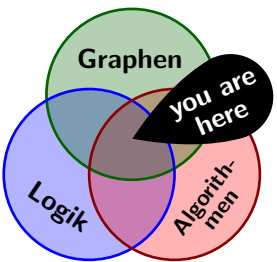
Monadisch Stabile und Monadisch Abhängige Graphklassen

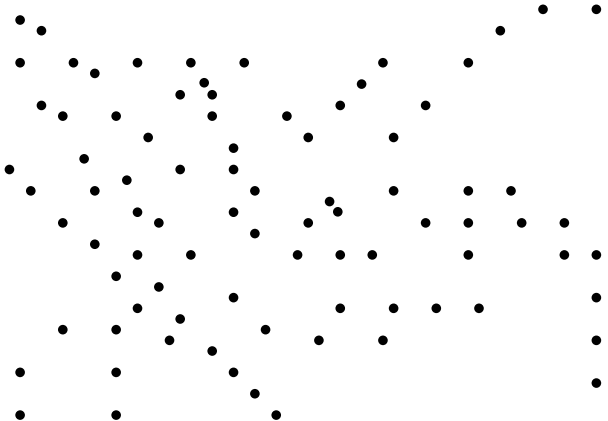
Charakterisierungen und Algorithmische Meta-Theoreme

Nikolas Mählmann, Universität Bremen

Kolloquium zum GI-Dissertationspreis 2024

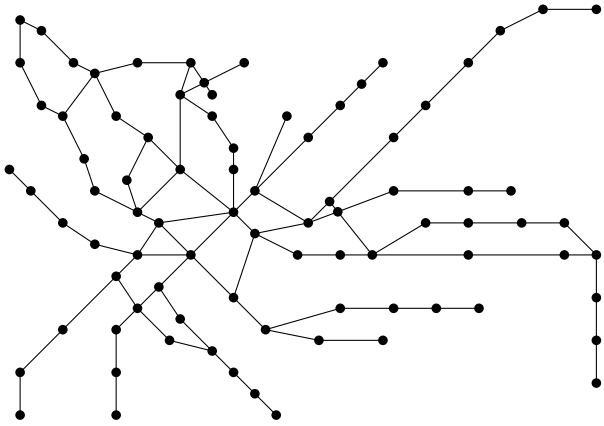


Graphen



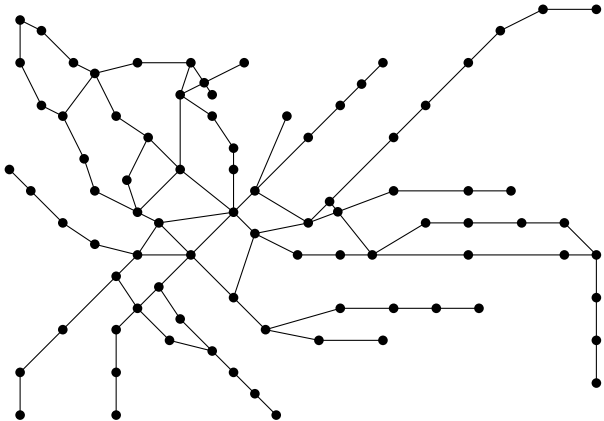
Graphen bestehen aus *Knoten*, verbunden durch *Kanten*.

Graphen



Graphen bestehen aus *Knoten*, verbunden durch *Kanten*.

Graphen



Graphen bestehen aus *Knoten*, verbunden durch *Kanten*.

Sie eignen sich zur Modellierung verschiedenster Systeme, z.B.:

- Straßen- und Stromnetze
- Computer-Netzwerke
- Schaltkreise
- Molekülstrukturen

Graphen

Graphen bestehen aus *Knoten*, verbunden durch *Kanten*.

Sie eignen sich zur Modellierung verschiedenster Systeme, z.B.:

- Straßen- und Stromnetze
- Computer-Netzwerke
- Schaltkreise
- Molekülstrukturen

Graphen

Graphen bestehen aus *Knoten*, verbunden durch *Kanten*.

Sie eignen sich zur Modellierung verschiedenster Systeme, z.B.:

- Straßen- und Stromnetze
- Computer-Netzwerke
- Schaltkreise
- Molekülstrukturen

Eine *Graphklasse* ist eine (meist unendliche) Menge von Graphen. Beispiel:

$$\{ \bullet, \bullet\!\!-\!\!\bullet, \triangle, \square, \text{K}_5, \ldots \}$$

Das Model Checking Problem der Prädikatenlogik

Problem: Entscheide für einen Graphen G und eine prädikatenlogische Formel φ, ob

$$G \models \varphi.$$

Beispiel: G enthält ein DOMINATING SET der Größe k gdw.

$$G \models \exists x_1 \ldots \exists x_k \forall y : \bigvee_{1 \leqslant i \leqslant k} \big(y = x_i \vee \mathsf{Kante}(y, x_i)\big).$$

Das Model Checking Problem der Prädikatenlogik

Problem: Entscheide für einen Graphen G und eine prädikatenlogische Formel φ, ob

$$G \models \varphi.$$

Beispiel: G enthält ein Dominating Set der Größe k gdw.

$$G \models \exists x_1 \ldots \exists x_k \forall y : \bigvee_{1 \leqslant i \leqslant k} \big(y = x_i \vee \mathsf{Kante}(y, x_i)\big).$$

Weitere ausdrückbare Probleme: Independent Set, Subgraph Isomorphie, Independent Rot-Blau Distanz-7 Dominating Set, ...

Das Model Checking Problem der Prädikatenlogik

Problem: Entscheide für einen Graphen G und eine prädikatenlogische Formel φ, ob

$$G \models \varphi.$$

Beispiel: G enthält ein Dominating Set der Größe k gdw.

$$G \models \exists x_1 \dots \exists x_k \forall y : \bigvee_{1 \leqslant i \leqslant k} \big(y = x_i \vee \mathsf{Kante}(y, x_i)\big).$$

Weitere ausdrückbare Probleme: Independent Set, Subgraph Isomorphie, Independent Rot-Blau Distanz-7 Dominating Set, ...

Laufzeit: Für die Klasse *aller Graphen* ist die bestmögliche Laufzeit $n^{\mathcal{O}(|\varphi|)}$.

(n = Anzahl der Knoten in G; Komplexitätsannahme: ETH)

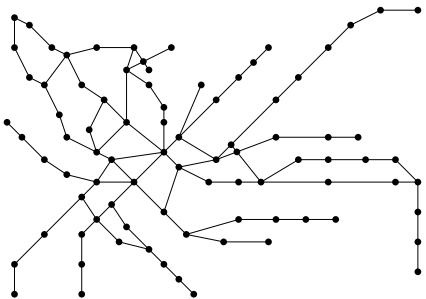
Strukturierte Graphklassen

Laufzeit: Für die Klasse *aller Graphen* ist die bestmögliche Laufzeit $n^{\mathcal{O}(|\varphi|)}$.

Strukturierte Graphklassen

Laufzeit: Für die Klasse *aller Graphen* ist die bestmögliche Laufzeit $n^{\mathcal{O}(|\varphi|)}$.

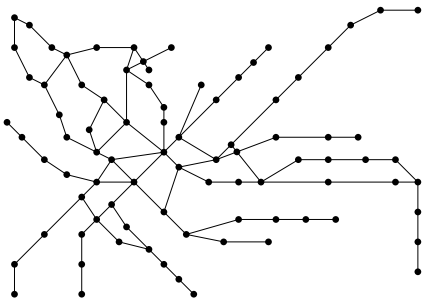
Aber: In der Praxis modellieren Graphen häufig stark strukturierte Systeme:



Strukturierte Graphklassen

Laufzeit: Für die Klasse *aller Graphen* ist die bestmögliche Laufzeit $n^{\mathcal{O}(|\varphi|)}$.

Aber: In der Praxis modellieren Graphen häufig stark strukturierte Systeme:

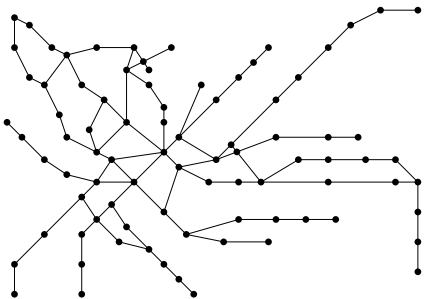


Keine kreuzenden Kanten
$\Rightarrow$ der Graph is *planar*
$\Rightarrow$ schnelleres Model Checking

Strukturierte Graphklassen

Laufzeit: Für die Klasse *aller Graphen* ist die bestmögliche Laufzeit $n^{\mathcal{O}(|\varphi|)}$.

Aber: In der Praxis modellieren Graphen häufig stark strukturierte Systeme:



Keine kreuzenden Kanten
$\Rightarrow$ der Graph is *planar*
$\Rightarrow$ schnelleres Model Checking

Frage: Für welche Klassen ist Model Checking lösbar in Zeit $f(|\varphi|) \cdot n^c$?

Stand der Technik

Theorem [Grohe, Kreutzer, Siebertz, 2014]

Model Checking ist lösbar in Zeit $f(|\varphi|) \cdot n^{1.00001}$ für jede **nowhere dense** Klasse.

Stand der Technik

Theorem [Grohe, Kreutzer, Siebertz, 2014]

Model Checking ist lösbar in Zeit $f(|\varphi|) \cdot n^{1.00001}$ für jede **nowhere dense** Klasse.

Viele Klassen mit geringer Kantendichte sind nowhere dense, z.B.:

- planare Graphen,
- Klassen mit beschränkter Baumweite,
- Klassen mit beschränktem Maximalgrad.

Stand der Technik

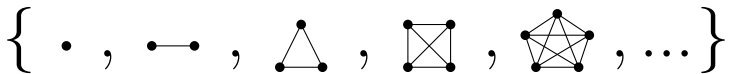
Theorem [Grohe, Kreutzer, Siebertz, 2014]

Model Checking ist lösbar in Zeit $f(|\varphi|) \cdot n^{1.00001}$ für jede **nowhere dense** Klasse.

Viele Klassen mit geringer Kantendichte sind nowhere dense, z.B.:

- planare Graphen,
- Klassen mit beschränkter Baumweite,
- Klassen mit beschränktem Maximalgrad.

Aber **keine** Klasse mit hoher Kantendichte ist nowhere dense:



Stand der Technik

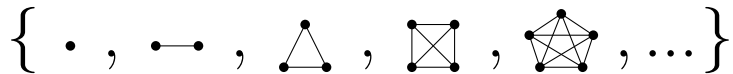
Theorem [Grohe, Kreutzer, Siebertz, 2014]

Model Checking ist lösbar in Zeit $f(|\varphi|) \cdot n^{1.00001}$ für jede **nowhere dense** Klasse.

Viele Klassen mit geringer Kantendichte sind nowhere dense, z.B.:

- planare Graphen,
- Klassen mit beschränkter Baumweite,
- Klassen mit beschränktem Maximalgrad.

Aber **keine** Klasse mit hoher Kantendichte ist nowhere dense:



Frage: Welche Struktureigenschaften helfen auf dichten Graphen?

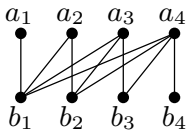
Monadische Stabilität und monadische Abhängigkeit

Definition: Eine Graphklasse $\mathcal{C}$ ist **monadisch stabil**, wenn Prädikatenlogik nicht alle linearen Ordnungen in $\mathcal{C}$ kodiert.

Monadische Stabilität und monadische Abhängigkeit

Definition: Eine Graphklasse $\mathcal{C}$ ist **monadisch stabil**, wenn Prädikatenlogik nicht alle linearen Ordnungen in $\mathcal{C}$ kodiert.

Beispiel für eine Kodierung: $\varphi(x, y) :=$ "Nachbarn$(x) \subsetneq$ Nachbarn(y)"

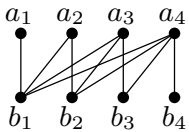


$a_1 \prec_\varphi a_2 \prec_\varphi a_3 \prec_\varphi a_4$

Monadische Stabilität und monadische Abhängigkeit

Definition: Eine Graphklasse $\mathcal{C}$ ist **monadisch stabil**, wenn Prädikatenlogik nicht alle linearen Ordnungen in $\mathcal{C}$ kodiert.

Beispiel für eine Kodierung: $\varphi(x, y) :=$ "Nachbarn$(x) \subsetneq$ Nachbarn(y)"



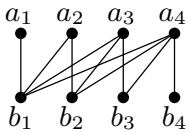
$a_1 \prec_\varphi a_2 \prec_\varphi a_3 \prec_\varphi a_4$

Definition: Eine Graphklasse $\mathcal{C}$ ist **monadisch abhängig**, wenn Prädikatenlogik nicht alle Graphen in $\mathcal{C}$ kodiert.

Monadische Stabilität und monadische Abhängigkeit

Definition: Eine Graphklasse $\mathcal{C}$ ist **monadisch stabil**, wenn Prädikatenlogik nicht alle linearen Ordnungen in $\mathcal{C}$ kodiert.

Beispiel für eine Kodierung: $\varphi(x, y) :=$ "Nachbarn$(x) \subsetneq$ Nachbarn(y)"



$a_1 \prec_\varphi a_2 \prec_\varphi a_3 \prec_\varphi a_4$

Definition: Eine Graphklasse $\mathcal{C}$ ist **monadisch abhängig**, wenn Prädikatenlogik nicht alle Graphen in $\mathcal{C}$ kodiert.

nowhere dense Klassen $\subsetneq$ monadisch stabile Klassen $\subsetneq$ monadisch abhängige Klassen

Algorithmische Ergebnisse

Die Model Checking Vermutung

Sei $\mathcal{C}$ eine hereditäre Graphklasse.

- $\mathcal{C}$ is monadisch abhängig $\Rightarrow$ Model Checking ist effizient für $\mathcal{C}$.
- $\mathcal{C}$ is monadisch unabhängig $\Rightarrow$ Model Checking ist schwer für $\mathcal{C}$.

(hereditär: abgeschlossen unter Löschen von Knoten)

Algorithmische Ergebnisse

Die Model Checking Vermutung

Sei $\mathcal{C}$ eine hereditäre Graphklasse.

- $\mathcal{C}$ is monadisch abhängig $\Rightarrow$ Model Checking ist effizient für $\mathcal{C}$.
- $\mathcal{C}$ is monadisch unabhängig $\Rightarrow$ Model Checking ist schwer für $\mathcal{C}$.

(hereditär: abgeschlossen unter Löschen von Knoten)

Wir zeigen:

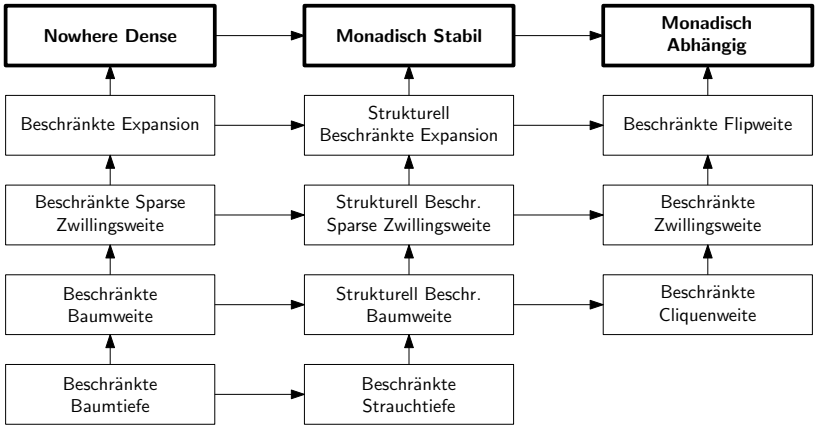
Theorem

Sei $\mathcal{C}$ eine hereditäre Graphklasse.

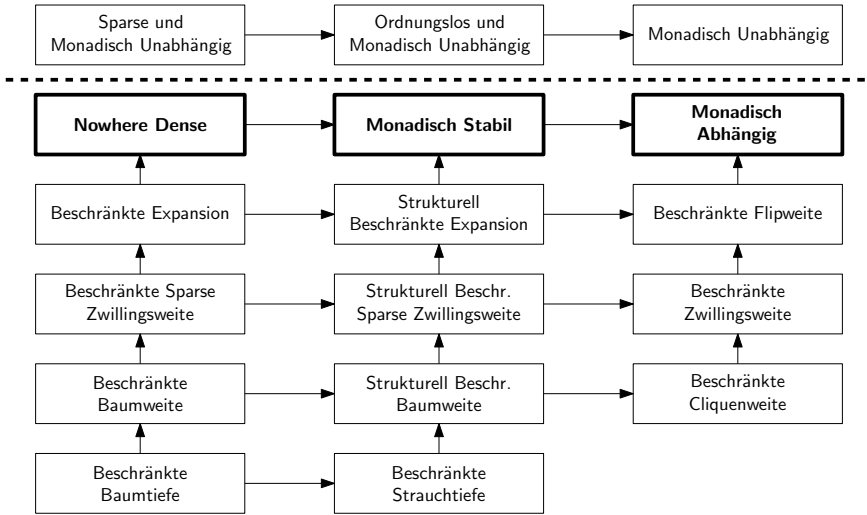
- $\mathcal{C}$ is monadisch stabil $\Rightarrow$ Model Checking ist lösbar in Zeit $f(|\varphi|) \cdot n^{6.00001}$ für $\mathcal{C}$.
- $\mathcal{C}$ is monadisch unabhängig $\Rightarrow$ Model Checking ist $\mathrm{AW}[*]$-schwer für $\mathcal{C}$.

Model Checking in hereditären Graphklassen

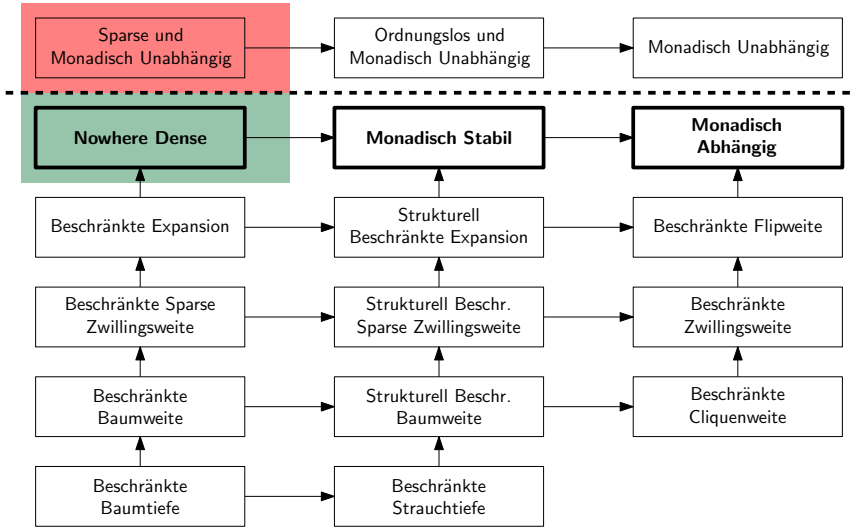
Model Checking in hereditären Graphklassen



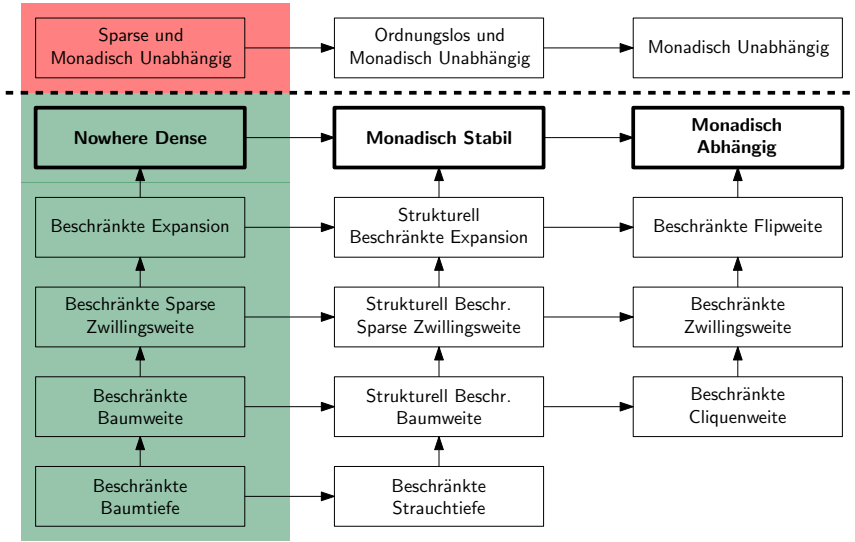
Model Checking in hereditären Graphklassen



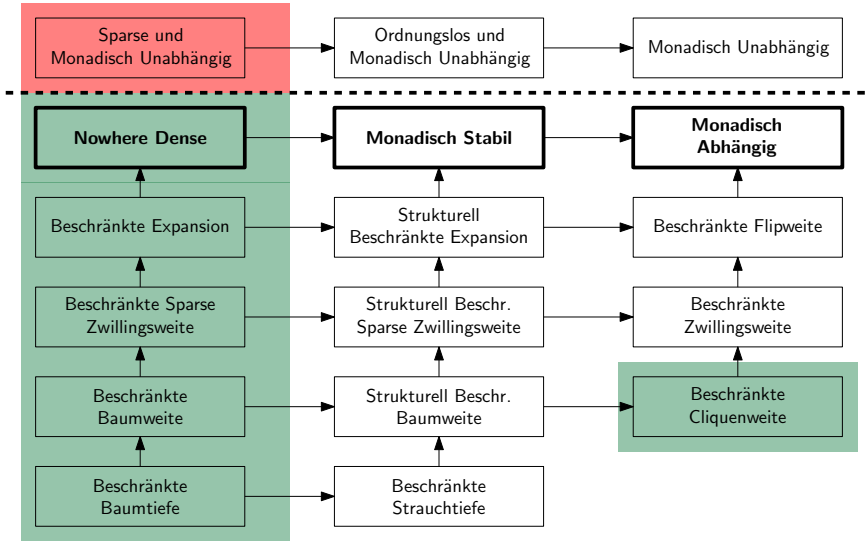
Model Checking in hereditären Graphklassen



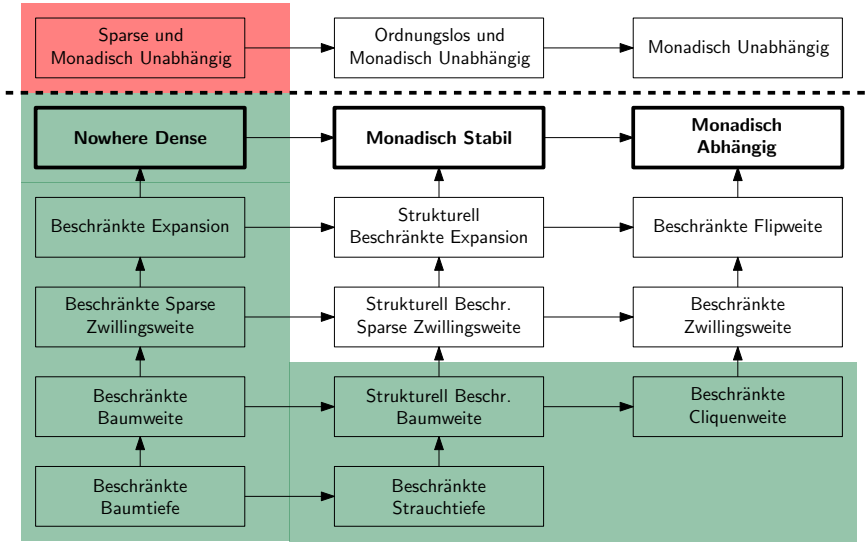
Model Checking in hereditären Graphklassen



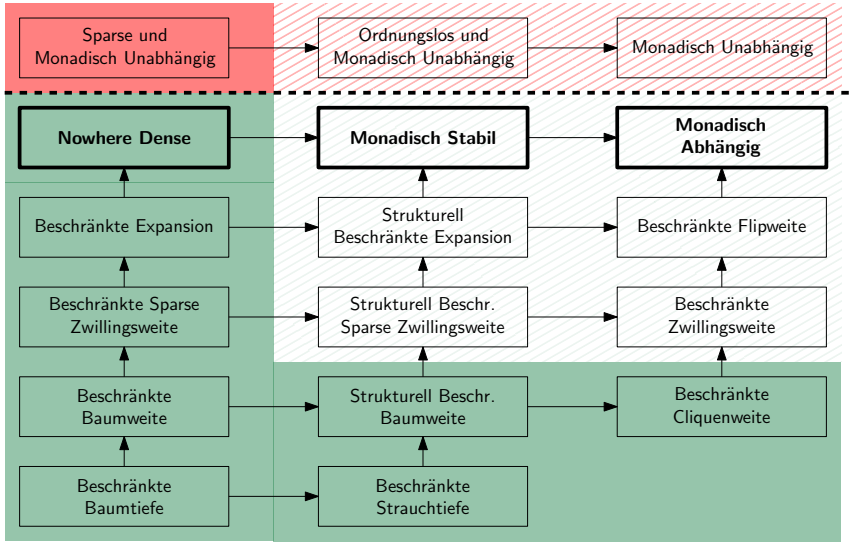
Model Checking in hereditären Graphklassen



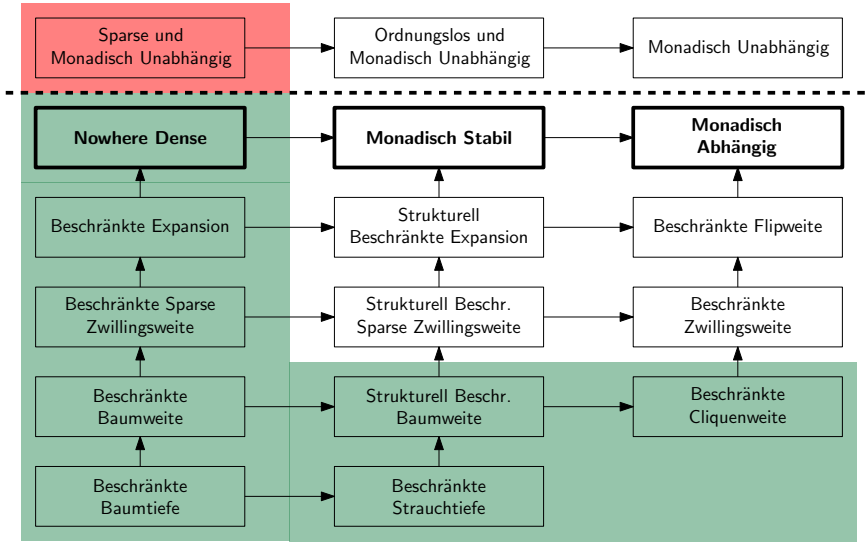
Model Checking in hereditären Graphklassen



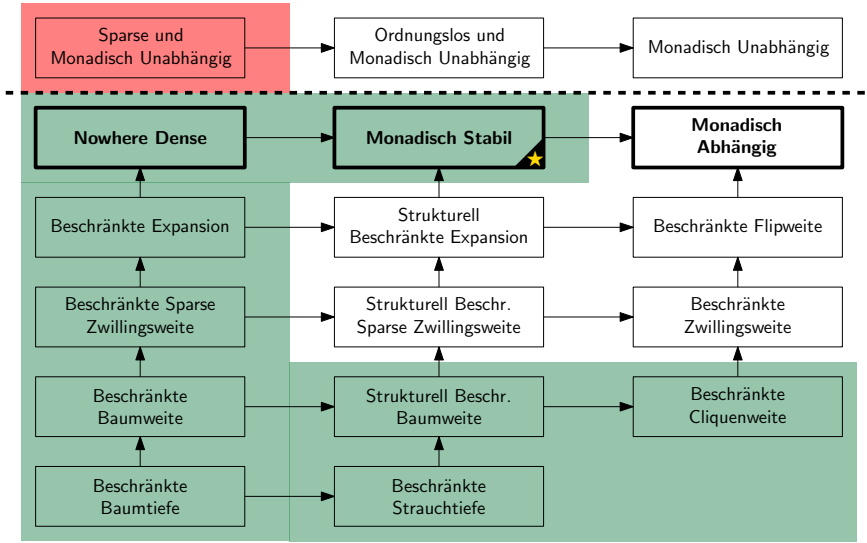
Vermutete Grenzen der eff. Lösbarkeit



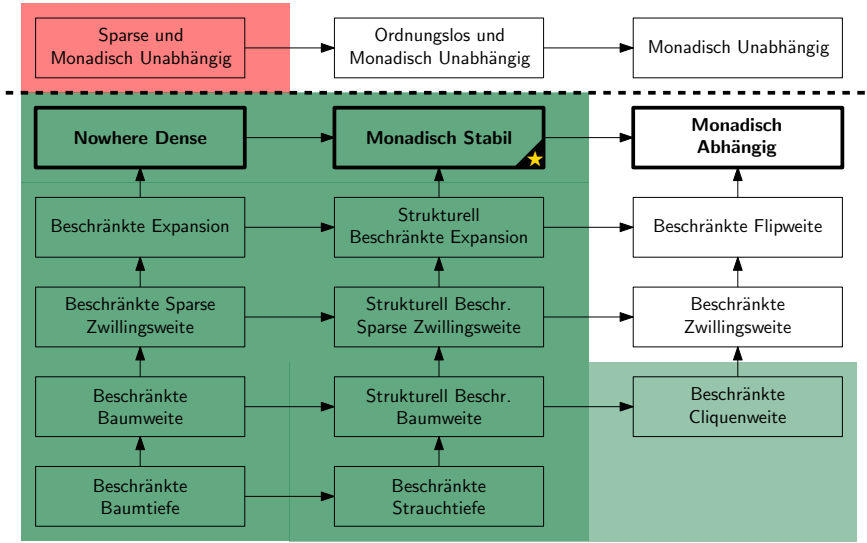
Unsere Ergebnisse ★



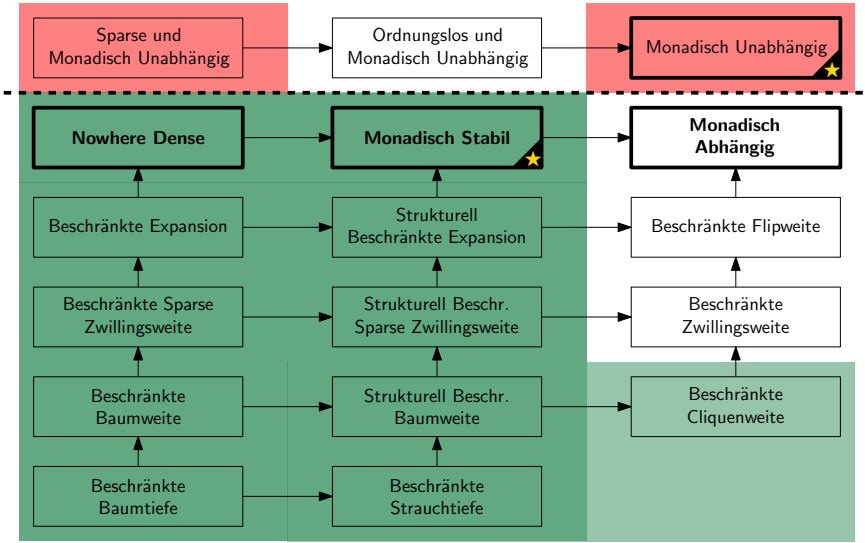
Unsere Ergebnisse ★



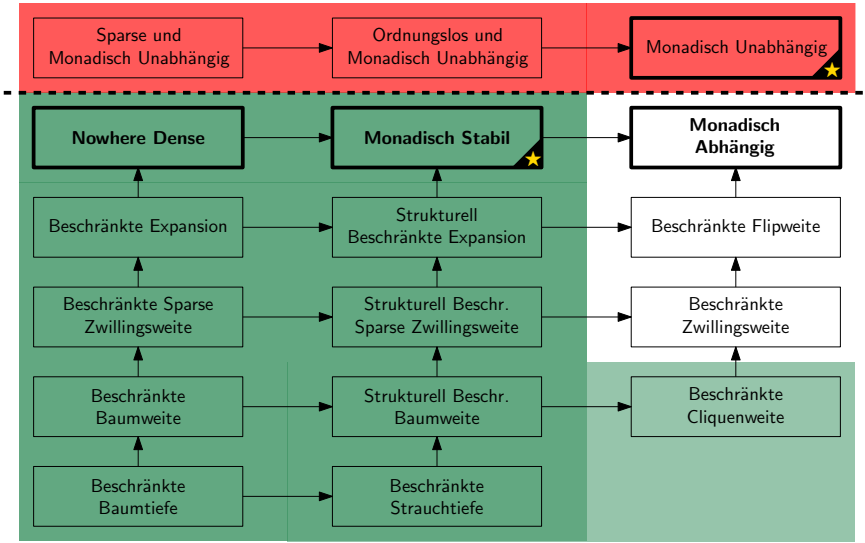
Unsere Ergebnisse ★



Unsere Ergebnisse ★



Unsere Ergebnisse ★



Kombinatorische Ergebnisse

Monadische Stabilität und Abhängigkeit sind mithilfe von **Logik** definiert.

Kombinatorische Ergebnisse

Monadische Stabilität und Abhängigkeit sind mithilfe von **Logik** definiert.

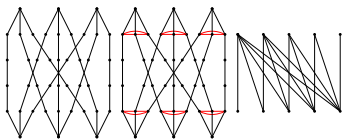
Hauptteil der Arbeit: **kombinatorische** Charakterisierungen durch:

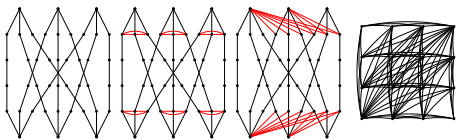
Kombinatorische Ergebnisse

Monadische Stabilität und Abhängigkeit sind mithilfe von **Logik** definiert.

Hauptteil der Arbeit: **kombinatorische** Charakterisierungen durch:

1. **Ausgeschlossene induzierte Subgraphen:**



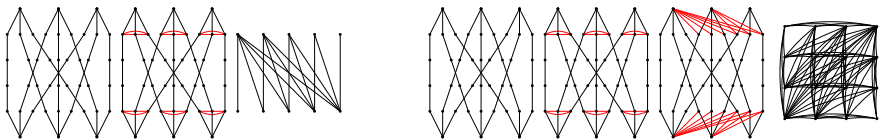


Kombinatorische Ergebnisse

Monadische Stabilität und Abhängigkeit sind mithilfe von **Logik** definiert.

Hauptteil der Arbeit: **kombinatorische** Charakterisierungen durch:

1. **Ausgeschlossene induzierte Subgraphen:**



2. **Ramsey-Eigenschaften**:

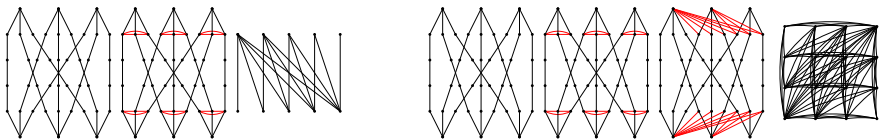
		Flatness	Breakability
dist-r	Flip-	**mon. Stabilität**	**mon. Abhängigkeit**
	Deletion-	**Nowhere Denseness**	**Nowhere Denseness**
dist-∞	Flip-	**Strauchtiefe**	**Cliquenweite**
	Deletion-	**Baumtiefe**	**Baumweite**

Kombinatorische Ergebnisse

Monadische Stabilität und Abhängigkeit sind mithilfe von **Logik** definiert.

Hauptteil der Arbeit: **kombinatorische** Charakterisierungen durch:

1. **Ausgeschlossene induzierte Subgraphen:**



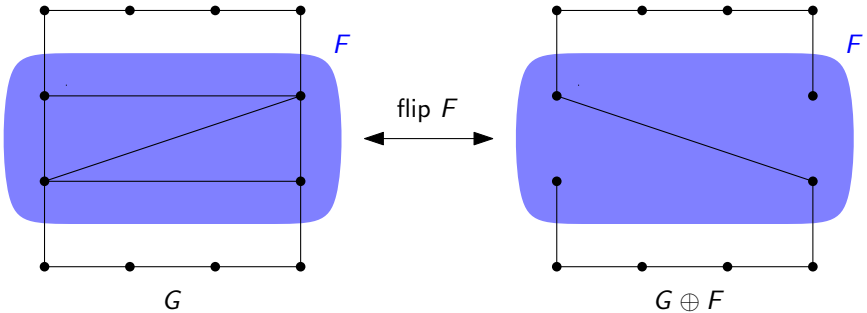
2. **Ramsey-Eigenschaften**:

		Flatness	Breakability
dist-r	Flip-	**mon. Stabilität**	**mon. Abhängigkeit**
	Deletion-	**Nowhere Denseness**	**Nowhere Denseness**
dist-∞	Flip-	**Strauchtiefe**	**Cliquenweite**
	Deletion-	**Baumtiefe**	**Baumweite**

3. **Flipper Spiel** (nur für monadische Stabilität)

Flips

Sei $G \oplus F$ der Graph den wir erhalten indem wir in G die Kanten zwischen den Knoten in F komplementieren.

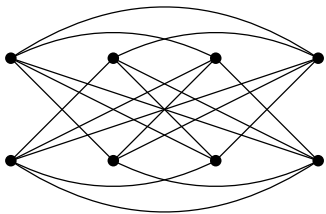


Das Flipper Spiel

Das *Radius-r Flipper Spiel* wird auf einem Graphen gespielt. In jeder Runde:

1. Flipper flippt eine Knotenmenge.
2. Localizer beschränkt den Graphen auf einen Radius-r Ball.

Flipper gewinnt, sobald nur noch einen Knoten übrig ist. Beispiel für $r = 2$:

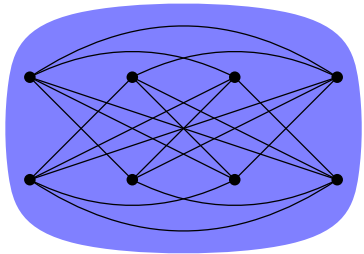


Das Flipper Spiel

Das *Radius-r Flipper Spiel* wird auf einem Graphen gespielt. In jeder Runde:

1. Flipper flippt eine Knotenmenge.
2. Localizer beschränkt den Graphen auf einen Radius-r Ball.

Flipper gewinnt, sobald nur noch einen Knoten übrig ist. Beispiel für $r = 2$:

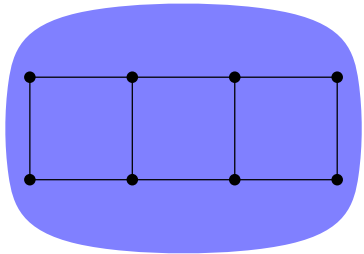


Das Flipper Spiel

Das *Radius-r Flipper Spiel* wird auf einem Graphen gespielt. In jeder Runde:

1. Flipper flippt eine Knotenmenge.
2. Localizer beschränkt den Graphen auf einen Radius-r Ball.

Flipper gewinnt, sobald nur noch einen Knoten übrig ist. Beispiel für $r = 2$:

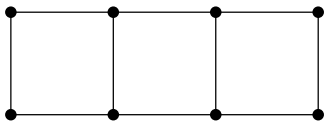


Das Flipper Spiel

Das *Radius-r Flipper Spiel* wird auf einem Graphen gespielt. In jeder Runde:

1. Flipper flippt eine Knotenmenge.
2. Localizer beschränkt den Graphen auf einen Radius-r Ball.

Flipper gewinnt, sobald nur noch einen Knoten übrig ist. Beispiel für $r = 2$:

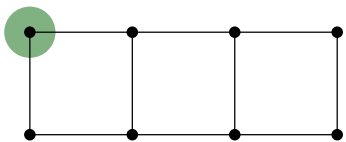


Das Flipper Spiel

Das *Radius-r Flipper Spiel* wird auf einem Graphen gespielt. In jeder Runde:

1. Flipper flippt eine Knotenmenge.
2. Localizer beschränkt den Graphen auf einen Radius-r Ball.

Flipper gewinnt, sobald nur noch einen Knoten übrig ist. Beispiel für $r = 2$:

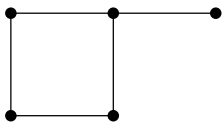


Das Flipper Spiel

Das *Radius-r Flipper Spiel* wird auf einem Graphen gespielt. In jeder Runde:

1. Flipper flippt eine Knotenmenge.
2. Localizer beschränkt den Graphen auf einen Radius-r Ball.

Flipper gewinnt, sobald nur noch einen Knoten übrig ist. Beispiel für $r = 2$:

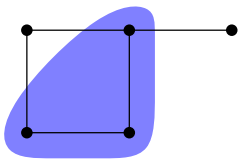


Das Flipper Spiel

Das *Radius-r Flipper Spiel* wird auf einem Graphen gespielt. In jeder Runde:

1. Flipper flippt eine Knotenmenge.
2. Localizer beschränkt den Graphen auf einen Radius-r Ball.

Flipper gewinnt, sobald nur noch einen Knoten übrig ist. Beispiel für $r = 2$:

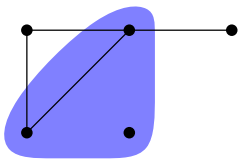


Das Flipper Spiel

Das *Radius-r Flipper Spiel* wird auf einem Graphen gespielt. In jeder Runde:

1. Flipper flippt eine Knotenmenge.
2. Localizer beschränkt den Graphen auf einen Radius-r Ball.

Flipper gewinnt, sobald nur noch einen Knoten übrig ist. Beispiel für $r = 2$:

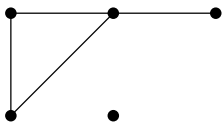


Das Flipper Spiel

Das *Radius-r Flipper Spiel* wird auf einem Graphen gespielt. In jeder Runde:

1. Flipper flippt eine Knotenmenge.
2. Localizer beschränkt den Graphen auf einen Radius-r Ball.

Flipper gewinnt, sobald nur noch einen Knoten übrig ist. Beispiel für $r = 2$:

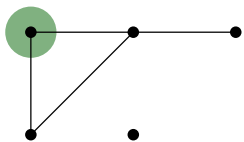


Das Flipper Spiel

Das *Radius-r Flipper Spiel* wird auf einem Graphen gespielt. In jeder Runde:

1. Flipper flippt eine Knotenmenge.
2. Localizer beschränkt den Graphen auf einen Radius-r Ball.

Flipper gewinnt, sobald nur noch einen Knoten übrig ist. Beispiel für $r = 2$:

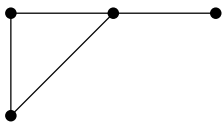


Das Flipper Spiel

Das *Radius-r Flipper Spiel* wird auf einem Graphen gespielt. In jeder Runde:

1. Flipper flippt eine Knotenmenge.
2. Localizer beschränkt den Graphen auf einen Radius-r Ball.

Flipper gewinnt, sobald nur noch einen Knoten übrig ist. Beispiel für $r = 2$:

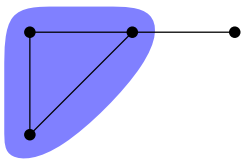


Das Flipper Spiel

Das *Radius-r Flipper Spiel* wird auf einem Graphen gespielt. In jeder Runde:

1. Flipper flippt eine Knotenmenge.
2. Localizer beschränkt den Graphen auf einen Radius-r Ball.

Flipper gewinnt, sobald nur noch einen Knoten übrig ist. Beispiel für $r = 2$:

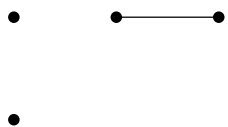


Das Flipper Spiel

Das *Radius-r Flipper Spiel* wird auf einem Graphen gespielt. In jeder Runde:

1. Flipper flippt eine Knotenmenge.
2. Localizer beschränkt den Graphen auf einen Radius-r Ball.

Flipper gewinnt, sobald nur noch einen Knoten übrig ist. Beispiel für $r = 2$:

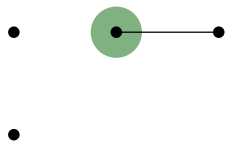


Das Flipper Spiel

Das *Radius-r Flipper Spiel* wird auf einem Graphen gespielt. In jeder Runde:

1. Flipper flippt eine Knotenmenge.
2. Localizer beschränkt den Graphen auf einen Radius-r Ball.

Flipper gewinnt, sobald nur noch einen Knoten übrig ist. Beispiel für $r = 2$:



Das Flipper Spiel

Das *Radius-r Flipper Spiel* wird auf einem Graphen gespielt. In jeder Runde:

1. Flipper flippt eine Knotenmenge.
2. Localizer beschränkt den Graphen auf einen Radius-r Ball.

Flipper gewinnt, sobald nur noch einen Knoten übrig ist. Beispiel für $r = 2$:

Das Flipper Spiel

Das *Radius-r Flipper Spiel* wird auf einem Graphen gespielt. In jeder Runde:

1. Flipper flippt eine Knotenmenge.
2. Localizer beschränkt den Graphen auf einen Radius-r Ball.

Flipper gewinnt, sobald nur noch einen Knoten übrig ist. Beispiel für $r = 2$:

Das Flipper Spiel

Das *Radius-r Flipper Spiel* wird auf einem Graphen gespielt. In jeder Runde:

1. Flipper flippt eine Knotenmenge.
2. Localizer beschränkt den Graphen auf einen Radius-r Ball.

Flipper gewinnt, sobald nur noch einen Knoten übrig ist. Beispiel für $r = 2$:

Das Flipper Spiel

Das *Radius-r Flipper Spiel* wird auf einem Graphen gespielt. In jeder Runde:

1. Flipper flippt eine Knotenmenge.
2. Localizer beschränkt den Graphen auf einen Radius-r Ball.

Flipper gewinnt, sobald nur noch einen Knoten übrig ist. Beispiel für $r = 2$:

Das Flipper Spiel

Das *Radius-r Flipper Spiel* wird auf einem Graphen gespielt. In jeder Runde:

1. Flipper flippt eine Knotenmenge.
2. Localizer beschränkt den Graphen auf einen Radius-r Ball.

Flipper gewinnt, sobald nur noch einen Knoten übrig ist. Beispiel für $r = 2$:

Das Flipper Spiel in monadisch stabilen Klassen

Theorem

Eine Graphklasse $\mathcal{C}$ ist monadisch stabil $\Leftrightarrow$

$\forall r \exists \ell$ sodass Flipper das Radius-r Spiel auf jedem Graph aus $\mathcal{C}$ in ℓ Runden gewinnt.

Das Flipper Spiel ist **rein kombinatorische** Charakterisierung!

Das Flipper Spiel in monadisch stabilen Klassen

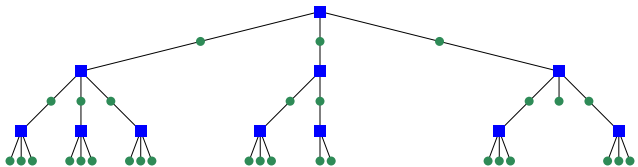
Theorem

Eine Graphklasse $\mathcal{C}$ ist monadisch stabil $\Leftrightarrow$

$\forall r \exists \ell$ sodass Flipper das Radius-r Spiel auf jedem Graph aus $\mathcal{C}$ in ℓ Runden gewinnt.

Das Flipper Spiel ist **rein kombinatorische** Charakterisierung!

Der Spielbaum ist eine Zerlegung der r-Bälle des Graphen mit **beschränkter Tiefe**.



Das Flipper Spiel in monadisch stabilen Klassen

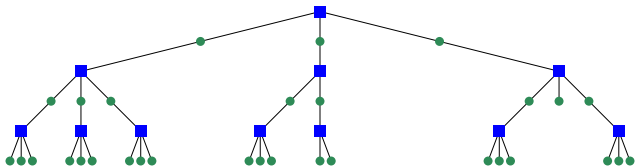
Theorem

Eine Graphklasse $\mathcal{C}$ ist monadisch stabil $\Leftrightarrow$

$\forall r \exists \ell$ sodass Flipper das Radius-r Spiel auf jedem Graph aus $\mathcal{C}$ in ℓ Runden gewinnt.

Das Flipper Spiel ist **rein kombinatorische** Charakterisierung!

Der Spielbaum ist eine Zerlegung der r-Bälle des Graphen mit **beschränkter Tiefe**.



Dynamische Programmierung liefert **effizientes Model Checking**.

Zusammenfassung

Monadische Stabilität und **Abhängigkeit** sind **logisch** definiert und generalisieren:

- Planarität, beschränkte Baum- und Cliquenweite, Nowhere Denseness, ...

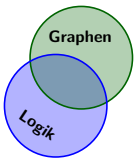
Zusammenfassung

Monadische Stabilität und **Abhängigkeit** sind **logisch** definiert und generalisieren:

- Planarität, beschränkte Baum- und Cliquenweite, Nowhere Denseness, ...

Wir entwickeln eine **kombinatorische** Strukturtheorie für diese Graphklassen:

- induzierte Subgraphen, Ramsey-Eigenschaften, Flipper Spiel



Zusammenfassung

Monadische Stabilität und **Abhängigkeit** sind **logisch** definiert und generalisieren:

- Planarität, beschränkte Baum- und Cliquenweite, Nowhere Denseness, ...

Wir entwickeln eine **kombinatorische** Strukturtheorie für diese Graphklassen:

- induzierte Subgraphen, Ramsey-Eigenschaften, Flipper Spiel

Algorithmische Anwendungen: Wir zeigen Model Checking ist ...

- ... lösbar in Zeit $f(|\varphi|) \cdot n^{6.00001}$ für jede monadisch stabile Graphklasse.
- ... AW[$*$]-schwer für jede hereditäre Klasse, die nicht monadisch abhängig ist.

Ausblick

Work in Progress:

- Flipper Spiele für monadische Abhängigkeit
- Stabilität und Abhängigkeit für MSO Logik

Weitere Anwendungen der entwickelten Strukturtheorie:

- Kernelisierungsalgorithmen
- Regularity Lemmas
- Ausdrucksstärke FO vs MSO
- Shallow Vertex Minors
- Rekonfigurationsprobleme
- Extension Preservation Theoreme
- neue Graphparameter
- Forking Independence

Publikationen 1/2

1. *Indiscernibles and Flatness in Monadically Stable and Monadically NIP Classes*
   joint work with Jan Dreier, Sebastian Siebertz, Szymon Toruńczyk
   ICALP 2023
2. *Flipper Games for Monadically Stable Graph Classes*
   joint work with Jakub Gajarský, Rose McCarty, Pierre Ohlmann, Michał Pilipczuk, Wojciech Przybyszewski, Sebastian Siebertz, Marek Sokołowski, Szymon Toruńczyk
   ICALP 2023
3. *First-Order Model Checking on Structurally Sparse Graph Classes*
   joint work with Jan Dreier, Sebastian Siebertz
   STOC 2023

Publikationen 2/2

4. *First-Order Model Checking on Monadically Stable Graph Classes*
   joint work with Jan Dreier, Ioannis Eleftheriadis, Rose McCarty, Michał Pilipczuk, Szymon Toruńczyk
   FOCS 2024

5. *Flip-Breakability: A Combinatorial Dichotomy for Monadically Dependent Graph Classes*
   joint work with Jan Dreier, Szymon Toruńczyk
   STOC 2024