$

Satz. Sei $\mathcal{A}$ eine Struktur mit mindestens zwei Elementen.

Dann ist $\mathrm{MC}(\mathrm{MSO}, \{\mathcal{A}\})$ PSPACE-vollständig.

Beweis. Reduktion von QBF auf $\mathrm{MC}(\mathrm{MSO}, \{\mathcal{A}\})$.

Zu jedem $\varphi \in$ QBF konstruieren wir (in Polynomialzeit) eine Formel $\varphi^* \in$ MSO, so dass: φ wird wahr gdw. $\mathcal{A} \models \varphi^*$.

(O.B.d.A. φ in NNF)

$$Q_1 X_1 \quad \cdots \quad Q_n X_n \quad \psi$$

$$\exists t \exists f \neg t = f \wedge \quad Q_1 x_1 \in \{t, f\} \cdots Q_n x_n \in \{t, f\} \quad \psi^*$$

ersetze X_i durch $x_i = t$ und $\neg X_i$ durch $x_i = f$

Folgerungen

Satz. Sei $\mathcal{A}$ eine Struktur mit mindestens zwei Elementen.

Dann ist $\text{MC}(\text{MSO}, \{\mathcal{A}\})$ PSPACE-vollständig.

Folgerung. Sei $\mathcal{C}$ eine Klasse von Strukturen die mindestens eine Struktur mit mehr als einem Element enthält.

Dann ist $\text{MC}(\text{MSO}, \mathcal{C})$ ebenso wie $\text{MC}(\text{FO}, \mathcal{C})$ PSPACE-vollständig.

Daten- und Formelkomplexität

Definition. Sei $\mathcal{C}$ eine Klasse von Strukturen und $\mathcal{L}$ eine Logik.

Instanzen von $\mathsf{MC}(\mathcal{L}, \mathcal{C})$ sind Paare $(\varphi \in \mathcal{L}, \mathcal{A} \in \mathcal{C})$.

- *Kombinierte Komplexität.*
  Berechne Zeit und Platz abhängig von $|A| + |\varphi|$.

  MSO: PSPACE-vollständig FO: PSPACE-vollständig.

- *Datenkomplexität.* Wir betrachten φ als konstant.
  Berechne Zeit und Platz abhängig von $|A|$.

  MSO: Polynomielle Hierarchie FO: Ptime.

- *Formelkomplexität.* Wir betrachten $\mathcal{A}$ als konstant.
  Berechne Zeit und Platz abhängig von $|\varphi|$.

  MSO: PSPACE-vollständig FO: PSPACE-vollständig.

Daten- und Formelkomplexität

Definition. Sei $\mathcal{C}$ eine Klasse von Strukturen und $\mathcal{L}$ eine l

Instanzen von $\text{MC}(\mathcal{L}, \mathcal{C})$ sind Paare $(\varphi \in \mathcal{L}, \mathcal{A} \in \mathcal{C})$.

- *Kombinierte Komplexität.*
  Berechne Zeit und Platz abhängig von $|A| + |\varphi|$.

  MSO: PSPACE-vollständig FO: PSPACE-vollständig.

- *Datenkomplexität.* Wir betrachten φ als konstant.
  Berechne Zeit und Platz abhängig von $|A|$.

  MSO: Polynomielle Hierarchie FO: Ptime.

- *Formelkomplexität.* Wir betrachten $\mathcal{A}$ als konstant.
  Berechne Zeit und Platz abhängig von $|\varphi|$.

  MSO: PSPACE-vollständig FO: PSPACE-vollständig.

Komplexität.

$$2^{|A|} \cdot \text{Anzahl Mengenquantoren} \cdot$$
$$|A|^{\text{Anzahl Elementquantoren}} \cdot p(|A|, |\varphi|)$$

Der Algorithmus benötigt polynomiellen Platz und exponentielle Zeit.

Parametrische Komplexität

Parametrische Komplexität

Auswertungsproblem $\mathsf{MC}(\mathcal{L}, \mathcal{C})$. *Eingabe:* $\varphi \in \mathcal{L}$ $\quad\quad \mathcal{A} \in \mathcal{C}$

Untersuche Komplexität in Abhängigkeit bestimmter Teile der Eingabe, dem *Parameter*.

Beispiele für Parameter bei Auswertungsproblemen.

- $|\varphi|$
- $|A|$
- Wenn $\mathcal{C}$ Klasse von Graphen: *Maximalgrad* $\Delta(\mathcal{A})$

Parametrische Komplexitätstheorie. Klassifikation der Komplexität von Problemen relativ zu dem gewählten Parameter.

Parametrische Probleme

Definition. Ein *parametrisches Problem* ist eine Sprache $\mathcal{L} \subseteq \Sigma^* \times \mathbb{N}$ für ein endliches Alphabet Σ.

Eingabe: (w, k) $\quad$ k: *Parameter*

Klassische Komplexität.
Ein Problem ist eine Sprache $\mathcal{L} \subseteq \Sigma^*$.

Beispiele.

- *Vertex Cover*.
  Eingabe: (G, k)
  Parameter. k
  Problem: existiert $X \subseteq V(G)$, $|X| \leq k$, so dass jede Kante $e \in E(G)$ einen Endpunkt in X hat.
- *Model-Checking* $\mathrm{MC}(\mathcal{L}, \mathcal{C})$.
  Eingabe: $\mathcal{A} \in \mathcal{C}$, $\varphi \in \mathcal{L}$
  Parameter. $k = |\varphi|$
  Problem: $\mathcal{A} \models \varphi$

Parametrische Probleme

Definition. Ein *parametrisches Problem* ist eine Sprache $\mathcal{L} \subseteq \Sigma^* \times \mathbb{N}$ für ein endliches Alphabet Σ.

Eingabe: (w, k) k: *Parameter*

Klassische Komplexität.
Ein Problem ist eine Sprache $\mathcal{L} \subseteq \Sigma^*$.

Beispiele.

- *Vertex Cover*.
  Eingabe: (G, k)
  Parameter. k
  Problem: existiert $X \subseteq V(G)$, $|X| \leq k$, so dass jede Kante $e \in E(G)$ einen Endpunkt in X hat.
- *Model-Checking* $\text{MC}(\mathcal{L}, \mathcal{C})$.
  Eingabe
  Parame
  Problem

Anmerkung.
Wir fixieren eine Kodierung von Graphen, Formeln usw. über einem Alphabet Σ.

Die Klasse FPT

Definition. Ein parametrisches Problem $\mathcal{P}$ ist *fixed-parameter tractable*, oder in der Klasse *FPT*, wenn es einen Algorithmus gibt, der auf Eingabe (w, k) in Zeit $f(k) \cdot |w|^c$ entscheidet, ob $(w, k) \in \mathcal{P}$. $c \in \mathbb{N}$ und $f : \mathbb{N} \to \mathbb{N}$ berechenbare Funktion.

Komplexität	
klassisch	parametrisiert
Ptime	FPT
NP	*W-Hierarchie* $W[1] \subseteq W[2] \subseteq \dots$

Achtung. Zugehörigkeit zu FPT hängt von der Wahl des Parameters ab.

$2^{|\varphi|} \cdot |A|^{99}$

$$2^{\left.2^{\cdot^{\cdot^{2^{|\varphi|}}}}\right\}|\varphi|} \cdot |A|$$

$|A|^{|\varphi|}$

$|A|^{\log|\varphi|}$

Beispiel

Satz. Sei $\mathtt{Tree}$ die Klasse aller endlichen Bäume.

$\mathrm{MC}(\textsc{mso}, \mathtt{Tree}) \in \mathrm{FPT}$.

Beweis. Eingabe: $T \in \mathtt{Tree}$ und $\varphi \in \textsc{mso}$.

1. Übersetze φ in einen $\mathrm{DBA}_{\uparrow}\mathcal{A}$, der dieselbe Baumsprache erkennt.
2. Lasse $\mathcal{A}$ auf T laufen und gib das Ergebnis aus.

Laufzeit

$f(|\varphi|)$

$O(|A|)$

Vergleiche. $\mathrm{MC}(\textsc{mso}, \mathtt{Tree})$ ist $\mathtt{Pspace}$-vollständig.

(es gibt Bäume mit 2 Knoten).

Weitere Beispiele

Probleme in FPT

- `Vertex Cover` parametrisiert durch die Größe der Lösung.
- `Disjoint Paths` auf ungerichteten Graphen parametrisiert durch Zahl der Terminalpaare.

W-Harte Probleme

- `Independet Set` und `Clique` parametrisiert durch Lösungsgröße sind $W[1]$-hart.
- `Dominating Set` parametrisiert durch Lösungsgröße ist $W[2]$-hart.

`Clique`
Eingabe. (G, k)
Parameter. k
Problem. $K_k \subseteq G$

`Dominating Set`
Eingabe. (G, k)
Parameter. k
Problem. ex $X \subseteq V(G)$, $|X| = k$, $X \cup N(X) = V(G)$.

`Disjoint Paths`
Eingabe. $(G, s_1, t_1, \dots, s_k, t_k)$
Parameter. k
Problem. ex. disjunkte Pfade $P_1, \dots, P_k$, so dass P_i s_i und t_i verbindet.

Parametrische Reduktion

Definition. Ein parametrisiertes Problem $\mathcal{P} \subseteq \sigma - \texttt{Struct} \times \mathbb{N}$ ist *parametrisiert $\mathcal{L}$-definierbar*, wenn für jedes $k \in \mathbb{N}$ ein $\varphi_k \in \mathcal{L}$ berechnet werden kann, so dass für alle Eingaben $(\mathcal{A}, k)$ gilt:

$$(\mathcal{A}, k) \in \mathcal{P} \quad \Leftrightarrow \quad (\mathcal{A}, \varphi_k).$$

Beispiel. `Dominating Set` ist parametrisiert FO-definierbar.

$$\varphi_k := \exists x_1 \dots \exists x_k \forall y \left(\bigvee_{i=1}^{k} (x_i = y \vee E(y, x_i)) \right)$$

Beobachtung. Sei $\mathcal{P}$ ein $W[i]$-hartes Problem, definiert auf einer Klasse $\mathcal{C}$ von Strukturen.

Wenn $\mathcal{P}$ parametrisiert $\mathcal{L}$ definierbar ist, dann ist $\mathrm{MC}(\mathcal{L}, \mathcal{C})$ ebenfalls $W[i]$-hart.

Folgerung. $\mathrm{MC}(\mathrm{FO}, \texttt{Graph})$ ist also $W[2]$-hart.

$\mathcal{L}$: eine Logik

σ: Signatur

σ-*Struct*: Klasse aller σ-Strukturen

Zusammenfassung

Parametrische Komplexität.

- Parametrische Komplexitätstheorie klassifiziert Probleme in Bezug auf bestimmte Parameter der Eingabe.
- Erlaubt eine feinere Klassifikation als die klassische Komplexitätstheorie.

Logik.

- Formelgröße als Parameter
- kombinierte Parameter: $|\varphi| + k$ $\quad$ k Strukturparameter

Komplexität logischer Auswertungsprobleme.

- $\mathrm{MC}(\mathrm{FO}, \mathtt{Graph})$ $W[2]$-hart und daher (verm.) nicht FPT
- Aber $\mathrm{MC}(\mathrm{MSO}, \mathtt{Tree}) \in \mathtt{FPT}$

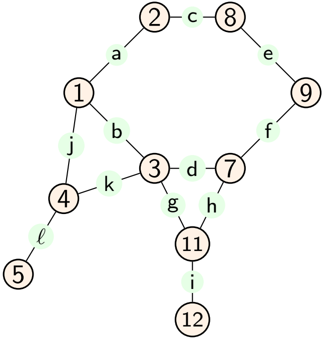
Inzidenz- und Standardmodellierung von Graphen

Modellierung durch Strukturen

Modellierung durch Strukturen. Um mit Formeln über mathematische Objekte wie Graphen etc. sprechen zu können, müssen wir sie geeignet durch logische Strukturen modellieren.

Dabei gibt es oft verschiedene Möglichkeiten der Modellierung.

Unter Umständen ändert die Art der Kodierung die Ausdrucksstärke der Logik auf den mathematischen Objekten.

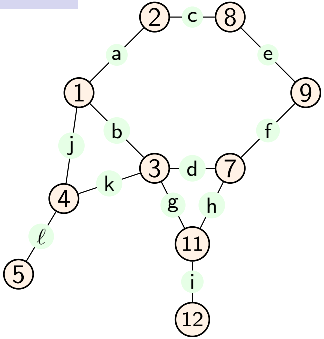


Wichtig. Die Modellierung sollte nach Möglichkeit keine „wesentlichen“ Informationen hinzufügen oder vergessen.

Modellierung von Bäumen durch *Sibling Unranked Trees*.

Hierbei wurden Informationen hinzugefügt, was nicht immer ohne Probleme ist.

Standardmodellierung und Inzidenzmodellierung



Standardmodellierung. Signatur $\sigma_{\texttt{Graph}} := \{E\}$ E: 2-stellig

Modelliere Graph G durch $\sigma_{\texttt{Graph}}$-Struktur $\mathcal{G} = (V(G), E^{\mathcal{G}})$ mit Universum $V(G)$ und $E^{\mathcal{G}} := \{(u, v) : \{u, v\} \in E(G)\}$.

Beispiel. $\mathcal{G} := (\{1, \dots, 12\}, \{(1,2), (2,1), (2,8), (8,2), \dots\})$.

Inzidenzmodellierung. Signatur $\sigma_{\texttt{inc}} := \{E, V, inc\}$
E, V: 1-stellig, inc: 2-stellig

Modelliere Graph G durch σ_{inc}-Struktur $\mathcal{G} = (U^{\mathcal{G}}, V^{\mathcal{G}}, E^{\mathcal{G}}, inc^{\mathcal{G}})$ mit

- Universum $U^{\mathcal{G}} = V(G) \cup E(G)$
- $E^{\mathcal{G}} := E(G)$ $V^{\mathcal{G}} := V(G)$ und
- $inc^{\mathcal{G}} := \{(v, e) : v \in V(G), e \in E(G), v \in e\}$.

Inzidenz. $\mathcal{G} := (U^{\mathcal{G}}, V^{\mathcal{G}}, E^{\mathcal{G}}, inc^{\mathcal{G}})$

$$\begin{aligned} U^{\mathcal{G}} &:= \{1, \dots, 12, a, b, \dots, \ell\}, \\ V^{\mathcal{G}} &:= \{1, \dots, 12\} \\ E^{\mathcal{G}} &:= \{a, \dots, \ell\} \\ inc^{\mathcal{G}} &:= \{(2, a), (2, c), (1, a), \dots\} \end{aligned}$$

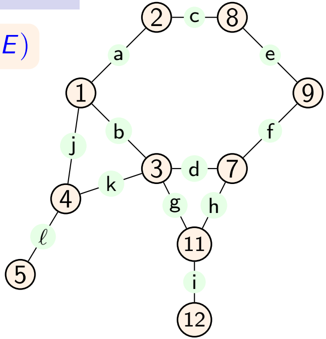
Beispiele

$M \subseteq E \mathrel{\hat{=}} \forall e(e \in M \rightarrow e \in E)$

Perfekte Matchings.

$$\varphi := \exists M \ \big(M \subseteq E \wedge \forall v \in V\, \exists^{=1} e(e \in M \wedge inc(v,e))\big)$$

$\exists^{=1} e\, \psi(e) \mathrel{\hat{=}} \exists e\, \psi(e) \wedge \neg \exists f(\psi(f) \wedge e \neq f)$
„es ex. genau 1 e mit ψ"



Inzidenz.
$\mathcal{G} := (U^{\mathcal{G}}, V^{\mathcal{G}}, E^{\mathcal{G}}, inc^{\mathcal{G}})$

$U^{\mathcal{G}} := \{1,\dots,12, a, b, \dots, \ell\}$,
$V^{\mathcal{G}} := \{1,\dots,12\}$
$E^{\mathcal{G}} := \{a,\dots,\ell\}$
$inc^{\mathcal{G}} := \{(2,a),(2,c),(1,a),\dots\}$

Beispiele

Perfekte Matchings.

$$\varphi := \exists M\ \big(M \subseteq E \land \forall v \in V\, \exists^{=1} e(e \in M \land inc(v,e))\big)$$

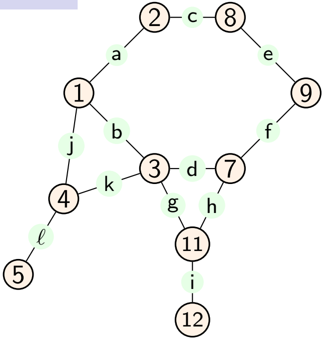
Vereinfachungen. Seien X, P Mengenvariablen.

- $P \subseteq E \ := \ \forall e(P(e) \rightarrow E(e))$
- $X \subseteq V \ := \ \forall x(X(x) \rightarrow V(x))$
- $v \in V(P)$: Kurzschreibweise für $\exists e \in P\ inc(v,e)$.
- Notation: $x \in X$, $\quad X \cap V(P) \neq \emptyset$, $\quad \dots$.
- Für $k \geq 0$ definieren wir $\exists^{=k} x\, \psi(x)$ als

$$\exists x_1 \dots \exists x_k \big(\bigwedge_{1 \leq i < j \leq k} \neg x_i = x_j \land \bigwedge_{i=1}^{k} \psi(x_i) \ \land$$
$$\forall y(\psi(y) \rightarrow \bigvee_{i=1}^{k} y = x_i)\big).$$

„*Es gibt genau k Elemente, die ψ erfüllen.*“

- Entsprechend $\exists^{\leq k} x \psi(x), \dots$.



Inzidenz.
$\mathcal{G} := (U^{\mathcal{G}}, V^{\mathcal{G}}, E^{\mathcal{G}}, inc^{\mathcal{G}})$

$U^{\mathcal{G}} := \{1, \dots, 12, a, b, \dots, \ell\}$,
$V^{\mathcal{G}} := \{1, \dots, 12\}$
$E^{\mathcal{G}} := \{a, \dots, \ell\}$
$inc^{\mathcal{G}} := \{(2,a), (2,c), (1,a), \dots\}$

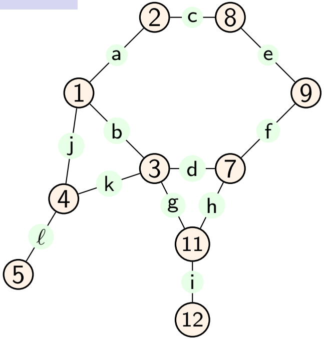
Beispiele

P induziert zusammenhängenden Untergraph.

$$\varphi_{con}(P) := P \subseteq E \wedge \forall P' \subseteq P\Big(P' \neq \emptyset \wedge \forall e, e' \in P$$
$$\big(\big(e \in P' \wedge \exists v(inc(v,e) \wedge inc(v,e'))\big) \to e' \in P'\big)\Big)$$
$$\to P = P'$$

P Baum.

$$\varphi_{tree}(P) := \varphi_{con}(P) \wedge \forall e \in P \neg\varphi_{con}(P \setminus \{e\}).$$



Inzidenz.

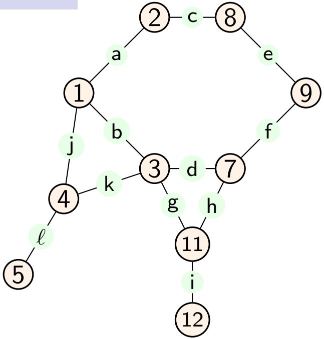
$\mathcal{G} := (U^{\mathcal{G}}, V^{\mathcal{G}}, E^{\mathcal{G}}, inc^{\mathcal{G}})$

$U^{\mathcal{G}} := \{1, \dots, 12, a, b, \dots, \ell\},$
$V^{\mathcal{G}} := \{1, \dots, 12\}$
$E^{\mathcal{G}} := \{a, \dots, \ell\}$
$inc^{\mathcal{G}} := \{(2,a), (2,c), (1,a), \dots\}$

Beispiele

Pfad P.

$$\varphi_{path}(P) := \varphi_{tree}(P) \wedge \forall x \in V(P) \exists^{\leq 2} e \in E(P)\, inc(x, e).$$

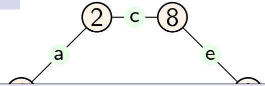


Inzidenz.

$\mathcal{G} := (U^{\mathcal{G}}, V^{\mathcal{G}}, E^{\mathcal{G}}, inc^{\mathcal{G}})$

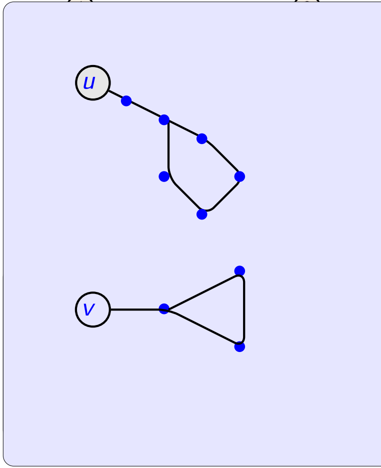
$$\begin{aligned} U^{\mathcal{G}} &:= \{1, \dots, 12, a, b, \dots, \ell\}, \\ V^{\mathcal{G}} &:= \{1, \dots, 12\} \\ E^{\mathcal{G}} &:= \{a, \dots, \ell\} \\ inc^{\mathcal{G}} &:= \{(2, a), (2, c), (1, a), \dots\} \end{aligned}$$

Beispiele



Pfad P.

$$\varphi_{path}(P) := \varphi_{tree}(P) \wedge \forall x \in V(P) \exists^{\leq 2} e \in E(P) inc(x, e).$$



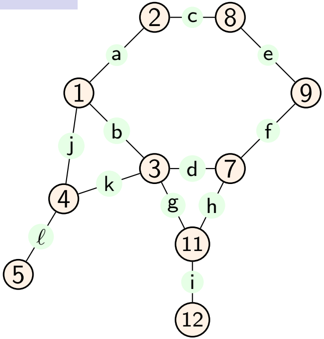
Beispiele

Pfad P.

$$\varphi_{path}(P) := \varphi_{tree}(P) \wedge \forall x \in V(P) \exists^{\leq 2} e \in E(P) inc(x, e).$$

Pfad P zwischen u und v.

$$\varphi_{uv\text{-}path}(P, u, v) := \varphi_{path}(P) \wedge \exists^{=1} e \in P\ inc(u, e) \wedge \exists^{=1} e' \in P inc(v, e').$$



Inzidenz.

$\mathcal{G} := (U^{\mathcal{G}}, V^{\mathcal{G}}, E^{\mathcal{G}}, inc^{\mathcal{G}})$

$U^{\mathcal{G}} := \{1, \dots, 12, a, b, \dots, \ell\},$
$V^{\mathcal{G}} := \{1, \dots, 12\}$
$E^{\mathcal{G}} := \{a, \dots, \ell\}$
$inc^{\mathcal{G}} := \{(2, a), (2, c), (1, a), \dots\}$

MSO *vs* MSO_2

MSO_2 und MSO_1. Wir schreiben MSO_2 wenn wir MSO auf Inzidenzstrukturen meinen und MSO_1, um festzulegen, dass die Standardmodellierung gemeint ist.

Ausdrucksstärke. MSO_2 ist ausdrucksstärker als MSO_1.

Z.B. kann man in MSO_2 Cliquen gerader Ordnung definieren, nicht aber in MSO_1. (Benutze die perfekte matching Formel.)

Ebenso kann in MSO_2 die Existenz eines Hamiltonpfades definiert werden.

Zusammenfassung: Nützliche MSO$_2$-Formeln

Formeln über Graphen.

- $\varphi_{pfad(P)}$: P ist die Kantenmenge eines Pfades.
- $\varphi_{Kreis}(C)$: C ist die Kantenmenge eines Kreises.
- $\varphi_{Baum}(T)$: T ist die Kantenmenge eines Baums.

Vereinfachungen. Sei P eine Mengenvariable.

- $v \in V(P)$: Kurzschreibweise für $\exists e \in P\ inc(v, e)$.
- Verwenden auch Schreibweisen wie $X \cap V(P) \neq \emptyset$.

Beispiel. $\varphi(P, P') := \varphi_{pfad}(P) \wedge \varphi_{pfad}(P') \wedge V(P) \cap V(P') = \emptyset$.

Zusammenfassung

Inzidenz- und Standardmodellierung. Graphen können auf verschiedene Arten als Strukturen modelliert werden.

Standardmodellierung. Universum ist die Knotenmenge.

Inzidenzmodellierung. Knoten und Kanten sind gleichberechtigte Elemente des Universums.

Ausdrucksstärke.

- FO ist gleich ausdrucksstark auf beiden Modellierungen.
- MSO ist sehr viel ausdruckstärker auf Inzidenzstrukturen.

MSO-Model-Checking auf Bäumen

Model-Checking auf Bäumen

Satz. Sei $\Gamma\text{-}\textsc{Tree}^{r}$ die Klasse aller endlichen Γ-Bäume über einem festem Rangalphabet Γ.

Dann ist $\mathsf{MC}(\textsc{mso}, \Gamma\text{-}\textsc{Tree}^{r}) \in \mathsf{FPT}$.

Beweis. Eingabe: $T \in \Gamma\text{-}\textsc{Tree}^{r}$ und $\varphi \in \textsc{mso}$.

1. Übersetze φ in einen $\mathsf{DBA}_{\uparrow}\mathcal{A}$, der dieselbe Baumsprache erkennt.
2. Lasse $\mathcal{A}$ auf T laufen und gebe das Ergebnis aus.

Laufzeit

$f(|\varphi|)$

$O(|T|)$

Vergleiche. $\mathsf{MC}(\textsc{mso}, \Gamma\text{-}\textsc{Tree}^{r})$ ist PSPACE-vollständig.

(es gibt Bäume mit 2 Knoten).

MSO-Model-Checking auf unbeschränkten Bäumen

Sei $\Sigma\text{-}\textsc{Tree}^{\mathrm{u}}$ die Klasse aller endlichen Bäume beliebiger Stelligkeit über einem festen Alphabet Σ.

Satz. $\mathrm{MC}(\textsc{mso}, \Sigma\text{-}\textsc{Tree}^{\mathrm{u}}) \in \mathrm{FPT}$.

Beweisidee. Reduziere $\mathrm{MC}(\textsc{mso}, \Sigma\text{-}\textsc{Tree}^{\mathrm{u}})$ auf $\mathrm{MC}(\textsc{mso}, \Gamma\text{-}\textsc{Tree}^{\mathrm{r}})$, wobei $\Gamma = \Sigma \dot\cup \{\circ\}$, indem Bäume durch SUTs kodiert werden.

$$\begin{array}{lccc}
 & \text{unranked} & & \text{ranked} \\
\text{tree} & T & \rightsquigarrow & T' \\
 & \rotatebox[origin=c]{90}{$\models$} & & \rotatebox[origin=c]{90}{$\models$} \\
\text{MSO} & \varphi & \rightsquigarrow & \varphi'
\end{array}$$

MSO-Model-Checking auf unbeschränkten Bäumen

Beweisidee. Betrachte $\mathtt{SUT} : \Sigma\text{-}\textsc{Tree}^{\mathrm{u}} \to \Gamma\text{-}\textsc{Tree}^{\mathrm{r}}$ als Abbildung.

Suche Abbildung

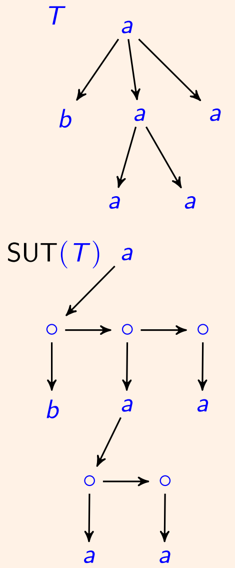
$$\Theta : \mathsf{MSO}[\{E, P_a : a \in \Sigma\}] \to \mathsf{MSO}[\{S_1, S_2, P_\circ, P_a : a \in \Sigma\}],$$

so dass für alle $T \in \Sigma\text{-}\textsc{Tree}^{\mathrm{u}}$ und $\varphi \in \mathsf{MSO}$ gilt:

$$T \models \varphi \quad \text{gdw.} \quad \mathtt{SUT}(T) \models \Theta(\varphi). \qquad (\star)$$

Übersetzung. Kanten $(u, v) \in E(T)$ entsprechen $\circ$-Pfaden von u nach v in $\mathtt{SUT}(T)$.

$$\varphi_E(u, v) := \exists P\Big(\varphi_p(P, u, v) \wedge \neg P_\circ(u) \wedge \neg P_\circ(v) \wedge \\ \forall x \in V(P)\big(x \notin \{u, v\} \to P_\circ(x)\big)\Big).$$



MSO-Model-Checking auf unbeschränkten Bäumen

Beweisidee. Betrachte $\mathtt{SUT} : \Sigma\text{-}\mathrm{TREE}^{u} \to \Gamma\text{-}\mathrm{TREE}^{r}$ als Abbildung.

$$T \models \varphi \quad \text{gdw.} \quad \mathtt{SUT}(T) \models \Theta(\varphi). \qquad (\star)$$

Übersetzung. Kanten $(u, v) \in E(T)$ entsprechen $\circ$-Pfaden von u nach v in $\mathtt{SUT}(T)$.

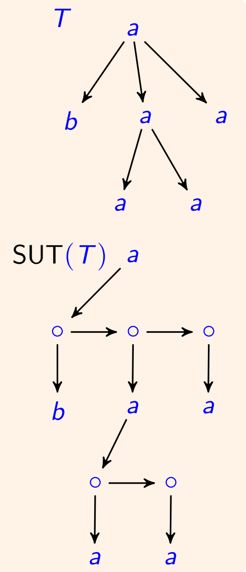
$$\varphi_E(u, v) := \exists P\Big(\varphi_p(P, u, v) \wedge \neg P_\circ(u) \wedge \neg P_\circ(v) \wedge \forall x \in V(P)\big(x \notin \{u, v\} \to P_\circ(x)\big)\Big).$$

Ersetze nun in φ jede Unterformel der Form $E(x, y)$ durch $\varphi_E(x, y)$. Dann gilt $(\star)$. $\square$

Anmerkung. Die gleiche Aussage gilt für auch MSO_2.

Beispiel. $\varphi := \exists x \exists y\big(P_a(x) \wedge P_b(y) \wedge E(x, y)\big)$

$$\rightsquigarrow \quad \Theta(\varphi) := \exists x \exists y\big(P_a(x) \wedge P_b(y) \wedge \varphi_E(x, y)\big)$$



Effizientes Model-Checking

Wir wollen $\mathsf{MC}(\mathrm{MSO}, \Sigma\text{-}\mathrm{TREE}^u) \in \mathsf{FPT}$ auf größere Klassen von Strukturen und stärkere Logiken erweitern.

Ziel. Finde für Logiken $\mathcal{L}$ wie $\mathrm{MSO}_1, \mathrm{MSO}_2, \mathrm{FO}, \ldots$ möglichst genaue *„strukturelle“* Charakterisierung der Klassen $\mathcal{C}$ für die $\mathsf{MC}(\mathcal{L}, \mathcal{C}) \in \mathsf{FPT}$.

Beobachtung. Je ausdruckstärker $\mathcal{L}$ ist, desto kleiner werden die Klassen $\mathcal{C}$ auf denen $\mathsf{MC}(\mathcal{L}, \mathcal{C}) \in \mathsf{FPT}$.

Model-Checking jenseits von Bäumen

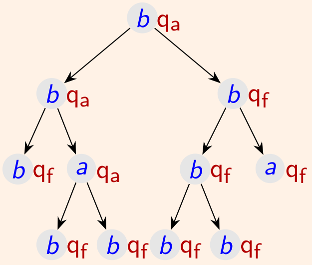
Frage.
Warum ist MSO-Model-Checking auf Bäumen handhabbar?

Algorithmen auf Bäumen. Kombiniere rekursiv Teillösungen für Unterbäume an den Nachfolgern der Wurzel zur Gesamtlösung des Problems.

Da die Unterbäume disjunkt sind, beeinflussen sich die Teilergebnisse nicht.

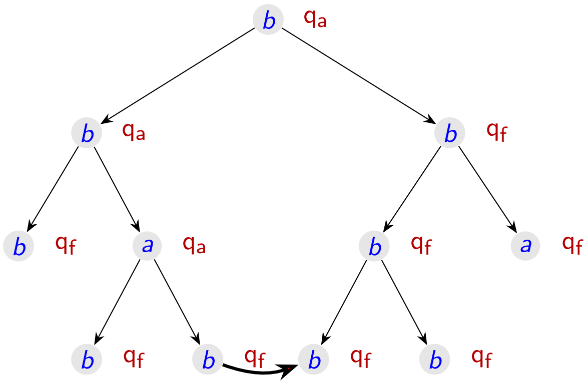
Idee. Kann man diese Eigenschaft der *rekursiven Zerlegbarkeit* von Bäumen auf allgemeinere Graphen erweitern?

Baum (T, β).



Erweiterung auf größere Graphklassen

Idee. Kann man diese Eigenschaft von Bäumen auf allgemeinere Graphen erweitern?



Separationen.
Getrennte Berechnung in den Teilbäumen möglich.

Die „*Wurzel*“ ist ein Knotenpaar.

Zusammenfassung

Model-Checking.

- MSO-Model-Checking erweitert von Bäumen mit festem Rang auf unbeschränkte Bäume.
- Technik funktioniert für MSO_1 und MSO_2.
- Später werden wir den Begriff der *Interpretation* kennen lernen, der diese Technik stark verallgemeinert.

Erweiterung auf größere Graphklassen.

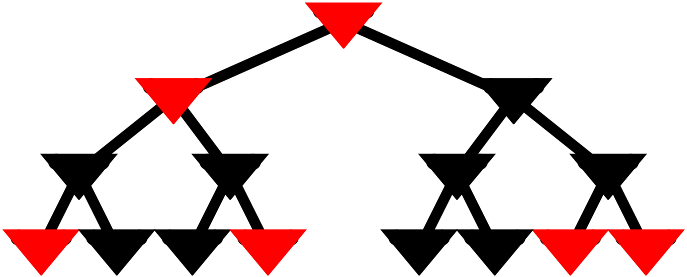
- Idee des Model-Checkings auf Bäumen kann auf größere Graphklassen erweitert werden, wenn wir „Trenner“ mit mehreren Knoten zulassen.
- Dies führt zum Begriff der *Baumweite*.

Baumweite

Von Bäumen zu allgemeineren Graphen

Idee. Bäume lassen sich rekursiv in disjunkte Teilbäume zerlegen.

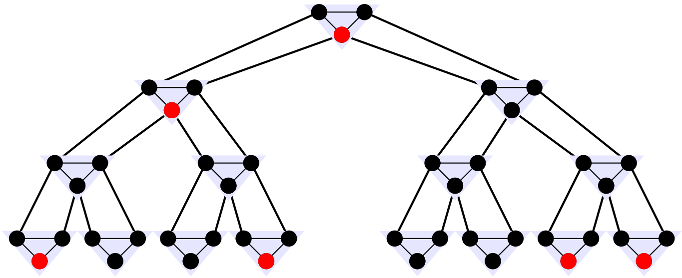
Kann man diese Eigenschaft auf allgemeinere Graphen erweitern?



Von Bäumen zu allgemeineren Graphen

Idee. Bäume lassen sich rekursiv in disjunkte Teilbäume zerlegen.

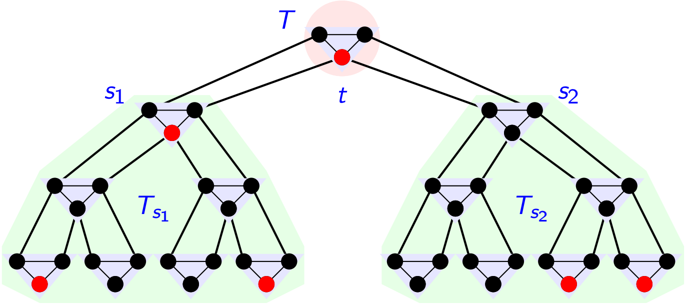
Kann man diese Eigenschaft auf allgemeinere Graphen erweitern?



Von Bäumen zu allgemeineren Graphen

Idee. Bäume lassen sich rekursiv in disjunkte Teilbäume zerlegen.

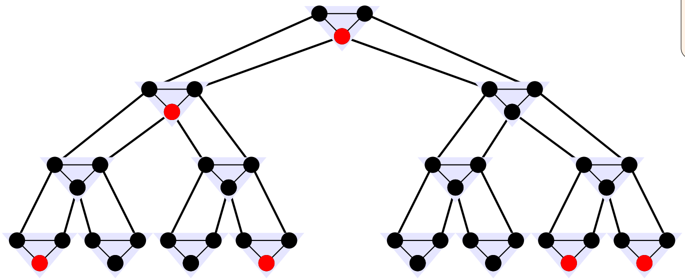
Kann man diese Eigenschaft auf allgemeinere Graphen erweitern?



Von Bäumen zu allgemeineren Graphen

Idee. Bäume lassen sich rekursiv in disjunkte Teilbäume zerlegen.

Kann man diese Eigenschaft auf allgemeinere Graphen erweitern?

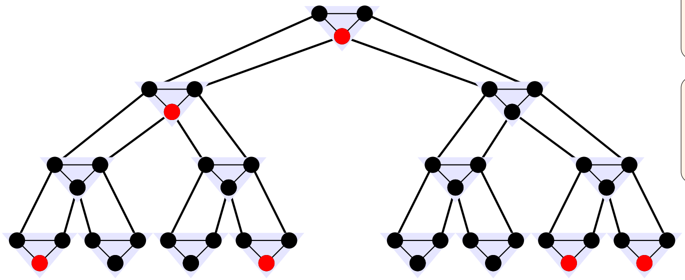


Anforderungen.
Wenn die Knoten in t gelöscht werden, gibt es *keine Verbindung* mehr zwischen T_{s_1} und T_{s_2}.

Von Bäumen zu allgemeineren Graphen

Idee. Bäume lassen sich rekursiv in disjunkte Teilbäume zerlegen.

Kann man diese Eigenschaft auf allgemeinere Graphen erweitern?



Anforderungen.
Wenn die Knoten in t gelöscht werden, gibt es *keinen Pfad* mehr zwischen *Knoten aus* T_{s_1} und T_{s_2}.

Äquivalent.
Es gibt keine Kante $\{u, v\}$ mit $u \in V(T_{s_1})$ und $v \in V(T_{s_2})$.

Baumzerlegungen

Definition. Sei G ein Graph.

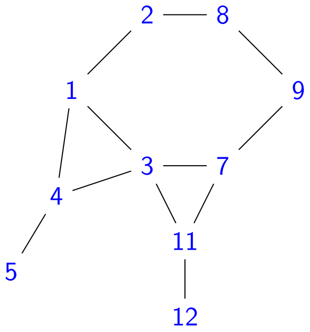
Eine *Baumzerlegung* von G ist ein Paar $\mathcal{T} = (T, \beta)$, wobei T ein Baum ist und $\beta : V(T) \to 2^{V(G)}$, so dass

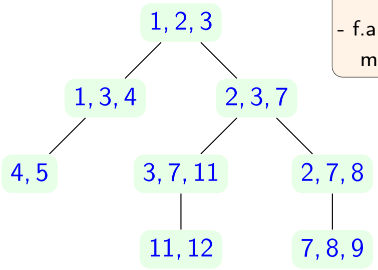
- für alle $v \in V(G)$ ist
$$\beta^{-1}(v) := \{t \in V(T) : v \in \beta(t)\} \neq \emptyset$$
und induziert einen Unterbaum von T.
- für jede Kante $e \in E(G)$ existiert ein $t \in V(T)$ mit $e \subseteq \beta(t)$.

Wir nennen die $\beta(t)$ *Taschen*.

Baumzerlegungen

Ein Graph G und eine Baumzerlegung von G.





Baumzerlegung (T, β).

T: Baum $\quad \beta : V(T) \to 2^{V(G)}$

- f.a. $v \in V(G)$ ist
  $\{t \in V(T) : v \in \beta(t)\}$
  nicht leer und zshgd. in T.

- f.a. $e \in E(G)$ ex. $t \in V(T)$
  mit $e \subseteq \beta(t)$.

Baumweite

Definition. Sei $\mathcal{T} = (T, \beta)$ eine Baumzerlegung von G.

Die *Weite* von $\mathcal{T}$ ist

$$w(\mathcal{T}) = \max\{|\beta(t)| : t \in V(T)\} - 1.$$

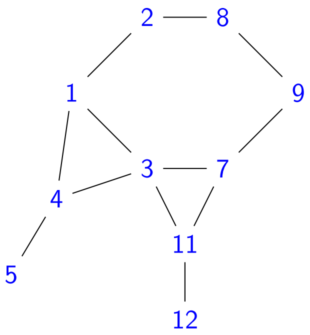
Die *Baumweite* von G ist

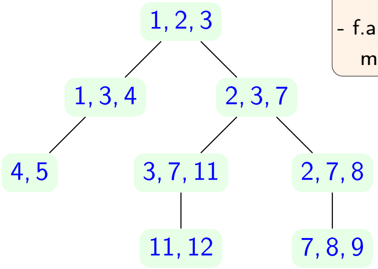
$$\mathrm{bw}(G) = \min\{w(\mathcal{T}) : \mathcal{T} \text{ Baumzerlegung von } G\}.$$

Eine Klasse $\mathcal{C}$ von Graphen hat *beschränkte Baumweite*, wenn es ein $k \in \mathbb{N}$ gibt mit $\mathrm{bw}(G) \leq k$ für alle $G \in \mathcal{C}$.

Baumzerlegungen

Ein Graph G und eine Baumzerlegung von G.





Baumzerlegung (T, β).

T: Baum $\quad \beta : V(T) \to 2^{V(G)}$

- f.a. $v \in V(G)$ ist
  $\{t \in V(T) : v \in \beta(t)\}$
  nicht leer und zshgd. in T.
- f.a. $e \in E(G)$ ex. $t \in V(T)$
  mit $e \subseteq \beta(t)$.

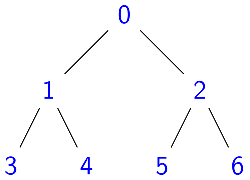
Beispiele

Baumzerlegung (T, β).

T: Baum $\quad \beta : V(T) \rightarrow 2^{V(G)}$

- f.a. $v \in V(G)$ ist
  $\{t \in V(T) : v \in \beta(t)\}$
  nicht leer und zshgd. in T.
- f.a. $e \in E(G)$ ex. $t \in V(T)$
  mit $e \subseteq \beta(t)$.

Bäume. Die Baumweite eines Baums ist 1.



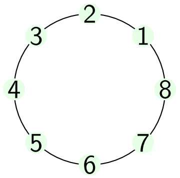
Beispiele

Baumzerlegung (T, β).

T: Baum $\quad \beta : V(T) \to 2^{V(G)}$

- f.a. $v \in V(G)$ ist $\{t \in V(T) : v \in \beta(t)\}$ nicht leer und zshgd. in T.
- f.a. $e \in E(G)$ ex. $t \in V(T)$ mit $e \subseteq \beta(t)$.

Kreise. Die Baumweite eines Kreises ist 2.



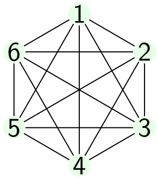
Beispiele

Baumzerlegung (T, β).

T: Baum $\beta : V(T) \rightarrow 2^{V(G)}$

- f.a. $v \in V(G)$ ist
  $\{t \in V(T) : v \in \beta(t)\}$
  nicht leer und zshgd. in T.
- f.a. $e \in E(G)$ ex. $t \in V(T)$
  mit $e \subseteq \beta(t)$.

Cliquen. Die Baumweite von K_n ist $n-1$.



Anwendungsbeispiele

Graphen relativ kleiner Baumweite kommen in ganz verschiedenen Anwendungsgebieten vor.

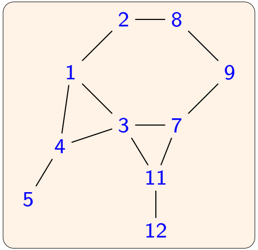
Beispiele.

- Seriell-parallele Graphen haben Baumweite höchstens 2.
- Kontrollflussgraphen (engl.: control flow graphs) von C- oder Java-Programmen haben Baumweite höchstens 8 (und höchstens 6, wenn sie kein goto benutzen).
- Backbone-Netze großer Internetprovider haben eine Baumweite von etwa 40.

Separationen

Definition. Sei G ein Graph und seien $X, Y, S \subseteq V(G)$.

1. S ist ein *X-Y-Trenner*, wenn es keinen Pfad in $G - S$ von X nach Y gibt.
2. Eine *Separation* in G ist ein Paar (A, B) mit $A, B \subseteq G$ und $A \cup B = G$.
3. Die *Ordnung von* (A, B) ist $|V(A) \cap V(B)|$.



Separationen und Baumweite

Lemma. Sei

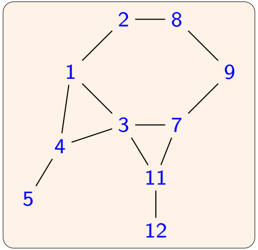
- $\mathcal{T} = (T, \beta)$ eine Baumzerlegung von G,
- $e = \{s, t\} \in E(T)$ und T_s, T_t die Komponenten von $T - e$ so dass $s \in V(T_s)$ und $t \in V(T_t)$.

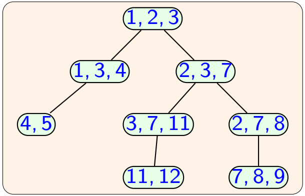
Sei

- Sei $S = \beta(s) \cap \beta(t)$ und
- $\beta(T_s) := \bigcup\{v \in V(G) :$ es ex. $t \in V(T_s)$ mit $v \in \beta(t)\}$
  $\beta(T_t) := \bigcup\{v \in V(G) :$ es ex. $t \in V(T_s)$ mit $v \in \beta(t)\}$.

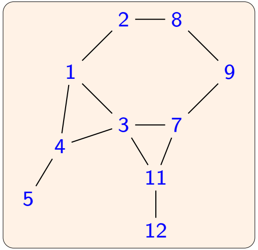
$(G[\beta(T_s)], G[\beta(T_t)])$ ist eine Separation mit Separator S und

S ist ein $\beta(T_s)$-$\beta(T_t)$-Trenner.



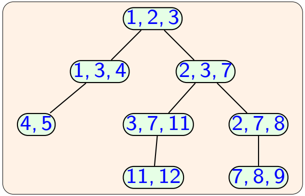


Zusammenfassung



Baumweite.

- Baumweite ist ein Graphparameter der die *Konnektivität* oder *Ähnlichkeit zu einem Baum* misst.
- Ein Graph hat kleine Baumweite wenn er baumartig durch Löschen weniger Knoten in Untergraphen konstanter Größe zerlegt werden kann.
- Graphen und Strukturen kleiner Baumweite kommen in einer Reihe sehr unterschiedlicher Anwendungen vor.
- Viele NP-schwere Probleme können effizient auf Graphen mit kleiner Baumweite gelöst werden.



Algorithmen auf Graphen kleiner Baumweite

Der Satz von Courcelle

Ziel. $\text{MC}(\text{MSO}_2, \mathcal{C}) \in \text{FPT}$ für jede Klasse $\mathcal{C}$ von Graphen mit beschränkter Baumweite. (Courcelle 1990)

Vorbereitung. Als Vorbereitung zu dem Beweis des Satzes zeigen wir zunächst folgende algorithmische Anwendung.

Satz. Das Problem

3-Col

Eingabe.	Graph $G \in \mathcal{C}$
Problem.	Ist G 3-färbbar?

kann in Zeit $2^{p(bw(G))} \cdot |G|$ gelöst werden. (p: Polynom)

3-*Färbbarkeit*

3-Färbbarkeit. Wir wollen 3-Col in Zeit $2^{p(\mathrm{bw}(G))} \cdot |G|$ lösen.

Eingabe. Graph $G \in \mathcal{C}$

Problem. Ist G 3-färbbar?

Idee. Benutze eine Baumzerlegung von G für einen Divide-and-Conquer Algorithmus.

Berechnen von Baumzerlegungen

Berechnen von Baumzerlegungen.

Das Problem

BAUMWEITE

Eingabe. Graph G und $k \in \mathbb{N}$

Problem. Gilt $\mathrm{bw}(G) \leq k$?

ist NP-vollständig.

Satz (Bodlaender).

Es gibt einen Algorithmus der, gegeben ein Graph G und $k \in \mathbb{N}$, in Zeit

$$2^{p(bw(G))} \cdot |G|$$

eine Baumzerlegung von G der Weite $\leq k$ ausrechnet oder korrekt entscheidet, dass $\mathrm{bw}(G) > k$. (p: Polynom)

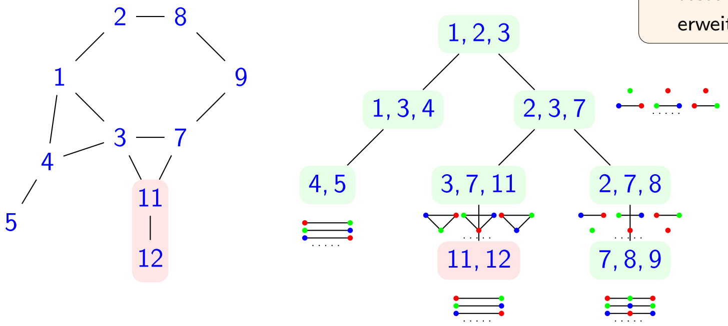
3-Färbbarkeit

3-Färbbarkeit. Wir wollen 3-Col in Zeit $2^{p(\mathrm{bw}(G))} \cdot |G|$ lösen.

Eingabe. Graph $G \in \mathcal{C}$

Problem. Ist G 3-färbbar?

Schritt 1. Berechne Baumzerlegung von G der Weite $k = bw(G)$.



Schritt 2 (Dynamische Programmierung).

Von Blättern zur Wurzel: Für alle $t \in V(T)$:

1. Berechne alle 3-Färb. von $G[\beta(t)]$.
2. Für jede 3-Färb. c von $G[\beta(t)]$, teste ob c zu 3-Färbung von $G[\beta(T_t)]$ erweitert werden kann.

3-Färbbarkeit

Satz. Das Problem

3-COL

Eingabe. Graph $G \in \mathcal{C}$

Problem. Ist G 3-färbbar?

kann in Zeit $2^{p(\mathrm{bw}(G))} \cdot |G|$ gelöst werden.

Algorithmus.

1. Berechne Baumzerlegung von G minimaler Weite.
2. Löse das Problem bottom-up auf den einzelnen Teilbäumen.

Zusammenfassung

Baumweite.

- Das Berechnen der Baumweite eines Graphen ist NP-vollständig.
- Aber es gibt fixed-parameter lineare Algorithmen um eine optimale Baumzerlegung auszurechnen.

Algorithmen.

- Sehr viele NP-schwere Graphenprobleme sind fixed-parameter tractable wenn die Baumweite als Parameter genommen wird.
- D.h. auf Klassen mit beschränkter Baumweite sind diese Probleme effizient lösbar.

Der Satz von Courcelle

Der Satz von Courcelle

Satz. $\mathrm{MC}(\mathrm{MSO}_2, \mathcal{C}) \in \mathrm{FPT}$ für jede Klasse $\mathcal{C}$ von Graphen mit beschränkter Baumweite. (Courcelle 1990)

Genauer: $\mathrm{MC}(\mathrm{MSO}_2, \mathtt{GRAPH}) \in \mathrm{FPT}$ param. durch $bw(G) + |\varphi|$.

Beweisidee. Gegeben $G \in \mathcal{C}$ und $\varphi \in \mathrm{MSO}$.

1. Berechne Baumzerlegung (T, β) von G.
2. Übersetze φ in Formel φ', so dass

$$\begin{array}{ccc} G & \rightsquigarrow & (T, \beta) \\ \rotatebox{-90}{$\models$} & & \rotatebox{-90}{$\models$} \\ \varphi & \rightsquigarrow & \varphi' \end{array}$$

(T, β): Beschrifteter Baum.

Problem
(T, β) ist keine Beschriftung über einem festen Alphabet.

Eine Kodierung von Baumzerlegungen

Kodieren durch Σ-Bäume. Wir wollen Baumzerlegungen der Weite k durch Σ_k-Bäume für ein festes Alphabet Σ_k kodieren.

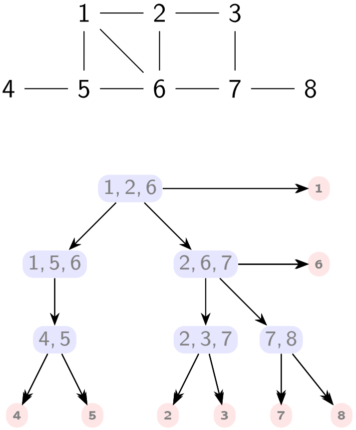
Schritt 1. Jeder Graph G hat eine Baumzerlegung (T, β) der Weite $bw(G)$ so dass

- $|\beta(t)| = 1$, wenn t ein Blatt ist, und
- für alle $v \in V(G)$ existiert genau ein Blatt $t \in V(T)$ mit $\beta(t) = \{v\}$.

Schritt 2. Die Taschen der inneren Knoten werden geordnete Tupel der Stelligkeit $\leq k+1$.

Schritt 3. Repräsentiere die "Knoten" in den Taschen durch Informationen, an welchen Stellen zwischen benachbarten Taschen „derselbe" Knoten steht.

Beispiel

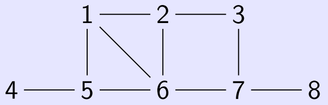


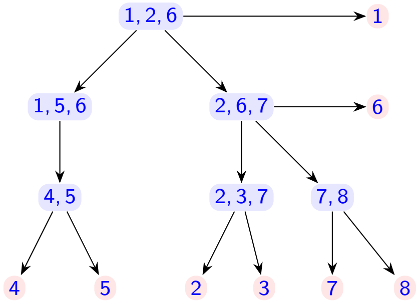
Eine Kodierung von Baumzerlegungen

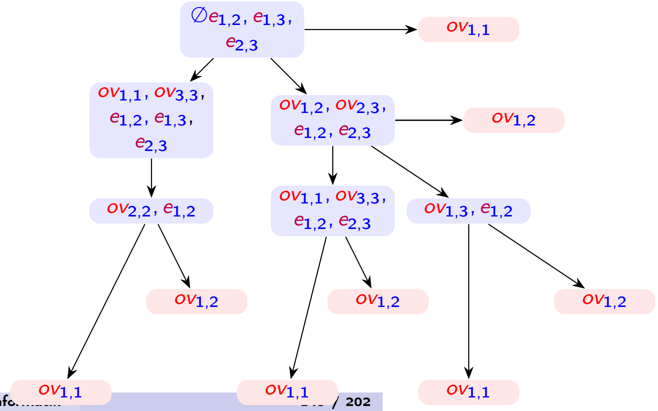
Ziel. Wir wollen Baumzerlegungen (T,β) der Weite k durch Σ_k-Bäume $\mathcal{T} = \mathcal{T}(T,\beta)$ für ein festes Alphabet Σ_k kodieren.

Für $k \in \mathbb{N}$ definiere $\Sigma_k := \mathcal{P}(\{ov_{i,j}, e_{i,j} : 1 \leq i \leq k+1\})$.

Beispiel







Eine Kodierung von Baumzerlegungen

Alphabet Σ_k.

Für $k \in \mathbb{N}$ sei $\Sigma_k := \mathcal{P}(\{ov_{i,j}, e_{i,j} : 1 \leq i \leq k+1\})$.

Beschriftung $\sigma : V(T) \to \Sigma_k$.

Für $t \in V(T)$ mit $\beta(t) = (t_1, \dots, t_l)$ sei

$$X_e := \{e_{i,j} : \{t_i, t_j\} \in E(G)\}.$$

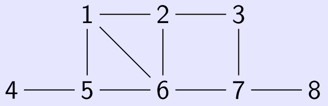
Wenn $(s,t) \in E(T)$ mit $\beta(s) = (s_1, \dots, s_{l'})$, dann

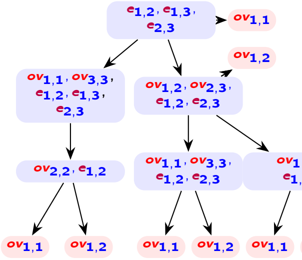
$$X_{ov} := \{ov_{i,j} : t_i = s_j\}.$$

Anderenfalls ist $X_{ov} = \emptyset$.

Wir definieren $\sigma(t) := X_e \cup X_{ov}$.

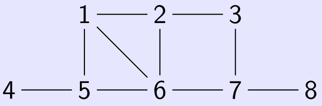
Beispiel





Eine Kodierung von Baumzerlegungen

Beispiel



Alphabet Σ_k.

Für $k \in \mathbb{N}$ sei $\Sigma_k := \mathcal{P}(\{ov_{i,j}, e_{i,j} : 1 \leq i \leq k+1\})$.

Knoten $v \in V(G)$. Bijektion zwischen $V(G)$ und $\{t \in V(T) : t \text{ Blatt } \}$.

Abschluss von $t \in V(T)$.

Kleinste Menge $\mathrm{cl}(t) \subseteq V(T) \times \{1, \dots, k+1\}$ für die gilt:

- $(t, 1) \in \mathrm{cl}(t)$ und
- für alle $(s, s') \in E(T)$ und $ov_{i,j} \in \beta(s')$:

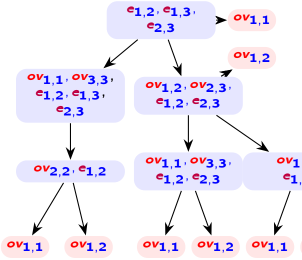
  $(s', i) \in \mathrm{cl}(t)$ gdw. $(s, j) \in \mathrm{cl}(t)$.

$v \in V(G)$ entspricht Abschluss des Blatts $t = t(v)$ mit $\beta(t) = (v)$.

Kanten $e \in E(G)$. Es gibt Kante $\{u, v\} \in E(G)$ gdw. es gibt

$s \in V(T)$ und $1 \leq i, j \leq k+1$ mit

$(s, i) \in \mathrm{cl}(t(u)), \quad (s, j) \in \mathrm{cl}(t(v))$ und $e_{i,j} \in \sigma(s)$.



Transformation der Formeln

Ziel. Gegeben $\varphi \in \text{MSO}$ und Graph G mit Baumzerlegung kodiert als Σ_k-Baum T. Konstruiere $\varphi' \in \text{MSO}$, so dass

$$G \models \varphi \text{ gdw. } T \models \varphi'.$$

Transformation der Formeln

Ziel. Gegeben $\varphi \in \mathsf{MSO}$ und Graph G mit Baumzerlegung kodiert als Σ_k-Baum T. Konstruiere $\varphi' \in \mathsf{MSO}$, so dass

$$G \models \varphi \text{ gdw. } T \models \varphi'.$$

Repräsentation $X \subseteq \mathcal{P}(\{(t,i) : t \in V(T), 1 \leq i \leq k+1\})$.

Tupel $(X_1, \ldots, X_{k+1})$, wobei $t \in X_i$ gdw. $(t,i) \in X$.

Abschluss $t \in V(T)$.
Kleinste $\mathrm{cl}(t) \subseteq V(T) \times \{1, \ldots, k+1\}$:
- $(t,1) \in \mathrm{cl}(t)$ und
- für $(s,s') \in E(T)$ und $ov_{i,j} \in \beta(s')$:
  $(s',i) \in \mathrm{cl}(t) \Leftrightarrow (s,j) \in \mathrm{cl}(t)$.

Transformation der Formeln

Ziel. Gegeben $\varphi \in \text{MSO}$ und Graph G mit Baumzerlegung kodiert als Σ_k-Baum T. Konstruiere $\varphi' \in \text{MSO}$, so dass

$$G \models \varphi \text{ gdw. } T \models \varphi'.$$

Repräsentation $X \subseteq \mathcal{P}(\{(t,i) : t \in V(T), 1 \leq i \leq k+1\})$.

Tupel $(X_1, \dots, X_{k+1})$, wobei $t \in X_i$ gdw. $(t,i) \in X$.

Hilfsformeln.

$$\begin{aligned}
(X_1,\dots,X_{k+1}) \subseteq (Y_1,\dots,Y_{k+1}) \quad &:= \quad \bigwedge_{i=1}^{k+1} X_i \subseteq Y_i \\
(X_1,\dots,X_{k+1}) \neq (Y_1,\dots,Y_{k+1}) \quad &:= \quad \bigvee_{i=1}^{k+1} Y_i \neq X_i \\
(X_1,\dots,X_{k+1}) \neq \emptyset \quad &:= \quad \bigvee_{i=1}^{k+1} X_i \neq \emptyset \\
ov_{i,j} \in \sigma(t) \quad &:= \quad \bigvee_{a \in \Sigma_k, ov_{i,j} \in a} P_a(t) \\
\varphi_{blatt}(t) \quad &:= \quad \neg\exists s (t,s) \in E
\end{aligned}$$

Abschluss $t \in V(T)$.
Kleinste $\mathrm{cl}(t) \subseteq V(T) \times \{1,\dots,k+1\}$:
- $(t,1) \in \mathrm{cl}(t)$ und
- für $(s,s') \in E(T)$ und $ov_{i,j} \in \beta(s')$:
  $(s',i) \in \mathrm{cl}(t) \Leftrightarrow (s,j) \in \mathrm{cl}(t)$.

Transformation der Formeln

Ziel. Gegeben $\varphi \in \text{MSO}$ und Graph G mit Baumzerlegung kodiert als Σ_k-Baum T. Konstruiere $\varphi' \in \text{MSO}$, so dass

$$G \models \varphi \text{ gdw. } T \models \varphi'.$$

Knoten. $v \in V(G)$ entspricht Blatt $t \in V(T)$ mit $\beta(t) := \{v\}$.

$$\varphi_v(t) := \varphi_{blatt}(t).$$

Hilfsformeln.
$ov_{i,j} \in \sigma(t)$
$e_{i,j} \in \sigma(t)$
$\overline{X} \neq \overline{Y}$
$\overline{X} \neq \emptyset$
$\overline{X} \subseteq \overline{Y}$
φ_{blatt}
$\varphi_{\text{cl}}(X_1, \ldots, X_{k+1})$

Transformation der Formeln

Ziel. Gegeben $\varphi \in \text{MSO}$ und Graph G mit Baumzerlegung kodiert als Σ_k-Baum T. Konstruiere $\varphi' \in \text{MSO}$, so dass

$$G \models \varphi \text{ gdw. } T \models \varphi'.$$

Knoten.
$v \in V(G) \mathrel{\hat{=}}$ Blatt $t \in V(T)$.
$\varphi_v(t) := \varphi_{blatt}(t)$.

Abschluss $t \in V(T)$.
Kleinste $\mathrm{cl}(t) \subseteq V(T) \times \{1, \dots, k+1\}$:
- $(t, 1) \in \mathrm{cl}(t)$ und
- für $(s, s') \in E(T)$ und $ov_{i,j} \in \beta(s')$: $(s', i) \in \mathrm{cl}(t) \Leftrightarrow (s, j) \in \mathrm{cl}(t)$.

Hilfsformeln.
$ov_{i,j} \in \sigma(t)$
$e_{i,j} \in \sigma(t)$
$\overline{X} \neq \overline{Y}$
$\overline{X} \neq \emptyset$
$\overline{X} \subseteq \overline{Y}$
φ_{blatt}
$\varphi_{\mathrm{cl}}(X_1, \dots, X_{k+1})$

Formel φ_{cl}.

$$\varphi'_{\mathrm{cl}}(X_1, \dots, X_{k+1}) := \forall (s, s') \in E \left(\bigwedge_{i,j} ov_{i,j} \in \sigma(s') \rightarrow (s \in X_j \leftrightarrow s' \in X_i)\right)$$

$$\varphi_{\mathrm{cl}}(X_1, \dots, X_{k+1}) := \varphi'_{\mathrm{cl}}(\overline{X}) \wedge \overline{X} \neq \emptyset \wedge \forall Y_1 \dots Y_{k+1} \left(\overline{Y} \neq \emptyset \wedge \overline{Y} \subsetneq \overline{X} \rightarrow \neg \varphi'_{\mathrm{cl}}(\overline{Y})\right)$$

Transformation der Formeln

Ziel. Gegeben $\varphi \in \mathrm{MSO}$ und Graph G mit Baumzerlegung kodiert als Σ_k-Baum T. Konstruiere $\varphi' \in \mathrm{MSO}$, so dass

$$G \models \varphi \text{ gdw. } T \models \varphi'.$$

Kanten. Seien $s, t \in V(T)$ Blätter und $u, v \in V(G)$ die repräsentierten Knoten.

$\{u, v\} \in E(G)$ gdw.

es gibt abgeschlossene Mengen X_s, X_t

mit s bzw. t als einzigem Blatt

und $u \in V(T)$ sowie i, j, so dass

$(u, i) \in X_s$, $(u, j) \in X_t$ und $e_{i,j} \in \sigma(u)$.

.

Knoten.
$v \in V(G) \hat{=}$ Blatt $t \in V(T)$.
$\varphi_v(t) := \varphi_{blatt}(t)$.

Hilfsformeln.
$ov_{i,j} \in \sigma(t)$
$e_{i,j} \in \sigma(t)$
$\overline{X} \neq \overline{Y}$
$\overline{X} \neq \emptyset$
$\overline{X} \subseteq \overline{Y}$
φ_{blatt}
$\varphi_{\mathrm{cl}}(X_1, \ldots, X_{k+1})$

Transformation der Formeln

Ziel. Gegeben $\varphi \in \text{MSO}$ und Graph G mit Baumzerlegung kodiert als Σ_k-Baum T. Konstruiere $\varphi' \in \text{MSO}$, so dass

$$G \models \varphi \text{ gdw. } T \models \varphi'.$$

Knoten.
$v \in V(G) \hat{=}$ Blatt $t \in V(T)$.
$\varphi_v(t) := \varphi_{blatt}(t)$.

Kanten. Seien $s, t \in V(T)$ Blätter und $u, v \in V(G)$ die repräsentierten Knoten.

$\{u, v\} \in E(G)$ gdw.

es gibt abgeschlossene Mengen X_s, X_t

mit s bzw. t als einzigem Blatt

und $u \in V(T)$ sowie i, j, so dass
$(u, i) \in X_s$, $(u, j) \in X_t$ und $e_{i,j} \in \sigma(u)$.

$\varphi_e(s, t) :=$

$\exists \bar{X} \exists \bar{Y} \varphi_{cl}(\bar{X}) \wedge \varphi_{cl}(\bar{Y}) \wedge$

.

Hilfsformeln.
$ov_{i,j} \in \sigma(t)$
$e_{i,j} \in \sigma(t)$
$\overline{X} \neq \overline{Y}$
$\overline{X} \neq \emptyset$
$\overline{X} \subseteq \overline{Y}$
φ_{blatt}
$\varphi_{\text{cl}}(X_1, \ldots, X_{k+1})$

Transformation der Formeln

Ziel. Gegeben $\varphi \in \text{MSO}$ und Graph G mit Baumzerlegung kodiert als Σ_k-Baum T. Konstruiere $\varphi' \in \text{MSO}$, so dass

$$G \models \varphi \text{ gdw. } T \models \varphi'.$$

Kanten. Seien $s, t \in V(T)$ Blätter und $u, v \in V(G)$ die repräsentierten Knoten.

$\{u, v\} \in E(G)$ gdw.

es gibt abgeschlossene Mengen X_s, X_t

mit s bzw. t als einzigem Blatt

und $u \in V(T)$ sowie i, j, so dass

$(u, i) \in X_s$, $(u, j) \in X_t$ und $e_{i,j} \in \sigma(u)$.

$$\varphi_e(s,t) := \exists \bar{X} \exists \bar{Y} \varphi_{cl}(\bar{X}) \wedge \varphi_{cl}(\bar{Y}) \wedge \forall x \big((\varphi_{blatt}(x) \wedge \bigvee_{i=1}^{k+1} x \in X_i) \leftrightarrow x = t \big) \wedge \forall x \big((\varphi_{blatt}(x) \wedge \bigvee_{i=1}^{k+1} x \in Y_i) \leftrightarrow x = s \big) \wedge$$

.

Knoten.
$v \in V(G) \mathrel{\hat{=}}$ Blatt $t \in V(T)$.
$\varphi_v(t) := \varphi_{blatt}(t)$.

Hilfsformeln.
$ov_{i,j} \in \sigma(t)$
$e_{i,j} \in \sigma(t)$
$\overline{X} \neq \overline{Y}$
$\overline{X} \neq \emptyset$
$\overline{X} \subseteq \overline{Y}$
φ_{blatt}
$\varphi_{\text{cl}}(X_1, \ldots, X_{k+1})$

Transformation der Formeln

Ziel. Gegeben $\varphi \in$ MSO und Graph G mit Baumzerlegung kodiert als Σ_k-Baum T. Konstruiere $\varphi' \in$ MSO, so dass

$$G \models \varphi \text{ gdw. } T \models \varphi'.$$

Knoten.
$v \in V(G) \hat{=}$ Blatt $t \in V(T)$.
$\varphi_v(t) := \varphi_{blatt}(t)$.

Kanten. Seien $s, t \in V(T)$ Blätter und $u, v \in V(G)$ die repräsentierten Knoten.

$\{u, v\} \in E(G)$ gdw. $\qquad \varphi_e(s,t) :=$

es gibt abgeschlossene Mengen X_s, X_t $\qquad \exists \bar{X} \exists \bar{Y} \varphi_{cl}(\bar{X}) \wedge \varphi_{cl}(\bar{Y}) \wedge$

mit s bzw. t als einzigem Blatt $\forall x \big((\varphi_{blatt}(x) \wedge \bigvee_{i=1}^{k+1} x \in X_i) \leftrightarrow x = t \big) \wedge$

$\forall x \big((\varphi_{blatt}(x) \wedge \bigvee_{i=1}^{k+1} x \in Y_i) \leftrightarrow x = s \big) \wedge$

und $u \in V(T)$ sowie i, j, so dass $\qquad \exists u \bigvee_{1 \le i,j \le k+1} \big(u \in X_i \wedge u \in Y_j \wedge e_{i,j} \in \sigma(u) \big)$.

$(u,i) \in X_s$, $(u,j) \in X_t$ und $e_{i,j} \in \sigma(u)$.

Hilfsformeln.
$ov_{i,j} \in \sigma(t)$
$e_{i,j} \in \sigma(t)$
$\overline{X} \neq \overline{Y}$
$\overline{X} \neq \emptyset$
$\overline{X} \subseteq \overline{Y}$
φ_{blatt}
$\varphi_{\mathrm{cl}}(X_1, \ldots, X_{k+1})$

Transformation der Formeln

Ziel. Gegeben $\varphi \in \mathsf{MSO}$ und Graph G mit Baumzerlegung kodiert als Σ_k-Baum T. Konstruiere $\varphi' \in \mathsf{MSO}$, so dass

$$G \models \varphi \text{ gdw. } T \models \varphi'.$$

Knotenformel $\varphi_v(t)$.

Kantenformel $\varphi_e(s,t)$.

Knoten.
$v \in V(G) \hat{=}$ Blatt $t \in V(T)$.
$\varphi_v(t) := \varphi_{blatt}(t)$.

Hilfsformeln.
$ov_{i,j} \in \sigma(t)$
$e_{i,j} \in \sigma(t)$
$\overline{X} \neq \overline{Y}$
$\overline{X} \neq \emptyset$
$\overline{X} \subseteq \overline{Y}$
φ_{blatt}
$\varphi_{\mathrm{cl}}(X_1, \dots, X_{k+1})$

Transformation der Formeln

Ziel. Gegeben $\varphi \in \mathrm{MSO}$ und Graph G mit Baumzerlegung kodiert als Σ_k-Baum T. Konstruiere $\varphi' \in \mathrm{MSO}$, so dass

$$G \models \varphi \text{ gdw. } T \models \varphi'.$$

Knotenformel $\varphi_v(t)$.

Kantenformel $\varphi_e(s,t)$.

Übersetzung von $\varphi \in \mathrm{MSO}$.

- $\exists x \psi \quad \rightsquigarrow \exists x \varphi_v(x) \wedge \psi'$
- $\exists X \psi \quad \rightsquigarrow \exists X (\forall x \in X \varphi_v(x) \wedge \psi')$
- $E(s,t) \quad \rightsquigarrow \varphi_e(s,t)$

Knoten.
$v \in V(G) \hat{=}$ Blatt $t \in V(T)$.
$\varphi_v(t) := \varphi_{blatt}(t)$.

Hilfsformeln.
$ov_{i,j} \in \sigma(t)$
$e_{i,j} \in \sigma(t)$
$\overline{X} \neq \overline{Y}$
$\overline{X} \neq \emptyset$
$\overline{X} \subseteq \overline{Y}$
φ_{blatt}
$\varphi_{\mathrm{cl}}(X_1, \dots, X_{k+1})$

Der Satz von Courcelle

Satz. $\mathrm{MC}(\mathrm{MSO}_2, \mathcal{C}) \in \mathrm{FPT}$ für jede Klasse $\mathcal{C}$ von Graphen mit beschränkter Baumweite. (Courcelle 1990)

Genauer: $\mathrm{MC}(\mathrm{MSO}_2, \texttt{GRAPH}) \in \mathrm{FPT}$ param durch $bw(G) + |\varphi|$.

Beweis. Gegeben $G \in \mathcal{C}$ und $\varphi \in \mathrm{MSO}$.

1. Berechne Baumzerlegung (T, β) von G und Σ_k-Baum (T, σ).
2. Übersetze φ in Formel φ', so dass $G \models \varphi \Leftrightarrow (T, \sigma) \models \varphi'$.

$$\begin{tikzcd}
G \arrow[r, rightsquigarrow] & (T, \beta) \\
\Vvdash & \Vvdash \\
\varphi \arrow[r, rightsquigarrow] & \varphi'
\end{tikzcd}$$

Der Satz von Courcelle

Satz. $\mathrm{MC}(\mathrm{MSO}_2, \mathcal{C}) \in \mathrm{FPT}$ für jede Klasse $\mathcal{C}$ von Graphen mit beschränkter Baumweite. (Courcelle 1990)

Genauer: $\mathrm{MC}(\mathrm{MSO}_2, \mathtt{GRAPH}) \in \mathrm{FPT}$ param durch $bw(G) + |\varphi|$.

Beweis. Gegeben $G \in \mathcal{C}$ und $\varphi \in \mathrm{MSO}$.

1. Berechne Baumzerlegung (T, β) von G und Σ_k-Baum (T, σ).
2. Übersetze φ in Formel φ', so dass $G \models \varphi \Leftrightarrow (T, \sigma) \models \varphi'$.

$$\begin{array}{ccccc} G & \rightsquigarrow & (T,\beta) & \rightsquigarrow & (T,\sigma) \\ \rotatebox{-90}{$\models$} & & & & \rotatebox{-90}{$\models$} \\ \varphi & & \rightsquigarrow & & \varphi' \end{array}$$

Zusammenfassung

MSO-Model-Checking auf Graphen kleiner Baumweite.

- $\text{MC}(\text{MSO}) \in \text{FPT}$ parametrisiert durch $bw(G) + |\varphi|$.
- Das Sequoia-Projekt der RWTH Aachen demonstriert, dass ein entsprechender Model-Checker auch in praktischen Anwendungen bestehen kann.
- Bisher haben wir allerdings nur die Standardkodierung von Graphen verwendet, also nur MSO_1.
- Die (leichte) Erweiterung auf MSO_2 behandeln wir im nächsten Video.

Interpretationen. Die hier verwendete Methode ist ein Spezialfall des Konzepts der *Interpretationen*.

Erweiterungen des Satzes von Courcelle

Der Satz von Courcelle

Satz. $\mathsf{MC}(\text{MSO}_2, \mathcal{C}) \in \mathsf{FPT}$ für jede Klasse $\mathcal{C}$ von Graphen mit beschränkter Baumweite. (Courcelle 1990)

Genauer: $\mathsf{MC}(\text{MSO}_2, \texttt{GRAPH}) \in \mathsf{FPT}$ param. durch $bw(G) + |\varphi|$.

Erweiterung auf MSO_2. Um den Beweis auf MSO_2, d.h. auf die Inzidenzkodierung von Graphen, erweitern zu können, benötigen wir noch das Konzept der Baumweite beliebiger Strukturen.

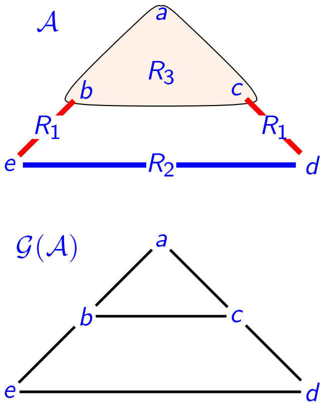
Gaifman-Graphen

Definition. Sei $\sigma := \{R_1, \dots, R_k\}$ eine relationale Signatur mit Relationssymbolen R_i der Stelligkeit r_i.

Der *Gaifman-Graph* einer σ-Struktur $\mathcal{A} = (A, (R_i^{\mathcal{A}})_{1 \leq i \leq k})$ ist der Graph $\mathcal{G}(\mathcal{A}) = (V, E)$ mit

- *Knotenmenge* $V = A$ und
- *Kantenmenge*

$$E = \{\{a_i, a_j\} : (a_1, \dots, a_{r_l}) \in R_l^{\mathcal{A}}, \text{ für ein } 1 \leq l \leq k\}.$$

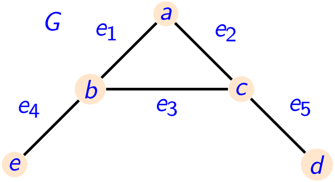


Beispiel. $\sigma := \{R_1, R_2, R_3\}$ mit $r_1 = r_2 = 2, \quad r_3 = 3$.

$$\mathcal{A} := \{\{a, \dots, e\}, \underbrace{\{(b,e), (c,d)\}}_{R_1^{\mathcal{A}}}, \underbrace{\{(d,e)\}}_{R_2^{\mathcal{A}}}, \underbrace{\{(a,b,c)\}}_{R_3^{\mathcal{A}}},$$

Dann ist $\mathcal{G}(\mathcal{A}) = (\{a, \dots, e\},$
$\{(a,b), (a,c), (c,b), (b,e), (c,d), (e,d), \quad (b,a), (c,a), (b,c), \dots\})$.

Gaifman-Graph von Inzidenzstrukturen

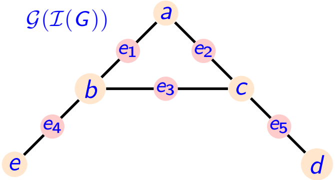


Beispiel. $\sigma_{inc} := \{V, E, inc\}$.

Dann ist $\mathcal{I} = \mathcal{I}(G) = (\{a, \dots, e\}, V^{\mathcal{I}}, E^{\mathcal{I}}, inc^{\mathcal{I}})$ mit

$inc^{\mathcal{I}} := \{(a, e_1), (a, e_2), \dots\}$ Inzidenzrelation

$E^{\mathcal{I}} := \{e_1, \dots, e_5\}$ $\quad V^{\mathcal{I}} := \{a, \dots, e\}$



Gaifman-Graph von Inzidenzstrukturen. $\mathcal{G}(\mathcal{I}(G))$ entsteht aus G, indem jede Kante von G einmal unterteilt wird.

Lemma. $bw(G) = bw(\mathcal{G}(\mathcal{I}(G)))$.

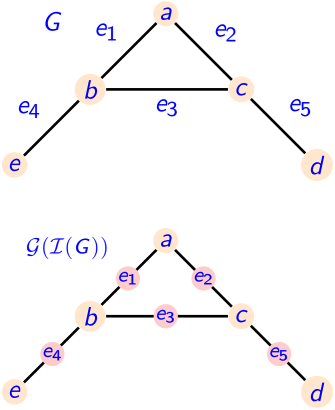
Baumweite von Inzidenzstrukturen

Lemma. $bw(G) = bw(\mathcal{G}(\mathcal{I}(G)))$.

Beweis. Sei $H = \mathcal{G}(\mathcal{I}(G))$.

1. Wenn G keine Kante enthält, gilt $bw(G) = bw(H) = 0$.
2. Wenn $bw(G) = 1$, dann ist G azyklisch und somit auch H azyklisch. Es gilt also $bw(H) = 1$.
3. Sei nun $bw(G) > 1$ und sei (T, β) eine BZ von G.
   Für jede Kante $e = \{u, v\} \in E(G)$ wähle
   - $t(e) \in V(T)$ mit $u, v \in \beta(t)$ und
   - füge neuen Knoten t_e mit $\beta(t_e) = \{e, u, v\}$ und die Kante $(t(e), t_e)$ zu T hinzu.

   Sei (T', β') die resultierende BZ. Dann ist (T', β') eine BZ von H der gleichen Weite wie (T, β). □



Der Satz von Courcelle für MSO$_2$

Satz. $MC(\text{MSO}_2, \mathcal{C}) \in \text{FPT}$ für jede Klasse $\mathcal{C}$ von Inzidenzstrukturen von Graphen mit beschränkter Baumweite. (Courcelle 1990)

Erinnerung: Alphabet Σ_k.
Für $k \in \mathbb{N}$ sei $\Sigma_k := \mathcal{P}(\{ov_{i,j}, e_{i,j} inc_{i,j}, v_i, e_i : 1 \leq i \leq k+1\})$.

Beschriftung $\sigma : V(T) \to \Sigma_k$.
Für $t \in V(T)$ mit $\beta(t) = (t_1, \dots, t_l)$ sei

$$X_e := \{inc_{i,j} : \{t_i, t_j\} \in inc^{\mathcal{I}}\} \cup \{v_i : t_i \in V^{\mathcal{I}}\} \cup \{e_i : t_i \in E^{\mathcal{I}}\}.$$

Wenn $(s,t) \in E(T)$ mit $\beta(s) = (s_1, \dots, s_{l'})$, dann $X_{ov} := \{ov_{i,j} : t_i = s_j\}$.

Anderenfalls ist $X_{ov} = \emptyset$. Wir definieren $\sigma(t) := X_e \cup X_{ov}$.

Beweis. Berechne aus der BZ von $\mathcal{G}(\mathcal{I})$ einen Σ_k-Baum (T, σ) und aus φ eine Formel φ', so dass gilt: $\mathcal{I} \models \varphi$ gdw. $(T, \sigma) \models \varphi'$.

Zusammenfassung

Gaifman-Graphen.

- Mit Hilfe von Gaifman-Graphen können wir die Baumweite beliebiger Strukturen definieren.
- Der Beweis des Satzes von Courcelle kann dann leicht von Graphen in Standardkodierung auf Inzidenzstrukturen angepasst werden.

Lokalität der Prädikatenlogik

Einleitung

Lokalität. Wir werden in diesem Teil der Vorlesung zeigen, dass in FO nur *lokale* Eigenschaften von Strukturen oder Elementen in Strukturen ausgedrückt werden können.

lokal: Eigenschaften, die nur von der *„Umgebung"* des Elements abhängt.

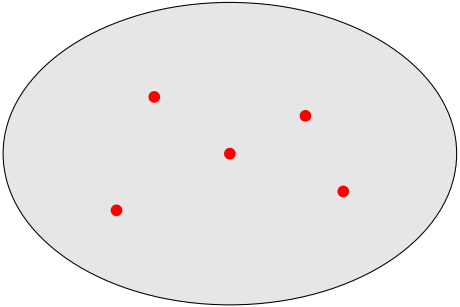
Lokale Eigenschaften	Nicht-lokale Eigenschaften
$\exists x \big(Rot(x) \wedge \exists y (E(x,y) \wedge Gruen(y))\big)$ Es gibt einen roten Knoten mit einem grünen Nachbarn.	Es gibt einen roten Knoten der mit einem Pfad zu einem grünen Knoten verbunden ist.
$\exists x\, Rot(x) \wedge \exists x\, Gruen(x)$ Es gibt einen roten Knoten und es gibt einen grünen Knoten.	Der Graph enthält eine gerade Anzahl von Knoten.

Dominierende Mengen

Beispiel.

$$\exists x_1 \dots \exists x_k \,\forall y \bigvee_{i=1}^{k} \big(y = x_i \vee E(x_i, y)\big)$$

Der Graph enthält eine dominierende Menge der Größe $\leq k$.



Lokal?

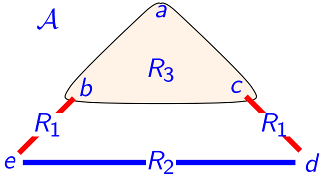
Erinnerung: Gaifman-Graph

Definition. Sei $\sigma := \{R_1, \dots, R_k\}$ eine relationale Signatur mit Relationssymbolen R_i der Stelligkeit r_i.

Der *Gaifman-Graph* einer σ-Struktur $\mathcal{A} = (A, (R_i^{\mathcal{A}})_{1 \le i \le k})$ ist der Graph $\mathcal{G}(\mathcal{A}) = (V, E)$ mit

- *Knotenmenge* $V = A$ und
- *Kantenmenge*

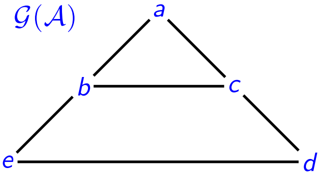
$$E = \{\{a_i, a_j\} : (a_1, \dots, a_{r_l}) \in R_l^{\mathcal{A}}, \text{ für ein } 1 \le l \le k\}.$$



Beispiel. $\sigma := \{R_1, R_2, R_3\}$ mit $r_1 = r_2 = 2, \quad r_3 = 3$.

$$\mathcal{A} := \{\{a, \dots, e\}, \underbrace{\{(b, e), (c, d)\}}_{R_1^{\mathcal{A}}}, \underbrace{\{(d, e)\}}_{R_2^{\mathcal{A}}}, \underbrace{\{(a, b, c)\}}_{R_3^{\mathcal{A}}},$$

Dann ist $\mathcal{G}(\mathcal{A}) = (\{a, \dots, e\},$
$\{(a, b), (a, c), (c, b), (b, e), (c, d), (e, d), \quad (b, a), (c, a), (b, c), \dots\})$.



Distanz und r-Nachbarschaften

Sei σ eine relationale Signature und $\mathcal{A}$ eine σ-Struktur.

Definition. Sei $r \in \mathbb{N}$.

1. Die *Distanz* $\mathrm{dist}_{\mathcal{A}}(a, b)$ von $a, b \in A$ ist die Distanz von a und b im Gaifman-Graph $\mathcal{G}(\mathcal{A})$.
2. Die *r-Nachbarschaft* von a in $\mathcal{A}$ ist die Menge $N_r^{\mathcal{A}}(a) := \{b \in A : \mathrm{dist}_{\mathcal{A}}(a, b) \leq r\}$.
3. $\mathcal{A}_r(a)$ ist die von $N_r^{\mathcal{A}}(a)$ induzierte Substruktur von $\mathcal{A}$.

   $\mathcal{A}_r(a) := \Big(N_r^{\mathcal{A}}(a), \big(R^{\mathcal{A}_r(a)}\big)_{R\in\sigma}\Big)$, wobei

   $R^{\mathcal{A}_r(a)} = \{(a_1, \dots, a_{ar(R)}) : a_i \in N_r^{\mathcal{A}}(a), (a_1, \dots, a_{ar(R)}) \in R^{\mathcal{A}}\}$.

Anmerkung.
Für $r \in \mathbb{N}$ existiert

$$dist_{\leq r}(x, y) \in \mathrm{FO}$$

die besagt, dass die Distanz zwischen x und y in $\mathcal{G}(\mathcal{A})$ höchstens r ist.

Schreibweise.

$dist(x, y) \leq r$

$dist(x, y) = r$

$dist(x, y) > r$

usw.

Lokale Sätze

Definition. Sei $\varphi(x) \in FO[\sigma]$. φ ist r-lokal um x, wenn für alle σ-Strukturen $\mathcal{A} := (A, \sigma)$ und alle $a \in A$ gilt:

$$\mathcal{A} \models \varphi(a) \iff \mathcal{A}_r(a) \models \varphi(a).$$

Beispiele. Sei $\sigma = \{R, E\}$, mit R 1-stellig und E 2-stellig.

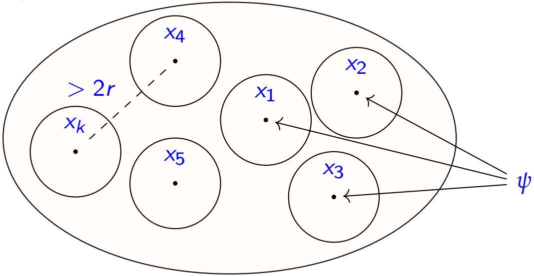
- $\varphi(x) = R(x)$ ist 0-lokal
- $\varphi(x) = \exists y \; (E(x, y) \wedge R(y))$ ist 1-lokal
- Für $r \in \mathbb{N}$ ist $\varphi(x) = \exists y \; R(y)$ nicht r-lokal.

Basis-Lokale Sätze

Definition. Ein *basis-lokaler Satz* ist ein Satz, der Form

$$\varphi = \exists x_1 \dots \exists x_k \left(\bigwedge_{1 \leq i < j \leq k} dist(x_i, x_j) > 2r \wedge \bigwedge_{i=1}^{k} \psi(x_i) \right),$$

wobei $\psi(x)$ eine r-lokale Formel ist für ein $r \in \mathbb{N}$.



Der Satz von Gaifman

Definition. Ein Satz $\varphi \in \mathsf{FO}$ ist in *Gaifman-Normalform*, wenn φ eine Boole'schen Kombination basis-lokaler Sätze ist.

Satz. Jeder Satz $\varphi \in FO$ ist äquivalent zu einem Satz in Gaifman-Normalform.

$$\bigvee_{s} \bigwedge_{t_s} (\neg)\exists x_1 \dots \exists x_{k_{s,t_s}} \left(\bigwedge_{1 \le i < j \le k_{s,t_s}} dist(x_i, x_j) > 2r_{s,t_s} \wedge \bigwedge_{i=1}^{k_{s,t_s}} \psi_{s,t_s}(x_i) \right)$$

Beispiele

Beispiel. Die Klasse der Graphen vom Durchmesser > 4.

$$\exists x_1 \exists x_2 \Big(\text{dist}(x_1, x_2) > 2 \cdot 2 \Big)$$

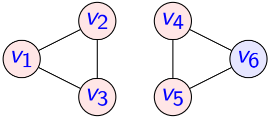
Beispiele

Beispiel. Die Klasse der Graphen, die ein rotes Dreieck enthalten.

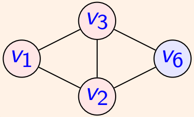
$$\varphi_3 := \exists x_1 \Big(\exists y \exists z \big(E(x_1, y) \wedge E(y, z) \wedge E(z, x_1) \wedge Rot(x_1) \wedge Rot(y) \wedge Rot(z) \big) \Big)$$

Beispiele

Beispiel. Die Klasse der Graphen, die folgenden induzierten Untergraph enthalten.



Falsch!
Die Formel würde auch hier gelten:



Idee.

$$\exists x_1\Big(\exists y \exists z\big(E(x_1,y)\wedge E(y,z)\wedge E(z,x_1)\wedge Rot(x_1)\wedge Rot(y)\wedge Rot(z)\big)\Big)$$

$$\wedge$$

$$\exists x_1\Big(\exists y \exists z\big(E(x_1,y)\wedge E(y,z)\wedge E(z,x_1)\wedge Rot(x_1)\wedge Rot(y)\wedge Blau(z)\big)\Big)$$

Zusammenfassung

Lokalität der Prädikatenlogik. Der Satz von Gaifman formalisiert die Intuition, dass in der Prädikatenlogik keine wirklich *globalen* Aussagen über Graphen getroffen werden können.

Der Satz hat sich als extrem nützlich für die Konstruktion von Auswertungsalgorithmen auf speziellen Graphklassen erwiesen.

Das Beispiel der *dominierenden Menge* zeigt aber, dass lokale Aussagen durchaus global aussehen können.

Der Satz von Gaifman

Der Satz von Gaifman

Definition. Ein Satz $\varphi \in \mathrm{FO}$ ist in *Gaifman-Normalform*, wenn φ eine boolesche Kombination basis-lokaler Sätze ist.

Satz Jeder Satz $\varphi \in FO$ ist äquivalent zu einem Satz in Gaifman-Normalform.

$$\bigvee_{s} \bigwedge_{t_s} (\neg)\exists x_1 \ldots \exists x_{k_{s,t_s}} \left(\bigwedge_{1\leq i<j\leq k_{s,t_s}} dist(x_i, x_j) > 2r_{s,t_s} \wedge \bigwedge_{i=1}^{k_{s,t_s}} \psi_{s,t_s}(x_i) \right)$$

Beweis des Satzes

Satz. Jeder Satz $\varphi \in FO$ ist äquivalent zu einem Satz in Gaifman-Normalform.

Lemma A. Für jedes $m \in \mathbb{N}$ gibt es ein $M = M(m) \in \mathbb{N}$, so dass für alle σ-Strukturen $\mathcal{A}, \mathcal{B}$ gilt:

Falls $\mathcal{A}$ und $\mathcal{B}$ dieselben basis-lokalen $\text{FO}[\sigma]$-Sätze vom Quantorenrang höchstens M erfüllen, dann gilt $\mathcal{A} \equiv_m \mathcal{B}$.

Lemma B. Sei φ ein Satz und $m := \text{qr}(\varphi)$.

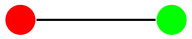
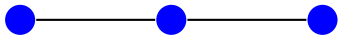
Sei $M = M(m)$ wie in Lemma A und $\chi_1^M, \ldots, \chi_{h(M)}^M$ eine (bis auf Äquivalenz) vollständige Liste aller basis-lokalen Sätze vom Quantorenrang $\leq M$.

Dann ist φ äquivalent zur folgenden Booleschen Kombination basis-lokaler Sätze:

$$\tilde{\varphi} := \bigvee \left\{ \left(\bigwedge_{i \in I} \chi_i^M \wedge \bigwedge_{i \notin I} \neg\chi_i^M\right) : \begin{array}{l} \text{es existiert } I \subseteq \{1, \ldots, h(M)\} \\ \text{und eine } \sigma\text{-Struktur } \mathcal{A}\text{, so dass } \mathcal{A} \models \varphi \\ \text{und für alle } 1 \leq i \leq M : \mathcal{A} \models \chi_i^M \iff i \in I \end{array} \right\}$$

Beispiel zu Lemma B

Sei $\mathcal{L} := \left\{ \begin{array}{l} \varphi_1 := \exists x R(x), \\ \varphi_2 := \exists x_1 \exists x_2 \exists x_3 \bigwedge_i B(x_i) \wedge E(x_1, x_2) \wedge E(x_2, x_3), \\ \varphi_3 := \forall y \big(G(y) \vee R(y) \big) \end{array} \right\}$.

Sei $\mathcal{A}_1$ die Struktur  und $\mathcal{A}_2$ die Struktur 

Betrachte den Satz

$$\psi := \Big(\varphi_1 \wedge \varphi_3 \wedge \neg\varphi_2 \Big) \vee \Big(\varphi_2 \wedge \neg\varphi_1 \wedge \neg\varphi_3 \Big)$$

Nun gilt: Wenn $\mathcal{B} \models \psi$ dann erfüllt $\mathcal{B}$ entweder genau die gleichen Sätze aus $\mathcal{L}$ wie $\mathcal{A}_1$ oder $\mathcal{B}$ erfüllt genau die gleichen Sätze aus $\mathcal{L}$ wie $\mathcal{A}_2$.

Lemma B.
Sei $\varphi \in \mathrm{FO}$ und $m := \mathrm{qr}(\varphi)$.
$M = M(m)$ und
$\chi_1^M, \dots, \chi_{h(M)}^M$ Liste aller basis-lokalen Sätze vom Quantorenrang $\leq M$.

φ ist äquivalent zu

$$\tilde{\varphi} := \bigvee \Big\{ \big(\bigwedge_{i \in I} \chi_i^M \wedge \bigwedge_{i \notin I} \neg\chi_i^M\big) : \text{es ex. } I \subseteq \{1, \dots, h(M)\} \text{ und } \mathcal{A}, \text{ so dass } \mathcal{A} \models \varphi \text{ und für } 1 \leq i \leq M: \quad \mathcal{A} \models \chi_i^M \iff i \in I \Big\}$$

Beweis von Lemma B

Beweis. Sei $\mathcal{B}$ eine σ-Struktur.
Wir zeigen, dass $\mathcal{B} \models \varphi$ genau dann, wenn $\mathcal{B} \models \tilde{\varphi}$.
Die Hinrichtung ist dabei klar.

Zum Beweis der Rückrichtung gelte also $\mathcal{B} \models \tilde{\varphi}$.
Also existiert eine Indexmenge $I \subseteq \{1, \dots, h(M)\}$ so dass

1. $\mathcal{B} \models \bigwedge_{i \in I} \chi_i^M \wedge \bigwedge_{i \notin I} \neg\chi_i^M$,
   d.h. $\mathcal{B}$ erfüllt genau die Sätze χ_i^M für die $i \in I$, und
2. es ex. σ-Struktur $\mathcal{A}$ mit $\mathcal{A} \models \varphi$ und $\mathcal{A} \models \chi_i^M$ gdw. $i \in I$.

Also erfüllen $\mathcal{A}$ und $\mathcal{B}$ dieselben basis-lokalen Sätze vom Quantorenrang höchstens M.

Nach Lemma A gilt also $\mathcal{A} \equiv_m \mathcal{B}$ und somit $\mathcal{B} \models \varphi$ $\square$

Lemma B.
Sei $\varphi \in \text{FO}$ und $m := \text{qr}(\varphi)$.
$M = M(m)$ und
$\chi_1^M, \dots, \chi_{h(M)}^M$ Liste aller basis-lokalen Sätze vom Quantorenrang $\leq M$.

φ ist äquivalent zu

$$\tilde{\varphi} := \bigvee \Big\{ \big(\bigwedge_{i \in I} \chi_i^M \wedge \bigwedge_{i \notin I} \neg\chi_i^M\big) : \begin{array}{l} \text{es ex. } I \subseteq \{1, \dots, h(M)\} \text{ und } \mathcal{A}, \\ \text{so dass } \mathcal{A} \models \varphi \text{ und für } 1 \leq i \leq M: \\ \quad \mathcal{A} \models \chi_i^M \iff i \in I \end{array} \Big\}$$

Beweis des Satzes

Satz. Jeder Satz $\varphi \in FO$ ist äquivalent zu einem Satz in Gaifman-Normalform.

Lemma A. Für jedes $m \in \mathbb{N}$ gibt es ein $M = M(m) \in \mathbb{N}$, so dass für alle σ-Strukturen $\mathcal{A}, \mathcal{B}$ gilt:

Falls $\mathcal{A}$ und $\mathcal{B}$ dieselben basis-lokalen $\mathrm{FO}[\sigma]$-Sätze vom Quantorenrang höchstens M erfüllen, dann gilt $\mathcal{A} \equiv_m \mathcal{B}$.

Lemma B. Sei φ ein Satz und $m := \mathrm{qr}(\varphi)$.

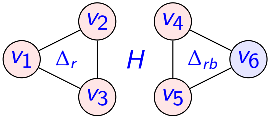
Sei $M = M(m)$ wie in Lemma A und $\chi_1^M, \ldots, \chi_{h(M)}^M$ eine (bis auf Äquivalenz) vollständige Liste aller basis-lokalen Sätze vom Quantorenrang $\leq M$.

Dann ist φ äquivalent zur folgenden Booleschen Kombination basis-lokaler Sätze:

$$\tilde{\varphi} := \bigvee \left\{ \left(\bigwedge_{i \in I} \chi_i^M \wedge \bigwedge_{i \notin I} \neg\chi_i^M\right) : \begin{array}{l} \text{es existiert } I \subseteq \{1, \ldots, h(M)\} \\ \text{und eine } \sigma \text{ -Struktur} \mathcal{A} \text{ , so dass } \mathcal{A} \models \varphi \\ \text{und für alle } 1 \leq i \leq M : \mathcal{A} \models \chi_i^M \iff i \in I \end{array} \right\}$$

Einschub: Beispiele für Gaifman-Normalform

Beispiel Gaifman-Normalform



Ziel. Wir wollen die Klasse der Graphen, die folgenden induzierten Untergraph H enthalten, durch einen Satz in Gaifman-Normalform definieren.

Hilfsformeln. Wir verwenden folgende Hilfsformeln.

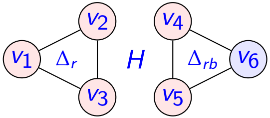
- $$\varphi_{\Delta_r}(x) := \exists y \exists z \big(\quad E(x,y) \wedge E(y,z) \wedge E(z,x) \wedge \\ Rot(x) \wedge Rot(y) \wedge Rot(z) \big)$$

  x liegt auf einem *roten Dreieck*.

- $$\varphi_{\Delta_{rb}}(x) := \exists y \exists z \Big(\quad E(x,y) \wedge E(y,z) \wedge E(z,x) \wedge \\ \big((Rot(x) \wedge Rot(y) \wedge Blau(z)) \vee \\ (Blau(x) \wedge Rot(y) \wedge Rot(z)) \big) \Big)$$

  x liegt auf einem *rot-blauen Dreieck*.

Beispiel Gaifman-Normalform



Ziel. Wir wollen die Klasse der Graphen, die folgenden induzierten Untergraph H enthalten, durch einen Satz in Gaifman-Normalform definieren.

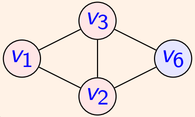
Idee 1.

$$\exists x_1 \varphi_{\Delta_r}(x_1) \wedge \exists x_1 \varphi_{\Delta_{rb}}(x_1)$$

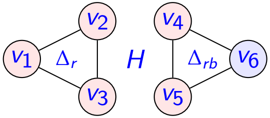
Hilfsformeln.
φ_{Δ_r}: x liegt auf einem *roten Dreieck*.

$\varphi_{\Delta_{rb}}$: x liegt auf einem *rot-blauen Dreieck*.

Falsch!
Die Formel würde auch hier gelten:



Beispiel Gaifman-Normalform



Ziel. Wir wollen die Klasse der Graphen, die folgenden induzierten Untergraph H enthalten, durch einen Satz in Gaifman-Normalform definieren.

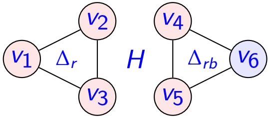
Wie können die Dreiecke zueinander liegen?

Hilfsformeln.

φ_{Δ_r}: x liegt auf einem *roten Dreieck*.

$\varphi_{\Delta_{rb}}$:x liegt auf einem *rot-blauen Dreieck*.

Beispiel Gaifman-Normalform



Ziel. Wir wollen die Klasse der Graphen, die folgenden induzierten Untergraph H enthalten, durch einen Satz in Gaifman-Normalform definieren.

Idee 2.

Fall 1. "Es ex. x_1 mit rotem und blau-rotem Dreieck in der 8-Nachbarschaft die nicht verbunden sind" oder

Fall 2. "Es ex. x_1 und es ex. x_2 mit Abstand $> 2 \cdot 2$ und x_1 liegt auf einem roten und x_2 auf einem rot-blauen Dreieck".

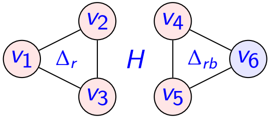
$$\exists x_1 \exists x_2 \Big(\mathrm{dist}(x_1, x_2) > 2 \cdot 2 \wedge \,?\, (x_1{:}\Delta_r \;\; x_2{:}\Delta_{rb}) \Big)$$

Hilfsformeln.

φ_{Δ_r}: x liegt auf einem *roten Dreieck*.

$\varphi_{\Delta_{rb}}$: x liegt auf einem *rot-blauen Dreieck*.

Beispiel Gaifman-Normalform



Ziel. Wir wollen die Klasse der Graphen, die folgenden induzierten Untergraph H enthalten, durch einen Satz in Gaifman-Normalform definieren.

Lösung.

1. „Es ex. x so dass $N_4(x)$ sowohl Δ_r als auch Δ_{rb} enthält"
   1.1 „Es ex x s.d. $N_8(x)$ enthält induziertes H" oder
   1.2 „Es ex. x_1 und x_2, $\text{dist}(x_1, x_2) > 2 \cdot 4$ und x_1 und x_2 liegen auf einem Δ_r oder
   1.3 „Es ex. x_1 und x_2, $\text{dist}(x_1, x_2) > 2 \cdot 4$ und x_1 und x_2 liegen auf einem Δ_{rb}.

   oder

2. „Es ex. kein x so dass $N_4(x)$ sowohl Δ_r als auch Δ_{rb} enthält" und „es ex. x auf Δ_r" und „es ex. x auf Δ_{rb}".

Hilfsformeln.

φ_{Δ_r}: x liegt auf einem *roten Dreieck*.

$\varphi_{\Delta_{rb}}$: x liegt auf einem *rot-blauen Dreieck*.

Lokale Formeln

Erinnerung. Eine Formel $\psi(x)$ ist *r-lokal*, wenn für alle Strukturen $\mathcal{A}$ und alle $a \in A$ gilt:

$$\mathcal{A} \models \psi[a] \qquad \text{gdw.} \qquad \mathcal{A}_r(a) \models \psi[a].$$

Die Gültigkeit von ψ in $\mathcal{A}$ mit Belegung $x \mapsto a$ hängt also nur von der r-Umgebung von a in $\mathcal{A}$ ab.

Relativierung. Sei $\psi(x)$ eine Formel.

Wir haben bereits Formeln $\text{dist}(x, y) \leq r$ konstruiert, so dass

$$\mathcal{A} \models \big(\text{dist}(x, y) \leq r\big)[x/a, y/b] \quad \text{gdw.} \quad \text{Distanz zwischen } a \text{ und } b \text{ in } \mathcal{G}(\mathcal{A}) \text{ ist } \leq r.$$

Sei $\psi^r(x)$ die Formel, die man aus ψ erhält, wenn rekursiv alle

- $\exists y\theta$ ersetzt werden durch $\exists y\big(\text{dist}(x, y) \leq r \wedge \theta^r\big)$
- $\forall y\theta$ ersetzt werden durch $\forall y\big(\text{dist}(x, y) \leq r \rightarrow \theta^r\big)$

Dann gilt $\mathcal{A} \models \psi^r[a]$ gdw. $\mathcal{A}_r(a) \models \psi[a]$.

Beweis des Satzes von Gaifman: Teil II

Der Satz von Gaifman

Satz. Jeder Satz $\varphi \in FO$ ist äquivalent zu einem Satz in Gaifman-Normalform.

Lemma A. Für jedes $m \in \mathbb{N}$ gibt es ein $M = M(m) \in \mathbb{N}$, so dass für alle σ-Strukturen $\mathcal{A}, \mathcal{B}$ gilt: Falls $\mathcal{A}$ und $\mathcal{B}$ dieselben basis-lokalen $\mathrm{FO}[\sigma]$-Sätze vom Quantorenrang höchstens M erfüllen, dann gilt $\mathcal{A} \equiv_m \mathcal{B}$.

Lemma B. Sei φ ein Satz und $m := \mathrm{qr}(\varphi)$.

Sei $M = M(m)$ wie in Lemma A und $\chi_1^M, \ldots, \chi_{h(M)}^M$ eine (bis auf Äquivalenz) vollständige Liste aller basis-lokalen Sätze vom Quantorenrang $\leq M$.

Dann ist φ äquivalent zur folgenden Booleschen Kombination basis-lokaler Sätze:

$$\tilde{\varphi} := \bigvee \left\{ (\bigwedge_{i \in I} \chi_i^M \wedge \bigwedge_{i \notin I} \neg\chi_i^M) : \begin{array}{l} \text{es existiert } I \subseteq \{1, \ldots, h(M)\} \\ \text{und eine } \sigma \text{ -Struktur} \mathcal{A} \text{ , so dass } \mathcal{A} \models \varphi \\ \text{und für alle } 1 \leq i \leq M : \mathcal{A} \models \chi_i^M \iff i \in I \end{array} \right\}$$

Beweis Lemma A

Lemma A. Für jedes $m \in \mathbb{N}$ gibt es ein $M = M(m) \in \mathbb{N}$, so dass für alle σ-Strukturen $\mathcal{A}, \mathcal{B}$ gilt: Falls $\mathcal{A}$ und $\mathcal{B}$ dieselben basis-lokalen $\mathrm{FO}[\sigma]$-Sätze vom Quantorenrang höchstens M erfüllen, dann gilt $\mathcal{A} \equiv_m \mathcal{B}$.

Erinnerung. $(\mathcal{A}, \overline{a}) \equiv_m (\mathcal{B}, \overline{b})$ gdw. für alle Formeln $\varphi(\overline{x})$ mit Quantorenrang $\mathrm{qr}(\varphi) \leq m$ und $|\overline{x}| = |\overline{a}| = |\overline{b}|$ gilt:

$$\mathcal{A} \models \varphi[\overline{a}] \quad \text{gdw.} \quad \mathcal{B} \models \varphi[\overline{b}].$$

EF-Spiele. Es gilt: $(\mathcal{A}, \overline{a}) \equiv_m (\mathcal{B}, \overline{b})$ gdw. die Duplikatorin (**D**) eine Gewinnstrategie im m-Runden Spiel $\mathfrak{G}_m(\mathcal{A}, \overline{a}, \mathcal{B}, \overline{b})$ hat.

Erinnerung. (Hintikka-Formeln) Sei $\mathcal{A}$ eine Struktur und $\overline{a} \in A^k$. Der *m-Isomorphietyp*, oder die *m-Hintikka-Formel*, $\varphi^m_{\mathcal{A},\overline{a}}(\overline{x})$ hat Quantorenrang $\mathrm{qr}(\varphi^m_{\mathcal{A},\overline{a}}) = m$ und es gilt:

$$\mathcal{B} \models \varphi^m_{\mathcal{A},\overline{a}}[\overline{b}] \text{ gdw. } (\mathcal{A}, \overline{a}) \equiv_m (\mathcal{B}, \overline{b}).$$

Beweis Lemma A

Lemma A. Für jedes $m \in \mathbb{N}$ gibt es ein $M = M(m) \in \mathbb{N}$, so dass für alle σ-Strukturen $\mathcal{A}, \mathcal{B}$ gilt: Falls $\mathcal{A}$ und $\mathcal{B}$ dieselben basis-lokalen $\text{FO}[\sigma]$-Sätze vom Quantorenrang höchstens M erfüllen, dann gilt $\mathcal{A} \equiv_m \mathcal{B}$.

Relativierte Hintikka-Formeln. Sei $\mathcal{A}$ eine Struktur und $\overline{a} \in A^k$.
$\varphi^{m,r}_{\mathcal{A},\overline{a}}(\overline{x})$: Einschränkung auf die r-Nachbarschaft um $\overline{x}$.

$$\mathcal{B} \models \varphi^{m,r}_{\mathcal{A},\overline{a}}[\overline{b}] \text{ gdw. } (\mathcal{A}_r(\overline{a}), \overline{a}) \equiv_m (\mathcal{B}_r(\overline{b}), \overline{b}).$$

Beweis Lemma A

Lemma A. Für jedes $m \in \mathbb{N}$ gibt es ein $M = M(m) \in \mathbb{N}$, so dass für alle σ-Strukturen $\mathcal{A}, \mathcal{B}$ gilt: Falls $\mathcal{A}$ und $\mathcal{B}$ dieselben basis-lokalen $\text{FO}[\sigma]$-Sätze vom Quantorenrang höchstens M erfüllen, dann gilt $\mathcal{A} \equiv_m \mathcal{B}$.

Voraussetzung. $\mathcal{A}, \mathcal{B}$ erfüllen die selben basis-lokalen Sätze φ mit $\mathrm{qr}(\varphi) \leq M(m)$.

Invariante. Wir zeigen, dass **D** so spielen kann, dass nach $1 \leq i \leq m$ gespielten Zügen $\overline{a} := a_1, \ldots, a_i \in A^i$ und $\overline{b} := b_1, \ldots, b_i \in B^i$ gilt:

$$(\mathcal{A}_{7^{m-i}}(\overline{a}), \overline{a}) \equiv_{g(m-i)} (\mathcal{B}_{7^{m-i}}(\overline{b}), \overline{b}).$$

($\mathcal{A}_{7^{m-i}}(\overline{a})$ die durch $N_{7^{m-i}}(\overline{a}) := \bigcup_{a \in \overline{a}} N_{7^{m-i}}(a)$ induzierte Substruktur.)

Die Funktion g. Definiere induktiv $g : \{0, \ldots, m\} \to \mathbb{N}$

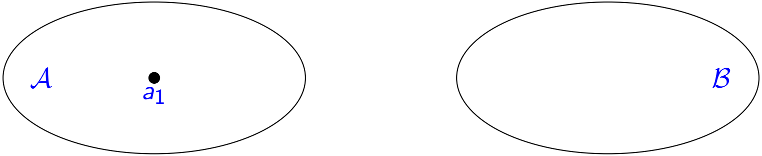
Basisfall. $g(0) = 1$. Definiere $M := g(m)$.

Berechne g nicht exakt, sondern formulieren Bedingungen an g.

Bedingung 1. $g(j) < g(j+1)$ für alle $j < m$.

Beweis Lemma A

Basisfall $i = 1$. Angenommen, **H** zieht $a_1 \in A$.



Invariante $(\star)$. Nach Runde i:

$$\equiv_{g(m-i)} \begin{matrix}(\mathcal{A}_{7^{m-i}}(\overline{a}), \overline{a}) \\ (\mathcal{B}_{7^{m-i}}(\overline{b}), \overline{b}).\end{matrix}$$

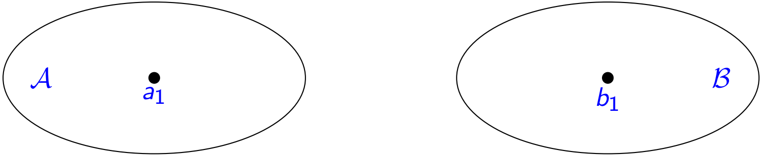
Funktion g.

$g(0) := 1 \quad M(m) = g(m)$

1. $g(j) < g(j+1)$

Beweis Lemma A

Basisfall $i = 1$. Angenommen, **H** zieht $a_1 \in A$.



Invariante $(\star)$. Nach Runde i:

$\equiv_{g(m-i)}$ $(\mathcal{A}_{7^{m-i}}(\overline{a}), \overline{a})$ $(\mathcal{B}_{7^{m-i}}(\overline{b}), \overline{b})$.

Funktion g.

$g(0) := 1 \quad M(m) = g(m)$

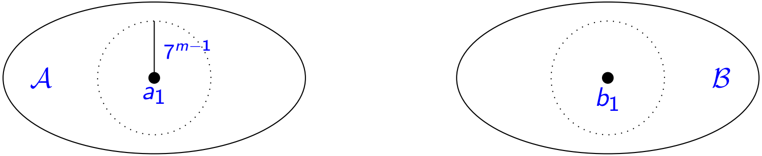
1. $g(j) < g(j+1)$

Beweis Lemma A

Basisfall $i = 1$. Angenommen, **H** zieht $a_1 \in A$.

Invariante $(\star)$. Nach Runde i:
$$\equiv_{g(m-i)} \begin{array}{l}(\mathcal{A}_{7^{m-i}}(\overline{a}), \overline{a})\\ (\mathcal{B}_{7^{m-i}}(\overline{b}), \overline{b}).\end{array}$$

Funktion g.
$g(0) := 1 \quad M(m) = g(m)$
1. $g(j) < g(j+1)$

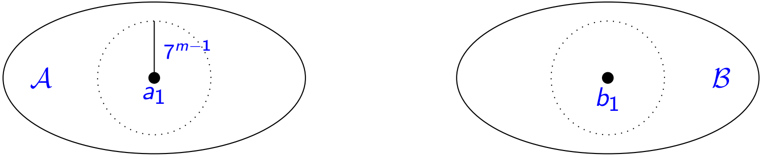


Beweis Lemma A

Basisfall $i = 1$. Angenommen, **H** zieht $a_1 \in A$.

Dann gilt $\mathcal{A} \models \psi_m := \exists x_1 \varphi^{g(m-1),7^{m-1}}_{\mathcal{A},a_1}(x_1)$.

($\varphi^{g(m-1),7^{m-1}}_{\mathcal{A},a_1}$: $g(m)$-Hintikka-Formel relativiert auf 7^{m-1}-Umgebung von x_1.)



Invariante $(\star)$. Nach Runde i:

$$\left(\mathcal{A}_{7^{m-i}}(\bar{a}), \bar{a}\right) \equiv_{g(m-i)} \left(\mathcal{B}_{7^{m-i}}(\bar{b}), \bar{b}\right).$$

Funktion g.

$g(0) := 1 \quad M(m) = g(m)$

1. $g(j) < g(j+1)$
2. $g(m) > \mathrm{qr}(\psi_m)$

Beweis Lemma A

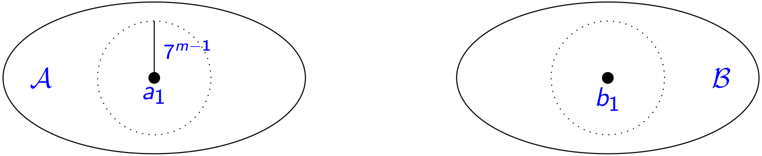
Basisfall $i = 1$. Angenommen, **H** zieht $a_1 \in A$.

Dann gilt $\mathcal{A} \models \psi_m := \exists x_1 \varphi^{g(m-1),7^{m-1}}_{\mathcal{A},a_1}(x_1)$.

($\varphi^{g(m-1),7^{m-1}}_{\mathcal{A},a_1}$: $g(m)$-Hintikka-Formel relativiert auf 7^{m-1}-Umgebung von x_1.)

Da ψ_m basis-lokal und $g(m) > \mathrm{qr}(\psi_m)$, folgt $\mathcal{B} \models \psi_m$.

Also existiert $b_1 \in B$, so dass $\mathcal{A}_{7^{m-1}}(a_1) \equiv_{g(m-1)} \mathcal{B}_{7^{m-1}}(b_1)$.



Invariante $(\star)$. Nach Runde i:

$$\begin{array}{l}(\mathcal{A}_{7^{m-i}}(\bar a), \bar a) \\ \equiv_{g(m-i)} (\mathcal{B}_{7^{m-i}}(\bar b), \bar b).\end{array}$$

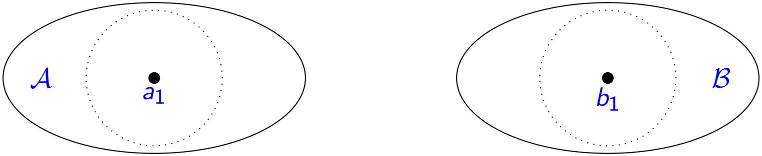
Funktion g.

$g(0) := 1 \quad M(m) = g(m)$

1. $g(j) < g(j+1)$
2. $g(m) > \mathrm{qr}(\psi_m)$

Beweis Lemma A

Induktionsschritt $i \to i+1$. Seien $a_1, \dots, a_i, b_1, \dots, b_i$ schon gezogen.
O.B.d.A. zieht **H** $a \in A$. Gesucht: $b \in B$, so dass $(\star)$ gilt.



Invariante $(\star)$. Nach Runde i:
$\equiv_{g(m-i)}$ $(\mathcal{A}_{7^{m-i}}(\overline{a}), \overline{a})$ $(\mathcal{B}_{7^{m-i}}(\overline{b}), \overline{b})$.

Funktion g.
$g(0) := 1$ $M(m) = g(m)$
1. $g(j) < g(j+1)$
2. $g(m) > \mathrm{qr}(\psi_m)$

Beweis Lemma A

Induktionsschritt $i \to i+1$. Seien $a_1, \dots, a_i, b_1, \dots, b_i$ schon gezogen.
O.B.d.A. zieht **H** $a \in A$. Gesucht: $b \in B$, so dass $(\star)$ gilt.

Fall 1. $a \in N_{2 \cdot 7^{m-(i+1)}}(\overline{a})$.

Invariante $(\star)$. Nach Runde i:
$$\equiv_{g(m-i)} \begin{array}{l}(\mathcal{A}_{7^{m-i}}(\overline{a}), \overline{a})\\(\mathcal{B}_{7^{m-i}}(\overline{b}), \overline{b}).\end{array}$$

Funktion g.
$g(0) := 1 \quad M(m) = g(m)$
1. $g(j) < g(j+1)$
2. $g(m) > \mathrm{qr}(\psi_m)$

Beweis Lemma A

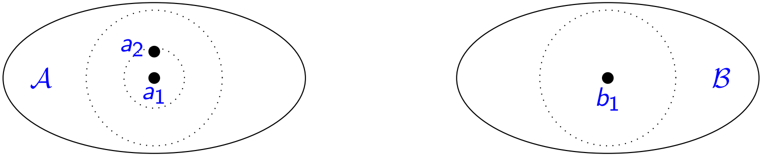
Invariante $(\star)$. Nach Runde i:
$\equiv_{g(m-i)} \begin{matrix}(\mathcal{A}_{7^{m-i}}(\overline{a}), \overline{a}) \\ (\mathcal{B}_{7^{m-i}}(\overline{b}), \overline{b}).\end{matrix}$

Funktion g.
$g(0) := 1 \quad M(m) = g(m)$
1. $g(j) < g(j+1)$
2. $g(m) > \mathrm{qr}(\psi_m)$

Induktionsschritt $i \to i+1$. Seien $a_1, \ldots, a_i, b_1, \ldots, b_i$ schon gezogen.
O.B.d.A. zieht **H** $a \in A$. Gesucht: $b \in B$, so dass $(\star)$ gilt.

Fall 1. $a \in N_{2\cdot 7^{m-(i+1)}}(\overline{a})$. Dann gilt $\mathcal{A}_{7^{m-i}}(\overline{a}) \models \psi_i^1[\overline{a}]$ mit

$$\psi_i^1 := \exists z\ \mathrm{dist}(\overline{a}, z) \leq 2 \cdot 7^{m-(i+1)} \wedge \varphi^{g(m-(i+1)),7^{m-(i+1)}}_{\mathcal{A}_{7^{m-(i+1)}}(\overline{a},a),\overline{a},a}(z)$$

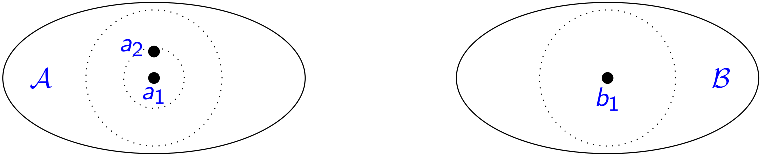


Beweis Lemma A

Induktionsschritt $i \to i+1$. Seien $a_1, \dots, a_i, b_1, \dots, b_i$ schon gezogen.
O.B.d.A. zieht **H** $a \in A$. Gesucht: $b \in B$, so dass $(\star)$ gilt.

Fall 1. $a \in N_{2 \cdot 7^{m-(i+1)}}(\bar{a})$. Dann gilt $\mathcal{A}_{7^{m-i}}(\bar{a}) \models \psi_i^1[\bar{a}]$ mit

$$\psi_i^1 := \exists z\, \mathrm{dist}(\bar{a}, z) \leq 2 \cdot 7^{m-(i+1)} \wedge \varphi^{g(m-(i+1)),7^{m-(i+1)}}_{\mathcal{A}_{7^{m-(i+1)}}(\bar{a},a),\bar{a},a}(z)$$



Invariante $(\star)$. Nach Runde i:
$\equiv_{g(m-i)}$ $(\mathcal{A}_{7^{m-i}}(\bar{a}), \bar{a})$
$(\mathcal{B}_{7^{m-i}}(\bar{b}), \bar{b})$.

Funktion g.
$g(0) := 1 \quad M(m) = g(m)$
1. $g(j) < g(j+1)$
2. $g(m) > \mathrm{qr}(\psi_m)$
3. $g(m-i) > \mathrm{qr}(\psi_i^1)$.

Beweis Lemma A

Induktionsschritt $i \to i+1$. Seien $a_1, \dots, a_i, b_1, \dots, b_i$ schon gezogen.
O.B.d.A. zieht **H** $a \in A$. Gesucht: $b \in B$, so dass $(\star)$ gilt.

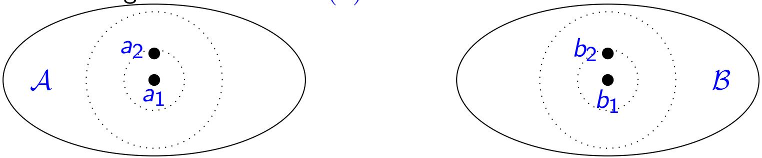
Fall 1. $a \in N_{2\cdot 7^{m-(i+1)}}(\overline{a})$. Dann gilt $\mathcal{A}_{7^{m-i}}(\overline{a}) \models \psi_i^1[\overline{a}]$ mit

$$\psi_i^1 := \exists z \ \mathrm{dist}(\overline{a}, z) \leq 2 \cdot 7^{m-(i+1)} \wedge \varphi^{g(m-(i+1)), 7^{m-(i+1)}}_{\mathcal{A}_{7^{m-(i+1)}}(\overline{a},a), \overline{a}, a}(z)$$

Nach $(\star)$ folgt $\mathcal{B}_{7^{m-i}}(\overline{b}) \models \psi_i^1$ und somit gibt es ein $b \in B$ mit

$$\mathcal{B}_{7^{m-(i+1)}}(\overline{b}, b) \models \varphi^{g(m-(i+1)), 7^{m-(i+1)}}_{\mathcal{A}_{7^{m-(i+1)}}(\overline{a},a), \overline{a}, a}[\overline{b}, b]$$

und daher gilt die Invariante $(\star)$.



Invariante $(\star)$. Nach Runde i:
$$\equiv_{g(m-i)} \begin{matrix} (\mathcal{A}_{7^{m-i}}(\overline{a}), \overline{a}) \\ (\mathcal{B}_{7^{m-i}}(\overline{b}), \overline{b}). \end{matrix}$$

Funktion g.
$g(0) := 1 \quad M(m) = g(m)$
1. $g(j) < g(j+1)$
2. $g(m) > \mathrm{qr}(\psi_m)$
3. $g(m-i) > \mathrm{qr}(\psi_i^1)$.

Beweis Lemma A

Fall 2. $a \notin N_{2\cdot 7^{m-(i+1)}}(\overline{a})$.

Invariante $(\star)$. Nach Runde i:
$$(\mathcal{A}_{7^{m-i}}(\overline{a}), \overline{a}) \equiv_{g(m-i)} (\mathcal{B}_{7^{m-i}}(\overline{b}), \overline{b}).$$

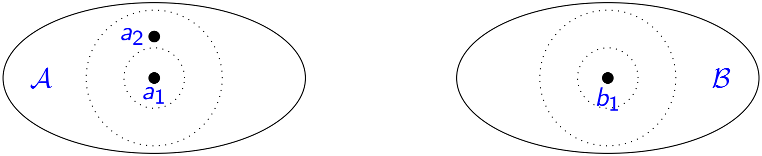
Funktion g.
$g(0) := 1 \quad M(m) = g(m)$
1. $g(j) < g(j+1)$
2. $g(m) > \mathrm{qr}(\psi_m)$
3. $g(m-i) > \mathrm{qr}(\psi_i^1)$.

Beweis Lemma A

Fall 2. $a \notin N_{2\cdot 7^{m-(i+1)}}(\overline{a})$. D.h. $N_{7^{m-(i+1)}}(\overline{a}) \cap N_{7^{m-(i+1)}}(a) = \emptyset$.

Invariante $(\star)$. Nach Runde i:
$\equiv_{g(m-i)} \begin{matrix}(\mathcal{A}_{7^{m-i}}(\overline{a}), \overline{a}) \\ (\mathcal{B}_{7^{m-i}}(\overline{b}), \overline{b}).\end{matrix}$

Funktion g.
$g(0) := 1 \quad M(m) = g(m)$
1. $g(j) < g(j+1)$
2. $g(m) > \mathrm{qr}(\psi_m)$
3. $g(m-i) > \mathrm{qr}(\psi_i^1)$.



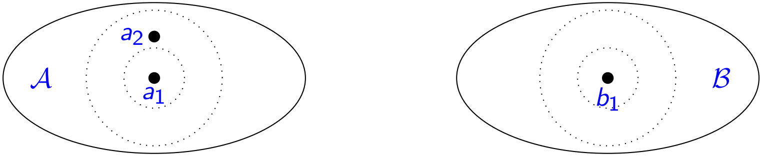
Beweis Lemma A

Fall 2. $a \notin N_{2\cdot 7^{m-(i+1)}}(\overline{a})$. D.h. $N_{7^{m-(i+1)}}(\overline{a}) \cap N_{7^{m-(i+1)}}(a) = \emptyset$.

Für $s \geq 1$ sei $\delta_s^{m-(i+1)}(x_1, \dots, x_s) :=$

$$\bigwedge_{1 \leq l < k \leq s} \mathrm{dist}(x_l, x_k) > 4 \cdot 7^{m-(i+1)} \wedge \bigwedge_{j=1}^{s} \varphi_{\mathcal{A},a}^{g(m-(i+1)),7^{m-(i+1)}}(x_j).$$

(Es gibt s Elemente mit Abst. $> 4 \cdot 7^{m-(i+1)}$ und $g(m-(i+1))$-äq. $7^{m-(i+1)}$-Umgeb.)



Invariante $(\star)$. Nach Runde i:
$(\mathcal{A}_{7^{m-i}}(\overline{a}), \overline{a}) \equiv_{g(m-i)} (\mathcal{B}_{7^{m-i}}(\overline{b}), \overline{b})$.

Funktion g.
$g(0) := 1 \quad M(m) = g(m)$
1. $g(j) < g(j+1)$
2. $g(m) > \mathrm{qr}(\psi_m)$
3. $g(m-i) > \mathrm{qr}(\psi_i^1)$.

Beweis Lemma A

Fall 2. $a \notin N_{2 \cdot 7^{m-(i+1)}}(\bar{a})$. D.h. $N_{7^{m-(i+1)}}(\bar{a}) \cap N_{7^{m-(i+1)}}(a) = \emptyset$.

Für $s \geq 1$ sei $\delta_s^{m-(i+1)}(x_1, \dots, x_s) :=$

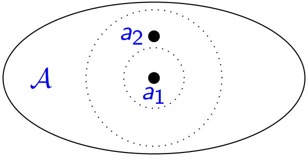
$$\bigwedge_{1 \leq l < k \leq s} \mathrm{dist}(x_l, x_k) > 4 \cdot 7^{m-(i+1)} \wedge \bigwedge_{j=1}^{s} \varphi_{\mathcal{A},a}^{g(m-(i+1)),7^{m-(i+1)}}(x_j).$$

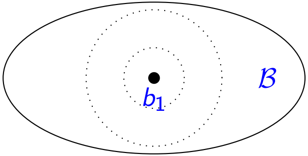
Sei $e_{\mathcal{A}}$ so, dass $\mathcal{A}_{7^{m-i}}(\bar{a}) \models \psi_i^2$ mit

$$\psi_i^2 := \exists x_1 \dots \exists x_{e_{\mathcal{A}}} \delta_{e_{\mathcal{A}}}^{m-(i+1)} \wedge \bigwedge_{j=1}^{e_{\mathcal{A}}} \mathrm{dist}(\bar{a}, x_l) \leq 2 \cdot 7^{m-(i+1)}$$

jedoch $\mathcal{A}_{7^{m-i}}(\bar{a}) \not\models \psi_i^3$ mit

$$\psi_i^3 := \exists x_1 \dots \exists x_{e_{\mathcal{A}}+1} \delta_{e_{\mathcal{A}}+1}^{m-(i+1)} \wedge \bigwedge_{j=1}^{e_{\mathcal{A}}+1} \mathrm{dist}(\bar{a}, x_l) \leq 2 \cdot 7^{m-(i+1)}.$$





Invariante $(\star)$. Nach Runde i:

$$\equiv_{g(m-i)} \begin{array}{l}(\mathcal{A}_{7^{m-i}}(\bar{a}), \bar{a}) \\ (\mathcal{B}_{7^{m-i}}(\bar{b}), \bar{b}).\end{array}$$

Funktion g.

$g(0) := 1 \quad M(m) = g(m)$

1. $g(j) < g(j+1)$
2. $g(m) > \mathrm{qr}(\psi_m)$
3. $g(m-i) > \mathrm{qr}(\psi_i^1)$.

Beweis Lemma A

Fall 2. $a \notin N_{2\cdot 7^{m-(i+1)}}(\bar{a})$. D.h. $N_{7^{m-(i+1)}}(\bar{a}) \cap N_{7^{m-(i+1)}}(a) = \emptyset$.

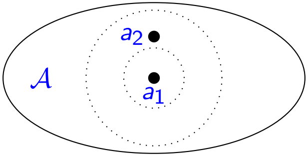
Sei $e_{\mathcal{A}}$ so, dass $\mathcal{A}_{7^{m-i}}(\bar{a}) \models \psi_i^2$ mit

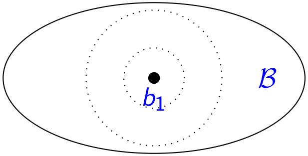
$$\psi_i^2 := \exists x_1 \dots \exists x_{e_{\mathcal{A}}} \delta_{e_{\mathcal{A}}}^{m-(i+1)} \wedge \bigwedge_{j=1}^{e_{\mathcal{A}}} \mathrm{dist}(\bar{a}, x_l) \leq 2 \cdot 7^{m-(i+1)}$$

jedoch $\mathcal{A}_{7^{m-i}}(\bar{a}) \not\models \psi_i^3$ mit

$$\psi_i^3 := \exists x_1 \dots \exists x_{e_{\mathcal{A}}+1} \delta_{e_{\mathcal{A}}+1}^{m-(i+1)} \wedge \bigwedge_{j=1}^{e_{\mathcal{A}}+1} \mathrm{dist}(\bar{a}, x_l) \leq 2 \cdot 7^{m-(i+1)}.$$

Da es in $N_{2\cdot 7^{m-(i+1)}}(a_i)$ keine zwei Elemente mit Abstand $> 4 \cdot 7^{m-(i+1)}$ geben kann, folgt $e_{\mathcal{A}} \leq |\bar{a}| = i \leq m$.





Invariante $(\star)$. Nach Runde i:
$$(\mathcal{A}_{7^{m-i}}(\bar{a}), \bar{a}) \equiv_{g(m-i)} (\mathcal{B}_{7^{m-i}}(\bar{b}), \bar{b}).$$

Funktion g.
$g(0) := 1 \quad M(m) = g(m)$

1. $g(j) < g(j+1)$
2. $g(m) > \mathrm{qr}(\psi_m)$
3. $g(m-i) > \mathrm{qr}(\psi_i^1)$.

Beweis Lemma A

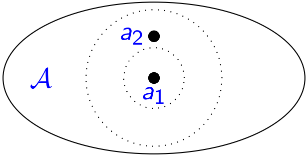
Fall 2. $a \notin N_{2 \cdot 7^{m-(i+1)}}(\bar{a})$. D.h. $N_{7^{m-(i+1)}}(\bar{a}) \cap N_{7^{m-(i+1)}}(a) = \emptyset$.

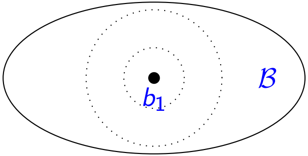
Sei $e_{\mathcal{A}}$ so, dass $\mathcal{A}_{7^{m-i}}(\bar{a}) \models \psi_i^2$ mit

$$\psi_i^2 := \exists x_1 \dots \exists x_{e_{\mathcal{A}}} \delta_{e_{\mathcal{A}}}^{m-(i+1)} \wedge \bigwedge_{j=1}^{e_{\mathcal{A}}} \mathrm{dist}(\bar{a}, x_l) \leq 2 \cdot 7^{m-(i+1)}$$

jedoch $\mathcal{A}_{7^{m-i}}(\bar{a}) \not\models \psi_i^3$ mit

$$\psi_i^3 := \exists x_1 \dots \exists x_{e_{\mathcal{A}}+1} \delta_{e_{\mathcal{A}}+1}^{m-(i+1)} \wedge \bigwedge_{j=1}^{e_{\mathcal{A}}+1} \mathrm{dist}(\bar{a}, x_l) \leq 2 \cdot 7^{m-(i+1)}.$$





Invariante $(\star)$. Nach Runde i:

$$\equiv_{g(m-i)} \begin{array}{l}(\mathcal{A}_{7^{m-i}}(\bar{a}), \bar{a}) \\ (\mathcal{B}_{7^{m-i}}(\bar{b}), \bar{b}).\end{array}$$

Funktion g.

$g(0) := 1 \quad M(m) = g(m)$

1. $g(j) < g(j+1)$
2. $g(m) > \mathrm{qr}(\psi_m)$
3. $g(m-i) > \mathrm{qr}(\psi_i^1)$.
4. $g(m-i) > \mathrm{qr}(\psi_i^2), \mathrm{qr}(\psi_i^3)$.

Beweis Lemma A

Fall 2. $a \notin N_{2\cdot 7^{m-(i+1)}}(\bar{a})$. D.h. $N_{7^{m-(i+1)}}(\bar{a}) \cap N_{7^{m-(i+1)}}(a) = \emptyset$.

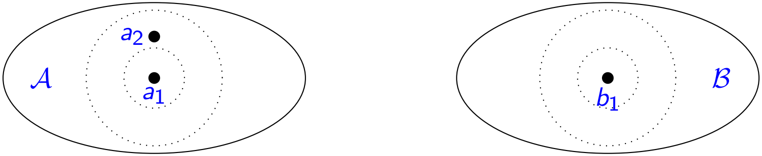
Sei $e_{\mathcal{A}}$ so, dass $\mathcal{A}_{7^{m-i}}(\bar{a}) \models \psi_i^2$ mit

$$\psi_i^2 := \exists x_1 \dots \exists x_{e_{\mathcal{A}}} \delta_{e_{\mathcal{A}}}^{m-(i+1)} \wedge \bigwedge_{j=1}^{e_{\mathcal{A}}} \mathrm{dist}(\bar{a}, x_l) \leq 2 \cdot 7^{m-(i+1)}$$

jedoch $\mathcal{A}_{7^{m-i}}(\bar{a}) \not\models \psi_i^3$ mit

$$\psi_i^3 := \exists x_1 \dots \exists x_{e_{\mathcal{A}}+1} \delta_{e_{\mathcal{A}}+1}^{m-(i+1)} \wedge \bigwedge_{j=1}^{e_{\mathcal{A}}+1} \mathrm{dist}(\bar{a}, x_l) \leq 2 \cdot 7^{m-(i+1)}.$$

Wir definieren $e_{\mathcal{B}}$ analog in $\mathcal{B}$.



Invariante $(\star)$. Nach Runde i:

$$\equiv_{g(m-i)} \begin{array}{l}(\mathcal{A}_{7^{m-i}}(\bar{a}), \bar{a}) \\ (\mathcal{B}_{7^{m-i}}(\bar{b}), \bar{b}).\end{array}$$

Funktion g.

$g(0) := 1 \quad M(m) = g(m)$

1. $g(j) < g(j+1)$
2. $g(m) > \mathrm{qr}(\psi_m)$
3. $g(m-i) > \mathrm{qr}(\psi_i^1)$.
4. $g(m-i) > \mathrm{qr}(\psi_i^2), \mathrm{qr}(\psi_i^3)$.

Beweis Lemma A

Fall 2. $a \notin N_{2\cdot 7^{m-(i+1)}}(\bar{a})$. D.h. $N_{7^{m-(i+1)}}(\bar{a}) \cap N_{7^{m-(i+1)}}(a) = \emptyset$.

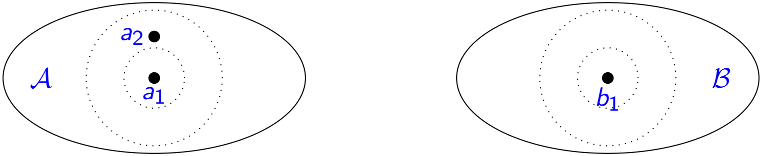
Sei $e_{\mathcal{A}}$ so, dass $\mathcal{A}_{7^{m-i}}(\bar{a}) \models \psi_i^2$ mit

$$\psi_i^2 := \exists x_1 \ldots \exists x_{e_{\mathcal{A}}} \delta_{e_{\mathcal{A}}}^{m-(i+1)} \wedge \bigwedge_{j=1}^{e_{\mathcal{A}}} \mathrm{dist}(\bar{a}, x_l) \leq 2 \cdot 7^{m-(i+1)}$$

jedoch $\mathcal{A}_{7^{m-i}}(\bar{a}) \not\models \psi_i^3$ mit

$$\psi_i^3 := \exists x_1 \ldots \exists x_{e_{\mathcal{A}}+1} \delta_{e_{\mathcal{A}}+1}^{m-(i+1)} \wedge \bigwedge_{j=1}^{e_{\mathcal{A}}+1} \mathrm{dist}(\bar{a}, x_l) \leq 2 \cdot 7^{m-(i+1)}.$$

Wir definieren $e_{\mathcal{B}}$ analog in $\mathcal{B}$. Wegen $(\star)$ folgt $e_{\mathcal{A}} = e_{\mathcal{B}} =: e$.



Invariante $(\star)$. Nach Runde i:

$\equiv_{g(m-i)}$ $(\mathcal{A}_{7^{m-i}}(\bar{a}), \bar{a})$
$(\mathcal{B}_{7^{m-i}}(\bar{b}), \bar{b})$.

Funktion g.

$g(0) := 1 \quad M(m) = g(m)$

1. $g(j) < g(j+1)$
2. $g(m) > \mathrm{qr}(\psi_m)$
3. $g(m-i) > \mathrm{qr}(\psi_i^1)$.
4. $g(m-i) > \mathrm{qr}(\psi_i^2), \mathrm{qr}(\psi_i^3)$.

Beweis Lemma A

Fall 2. $a \notin N_{2\cdot 7^{m-(i+1)}}(\bar{a})$. D.h. $N_{7^{m-(i+1)}}(\bar{a}) \cap N_{7^{m-(i+1)}}(a) = \emptyset$.

e: Anzahl $a' \in N_{2\cdot 7^{m-(i+1)}}(\bar{a})$ mit paar. Abst. $> 4 \cdot 7^{m-(i+1)}$ und

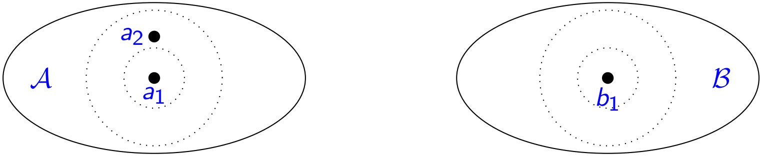
$$N_{7^{m-(i+1)}}(a') \equiv_{g(m-(i+1))} N_{7^{m-(i+1)}}(a).$$

Offensichtlich gilt $\mathcal{A} \models \psi_i^4 := \exists x_1 \dots \exists x_e \delta_e^{m-(i+1)}$

Falls auch noch $\mathcal{A} \models \psi_i^5 := \exists x_1 \dots \exists x_e \exists x_{e+1} \delta_{e+1}^{m-(i+1)}$ gilt,

so definieren wir $e'_{\mathcal{A}} := e + 1$., anderenfalls $e'_{\mathcal{A}} := e$.

Wiederum definieren wir $e'_{\mathcal{B}}$ analog.



Invariante $(\star)$. Nach Runde i:

$$\equiv_{g(m-i)} \begin{array}{l}(\mathcal{A}_{7^{m-i}}(\bar{a}), \bar{a})\\ (\mathcal{B}_{7^{m-i}}(\bar{b}), \bar{b}).\end{array}$$

Funktion g.

$g(0) := 1 \quad M(m) = g(m)$

1. $g(j) < g(j+1)$
2. $g(m) > \mathrm{qr}(\psi_m)$
3. $g(m-i) > \mathrm{qr}(\psi_i^1)$.
4. $g(m-i) > \mathrm{qr}(\psi_i^2), \mathrm{qr}(\psi_i^3)$.

Beweis Lemma A

Fall 2. $a \notin N_{2\cdot 7^{m-(i+1)}}(\bar{a})$. D.h. $N_{7^{m-(i+1)}}(\bar{a}) \cap N_{7^{m-(i+1)}}(a) = \emptyset$.

e: Anzahl $a' \in N_{2\cdot 7^{m-(i+1)}}(\bar{a})$ mit paar. Abst. $> 4 \cdot 7^{m-(i+1)}$ und

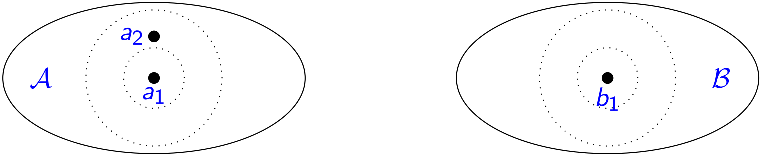
$$N_{7^{m-(i+1)}}(a') \equiv_{g(m-(i+1))} N_{7^{m-(i+1)}}(a).$$

Offensichtlich gilt $\mathcal{A} \models \psi_i^4 := \exists x_1 \dots \exists x_e \delta_e^{m-(i+1)}$

Falls auch noch $\mathcal{A} \models \psi_i^5 := \exists x_1 \dots \exists x_e \exists x_{e+1} \delta_{e+1}^{m-(i+1)}$ gilt,

so definieren wir $e'_{\mathcal{A}} := e + 1.$, anderenfalls $e'_{\mathcal{A}} := e$.

Wiederum definieren wir $e'_{\mathcal{B}}$ analog.



Invariante $(\star)$. Nach Runde i:

$\equiv_{g(m-i)}$ $(\mathcal{A}_{7^{m-i}}(\bar{a}), \bar{a})$
$(\mathcal{B}_{7^{m-i}}(\bar{b}), \bar{b})$.

Funktion g.

$g(0) := 1 \quad M(m) = g(m)$

1. $g(j) < g(j+1)$
2. $g(m) > \mathrm{qr}(\psi_m)$
3. $g(m-i) > \mathrm{qr}(\psi_i^1)$.
4. $g(m-i) > \mathrm{qr}(\psi_i^2), \mathrm{qr}(\psi_i^3)$.
5. $g(m-i) > \mathrm{qr}(\psi_i^4), \mathrm{qr}(\psi_i^5)$.

Beweis Lemma A

Fall 2. $a \notin N_{2\cdot 7^{m-(i+1)}}(\overline{a})$. D.h. $N_{7^{m-(i+1)}}(\overline{a}) \cap N_{7^{m-(i+1)}}(a) = \emptyset$.

e: Anzahl $a' \in N_{2\cdot 7^{m-(i+1)}}(\overline{a})$ mit paar. Abst. $> 4 \cdot 7^{m-(i+1)}$ und

$$N_{7^{m-(i+1)}}(a') \equiv_{g(m-(i+1))} N_{7^{m-(i+1)}}(a).$$

Offensichtlich gilt $\mathcal{A} \models \psi_i^4 := \exists x_1 \dots \exists x_e \delta_e^{m-(i+1)}$

Falls auch noch $\mathcal{A} \models \psi_i^5 := \exists x_1 \dots \exists x_e \exists x_{e+1} \delta_{e+1}^{m-(i+1)}$ gilt,

so definieren wir $e'_{\mathcal{A}} := e + 1.$, anderenfalls $e'_{\mathcal{A}} := e$.

Wiederum definieren wir $e'_{\mathcal{B}}$ analog.

Da ψ_i^4, ψ_i^5 basis-lokal sind, gilt $e'_{\mathcal{A}} = e'_{\mathcal{B}} =: e'$.

Invariante $(\star)$. Nach Runde i:

$$\equiv_{g(m-i)} \begin{array}{l}(\mathcal{A}_{7^{m-i}}(\overline{a}), \overline{a}) \\ (\mathcal{B}_{7^{m-i}}(\overline{b}), \overline{b}).\end{array}$$

Funktion g.

$g(0) := 1 \quad M(m) = g(m)$

1. $g(j) < g(j+1)$
2. $g(m) > \mathrm{qr}(\psi_m)$
3. $g(m-i) > \mathrm{qr}(\psi_i^1)$.
4. $g(m-i) > \mathrm{qr}(\psi_i^2), \mathrm{qr}(\psi_i^3)$.
5. $g(m-i) > \mathrm{qr}(\psi_i^4), \mathrm{qr}(\psi_i^5)$.

Beweis Lemma A

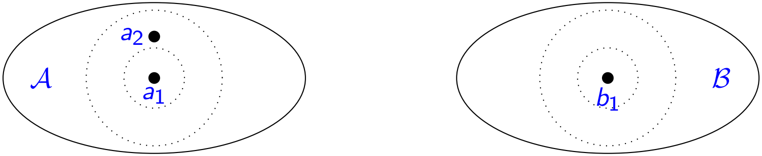
Fall 2. $a \notin N_{2\cdot 7^{m-(i+1)}}(\bar{a})$. D.h. $N_{7^{m-(i+1)}}(\bar{a}) \cap N_{7^{m-(i+1)}}(a) = \emptyset$.

e: Anzahl $a' \in N_{2\cdot 7^{m-(i+1)}}(\bar{a})$ mit paar. Abst. $> 4 \cdot 7^{m-(i+1)}$ und

$$N_{7^{m-(i+1)}}(a') \equiv_{g(m-(i+1))} N_{7^{m-(i+1)}}(a).$$

e': Anzahl a' mit Abst. $> 4 \cdot 7^{m-(i+1)}$ und

$$N_{7^{m-(i+1)}}(a') \equiv_{g(m-(i+1))} N_{7^{m-(i+1)}}(a).$$



Invariante $(\star)$. Nach Runde i:

$$\equiv_{g(m-i)} \begin{matrix} (\mathcal{A}_{7^{m-i}}(\bar{a}), \bar{a}) \\ (\mathcal{B}_{7^{m-i}}(\bar{b}), \bar{b}). \end{matrix}$$

Funktion g.

$g(0) := 1 \quad M(m) = g(m)$

1. $g(j) < g(j+1)$
2. $g(m) > \mathrm{qr}(\psi_m)$
3. $g(m-i) > \mathrm{qr}(\psi_i^1)$.
4. $g(m-i) > \mathrm{qr}(\psi_i^2), \mathrm{qr}(\psi_i^3)$.
5. $g(m-i) > \mathrm{qr}(\psi_i^4), \mathrm{qr}(\psi_i^5)$.

Beweis Lemma A

Fall 2. $a \notin N_{2\cdot 7^{m-(i+1)}}(\bar{a})$. D.h. $N_{7^{m-(i+1)}}(\bar{a}) \cap N_{7^{m-(i+1)}}(a) = \emptyset$.

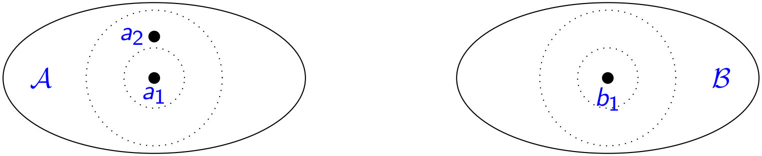
e: Anzahl $a' \in N_{2\cdot 7^{m-(i+1)}}(\bar{a})$ mit paar. Abst. $> 4 \cdot 7^{m-(i+1)}$ und

$$N_{7^{m-(i+1)}}(a') \equiv_{g(m-(i+1))} N_{7^{m-(i+1)}}(a).$$

e': Anzahl a' mit Abst. $> 4 \cdot 7^{m-(i+1)}$ und

$$N_{7^{m-(i+1)}}(a') \equiv_{g(m-(i+1))} N_{7^{m-(i+1)}}(a).$$

Fall a). $e' = e + 1$. Dann gilt $\mathcal{B} \models \psi_i^5$ und es ex. $b \in B$ mit $\mathcal{B} \models \varphi_a^{g(m-(i+1)),7^{m-(i+1)}}[b]$, so dass $N_{7^{m-(i+1)}}(\bar{b}) \cap N_{7^{m-(i+1)}}(b) = \emptyset$.



Invariante $(\star)$. Nach Runde i:

$$\equiv_{g(m-i)} \begin{array}{l}(\mathcal{A}_{7^{m-i}}(\bar{a}), \bar{a})\\ (\mathcal{B}_{7^{m-i}}(\bar{b}), \bar{b}).\end{array}$$

Funktion g.

$g(0) := 1 \quad M(m) = g(m)$

1. $g(j) < g(j+1)$
2. $g(m) > \mathrm{qr}(\psi_m)$
3. $g(m-i) > \mathrm{qr}(\psi_i^1)$.
4. $g(m-i) > \mathrm{qr}(\psi_i^2), \mathrm{qr}(\psi_i^3)$.
5. $g(m-i) > \mathrm{qr}(\psi_i^4), \mathrm{qr}(\psi_i^5)$.

Beweis Lemma A

Fall 2. $a \notin N_{2 \cdot 7^{m-(i+1)}}(\bar{a})$. D.h. $N_{7^{m-(i+1)}}(\bar{a}) \cap N_{7^{m-(i+1)}}(a) = \emptyset$.

e: Anzahl $a' \in N_{2 \cdot 7^{m-(i+1)}}(\bar{a})$ mit paar. Abst. $> 4 \cdot 7^{m-(i+1)}$ und

$$N_{7^{m-(i+1)}}(a') \equiv_{g(m-(i+1))} N_{7^{m-(i+1)}}(a).$$

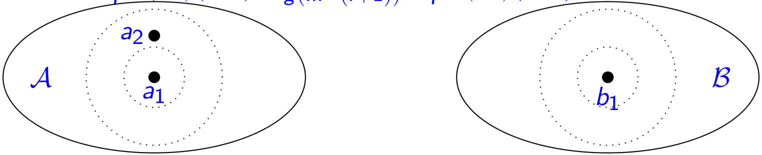
e': Anzahl a' mit Abst. $> 4 \cdot 7^{m-(i+1)}$ und

$$N_{7^{m-(i+1)}}(a') \equiv_{g(m-(i+1))} N_{7^{m-(i+1)}}(a).$$

Fall a). $e' = e + 1$. Dann gilt $\mathcal{B} \models \psi_i^5$ und es ex. $b \in B$ mit $\mathcal{B} \models \varphi_a^{g(m-(i+1)), 7^{m-(i+1)}}[b]$, so dass $N_{7^{m-(i+1)}}(\bar{b}) \cap N_{7^{m-(i+1)}}(b) = \emptyset$.

Es gilt also $\mathcal{A}_{7^{m-(i+1)}}(a) \equiv_{g(m-(i+1))} \mathcal{B}_{7^{m-(i+1)}}(b)$ und somit

$$\mathcal{A}_{7^{m-(i+1)}}(\bar{a}, a) \equiv_{g(m-(i+1))} \mathcal{B}_{7^{m-(i+1)}}(\bar{b}, b).$$



Invariante $(\star)$. Nach Runde i:

$$(\mathcal{A}_{7^{m-i}}(\bar{a}), \bar{a}) \equiv_{g(m-i)} (\mathcal{B}_{7^{m-i}}(\bar{b}), \bar{b}).$$

Funktion g.

$g(0) := 1 \quad M(m) = g(m)$

1. $g(j) < g(j+1)$
2. $g(m) > \mathrm{qr}(\psi_m)$
3. $g(m-i) > \mathrm{qr}(\psi_i^1)$.
4. $g(m-i) > \mathrm{qr}(\psi_i^2), \mathrm{qr}(\psi_i^3)$.
5. $g(m-i) > \mathrm{qr}(\psi_i^4), \mathrm{qr}(\psi_i^5)$.

Beweis Lemma A

Fall 2. $a \notin N_{2\cdot 7^{m-(i+1)}}(\bar{a})$. D.h. $N_{7^{m-(i+1)}}(\bar{a}) \cap N_{7^{m-(i+1)}}(a) = \emptyset$.

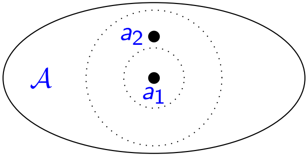
e: Anzahl $a' \in N_{2\cdot 7^{m-(i+1)}}(\bar{a})$ mit paar. Abst. $> 4 \cdot 7^{m-(i+1)}$ und

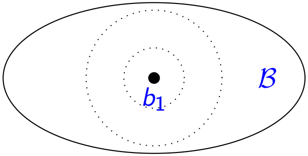
$$N_{7^{m-(i+1)}}(a') \equiv_{g(m-(i+1))} N_{7^{m-(i+1)}}(a).$$

e': Anzahl a' mit Abst. $> 4 \cdot 7^{m-(i+1)}$ und

$$N_{7^{m-(i+1)}}(a') \equiv_{g(m-(i+1))} N_{7^{m-(i+1)}}(a).$$

Fall b). $e = e'$ D.h. alle $a' \in A$, die $\varphi_{\mathcal{A},a}^{g(m-(i+1)),7^{m-(i+1)}}$ erfüllen, haben Abstand $\leq 6 \cdot 7^{m-(i+1)} < 7^{m-i}$.





Invariante $(\star)$. Nach Runde i:

$$\equiv_{g(m-i)} \begin{matrix} (\mathcal{A}_{7^{m-i}}(\bar{a}), \bar{a}) \\ (\mathcal{B}_{7^{m-i}}(\bar{b}), \bar{b}). \end{matrix}$$

Funktion g.

$g(0) := 1 \quad M(m) = g(m)$

1. $g(j) < g(j+1)$
2. $g(m) > \mathrm{qr}(\psi_m)$
3. $g(m-i) > \mathrm{qr}(\psi_i^1)$.
4. $g(m-i) > \mathrm{qr}(\psi_i^2), \mathrm{qr}(\psi_i^3)$.
5. $g(m-i) > \mathrm{qr}(\psi_i^4), \mathrm{qr}(\psi_i^5)$.

Beweis Lemma A

Fall 2. $a \notin N_{2\cdot 7^{m-(i+1)}}(\overline{a})$. D.h. $N_{7^{m-(i+1)}}(\overline{a}) \cap N_{7^{m-(i+1)}}(a) = \emptyset$.

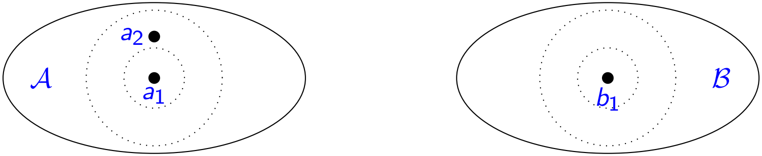
e: Anzahl $a' \in N_{2\cdot 7^{m-(i+1)}}(\overline{a})$ mit paar. Abst. $> 4 \cdot 7^{m-(i+1)}$ und

$$N_{7^{m-(i+1)}}(a') \equiv_{g(m-(i+1))} N_{7^{m-(i+1)}}(a).$$

e': Anzahl a' mit Abst. $> 4 \cdot 7^{m-(i+1)}$ und

$$N_{7^{m-(i+1)}}(a') \equiv_{g(m-(i+1))} N_{7^{m-(i+1)}}(a).$$

Fall b). $e = e'$ D.h. alle $a' \in A$, die $\varphi_{\mathcal{A},a}^{g(m-(i+1)),7^{m-(i+1)}}$ erfüllen, haben Abstand $\leq 6 \cdot 7^{m-(i+1)} < 7^{m-i}$. Denn sonst wären a' und die e Elemente in $N_{2\cdot 7^j}(\overline{a})$ insgesamt $e+1$ Elemente, Widerspruch zu $e = e'$.



Invariante $(\star)$. Nach Runde i:

$$\equiv_{g(m-i)} \begin{matrix}(\mathcal{A}_{7^{m-i}}(\overline{a}), \overline{a}) \\ (\mathcal{B}_{7^{m-i}}(\overline{b}), \overline{b}).\end{matrix}$$

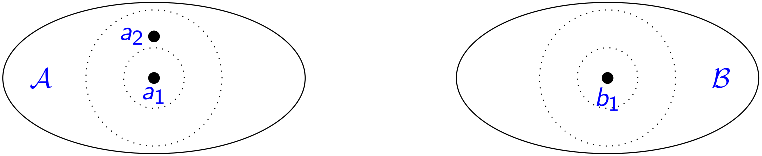
Funktion g.
$g(0) := 1 \quad M(m) = g(m)$
1. $g(j) < g(j+1)$
2. $g(m) > \mathrm{qr}(\psi_m)$
3. $g(m-i) > \mathrm{qr}(\psi_i^1)$.
4. $g(m-i) > \mathrm{qr}(\psi_i^2), \mathrm{qr}(\psi_i^3)$.
5. $g(m-i) > \mathrm{qr}(\psi_i^4), \mathrm{qr}(\psi_i^5)$.

Beweis Lemma A

Fall b). $e = e'$ D.h. alle $a' \in A$, die $\varphi^{g(m-(i+1)),7^{m-(i+1)}}_{\mathcal{A},a}$ erfüllen, haben Abstand $\leq 6 \cdot 7^{m-(i+1)} < 7^{m-i}$.

Es gilt also $\mathcal{A}_{7^{m(i+1)}}(\overline{a}) \models \psi_i^6[\overline{x}/\overline{a}]$ mit

$$\psi_i^6 := \exists z \text{dist}(\overline{x}, z) > 2 \cdot 7^{m-(i+1)} \wedge \text{dist}(\overline{x}, z) \leq 6 \cdot 7^{m-(i+1)} \wedge \varphi^{g(m-(i+1)),7^{m-(i+1)}}_{\mathcal{A},a}(z) \wedge \varphi^{g(m-(i+1)),7^{m-(i+1)}}_{\mathcal{A},\overline{a}}(\overline{x}).$$



Invariante $(\star)$. Nach Runde i:
$(\mathcal{A}_{7^{m-i}}(\overline{a}), \overline{a}) \equiv_{g(m-i)} (\mathcal{B}_{7^{m-i}}(\overline{b}), \overline{b})$.

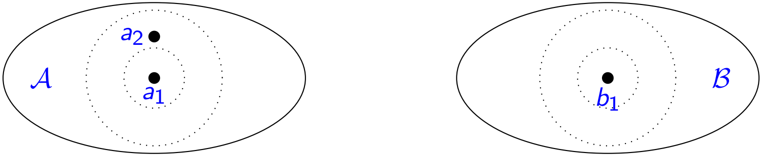
Funktion g.
$g(0) := 1 \quad M(m) = g(m)$
1. $g(j) < g(j+1)$
2. $g(m) > \mathrm{qr}(\psi_m)$
3. $g(m-i) > \mathrm{qr}(\psi_i^1)$.
4. $g(m-i) > \mathrm{qr}(\psi_i^2), \mathrm{qr}(\psi_i^3)$.
5. $g(m-i) > \mathrm{qr}(\psi_i^4), \mathrm{qr}(\psi_i^5)$.

Beweis Lemma A

Fall b). $e = e'$ D.h. alle $a' \in A$, die $\varphi^{g(m-(i+1)),7^{m-(i+1)}}_{\mathcal{A},a}$ erfüllen, haben Abstand $\leq 6 \cdot 7^{m-(i+1)} < 7^{m-i}$.

Es gilt also $\mathcal{A}_{7^{m(i+1)}}(\overline{a}) \models \psi_i^6[\overline{x}/\overline{a}]$ mit

$$\psi_i^6 := \exists z \mathrm{dist}(\overline{x}, z) > 2 \cdot 7^{m-(i+1)} \wedge \mathrm{dist}(\overline{x}, z) \leq 6 \cdot 7^{m-(i+1)} \wedge \varphi^{g(m-(i+1)),7^{m-(i+1)}}_{\mathcal{A},a}(z) \wedge \varphi^{g(m-(i+1)),7^{m-(i+1)}}_{\mathcal{A},\overline{a}}(\overline{x}).$$



Invariante $(\star)$. Nach Runde i:
$(\mathcal{A}_{7^{m-i}}(\overline{a}), \overline{a}) \equiv_{g(m-i)} (\mathcal{B}_{7^{m-i}}(\overline{b}), \overline{b})$.

Funktion g.
$g(0) := 1 \quad M(m) = g(m)$
1. $g(j) < g(j+1)$
2. $g(m) > \mathrm{qr}(\psi_m)$
3. $g(m-i) > \mathrm{qr}(\psi_i^1)$.
4. $g(m-i) > \mathrm{qr}(\psi_i^2), \mathrm{qr}(\psi_i^3)$.
5. $g(m-i) > \mathrm{qr}(\psi_i^4), \mathrm{qr}(\psi_i^5)$.
6. $g(m-i) > \mathrm{qr}(\psi_i^6)$.

Beweis Lemma A

Fall b). $e = e'$ D.h. alle $a' \in A$, die $\varphi^{g(m-(i+1)),7^{m-(i+1)}}_{\mathcal{A},a}$ erfüllen, haben Abstand $\leq 6 \cdot 7^{m-(i+1)} < 7^{m-i}$.

Es gilt also $\mathcal{A}_{7^{m(i+1)}}(\overline{a}) \models \psi_i^6[\overline{x}/\overline{a}]$ mit

$$\psi_i^6 := \exists z \operatorname{dist}(\overline{x}, z) > 2 \cdot 7^{m-(i+1)} \wedge \operatorname{dist}(\overline{x}, z) \leq 6 \cdot 7^{m-(i+1)} \wedge \varphi^{g(m-(i+1)),7^{m-(i+1)}}_{\mathcal{A},a}(z) \wedge \varphi^{g(m-(i+1)),7^{m-(i+1)}}_{\mathcal{A},\overline{a}}(\overline{x}).$$

Daher folgt aus der Voraussetzung $\mathcal{B}_{7^{m(i+1)}}(\overline{b}) \models \psi_i^6[\overline{x}/\overline{b}]$ und somit existiert ein $b \in B$ mit

$$2 \cdot 7^{m-(i+1)} < \operatorname{dist}(\overline{b}, b) \leq 6 \cdot 7^{m-(i+1)}$$

so dass $(\mathcal{A}_{7^{m-(i+1)}}(a), a) \equiv_{g(m-(i+1))} (\mathcal{B}_{7^{m-(i+1)}}(b), b)$.

Also gilt $(\mathcal{A}_{7^{m-(i+1)}}(\overline{a}, a), \overline{a}, a) \equiv_{g(m-(i+1))} (\mathcal{B}_{7^{m-(i+1)}}(\overline{b}, b), \overline{b}, b)$

Invariante $(\star)$. Nach Runde i:
$(\mathcal{A}_{7^{m-i}}(\overline{a}), \overline{a}) \equiv_{g(m-i)} (\mathcal{B}_{7^{m-i}}(\overline{b}), \overline{b})$.

Funktion g.
$g(0) := 1 \quad M(m) = g(m)$
1. $g(j) < g(j+1)$
2. $g(m) > \operatorname{qr}(\psi_m)$
3. $g(m-i) > \operatorname{qr}(\psi_i^1)$.
4. $g(m-i) > \operatorname{qr}(\psi_i^2), \operatorname{qr}(\psi_i^3)$.
5. $g(m-i) > \operatorname{qr}(\psi_i^4), \operatorname{qr}(\psi_i^5)$.
6. $g(m-i) > \operatorname{qr}(\psi_i^6)$.

Abschluss des Beweises des Satzes von Gaifman

Lemma A. Für jedes $m \in \mathbb{N}$ gibt es ein $M = M(m) \in \mathbb{N}$, so dass für alle σ-Strukturen $\mathcal{A}, \mathcal{B}$ gilt: Falls $\mathcal{A}$ und $\mathcal{B}$ dieselben basis-lokalen $\text{FO}[\sigma]$-Sätze vom Quantorenrang höchstens M erfüllen, dann gilt $\mathcal{A} \equiv_m \mathcal{B}$.

Satz. Jeder Satz $\varphi \in FO$ ist äquivalent zu einem Satz in Gaifman-Normalform.

Satz. Dawar, Grohe, K., Schweikardt, 2006 Für jedes $h \in \mathbb{N}$ gibt es einen $\text{FO}[E]$-Satz φ_h der Größe $\mathcal{O}(h^4)$, so dass jeder zu φ_h äquivalente $\text{FO}[E]$-Satz in GNF (Gaifman-Normalform) mindestens die Größe $\text{tow}(h)$ hat, wobei

$$\text{tow}(0) := 1 \quad \text{und} \quad \text{tow}(n+1) := 2^{\text{tow}(n)} \quad \text{für alle } n \in \mathbb{N}.$$

Beispiel für die Gaifman-Normalform

Beispiel Gaifman-Normalform

Dominierende Mengen.

$$\varphi_k := \exists x_1 \dots \exists x_k \forall y \bigvee_i (x_i = y \vee Ex_i y)$$

φ_k gilt in einem Graph G genau dann, wenn G eine dominierende Menge der Größe $\leq k$ enthält.

In zusammenhängenden Graphen, dies ist äquivalent zu folgendem Satz in Gaifman-Normalform

$$\neg\Big(\underbrace{\exists x_1 \exists x_2 \mathrm{dist}(x_1, x_2) > 3k}_{\textit{Durchmesser groß, d.h. keine dom. Menge exist.}} \Big) \quad \wedge \quad \underbrace{\exists x \varphi_{3k-local-DS}(x)}_{\textit{Duchmesser} \leq 3k}$$

wobei $\varphi_{3k-local-DS}(x)$ wie folgt definiert ist:

$$\varphi_{3k-local-DS}(x) := \exists x_1 \dots \exists x_k \bigwedge_i \mathrm{dist}(x, x_i) \leq 3k \wedge \forall y \bigvee_i (x_i = y \vee Ex_i y)$$

Anwendungen des Satzes von Gaifman

FO-Model-Checking auf Graphen beschränkten Grads

Theorem. (Seese, 1996)

Sei $\mathcal{C}$ eine Klasse von Graphen mit Maximalgrad $d \geq 1$.

MC(FO, $\mathcal{C}$)	
Eingabe:	Graph $G \in \mathcal{C}$, $\varphi \in$ FO.
Parameter:	$\lvert\varphi\rvert$.
Problem:	Decide $G \models \varphi$.

ist fixed-parameter tractable (Linearzeit fpt-Algorithmus).

Beweis. Nach dem Satz von Gaifman reicht es, basislokale Sätze zu betrachten, also Sätze der Form

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

für eine r-lokale Formel $\psi(x)$.

Beweis des Satzes von Seese

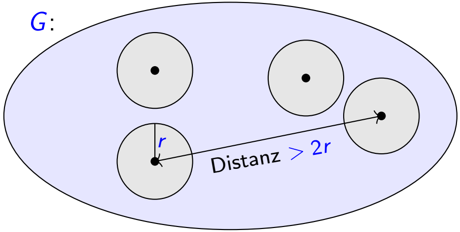
Angenommen

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

für eine r-lokale Formel $\psi(x)$.

Sei G ein Graph mit Maximalgrad d.

Wir suchen m Knoten mit Distanz $> 2r$,
deren r-Nachbarschaften ψ erfüllen.



Schritt 1

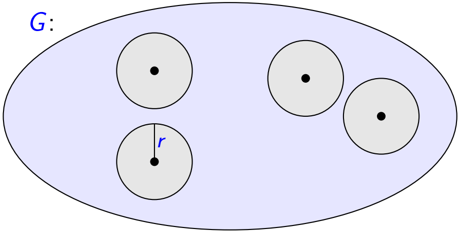
Algorithmus: Schritt 1

für alle $v \in V(G)$

- berechne $N_r(v)$
- entscheide, ob $N_r(v) \models \psi(v)$
  (Nachbarschaften konstanter Größe)

Wenn ja, färbe den Knoten rot.

G:



Laufzeit: $\mathcal{O}(n)$

Schritt 1

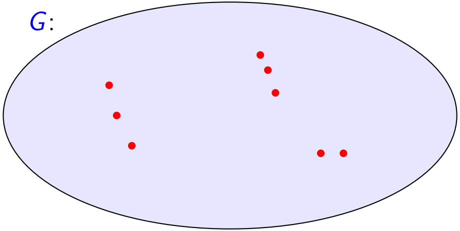
Algorithmus: Schritt 1

für alle $v \in V(G)$

- berechne $N_r(v)$
- entscheide, ob $N_r(v) \models \psi(v)$
  (Nachbarschaften konstanter Größe)

  Wenn ja, färbe den Knoten rot.

G:



Laufzeit: $\mathcal{O}(n)$

Schritt 1

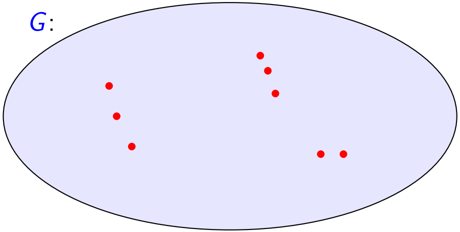
Algorithmus: Schritt 1

für alle $v \in V(G)$ $\mathcal{O}(n)$

- berechne $N_r(v)$ $\mathcal{O}(d^r) = \mathcal{O}(1)$
- entscheide, ob $N_r(v) \models \psi(v)$ $\mathcal{O}(1)$
  (Nachbarschaften konstanter Größe)

  Wenn ja, färbe den Knoten rot.

G:



Laufzeit: $\mathcal{O}(n)$

Schritt 2: Greedy-Ansatz

Sei Q die Menge der roten Knoten.

Algorithmus: Schritt 2.

$L := \emptyset$

while $Q \neq \emptyset$ **do**

wähle $v \in Q$

$L := L \cup \{v\}$

$Q := Q \setminus N_{2r}(v)$

od

if $|L| \geq m$ **then** akzeptiere

else

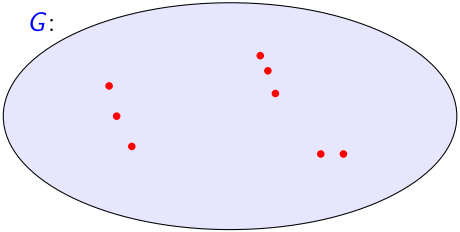
(alle roten Knoten sind in der $2r$-Nachbarschaft eines Elements aus L)

if $G[N_{2r}(L)] \models \exists x_1 \dots x_m (\bigwedge_{i \neq j} \mathrm{dist}(x_i, x_j) > 2r \wedge \bigwedge_i$ "x_i ist *rot*"$)$

akzeptiere **else** verwerfe

Angenommen $m = 4$

G:



Laufzeit: $\mathcal{O}(n)$

Schritt 2: Greedy-Ansatz

Sei Q die Menge der roten Knoten.

Algorithmus: Schritt 2.

$L := \emptyset$

while $Q \neq \emptyset$ **do**

wähle $v \in Q$

$L := L \cup \{v\}$

$Q := Q \setminus N_{2r}(v)$

od

if $|L| \geq m$ **then** akzeptiere

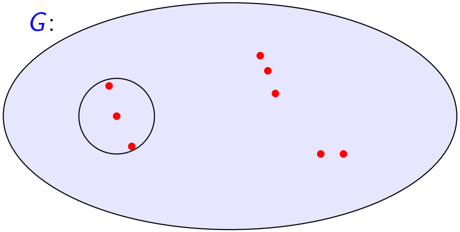
else

(alle roten Knoten sind in der $2r$-Nachbarschaft eines Elements aus L)

if $G[N_{2r}(L)] \models \exists x_1 \dots x_m (\bigwedge_{i \neq j} \text{dist}(x_i, x_j) > 2r \wedge \bigwedge_i$ "x_i ist *rot*"$)$

akzeptiere **else** verwerfe

Angenommen $m = 4$



Laufzeit: $\mathcal{O}(n)$

Schritt 2: Greedy-Ansatz

Sei Q die Menge der roten Knoten.

Algorithmus: Schritt 2.

$L := \emptyset$

while $Q \neq \emptyset$ **do**

wähle $v \in Q$

$L := L \cup \{v\}$

$Q := Q \setminus N_{2r}(v)$

od

if $|L| \geq m$ **then** akzeptiere

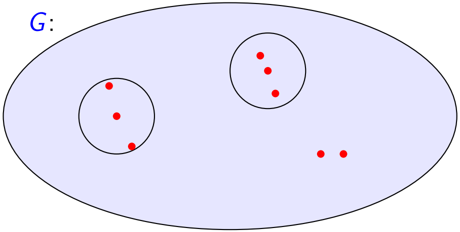
else

(alle roten Knoten sind in der $2r$-Nachbarschaft eines Elements aus L)

if $G[N_{2r}(L)] \models \exists x_1 \dots x_m (\bigwedge_{i \neq j} \mathrm{dist}(x_i, x_j) > 2r \wedge \bigwedge_i \text{“}x_i \text{ ist } rot\text{”})$

akzeptiere **else** verwerfe

Angenommen $m = 4$



Laufzeit: $\mathcal{O}(n)$

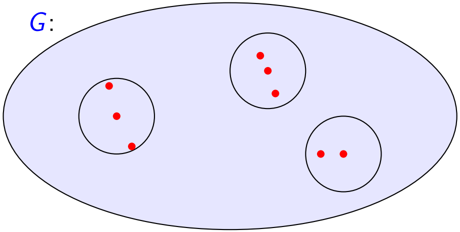
Schritt 2: Greedy-Ansatz

Sei Q die Menge der roten Knoten.

Algorithmus: Schritt 2.

$L := \emptyset$
while $Q \neq \emptyset$ **do**
  wähle $v \in Q$
  $L := L \cup \{v\}$
  $Q := Q \setminus N_{2r}(v)$
od
if $|L| \geq m$ **then** akzeptiere
else
  (alle roten Knoten sind in der $2r$-Nachbarschaft eines Elements aus L)
  if $G[N_{2r}(L)] \models \exists x_1 \dots x_m (\bigwedge_{i \neq j} \mathrm{dist}(x_i, x_j) > 2r \wedge \bigwedge_i$ "x_i ist *rot*")
  akzeptiere **else** verwerfe

Angenommen $m = 4$



Laufzeit: $\mathcal{O}(n)$

Schritt 2: Greedy-Ansatz

Sei Q die Menge der roten Knoten.

Algorithmus: Schritt 2.

$L := \emptyset$

while $Q \neq \emptyset$ **do**

wähle $v \in Q$

$L := L \cup \{v\}$

$Q := Q \setminus N_{2r}(v)$

od

if $|L| \geq m$ **then** akzeptiere

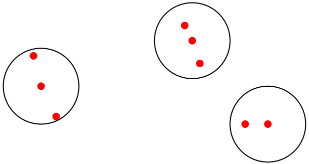
else

(alle roten Knoten sind in der $2r$-Nachbarschaft eines Elements aus L)

if $G[N_{2r}(L)] \models \exists x_1 \dots x_m (\bigwedge_{i \neq j} \mathrm{dist}(x_i, x_j) > 2r \wedge \bigwedge_i \text{“}x_i \text{ ist } \textit{rot}\text{”})$

akzeptiere **else** verwerfe

Angenommen $m = 4$



Laufzeit: $\mathcal{O}(n)$

FO-Model-Checking auf Graphen beschränkten Grads

Theorem. (Seese, 1996)

Sei $\mathcal{C}$ eine Klasse von Graphen mit Maximalgrad $d \geq 1$.

MC(FO, $\mathcal{C}$)	
Eingabe:	Graph $G \in \mathcal{C}$, $\varphi \in$ FO.
Parameter:	$\lvert\varphi\rvert$.
Problem:	Entscheide $G \models \varphi$.

ist fixed-parameter tractable (linearzeit fpt-Algorithmus).

Aber:

Der Beweis zeigt noch viel mehr ...

... denn, wo wurde der beschränkte Grad benutzt?

Schritt 1

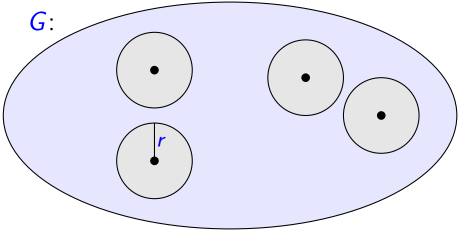
Algorithmus: Schritt 1

für alle $v \in V(G)$ $\mathcal{O}(n)$

- berechne $N_r(v)$ $\mathcal{O}(d^r) = \mathcal{O}(1)$
- entscheide, ob $N_r(v) \models \psi(v)$ $\mathcal{O}(1)$
  (Nachbarschaften konstanter Größe)

  Wenn ja, färbe den Knoten rot.

G:



Laufzeit: $\mathcal{O}(n)$

Schritt 2: Greedy-Ansatz

Sei Q die Menge der roten Knoten.

Algorithmus: Schritt 2.

$L := \emptyset$

while $Q \neq \emptyset$ **do**

wähle $v \in Q$

$L := L \cup \{v\}$

$Q := Q \setminus N_{2r}(v)$

od

if $|L| \geq m$ **then** akzeptiere

else

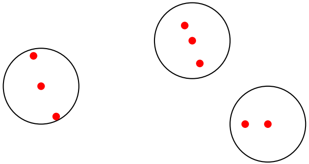
(alle roten Knoten sind in der $2r$-Nachbarschaft eines Elements aus L)

if $G[N_{2r}(L)] \models \exists x_1 \dots x_m (\bigwedge_{i \neq j} \mathrm{dist}(x_i, x_j) > 2r \wedge \bigwedge_i$ "x_i ist *rot*"$)$

akzeptiere **else** verwerfe

Laufzeit: $\mathcal{O}(n)$

Angenommen $m = 4$



Lokales Model-Checking

Im wesentlichen:

- Wir müssen r-lokale Formeln $\psi(x)$ inn r'-Nachbarschaften testen können.

Dabei: r, r' hängen nur von der Originalformel φ ab und sind daher konstant

(Teil des Parameters).

Lokales Model-Checking

Theorem. Sei $\mathcal{C}$ eine Klasse von Graphen auf denen folgendes Problem in FPT ist:

> Local-FO-MC($\mathcal{C}$)
> *Eingabe:* $\varphi \in \text{FO}$, Graph $G \in \mathcal{C}, v_1, \dots, v_k \in V(G)$, and $r \in \mathbb{N}$.
> *Parameter:* $r + k + |\varphi|$.
> *Problem:* Entscheide $G\left[N_r^G(v_1, \dots, v_k)\right] \models \varphi$.

Dann ist FO-Model-Checking auf $\mathcal{C}$ fixed-parameter tractable.

Folgerung: Für effizientes FO-Model-Checking reicht es, wenn jede r-Nachbarschaft in einem Graph *einfach* ist.

Nicht der gesamte Graph muss kleine Baumweite haben, sondern nur die r-Nachbarschaften.

Model-Checking auf Planaren Graphen

Theorem. (Frick und Grohe, 2001)

> MC(FO, Planar)
> *Eingabe:* Planarer Graph $G \in \mathcal{C}$, $\varphi \in$ FO.
> *Parameter:* $|\varphi|$.
> *Problem:* Entscheide $G \models \varphi$.

ist fixed-parameter tractable (linearzeit fpt-Algorithmus).

Beweis. Das Ergebnis folgt aus dem lokalen Model-Checking zusammen mit folgendem Satz.

Satz. (Robertson and Seymour)

Jeder planare Graph mit Durchmesser $\leq r$ hat Baumweite $\leq 3r$.

Zusammenfassung

Der Satz von Gaifman im Model-Checking.

- Die Gaifman-Normalform erlaubt es, das Model-Checking von FO-Formeln in allgemeinen Strukturen auf das Model-Checking in r-Nachbarschaften (im Gaifmangraph) zu reduzieren.
- Diese Technik erlaubt es Auswertungsalgorithmen auf vielen Klassen von Graphen zu entwickeln, indem gezeigt wird, dass die r-Nachbarschaften in irgendeiner Form kontrolliert werden können.
- Bei *dichten* Strukturen, z.B. Strukturen mit einer Ordnung, sind die r-Nachbarschaften oft sehr groß, überdecken z.B. die gesamte Struktur.

  Da kommt dieser Ansatz an seine Grenzen.

Satz. (Grohe, K., Siebertz '15)

Sei $\mathcal{C}$ eine Klasse von Graphen, die unter Untergraphen abgeschlossen ist. D.h., wenn $G \in \mathcal{C}$ und $H \subseteq G$, dann ist auch $H \in \mathcal{C}$.

Dann ist $\mathrm{MC}(\textsc{fo}, \mathcal{C}) \in \mathrm{FPT}$ genau dann, wenn $\mathcal{C}$ nowhere dense ist.