



**UNIVERSITÄT
WÜRZBURG**

Seminar: Algorithmen und Logik

Wintersemester 2026/27

Einführungsveranstaltung am 13. Oktober 2026

Lehrstuhl für Informatik I & Professur für Mathematik III

Anton Freund, Dominik Jilg, Philipp Keller, Boris Klemz,
Antonio Lauerbach, Samuel Wolf, Alexander Wolff

Ziele und Ablauf des Seminars

JedeR TeilnehmerIn arbeitet sich in ein abgegrenztes Thema ein. Dieses ist didaktisch aufzubereiten und den anderen KursteilnehmerInnen in einem **Vortrag** zu vermitteln, sowie in einer **schriftliche Ausarbeitung** darzustellen.

Ziele und Ablauf des Seminars

- Di, 13.10.2026: **Einführung**

Ziele und Ablauf des Seminars

- Di, 13.10.2026: **Einführung**
- Di, 20.10.2026: **Kurzvorträge** zu jedem Thema (etwa 5 Min., ca. 3 Folien)

Inhalte:

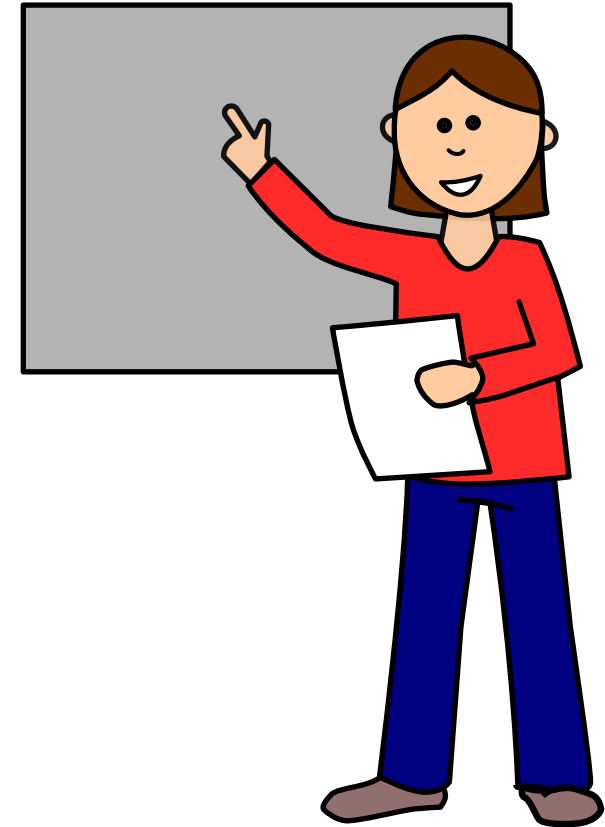
- Ausblick auf den eigentlichen Vortrag geben
- Problemstellung nennen & motivieren
- Wichtigste Resultate nennen & einordnen

Ziele:

- Zeitnah einarbeiten
- Themenauswahl prüfen
- Vortragen üben
- Feedback bekommen ohne Bewertung

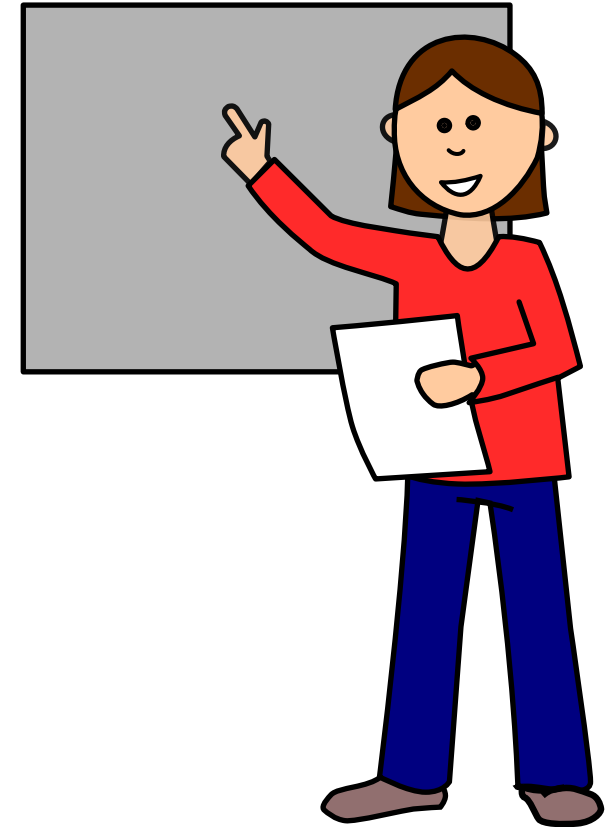
Ziele und Ablauf des Seminars

- Di, 13.10.2026: **Einführung**
- Di, 20.10.2026: **Kurzvorträge** zu jedem Thema (etwa 5 Min., ca. 3 Folien)
- Di, 03.11.2026: **Tutorial** von uns



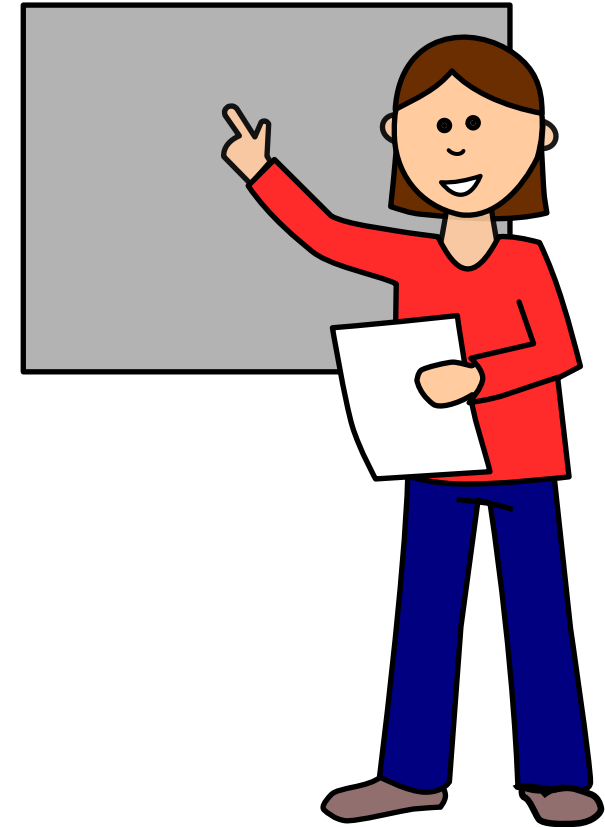
Ziele und Ablauf des Seminars

- Di, 13.10.2026: **Einführung**
- Di, 20.10.2026: **Kurzvorträge** zu jedem Thema (etwa 5 Min., ca. 3 Folien)
- Di, 03.11.2026: **Tutorial** von uns
- ab Di, 10.11.2026: **Hauptvorträge** (i.d.R. einer pro Woche)



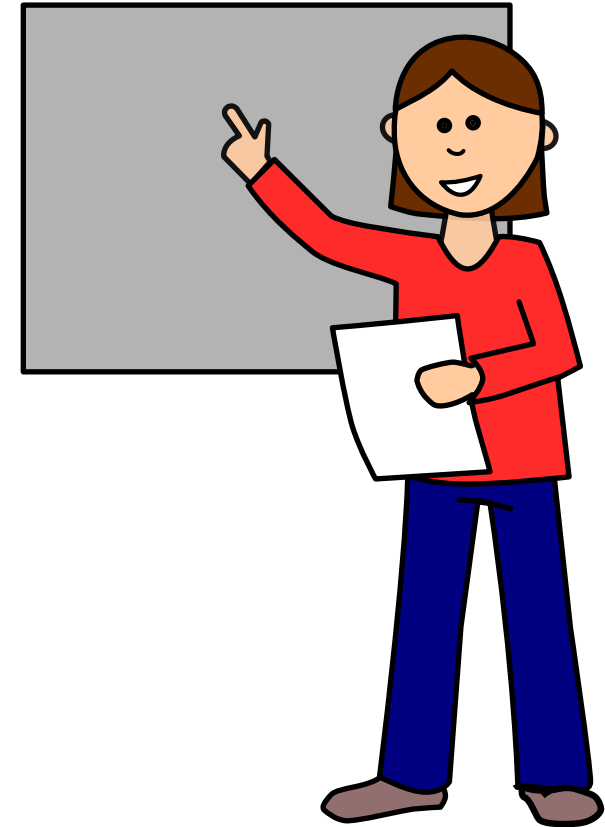
Ziele und Ablauf des Seminars

- Di, 13.10.2026: **Einführung**
- Di, 20.10.2026: **Kurzvorträge** zu jedem Thema (etwa 5 Min., ca. 3 Folien)
- Di, 03.11.2026: **Tutorial** von uns
- ab Di, 10.11.2026: **Hauptvorträge** (i.d.R. einer pro Woche)
- Mo, 08.02.2027: **Ausarbeitungen** abgeben



Ziele und Ablauf des Seminars

- Di, 13.10.2026: **Einführung**
- Di, 20.10.2026: **Kurzvorträge** zu jedem Thema (etwa 5 Min., ca. 3 Folien)
- Di, 03.11.2026: **Tutorial** von uns
- ab Di, 10.11.2026: **Hauptvorträge** (i.d.R. einer pro Woche)
- Mo, 08.02.2027: **Ausarbeitungen** abgeben
- irgendwann zwischendurch: Einführung in **Lean** + Übungssession



Introduction to LEAN

[nature](#) > [news](#) > article

NEWS | 07 September 2026

Anthropic AI ‘formalizes’ proof of Fermat’s last theorem – a milestone for mathematics

Claude produced a 13-million-line, computer-checked proof of the famed conjecture in just 11 days.

Mathematicians amazed

Mathematicians have been increasingly astonished by the pace at which AI’s mathematical skills have improved. This includes the technology’s ability to ‘formalize’ proofs – translate mathematical arguments from natural language into formal, computer-certifiable code, typically in an open-source programming language called [Lean](#).

Introduction to LEAN

[nature](#) > [news](#) > [article](#)

NEWS | 07 September 2026

Anthropic AI ‘for Fermat’s last theorem’ for mathematical reasoning

Claude produced a 13-million-line proof in just 11 days.

Mathematicians amazed

Mathematicians have been increasingly amazed as AI skills have improved. This includes mathematical arguments from formal languages typically in an open-source programming language.

Article

Olympiad-level formal mathematical reasoning with reinforcement learning

<https://doi.org/10.1038/s41586-025-09833-y>

Received: 3 June 2025

Accepted: 30 October 2025

Published online: 12 November 2025

Open access

 Check for updates

Thomas Hubert^{1✉}, Rishi Mehta¹, Laurent Sartran¹, Miklós Z. Horváth¹, Goran Žužić¹, Eric Wieser^{1✉}, Aja Huang¹, Julian Schrittwieser¹, Yannick Schroecker¹, Hussain Masoom¹, Ottavia Bertolli¹, Tom Zahavy¹, Amol Mandhane¹, Jessica Yung¹, Iuliya Beloshapka¹, Borja Ibarz¹, Vivek Veeriah¹, Lei Yu¹, Oliver Nash¹, Paul Lezeau¹, Salvatore Mercuri¹, Calle Sönne¹, Bhavik Mehta¹, Alex Davies¹, Daniel Zheng¹, Fabian Pedregosa¹, Yin Li¹, Ingrid von Glehn¹, Mark Rowland¹, Samuel Albanie¹, Ameya Velingker¹, Simon Schmitt¹, Edward Lockhart¹, Edward Hughes¹, Henryk Michalewski¹, Nicolas Sonnerat¹, Demis Hassabis¹, Pushmeet Kohli^{1✉} & David Silver¹

A long-standing goal of artificial intelligence (AI) is to build systems capable of complex reasoning in vast domains, a task epitomized by mathematics with its boundless concepts and demand for rigorous proof. Recent AI systems, often reliant on human data, typically lack the formal verification necessary to guarantee correctness. By contrast, formal languages such as **Lean** offer an interactive environment that grounds reasoning, and reinforcement learning (RL) provides a mechanism for learning in such environments. Here we present AlphaProof, an AlphaZero-inspired²

Introduction to LEAN

[nature](#) > [news](#) > [article](#)

NEWS | 07 September 2026

Anthropic AI 'cracks' Fermat's last theorem for mathematicians

Claude produced a 13-million-line proof in just 11 days.

Mathematicians amazed

Mathematicians have been incredulous that AI systems capable of complex mathematics with its boundless power are necessary to guarantee correctness. As **Lean** offers an interactive environment that combines reinforcement learning (RL) with a mechanism for formal verification, here we present AlphaProof, an AlphaZero-inspired

Article

Olympiad-level formal reasoning with

<https://doi.org/10.1016/j.sci.2026.101611>

Received: 3 June 2026

Accepted: 30 October 2026

Published online: 12 November 2026

Open access

 Check for updates



Terence Tao

tao@mathstodon.xyz

I have just launched a "**Lean** companion" to my real analysis undergraduate textbook "Analysis I" at github.com/teorth/analysis. This gives a Lean translation (or paraphrasing) of the various definitions, theorems, and exercises in the textbook into Lean, thus allowing for an alternate way for students to work through the text. It is also designed to gradually transition into the standard Lean library Mathlib, thus also potentially serving as an introduction to that library as well.

Further discussion at terrytao.wordpress.com/2025/05/05/

AI systems capable of complex mathematics with its boundless power are necessary to guarantee correctness. As **Lean** offers an interactive environment that combines reinforcement learning (RL) with a mechanism for formal verification, here we present AlphaProof, an AlphaZero-inspired

Introduction to LEAN

[nature](#) > [news](#) > article

NEWS | 07 September 2026

Anthropic AI ‘for Fermat’s last theorem’ for mathematicians

Claude produced a 13-million-line proof in just 11 days.

Mathematicians amazed

Mathematicians have been incredible. Their skills have improved. This includes mathematical arguments from typically in an open-source programming language.

Article

Olympiad-level formal reasoning with

<https://doi.org/10.1093>

Received: 3 June 2026

Accepted: 30 October 2026

Published online: 12 November 2026

Open access

 Check for updates



Terence Tao

@tao@mathstodon.xyz

I have just launched a "Lean companion" to my real analysis undergraduate textbook "Analysis I" at github.com/teorth/analysis. This gives a Lean translation (or paraphrasing) of the various definitions, theorems, and exercises allowing for an alternate way for formalizing theorems. It is also designed to gradually transition into Mathlib, thus also potentially useful as well.

Further discussion at [terrytao](#)



Terence Tao

@tao@mathstodon.xyz

As a consequence of my [#Lean4](#) formalization project I have found a small (but non-trivial) bug in my paper! While in the course of formalizing the arguments in page 6 of arxiv.org/pdf/2310.05328.pdf, I discovered that the expression $\frac{1}{2} \log \frac{n-1}{n-k-1}$ that appears in those arguments actually diverges in the case $n = 3, k = 2$! Fortunately this is an issue that is only present for small values of n , for which one can argue directly (with a worse constant), so I can fix the argument by changing some of the numerical constants on this page (the arguments here still work fine for $n \geq 8$, and the small n case can be handled by cruder methods).

...an AlphaZero-inspired²

Introduction to LEAN

nature > news > article

NEWS | 07 September 2026

Anthropic AI ‘for Fermat’s last theorem’ for mathematical reasoning

Claude produced a 13-million-line proof in just 11 days.

Mathematicians amazed

Mathematicians have been increasingly amazed as AI systems demonstrate skills that have improved. This includes formalizing mathematical arguments from scratch, typically in an open-source programming language.

Article

Olympiad-level formal reasoning with

https://doi.org/10.1093/natrev/nr1234

Received: 3 June 2026

Accepted: 30 October 2026

Published online: 12 November 2026

Open access

Check for updates

Terence Tao
@tao@mathstodon.xyz

I have just launched a "**Lean** companion" to my real analysis undergraduate textbook "Analysis I" at github.com/teorth/analysis. This gives a Lean translation (or paraphrasing) of the various definitions, theorems, and exercises, allowing for an alternate way for formalizing the arguments in page 6 of [arxiv](#). I discovered that the expression $\frac{1}{2} \log \frac{n-1}{n-k}$ arguments actually diverges in the case $n = k$, which is an issue that is only present for small values of n . I will argue directly (with a worse constant), so I am changing some of the numerical constants on this page (the arguments here still work fine for $n \geq 8$, and the small n case can be handled by cruder methods).

Terence Tao
@tao@mathstodon.xyz

As a consequence of my **#Lean4** formalization of the arguments in page 6 of [arxiv](#), I discovered that the expression $\frac{1}{2} \log \frac{n-1}{n-k}$ arguments actually diverges in the case $n = k$, which is an issue that is only present for small values of n . I will argue directly (with a worse constant), so I am changing some of the numerical constants on this page (the arguments here still work fine for $n \geq 8$, and the small n case can be handled by cruder methods).

Further discussion at [terrytao](#)

... as **Lean** reinforcement learning

... comments. Here we go

... an AlphaZero-inspired

CSLib: The **Lean** Computer Science Library

Clark Barrett^{1,2}, Swarat Chaudhuri^{3,4}, Fabrizio Montesi⁵, Jim Grundy¹, Pushmeet Kohli³, Leonardo de Moura^{6,1}, Alexandre Rademaker^{7,8}, and Sorrachai Yingchareonthawornchai⁹

- ¹Amazon
- ²Stanford University
- ³Google DeepMind
- ⁴The University of Texas at Austin
- ⁵FORM, University of Southern Denmark
- ⁶Lean FRO
- ⁷Atlas Computing
- ⁸Getulio Vargas Foundation
- ⁹Institute for Theoretical Studies, ETH Zürich

Abstract

We introduce CSLib, an open-source framework for proving computer-science-related theorems and writing formally verified code in the Lean proof assistant. CSLib aims to be for computer science what Lean's Mathlib is for mathematics. Mathlib has been tremendously impactful: it is a key reason for Lean's popularity within the mathematics research community, and it has also played a critical role in the training of AI systems for mathematical reasoning. However, the base of computer science knowledge in Lean is currently quite limited. CSLib will vastly enhance this knowledge base and provide infrastructure for using this knowledge in real-world verification projects. By doing so, CSLib will (1) enable the broad use of Lean in computer science education and research, and (2) facilitate the manual and AI-aided engineering of large-scale formally verified systems.

Introduction to LEAN

[nature](#) > [news](#) > article

NEWS | 07 September 2026

October 6, 2026 Research Publication

Sharing AI progress in mathematics

[View on GitHub ↗](#)

▶ Listen to article 2:04

🔗 Share

We're releasing a broad range of new mathematical results produced by an [internal frontier model](#).

Article

Olympiad-level formal reasoning with

Terence Tao
@tao@mathstodon.xyz

I launched a "[Lean](#) companion" to my real analysis graduate textbook "Analysis I" at github.com/teorth/analysis. It provides a Lean translation (or paraphrasing) of the various propositions, theorems, and exercises, designed to gradually transform the textbook into Lean, thus providing an alternate way for formalizing the text. It is also potentially useful as a library.

Further discussion at [terrytao.com](#)



Terence Tao
@tao@mathstodon.xyz

As a consequence of my [#Lean4](#) formalization of a small (but non-trivial) bug in my paper! While formalizing the arguments in page 6 of [arxiv](#), I discovered that the expression $\frac{1}{2} \log \frac{n-1}{n-k}$ arguments actually diverges in the case $n = k$, which is an issue that is only present for small values of n . I argue directly (with a worse constant), so I'm changing some of the numerical constants on this page (the arguments here still work fine for $n \geq 8$, and the small n case can be handled by cruder methods).

CSLib: The [Lean](#) Computer Science Library

Clark Barrett^{1,2}, Swarat Chaudhuri^{3,4}, Fabrizio Montesi⁵, Jim Grundy¹, Pushmeet Kohli³, Leonardo de Moura^{6,1}, Alexandre Rademaker^{7,8}, and Sorrachai Yingchareonthawornchai⁹

- ¹Amazon
- ²Stanford University
- ³Google DeepMind
- ⁴The University of Texas at Austin
- ⁵FORM, University of Southern Denmark
- ⁶Lean FRO
- ⁷Atlas Computing
- ⁸Getulio Vargas Foundation
- ⁹Institute for Theoretical Studies, ETH Zürich

Abstract

We introduce CSLib, an open-source framework for proving computer-science-related theorems and writing formally verified code in the Lean proof assistant. CSLib aims to be for computer science what Lean's Mathlib is for mathematics. Mathlib has been tremendously impactful: it is a key reason for Lean's popularity within the mathematics research community, and it has also played a critical role in the training of AI systems for mathematical reasoning. However, the base of computer science knowledge in Lean is currently quite limited. CSLib will vastly enhance this knowledge base and provide infrastructure for using this knowledge in real-world verification projects. By doing so, CSLib will (1) enable the broad use of Lean in computer science education and research, and (2) facilitate the manual and AI-aided engineering of large-scale formally verified systems.

Introduction to L3A7N

October 6, 2026 Research Publication

Sharing AI progress in mathematics

[View on GitHub ↗](#)

 Listen to article 2:04  Share

We're releasing a broad range of new mathematical results produced by an [internal frontier model](#).

Introduction to L_{EA}N

October 6, 2026 Research Publication

Sharing AI progress in mathematics

[View on GitHub ↗](#)

▶ Listen to article 2:04

🔗 Share

We're releasing a broad range of new mathematical results produced by an [internal frontier model](#).

102. The Unique Games Conjecture and optimal approximation thresholds. Proves Khot’s Unique Games Conjecture. Independent direct reductions also establish NP-hardness, on unweighted graphs, of approximation beyond the Goemans–Williamson ratio for Max-Cut, below factor two for Vertex Cover, and within any fixed constant factor for Min-UnCut and directed feedback vertex set. These direct proofs use established PCP and Label Cover hardness results.
[Unique Games theorem](#) · [Optimal Max-Cut hardness](#) · [Factor-two Vertex Cover hardness](#) · [Constant-factor Min-UnCut hardness](#) · [Directed feedback vertex set hardness](#)

Introduction to LEAN

October 6, 2026 Research Publication

Sharing AI progress in mathematics

View on GitHub ↗

▶ Listen to article 2:04

🔗 Share

We're releasing a broad range of new mathematical results produced by an [internal frontier model](#).

102. The Unique Games Conjecture and optimal approximation thresholds. Proves Khot’s Unique Games Conjecture. Independent direct reductions also establish NP-hardness, on unweighted graphs, of approximation beyond the Goemans–Williamson ratio for Max-Cut, below factor two for Vertex Cover, and within any fixed constant factor for Min-UnCut and directed feedback vertex set. These direct proofs use established PCP and Label Cover hardness results.

[Unique Games theorem](#) · [Optimal Max-Cut hardness](#) · [Factor-two Vertex Cover hardness](#) · [Constant-factor Min-UnCut hardness](#) · [Directed feedback vertex set hardness](#)

```
8 namespace UniqueGamesTheorem.Theorem
9
10 theorem exists_binaryGapReduction (ε δ : ℝ)
11   (hε : 0 < ε) (hεhalf : ε < 1 / 2)
12   (hδ : 0 < δ) (hδhalf : δ < 1 / 2) :
13     Nonempty (Explicit.MachineOutputContract.BinaryGapReduction ε δ) := by
14   obtain ⟨P⟩ := ParameterSelection.exists_outerParameters ε δ hε hδ
15   obtain ⟨M⟩ := Decoder.MatrixGap.exists_parameters P.pStar P.pStar_pos
16   exact ⟨FromMatrixGap.reduction P M⟩
17
18 end UniqueGamesTheorem.Theorem
19
20 namespace UniqueGamesTheorem
21
22
23 /-- For every fixed pair of errors in `(0, 1/2)`, binary 3SAT reduces in
24 polynomial time to translation Unique Games with completeness at least `1 - ε`
25 and soundness at most `δ`. The alphabet and machine depend only on the errors. -/
26 theorem theorem11 (ε δ : ℝ)
27   (hε : 0 < ε) (hεhalf : ε < 1 / 2)
28   (hδ : 0 < δ) (hδhalf : δ < 1 / 2) :
29     Nonempty (Explicit.MachineOutputContract.BinaryGapReduction ε δ) :=
30   Theorem.exists_binaryGapReduction ε δ hε hεhalf hδ hδhalf
31
32 end UniqueGamesTheorem
```

Introduction to LEAN

October 6, 2026 Research Publication

Sharing AI progress in mathematics

[View on GitHub ↗](#)

▶ Listen to article 2:04

🔗 Share

We're releasing a broad range of new mathematical results produced by an [internal frontier model](#).

In this seminar:

- A small introduction to LEAN

102. The Unique Games Conjecture and optimal approximation thresholds. Proves Khot's Unique Games Conjecture. Independent direct reductions also establish NP-hardness, on unweighted graphs, of approximation beyond the Goemans–Williamson ratio for Max-Cut, below factor two for Vertex Cover, and within any fixed constant factor for Min-UnCut and directed feedback vertex set. These direct proofs use established PCP and Label Cover hardness results.

[Unique Games theorem](#) · [Optimal Max-Cut hardness](#) · [Factor-two Vertex Cover hardness](#) · [Constant-factor Min-UnCut hardness](#) · [Directed feedback vertex set hardness](#)

```
8 namespace UniqueGamesTheorem.Theorem
9
10 theorem exists_binaryGapReduction (ε δ : ℝ)
11   (hε : 0 < ε) (hεhalf : ε < 1 / 2)
12   (hδ : 0 < δ) (hδhalf : δ < 1 / 2) :
13     Nonempty (Explicit.MachineOutputContract.BinaryGapReduction ε δ) := by
14   obtain ⟨P⟩ := ParameterSelection.exists_outerParameters ε δ hε hδ
15   obtain ⟨M⟩ := Decoder.MatrixGap.exists_parameters P.pStar P.pStar_pos
16   exact ⟨FromMatrixGap.reduction P M⟩
17
18 end UniqueGamesTheorem.Theorem
19
20 namespace UniqueGamesTheorem
21
22
23 /-- For every fixed pair of errors in `(0, 1/2)`, binary 3SAT reduces in
24 polynomial time to translation Unique Games with completeness at least `1 - ε`
25 and soundness at most `δ`. The alphabet and machine depend only on the errors. -/
26 theorem theorem11 (ε δ : ℝ)
27   (hε : 0 < ε) (hεhalf : ε < 1 / 2)
28   (hδ : 0 < δ) (hδhalf : δ < 1 / 2) :
29     Nonempty (Explicit.MachineOutputContract.BinaryGapReduction ε δ) :=
30   Theorem.exists_binaryGapReduction ε δ hε hεhalf hδ hδhalf
31
32 end UniqueGamesTheorem
```

Introduction to LEAN

October 6, 2026 Research Publication

Sharing AI progress in mathematics

[View on GitHub ↗](#)

▶ Listen to article 2:04

🔗 Share

We're releasing a broad range of new mathematical results produced by an [internal frontier model](#).

In this seminar:

- A small introduction to LEAN
- Hands-on exercise session

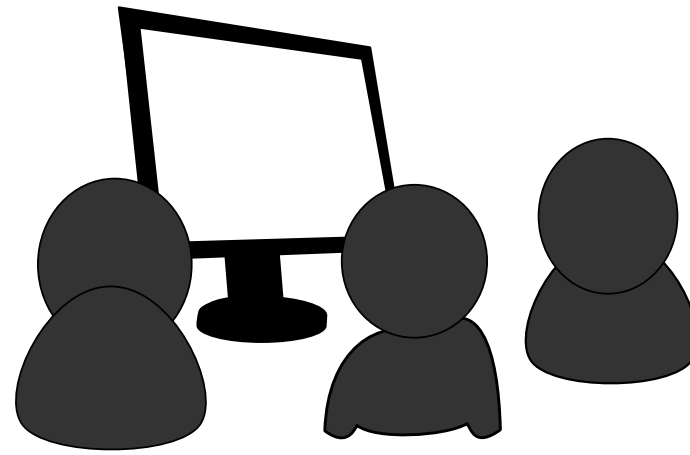
102. The Unique Games Conjecture and optimal approximation thresholds. Proves Khot's Unique Games Conjecture. Independent direct reductions also establish NP-hardness, on unweighted graphs, of approximation beyond the Goemans–Williamson ratio for Max-Cut, below factor two for Vertex Cover, and within any fixed constant factor for Min-UnCut and directed feedback vertex set. These direct proofs use established PCP and Label Cover hardness results.

[Unique Games theorem](#) · [Optimal Max-Cut hardness](#) · [Factor-two Vertex Cover hardness](#) · [Constant-factor Min-UnCut hardness](#) · [Directed feedback vertex set hardness](#)

```
8 namespace UniqueGamesTheorem.Theorem
9
10 theorem exists_binaryGapReduction (ε δ : ℝ)
11   (hε : 0 < ε) (hεhalf : ε < 1 / 2)
12   (hδ : 0 < δ) (hδhalf : δ < 1 / 2) :
13     Nonempty (Explicit.MachineOutputContract.BinaryGapReduction ε δ) := by
14   obtain ⟨P⟩ := ParameterSelection.exists_outerParameters ε δ hε hδ
15   obtain ⟨M⟩ := Decoder.MatrixGap.exists_parameters P.pStar P.pStar_pos
16   exact ⟨FromMatrixGap.reduction P M⟩
17
18 end UniqueGamesTheorem.Theorem
19
20 namespace UniqueGamesTheorem
21
22
23 /-- For every fixed pair of errors in `(0, 1/2)`, binary 3SAT reduces in
24 polynomial time to translation Unique Games with completeness at least `1 - ε`
25 and soundness at most `δ`. The alphabet and machine depend only on the errors. -/
26 theorem theorem11 (ε δ : ℝ)
27   (hε : 0 < ε) (hεhalf : ε < 1 / 2)
28   (hδ : 0 < δ) (hδhalf : δ < 1 / 2) :
29     Nonempty (Explicit.MachineOutputContract.BinaryGapReduction ε δ) :=
30   Theorem.exists_binaryGapReduction ε δ hε hεhalf hδ hδhalf
31
32 end UniqueGamesTheorem
```

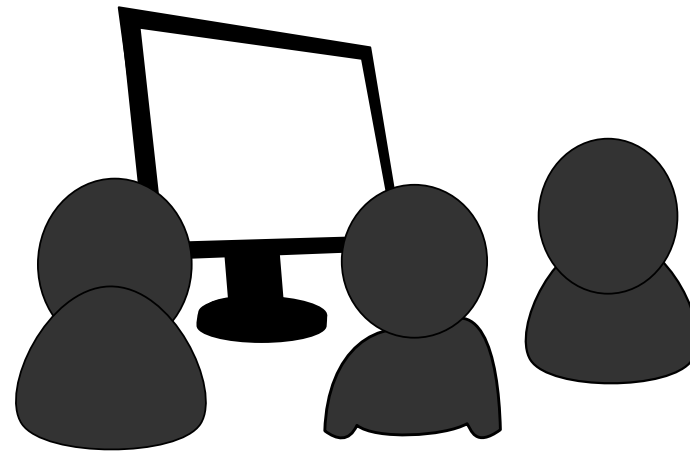
Vorträge

- etwa 45 Minuten **Vortrag**
(zu zweit etwa 60 Minuten)



Vorträge

- etwa 45 Minuten **Vortrag**
(zu zweit etwa 60 Minuten)

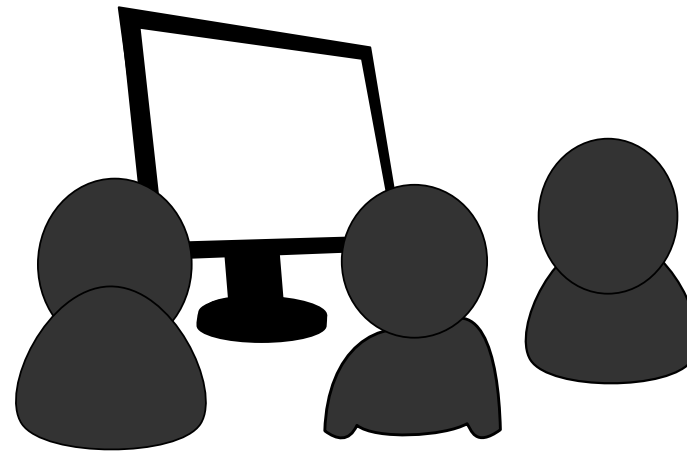


Das reicht i.d.R. nicht um alles im Detail zu besprechen!

→ wesentliche Teile identifizieren und ausführlich
behandeln, unwesentliche Teile skizzieren

Vorträge

- etwa 45 Minuten **Vortrag**
(zu zweit etwa 60 Minuten)



Das reicht i.d.R. nicht um alles im Detail zu besprechen!

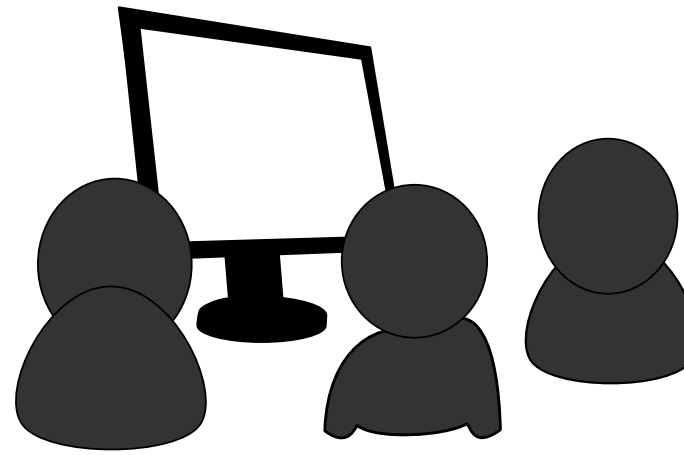
→ wesentliche Teile identifizieren und ausführlich behandeln, unwesentliche Teile skizzieren

Ausnahme: Einige Themen sind weniger umfangreich

→ verbleibende Zeit durch Inhalte angrenzender Literatur füllen (eigene Literaturrecherche!)

Vorträge

- etwa 45 Minuten **Vortrag**
(zu zweit etwa 60 Minuten)



Das reicht i.d.R. nicht um alles im Detail zu besprechen!

→ wesentliche Teile identifizieren und ausführlich behandeln, unwesentliche Teile skizzieren

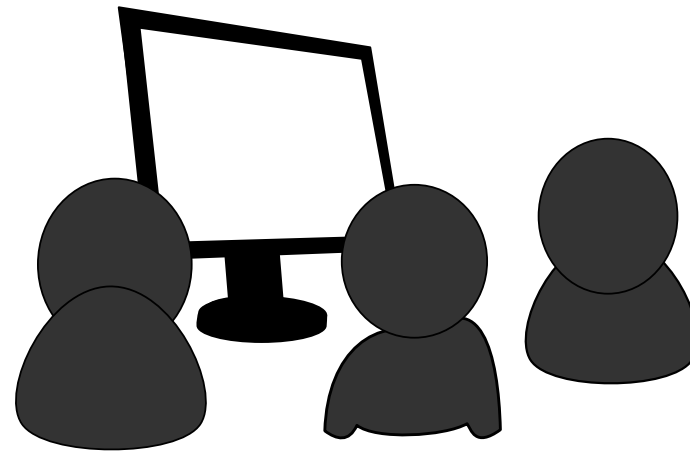
Ausnahme: Einige Themen sind weniger umfangreich

→ verbleibende Zeit durch Inhalte angrenzender Literatur füllen (eigene Literaturrecherche!)

In jedem Fall sollen die 45 / 60 Minuten stimmig ausgefüllt werden.

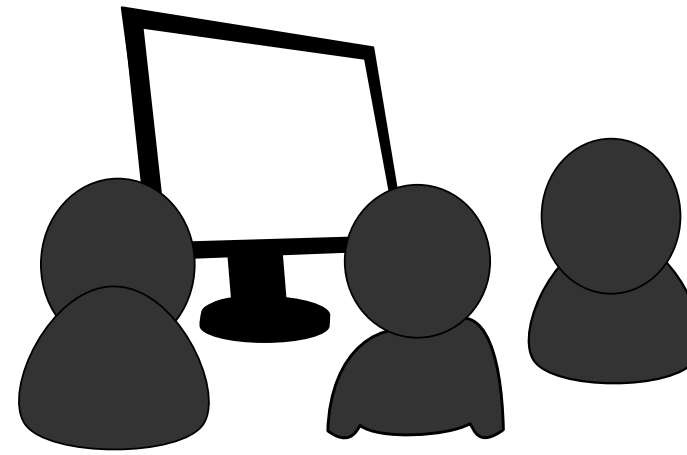
Vorträge

- etwa 45 Minuten **Vortrag**
(zu zweit etwa 60 Minuten)



Vorträge

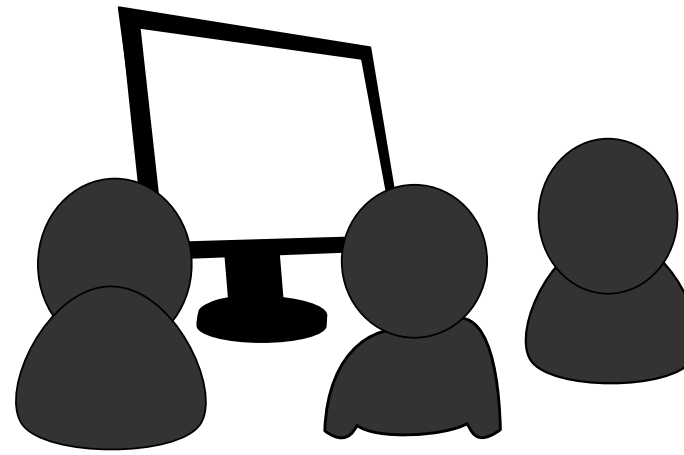
- etwa 45 Minuten **Vortrag**
(zu zweit etwa 60 Minuten)



- anschließend / währenddessen **Diskussion / Interaktion**
(Übungsaufgaben, interaktive Beispiele, Besprechung offener Probleme, etc.) (geht nicht in die Zeit ein)

Ideen aus der Diskussion in die Ausarbeitung mitaufnehmen!

Vorträge



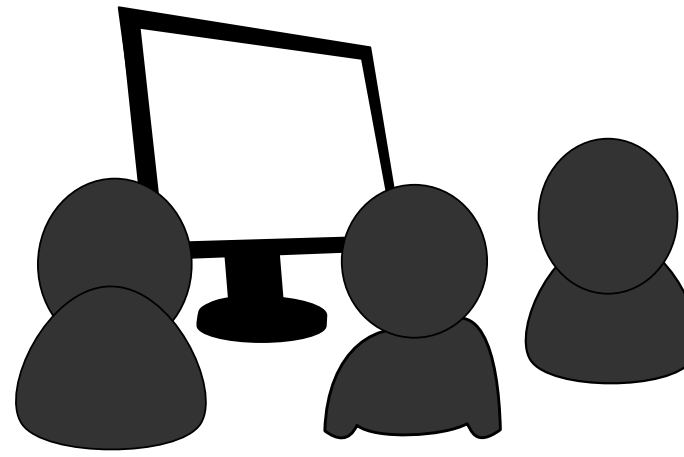
- etwa 45 Minuten **Vortrag**
(zu zweit etwa 60 Minuten)
- anschließend / währenddessen **Diskussion / Interaktion**
(Übungsaufgaben, interaktive Beispiele, Besprechung offener Probleme, etc.) (geht nicht in die Zeit ein)

Ideen aus der Diskussion in die Ausarbeitung mitaufnehmen!

Vorbesprechungen (verpflichtend):

- **Drei** Wochen vor dem eigenen Vortrag:
Besprechung der **Inhaltsübersicht** mit eurer BetreuerIn

Vorträge



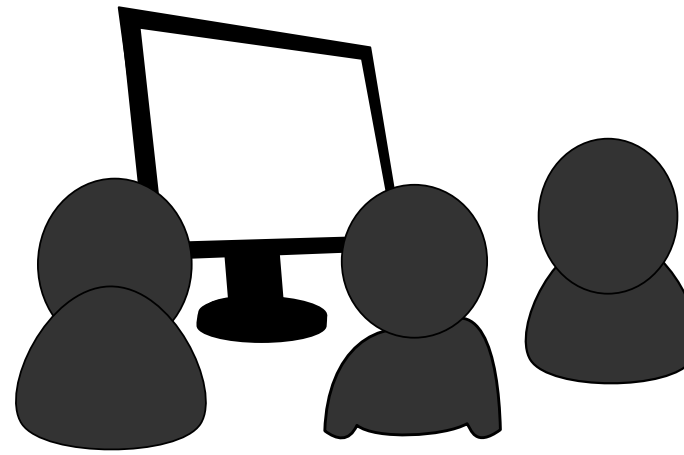
- etwa 45 Minuten **Vortrag**
(zu zweit etwa 60 Minuten)
- anschließend / währenddessen **Diskussion / Interaktion**
(Übungsaufgaben, interaktive Beispiele, Besprechung offener Probleme, etc.) (geht nicht in die Zeit ein)

Ideen aus der Diskussion in die Ausarbeitung mitaufnehmen!

Vorbesprechungen (verpflichtend):

- **Drei** Wochen vor dem eigenen Vortrag:
Besprechung der **Inhaltsübersicht** mit eurer BetreuerIn
- **Zwei** Wochen vor dem Vortrag:
Besprechung eurer **Folien** mit eurer BetreuerIn

Vorträge



- etwa 45 Minuten **Vortrag**
(zu zweit etwa 60 Minuten)
- anschließend / währenddessen **Diskussion / Interaktion**
(Übungsaufgaben, interaktive Beispiele, Besprechung offener Probleme, etc.) (geht nicht in die Zeit ein)

Ideen aus der Diskussion in die Ausarbeitung mitaufnehmen!

Vorbesprechungen (verpflichtend):

- **Drei Wochen** vor dem eigenen Vortrag:
Besprechung der **Inhaltsübersicht** mit eurer BetreuerIn
- **Zwei Wochen** vor dem Vortrag:
Besprechung eurer **Folien** mit eurer BetreuerIn

**Diese Termine sind strikt
(außer für den 1. Vortrag)!**

Ausarbeitung

- alleine 7–9, zu zweit 11–13 Seiten;



Ausarbeitung

- alleine 7–9, zu zweit 11–13 Seiten;

Wie schon beim Vortrag gilt auch hier:

Das reicht i.d.R. nicht um alles im Detail zu beschreiben!

→ wesentliche Teile identifizieren und ausführlich behandeln, unwesentliche Teile skizzieren



Ausarbeitung

- alleine 7–9, zu zweit 11–13 Seiten;

Wie schon beim Vortrag gilt auch hier:

Das reicht i.d.R. nicht um alles im Detail zu beschreiben!

→ wesentliche Teile identifizieren und ausführlich behandeln, unwesentliche Teile skizzieren

Ausnahme: Einige Themen sind weniger umfangreich.

→ durch geeignete eigene Inhalte erweitern

(siehe nächste Folie)



Ausarbeitung

- alleine 7–9, zu zweit 11–13 Seiten;

Wie schon beim Vortrag gilt auch hier:

Das reicht i.d.R. nicht um alles im Detail zu beschreiben!

→ wesentliche Teile identifizieren und ausführlich behandeln, unwesentliche Teile skizzieren

Ausnahme: Einige Themen sind weniger umfangreich.

→ durch geeignete eigene Inhalte erweitern

(siehe nächste Folie)

In jedem Fall sollen die 7–9 / 11–13 Seiten stimmig ausgefüllt werden.



Ausarbeitung

- alleine 7–9, zu zweit 11–13 Seiten;
Abbildungen sind hilfreich!



Ausarbeitung

- alleine 7–9, zu zweit 11–13 Seiten;
Abbildungen sind hilfreich! (und gehen nicht in das Seitenlimit ein)



Ausarbeitung

Bitte Vektorgrafiken, keine Bitmaps!

- alleine 7–9, zu zweit 11–13 Seiten;
Abbildungen sind hilfreich! (und gehen nicht in das Seitenlimit ein)



Ausarbeitung

Bitte Vektorgrafiken, keine Bitmaps!

- alleine 7–9, zu zweit 11–13 Seiten;
Abbildungen sind hilfreich! (und gehen nicht in das Seitenlimit ein)
- **keine reine Zusammenfassung** des Artikels; wir erwarten einen **eigenen Beitrag**. Z.B. manche Resultate weglassen, andere Beweise ausführlicher, offene Probleme diskutieren, eigene Literaturrecherche & Material aus angrenzender Literatur, Verbindungen zu anderen Vortragsthemen etc.



Ausarbeitung

Bitte Vektorgrafiken, keine Bitmaps!

- alleine 7–9, zu zweit 11–13 Seiten;
Abbildungen sind hilfreich! (und gehen nicht in das Seitenlimit ein)
- **keine reine Zusammenfassung** des Artikels; wir erwarten einen **eigenen Beitrag**. Z.B. manche Resultate weglassen, andere Beweise ausführlicher, offene Probleme diskutieren, eigene Literaturrecherche & Material aus angrenzender Literatur, Verbindungen zu anderen Vortragsthemen etc.
- L^AT_EX-Vorlage auf der WueCampus Seite!



Ausarbeitung

Bitte Vektorgrafiken, keine Bitmaps!

- alleine 7–9, zu zweit 11–13 Seiten;
Abbildungen sind hilfreich! (und gehen nicht in das Seitenlimit ein)
- **keine reine Zusammenfassung** des Artikels; wir erwarten einen **eigenen Beitrag**. Z.B. manche Resultate weglassen, andere Beweise ausführlicher, offene Probleme diskutieren, eigene Literaturrecherche & Material aus angrenzender Literatur, Verbindungen zu anderen Vortragsthemen etc.
- L^AT_EX-Vorlage auf der WueCampus Seite!
- **Vorabversion** der Ausarbeitung bis spätestens 2 Wochen nach dem eigenen Vortrag abgeben, um Feedback zu erhalten (freiwillig).



Bestehen & Bewertung

Voraussetzungen für das Bestehen des Seminars

- Halten einer Präsentation zum gewählten Thema
- Anfertigen einer Ausarbeitung
- Teilnahme an den Diskussionen

Bestehen & Bewertung

Voraussetzungen für das Bestehen des Seminars

- Halten einer Präsentation zum gewählten Thema
- Anfertigen einer Ausarbeitung
- Teilnahme an den Diskussionen

Bewertung

- Vortrag (Inhalte, Gestaltung der Folien, Verständlichkeit, Interaktivität)
- Ausarbeitung (Inhalte, roter Faden, sprachliche Darstellung, Rechtschreibung, eigener Beitrag)
- 50 : 50

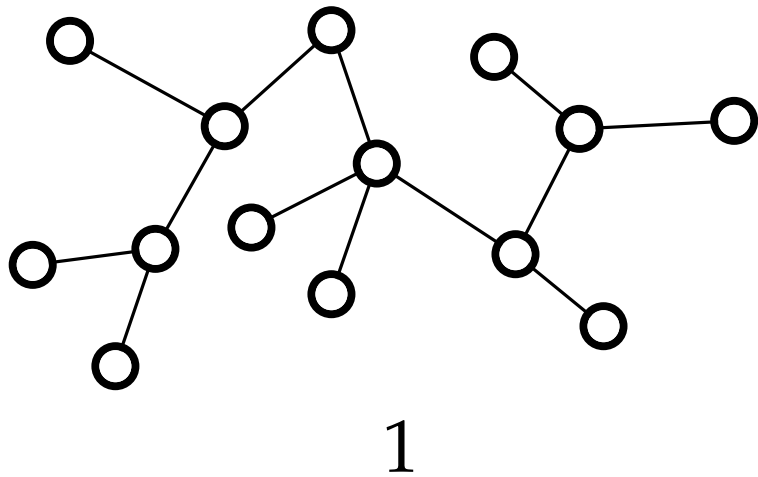
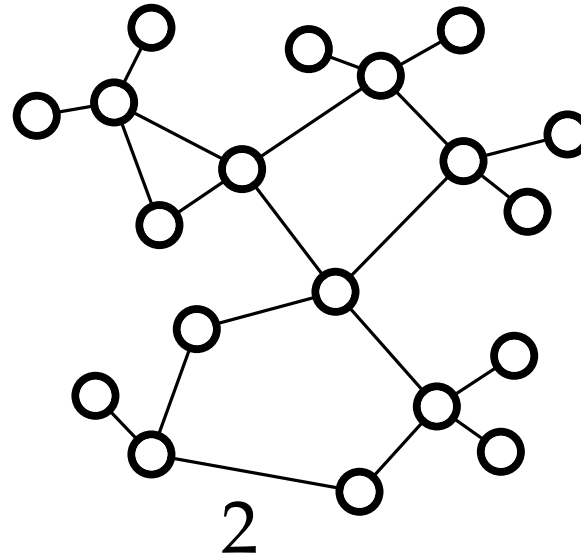
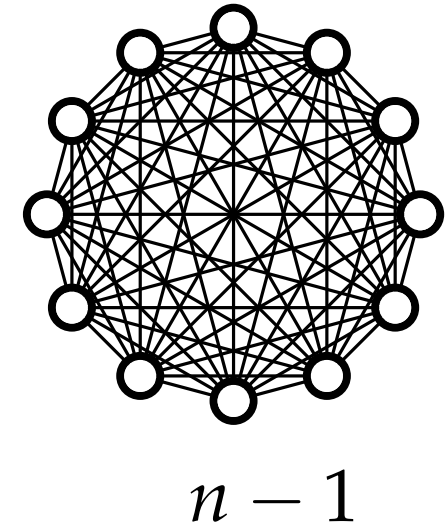
Themenübersicht

1. Monadic Second-Order Logic on Graphs

Die **Baumweite** ist ein Maß dafür, wie ähnlich ein Graph einem Baum ist.

klein = sehr ähnlich

groß = nicht ähnlich

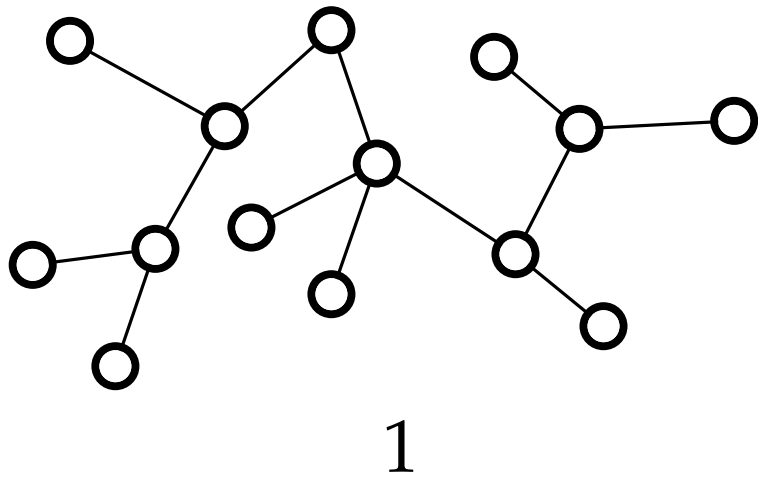
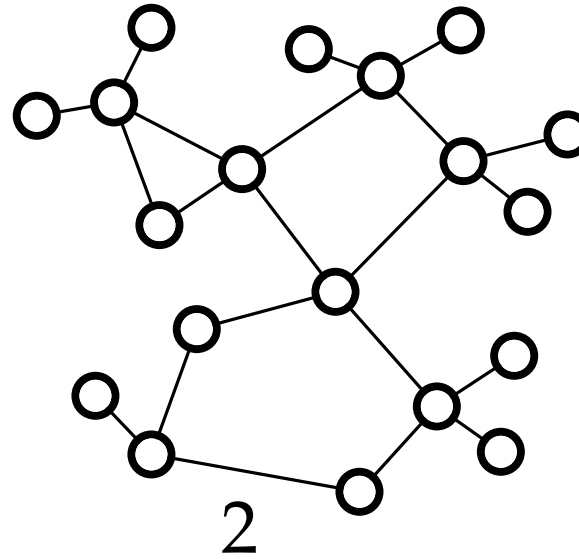
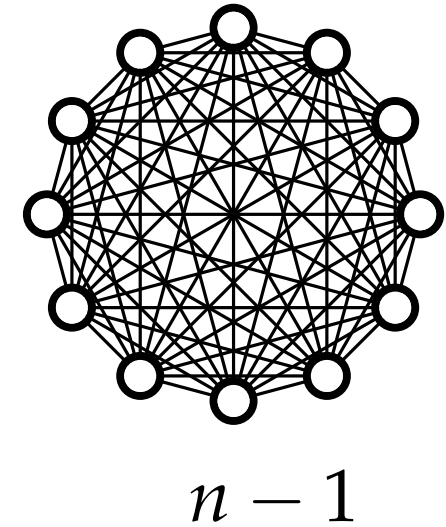


1. Monadic Second-Order Logic on Graphs

Die **Baumweite** ist ein Maß dafür, wie ähnlich ein Graph einem Baum ist.

klein = sehr ähnlich

groß = nicht ähnlich



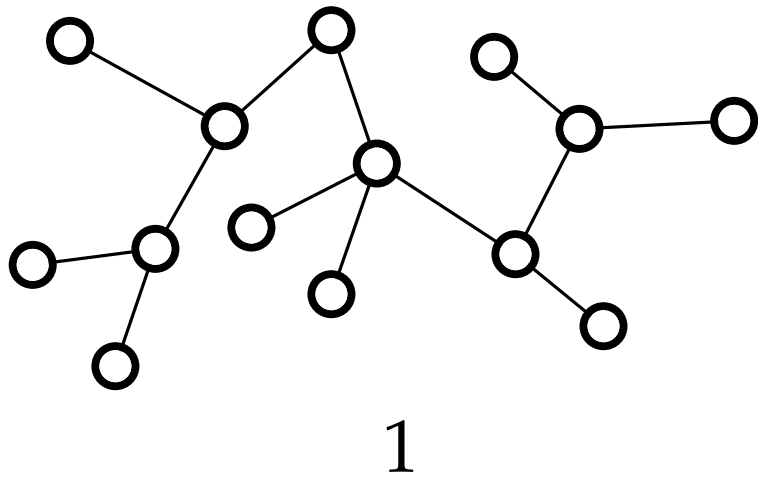
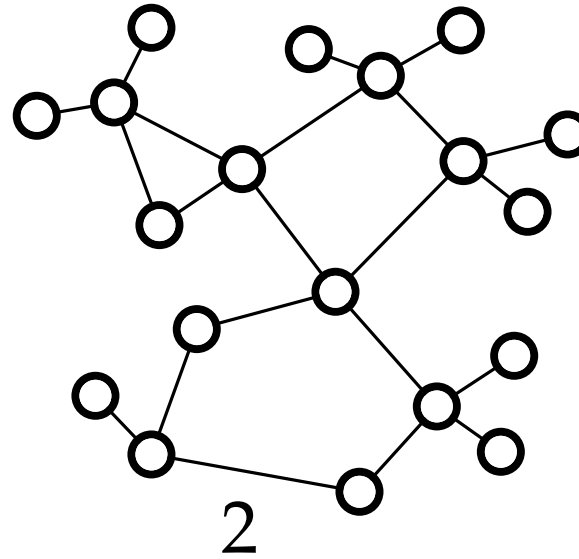
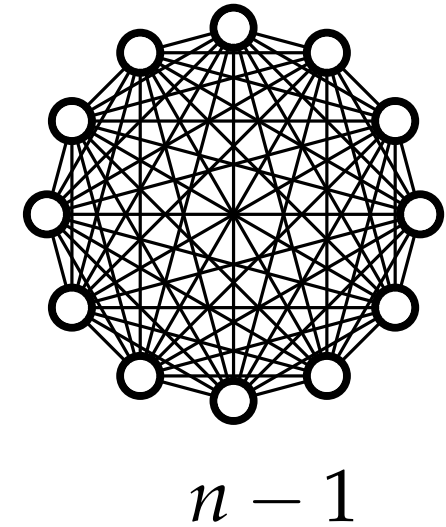
Der **Satz von Courcelle** besagt, dass *alle* graphentheoretischen Problem, die in der sogenannten monadischen Graph-Logik zweiter Stufe formuliert werden können, in Graphen mit kleiner Baumweite effizient gelöst werden können.

1. Monadic Second-Order Logic on Graphs

Die **Baumweite** ist ein Maß dafür, wie ähnlich ein Graph einem Baum ist.

klein = sehr ähnlich

groß = nicht ähnlich



Der **Satz von Courcelle** besagt, dass *alle* graphentheoretischen Problem, die in der sogenannten monadischen Graph-Logik zweiter Stufe formuliert werden können, in Graphen mit kleiner Baumweite effizient gelöst werden können.

Was ist diese Logik und welche Probleme kann man damit formulieren?

2. Metatheorem für Knotenüberdeckungen

Satz von Courcelle: Alle graphentheoretischen Probleme, die in der monadischen Graph-Logik zweiter Stufe (MSO_2) formuliert werden können, lassen sich in Graphen mit fester Baumweite effizient lösen.

2. Metatheorem für Knotenüberdeckungen

Satz von Courcelle: Alle graphentheoretischen Probleme, die in der monadischen Graph-Logik zweiter Stufe (MSO_2) formuliert werden können, lassen sich in Graphen mit fester Baumweite **effizient** lösen.

?

2. Metatheorem für Knotenüberdeckungen

Satz von Courcelle: Alle graphentheoretischen Probleme, die in der monadischen Graph-Logik zweiter Stufe (MSO_2) formuliert werden können, lassen sich in Graphen mit fester Baumweite in Linearzeit lösen.

2. Metatheorem für Knotenüberdeckungen

Satz von Courcelle: Alle graphentheoretischen Probleme, die in der monadischen Graph-Logik zweiter Stufe (MSO_2) formuliert werden können, lassen sich in Graphen mit fester Baumweite **in Linearzeit** lösen.

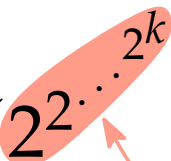
?

2. Metatheorem für Knotenüberdeckungen

Satz von Courcelle: Alle graphentheoretischen Probleme, die in der monadischen Graph-Logik zweiter Stufe (MSO_2) formuliert werden können, lassen sich in Graphen mit Baumweite k in $O(2^{2^{\dots^{2^k}}} \cdot n)$ Zeit lösen.

2. Metatheorem für Knotenüberdeckungen

Satz von Courcelle: Alle graphentheoretischen Probleme, die in der monadischen Graph-Logik zweiter Stufe (MSO_2) formuliert werden können, lassen sich in Graphen mit Baumweite k in $O(2^{2^{\dots^{2^k}}} \cdot n)$ Zeit lösen.

 # Quantorenalternierungen

2. Metatheorem für Knotenüberdeckungen

Satz von Courcelle: Alle graphentheoretischen Probleme, die in der monadischen Graph-Logik zweiter Stufe (MSO_2) formuliert werden können,

lassen sich in Graphen mit Baumweite k in $O(2^{2^{\dots^{2^k}}} \cdot n)$ Zeit lösen.

Quantorenalternierungen
lässt sich nicht vermeiden
[Frick & Grohe 04]

2. Metatheorem für Knotenüberdeckungen

[Lampis 12]:

~~Satz von Courcelle:~~ Alle graphentheoretischen Probleme, die in der monadischen Graph-Logik zweiter Stufe (MSO_2) formuliert werden können,

lassen sich in Graphen mit ~~Baumweite~~ k in $O(\overset{2^k}{\cancel{2^{2^k}}} \cdot n)$ Zeit lösen.

Knotenüberdeckungsanzahl 2^{2^k}

2. Metatheorem für Knotenüberdeckungen

[Lampis 12]:

~~Satz von Courcelle:~~ Alle graphentheoretischen Probleme, die in der monadischen Graph-Logik zweiter Stufe (MSO_2) formuliert werden können,

lassen sich in Graphen mit ~~Baumweite~~ k in $O(\overset{\text{Knotenüberdeckungsanzahl}}{2^{2^k}} \cdot n)$ Zeit lösen.

- Wie funktioniert das?
- Warum geht es nicht schneller?

3. Solving 2-SAT in Linear Time

SATISFIABILITY (SAT)

Gegeben: Boolesche Formel in konjunktiver Normalform.

Frage: Gibt es eine erfüllende Belegung?

$$(A \vee \neg B \vee \neg C) \wedge (B \vee C \vee \neg D) \wedge (B \vee \neg D \vee E) \wedge (\neg B \vee D \vee \neg E)$$

3. Solving 2-SAT in Linear Time

SATISFIABILITY (SAT)

Gegeben: Boolesche Formel in konjunktiver Normalform.

Frage: Gibt es eine erfüllende Belegung?

$$(A \vee \neg B \vee \neg C) \wedge (B \vee C \vee \neg D) \wedge (B \vee \neg D \vee E) \wedge (\neg B \vee D \vee \neg E)$$

k -SATISFIABILITY (k -SAT): Wie SAT, aber jede Klausel darf $\leq k$ Literale enthalten.

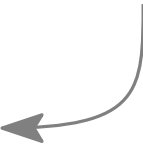
3. Solving 2-SAT in Linear Time

SATISFIABILITY (SAT)

Gegeben: Boolesche Formel in konjunktiver Normalform.

eine 3-SAT Formel

Frage: Gibt es eine erfüllende Belegung?

$$(A \vee \neg B \vee \neg C) \wedge (B \vee C \vee \neg D) \wedge (B \vee \neg D \vee E) \wedge (\neg B \vee D \vee \neg E)$$


k -SATISFIABILITY (k -SAT): Wie SAT, aber jede Klausel darf $\leq k$ Literale enthalten.

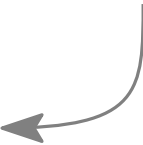
3. Solving 2-SAT in Linear Time

SATISFIABILITY (SAT)

Gegeben: Boolesche Formel in konjunktiver Normalform.

eine 3-SAT Formel

Frage: Gibt es eine erfüllende Belegung?

$$(A \vee \neg B \vee \neg C) \wedge (B \vee C \vee \neg D) \wedge (B \vee \neg D \vee E) \wedge (\neg B \vee D \vee \neg E)$$


k -SATISFIABILITY (k -SAT): Wie SAT, aber jede Klausel darf $\leq k$ Literale enthalten.

- NP-schwer für $k \geq 3$

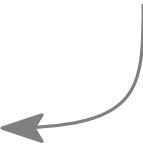
3. Solving 2-SAT in Linear Time

SATISFIABILITY (SAT)

Gegeben: Boolesche Formel in konjunktiver Normalform.

Frage: Gibt es eine erfüllende Belegung?

eine 3-SAT Formel

$$(A \vee \neg B \vee \neg C) \wedge (B \vee C \vee \neg D) \wedge (B \vee \neg D \vee E) \wedge (\neg B \vee D \vee \neg E)$$


k -SATISFIABILITY (k -SAT): Wie SAT, aber jede Klausel darf $\leq k$ Literale enthalten.

- NP-schwer für $k \geq 3$
- Lösbar in Linearzeit für $k \leq 2$

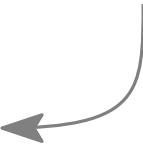
3. Solving 2-SAT in Linear Time

SATISFIABILITY (SAT)

Gegeben: Boolesche Formel in konjunktiver Normalform.

eine 3-SAT Formel

Frage: Gibt es eine erfüllende Belegung?

$$(A \vee \neg B \vee \neg C) \wedge (B \vee C \vee \neg D) \wedge (B \vee \neg D \vee E) \wedge (\neg B \vee D \vee \neg E)$$


k -SATISFIABILITY (k -SAT): Wie SAT, aber jede Klausel darf $\leq k$ Literale enthalten.

- NP-schwer für $k \geq 3$
- Lösbar in Linearzeit für $k \leq 2$ **Hier!**

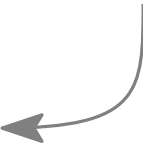
3. Solving 2-SAT in Linear Time

SATISFIABILITY (SAT)

Gegeben: Boolesche Formel in konjunktiver Normalform.

eine 3-SAT Formel

Frage: Gibt es eine erfüllende Belegung?

$$(A \vee \neg B \vee \neg C) \wedge (B \vee C \vee \neg D) \wedge (B \vee \neg D \vee E) \wedge (\neg B \vee D \vee \neg E)$$


k -SATISFIABILITY (k -SAT): Wie SAT, aber jede Klausel darf $\leq k$ Literale enthalten.

- NP-schwer für $k \geq 3$
- Lösbar in Linearzeit für $k \leq 2$ **Hier!**

Hauptidee: Jede 2-SAT-Formel entspricht einem gerichteten Implikationsgraphen.

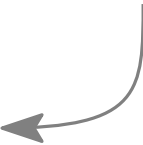
3. Solving 2-SAT in Linear Time

SATISFIABILITY (SAT)

Gegeben: Boolesche Formel in konjunktiver Normalform.

eine 3-SAT Formel

Frage: Gibt es eine erfüllende Belegung?

$$(A \vee \neg B \vee \neg C) \wedge (B \vee C \vee \neg D) \wedge (B \vee \neg D \vee E) \wedge (\neg B \vee D \vee \neg E)$$


k -SATISFIABILITY (k -SAT): Wie SAT, aber jede Klausel darf $\leq k$ Literale enthalten.

- NP-schwer für $k \geq 3$
- Lösbar in Linearzeit für $k \leq 2$ **Hier!**

Hauptidee: Jede 2-SAT-Formel entspricht einem gerichteten Implikationsgraphen.

$$(A \vee B)$$

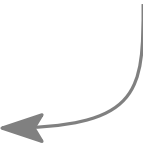
3. Solving 2-SAT in Linear Time

SATISFIABILITY (SAT)

Gegeben: Boolesche Formel in konjunktiver Normalform.

eine 3-SAT Formel

Frage: Gibt es eine erfüllende Belegung?

$$(A \vee \neg B \vee \neg C) \wedge (B \vee C \vee \neg D) \wedge (B \vee \neg D \vee E) \wedge (\neg B \vee D \vee \neg E)$$


k -SATISFIABILITY (k -SAT): Wie SAT, aber jede Klausel darf $\leq k$ Literale enthalten.

- NP-schwer für $k \geq 3$
- Lösbar in Linearzeit für $k \leq 2$ **Hier!**

Hauptidee: Jede 2-SAT-Formel entspricht einem gerichteten Implikationsgraphen.

$$(A \vee B) \rightsquigarrow \begin{array}{l} \neg A \Rightarrow B \\ \neg B \Rightarrow A \end{array}$$

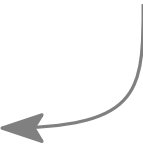
3. Solving 2-SAT in Linear Time

SATISFIABILITY (SAT)

Gegeben: Boolesche Formel in konjunktiver Normalform.

eine 3-SAT Formel

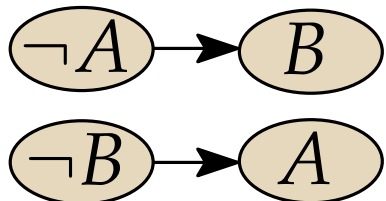
Frage: Gibt es eine erfüllende Belegung?

$$(A \vee \neg B \vee \neg C) \wedge (B \vee C \vee \neg D) \wedge (B \vee \neg D \vee E) \wedge (\neg B \vee D \vee \neg E)$$


k -SATISFIABILITY (k -SAT): Wie SAT, aber jede Klausel darf $\leq k$ Literale enthalten.

- NP-schwer für $k \geq 3$
- Lösbar in Linearzeit für $k \leq 2$ **Hier!**

Hauptidee: Jede 2-SAT-Formel entspricht einem gerichteten Implikationsgraphen.

$$(A \vee B) \rightsquigarrow \begin{array}{l} \neg A \Rightarrow B \\ \neg B \Rightarrow A \end{array} \rightsquigarrow \begin{array}{cc} \text{---} \neg A \text{---} & \longrightarrow & B \text{---} \\ \text{---} \neg B \text{---} & \longrightarrow & A \text{---} \end{array}$$


3. Solving 2-SAT in Linear Time

SATISFIABILITY (SAT)

Gegeben: Boolesche Formel in konjunktiver Normalform.

Frage: Gibt es eine erfüllende Belegung?

eine 3-SAT Formel

$$(A \vee \neg B \vee \neg C) \wedge (B \vee C \vee \neg D) \wedge (B \vee \neg D \vee E) \wedge (\neg B \vee D \vee \neg E)$$

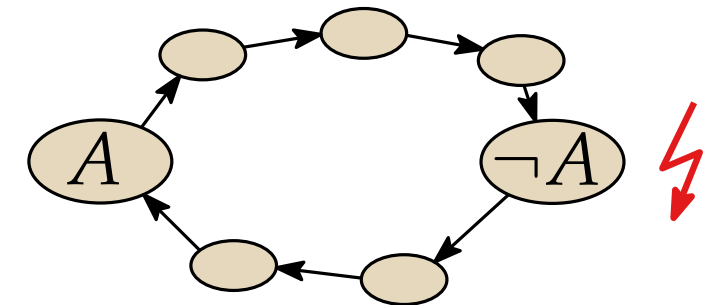
k -SATISFIABILITY (k -SAT): Wie SAT, aber jede Klausel darf $\leq k$ Literale enthalten.

- NP-schwer für $k \geq 3$
- Lösbar in Linearzeit für $k \leq 2$ **Hier!**

Hauptidee: Jede 2-SAT-Formel entspricht einem gerichteten Implikationsgraphen.

$$(A \vee B) \rightsquigarrow \begin{array}{l} \neg A \Rightarrow B \\ \neg B \Rightarrow A \end{array} \rightsquigarrow \begin{array}{cc} \neg A & \rightarrow & B \\ \neg B & \rightarrow & A \end{array}$$

Gerichtete Kreise
geben Aufschluss
über Lösbarkeit!



3. Solving 2-SAT in Linear Time

SATISFIABILITY (SAT)

Gegeben: Boolesche Formel in konjunktiver Normalform.

Frage: Gibt es eine erfüllende Belegung?

eine 3-SAT Formel

$$(A \vee \neg B \vee \neg C) \wedge (B \vee C \vee \neg D) \wedge (B \vee \neg D \vee E) \wedge (\neg B \vee D \vee \neg E)$$

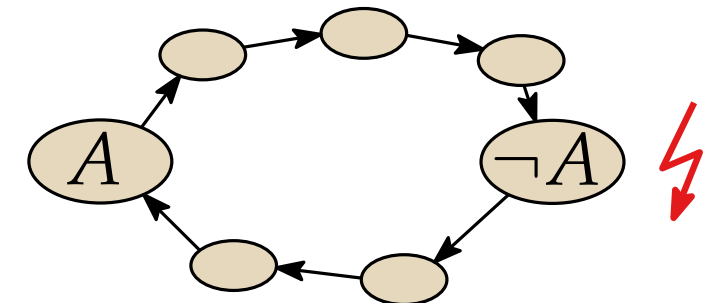
k -SATISFIABILITY (k -SAT): Wie SAT, aber jede Klausel darf $\leq k$ Literale enthalten.

- NP-schwer für $k \geq 3$
- Lösbar in Linearzeit für $k \leq 2$ **Hier!**

Hauptidee: Jede 2-SAT-Formel entspricht einem gerichteten Implikationsgraphen.

$$(A \vee B) \rightsquigarrow \begin{array}{l} \neg A \Rightarrow B \\ \neg B \Rightarrow A \end{array} \rightsquigarrow \begin{array}{cc} \neg A & \rightarrow & B \\ \neg B & \rightarrow & A \end{array}$$

Gerichtete Kreise
geben Aufschluss
über Lösbarkeit!



Hauptschwierigkeit: Berechnung der starken Zusammenhangskomponenten.

4. Lower Bounds Based on (S)ETH

Let A be an NP-hard problem.

4. Lower Bounds Based on (S)ETH

Let A be an NP-hard problem.

If $P \neq NP$, there is no polynomial time algorithm for A .

4. Lower Bounds Based on (S)ETH

Let A be an NP-hard problem.

If $P \neq NP$, there is no polynomial time algorithm for A .

With a polynomial time reduction from A to some other problem B , we can transfer this **conditional lower bound** to B , i.e., we can show:

4. Lower Bounds Based on (S)ETH

Let A be an NP-hard problem.

If $P \neq NP$, there is no polynomial time algorithm for A .

With a polynomial time reduction from A to some other problem B , we can transfer this **conditional lower bound** to B , i.e., we can show:

If $P \neq NP$, there is no polynomial time algorithm for B .

4. Lower Bounds Based on (S)ETH

Let A be an NP-hard problem.

If $P \neq NP$, there is no polynomial time algorithm for A .

With a polynomial time reduction from A to some other problem B , we can transfer this **conditional lower bound** to B , i.e., we can show:

If $P \neq NP$, there is no polynomial time algorithm for B .

Here: More fine-grained lower bounds based on other conditions:

4. Lower Bounds Based on (S)ETH

Let A be an NP-hard problem.

If $P \neq NP$, there is no polynomial time algorithm for A .

With a polynomial time reduction from A to some other problem B , we can transfer this **conditional lower bound** to B , i.e., we can show:

If $P \neq NP$, there is no polynomial time algorithm for B .

Here: More fine-grained lower bounds based on other conditions:

The **Strong Exponential Time Hypothesis (SETH)** implies that there is no algorithm for SATISFIABILITY with runtime $(2 - \varepsilon)^n \cdot \text{poly}(n)$, where n is the number of variables and $\varepsilon > 0$.

4. Lower Bounds Based on (S)ETH

Let A be an NP-hard problem.

If $P \neq NP$, there is no polynomial time algorithm for A .

With a polynomial time reduction from A to some other problem B , we can transfer this **conditional lower bound** to B , i.e., we can show:

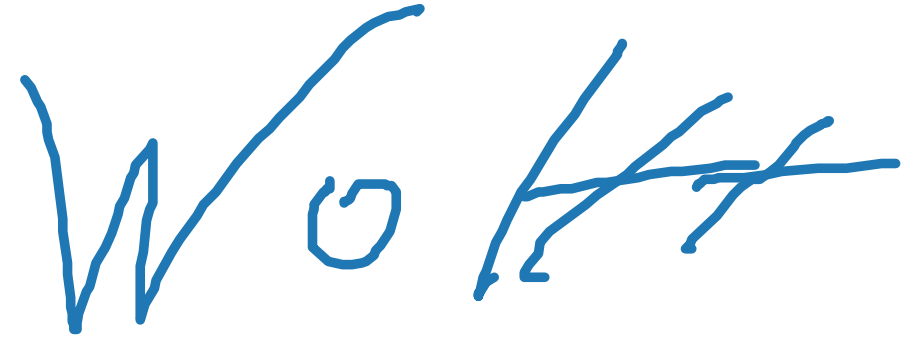
If $P \neq NP$, there is no polynomial time algorithm for B .

Here: More fine-grained lower bounds based on other conditions:

The **Strong Exponential Time Hypothesis (SETH)** implies that there is no algorithm for SATISFIABILITY with runtime $(2 - \varepsilon)^n \cdot \text{poly}(n)$, where n is the number of variables and $\varepsilon > 0$.

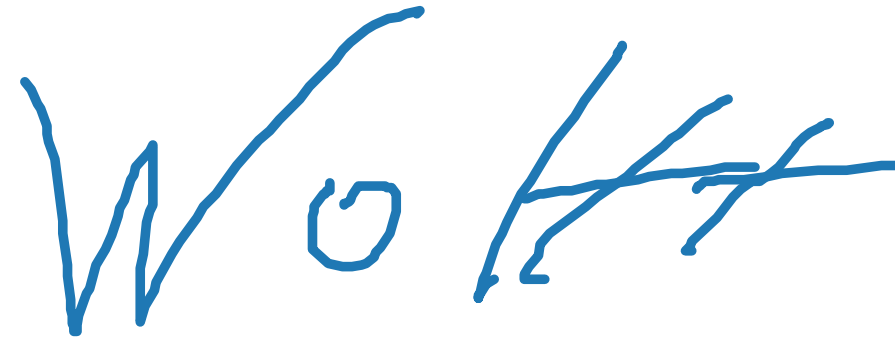
With suitable reductions from SATISFIABILITY to some other problem B , we can obtain similar *specific* lower bounds for the runtime of B , assuming SETH.

5. Why Walking the Dog Takes Time

A red handwritten signature that appears to read 'Wolter'. The letters are connected and stylized, with a prominent 'W' and 'L'.A blue handwritten signature that also appears to read 'Wolter'. It is very similar in style to the red signature, with a prominent 'W' and 'L'.

Stammen diese 2 Unterschriften von derselben Person?

5. Why Walking the Dog Takes Time

A red handwritten signature that appears to read 'Wolter'. The letters are connected and have a slightly jagged, cursive appearance.A blue handwritten signature that also appears to read 'Wolter'. The letters are more fluid and rounded than the red one, with a distinct 'o' in the second syllable.

Stammen diese 2 Unterschriften von derselben Person?

Die **Fréchet-Distanz** ist ein Maß dafür, wie ähnlich sich 2 Kurvenzüge sind.

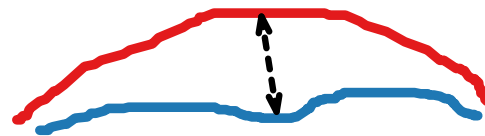
5. Why Walking the Dog Takes Time



Stammen diese 2 Unterschriften von derselben Person?

Die **Fréchet-Distanz** ist ein Maß dafür, wie ähnlich sich 2 Kurvenzüge sind.

Informell entspricht sie der Länge einer kürzesten Leine, mit der ein Hund die eine und seine Besitzerin die andere Kurve gemeinsam durchlaufen können:



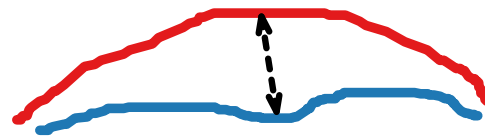
5. Why Walking the Dog Takes Time



Stammen diese 2 Unterschriften von derselben Person?

Die **Fréchet-Distanz** ist ein Maß dafür, wie ähnlich sich 2 Kurvenzüge sind.

Informell entspricht sie der Länge einer kürzesten Leine, mit der ein Hund die eine und seine Besitzerin die andere Kurve gemeinsam durchlaufen können:



Unter Annahme von SETH kann man eine polynomielle (fast scharfe!) untere Schranke für die Laufzeit zur Berechnung der Fréchet-Distanz zeigen.

6. Quantifier Elimination over the Real Numbers

Kann man die folgende Bedingung ohne Quantoren ausdrücken?

$$\exists x : x^2 + b \cdot x + c < 0$$

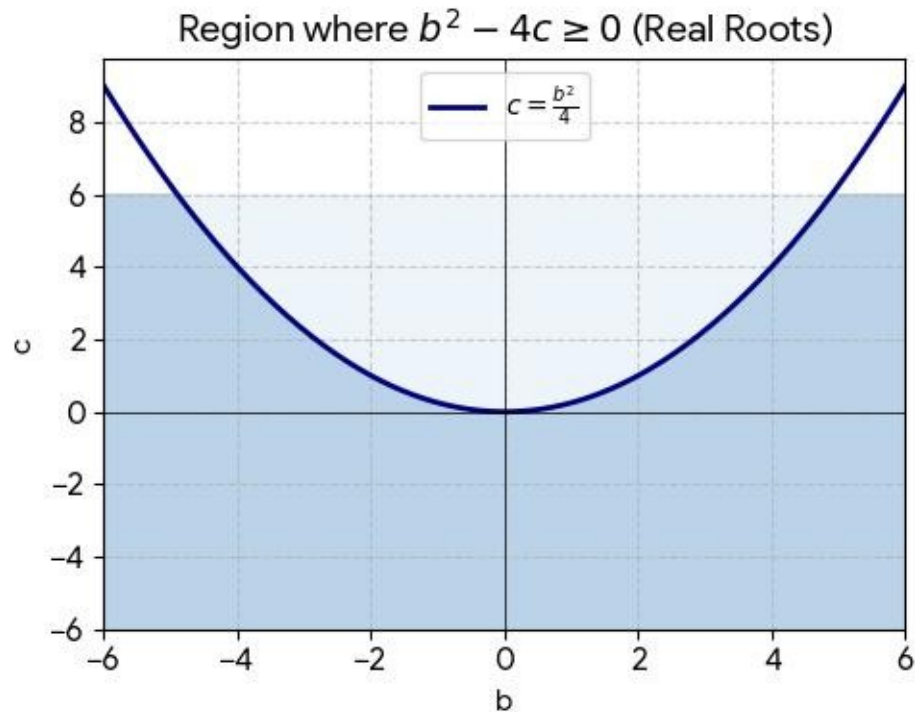
6. Quantifier Elimination over the Real Numbers

Kann man die folgende Bedingung ohne Quantoren ausdrücken?

$$\exists x : x^2 + b \cdot x + c < 0$$

Ja, wegen der Lösungsformel für quadratische Gleichungen ist äquivalent:

$$b^2 - 4c \geq 0$$



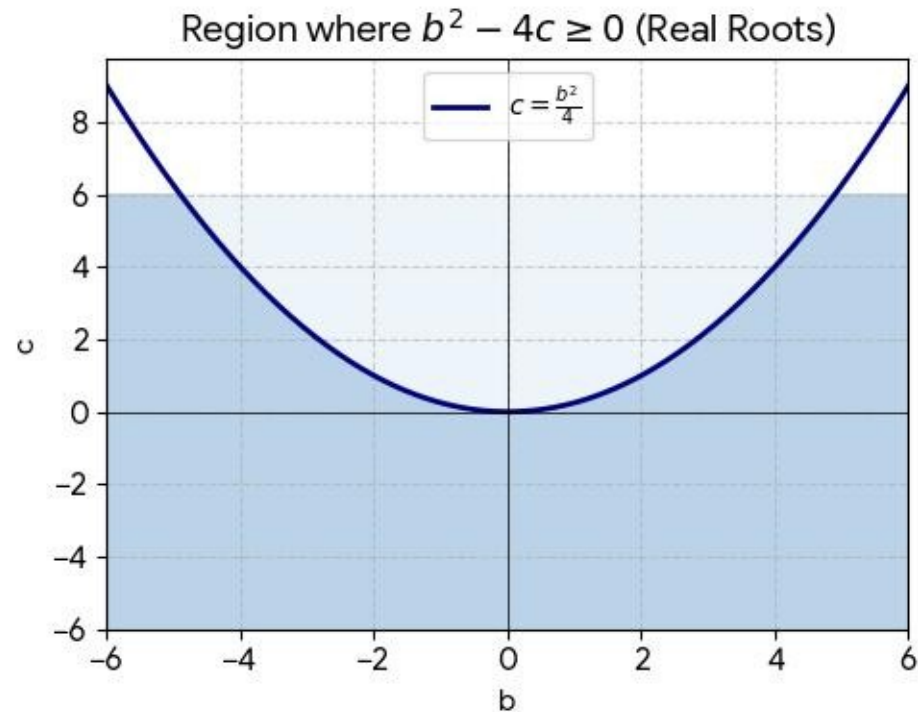
6. Quantifier Elimination over the Real Numbers

Kann man die folgende Bedingung ohne Quantoren ausdrücken?

$$\exists x : x^2 + b \cdot x + c < 0$$

Ja, wegen der Lösungsformel für quadratische Gleichungen ist äquivalent:

$$b^2 - 4c \geq 0$$



Das ist kein Zufall: In Aussagen über

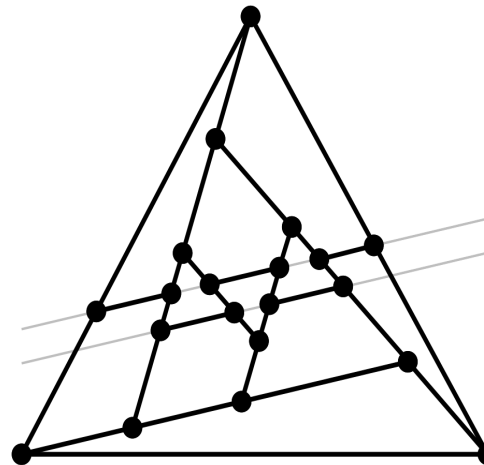
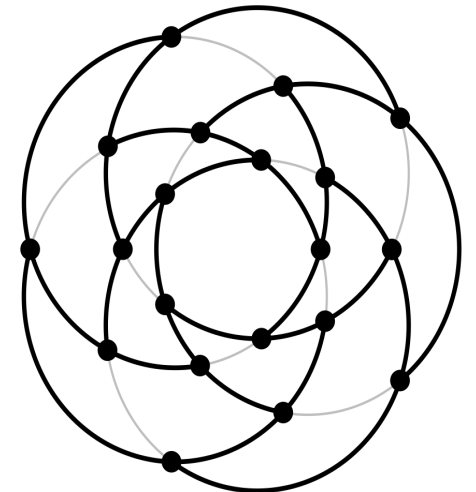
$$(\mathbb{R}, 0, 1, +, \cdot, <)$$

lassen sich stets die Quantoren eliminieren.

6. Quantifier Elimination over the Real Numbers

Programme for this part:

1. Session on prerequisites from logic (Anton).
2. A general criterion for quantifier elimination (student presentation).
3. Quantifier elimination for field of reals (student presentation).
4. Application to graph-drawing (student presentation).



7. The Weisfeiler–Leman Dimension of Planar Graphs

7. The Weisfeiler–Leman Dimension of Planar Graphs

The WL-algorithm is a fundamental algorithm used as a subroutine in graph isomorphism testing.

7. The Weisfeiler–Leman Dimension of Planar Graphs

The WL-algorithm is a fundamental algorithm used as a subroutine in graph isomorphism testing.

For every $k \in \mathbb{N}$, there is a k -dimensional version of the algorithm that colors all k -tuples of vertices in two given undirected input graphs and iteratively refines the color classes based on information of previously obtained colors.

7. The Weisfeiler–Leman Dimension of Planar Graphs

The WL-algorithm is a fundamental algorithm used as a subroutine in graph isomorphism testing.

For every $k \in \mathbb{N}$, there is a k -dimensional version of the algorithm that colors all k -tuples of vertices in two given undirected input graphs and iteratively refines the color classes based on information of previously obtained colors.

For graphs G and G' , if k is the smallest number such that the k -dimensional WL-algorithm distinguishes G and G' , then $k + 1$ is the smallest number of variables of a sentence in *first-order logic with counting* that distinguishes G & G' .

7. The Weisfeiler–Leman Dimension of Planar Graphs

The WL-algorithm is a fundamental algorithm used as a subroutine in graph isomorphism testing.

For every $k \in \mathbb{N}$, there is a k -dimensional version of the algorithm that colors all k -tuples of vertices in two given undirected input graphs and iteratively refines the color classes based on information of previously obtained colors.

For graphs G and G' , if k is the smallest number such that the k -dimensional WL-algorithm distinguishes G and G' , then $k + 1$ is the smallest number of variables of a sentence in *first-order logic with counting* that distinguishes G & G' .

There are pairs of graphs on n vertices that require a dimension of $\Omega(n)$ for the WL-algorithm to distinguish them. [Cai, Fürer, and Immerman 1992]

7. The Weisfeiler–Leman Dimension of Planar Graphs

The WL-algorithm is a fundamental algorithm used as a subroutine in graph isomorphism testing.

For every $k \in \mathbb{N}$, there is a k -dimensional version of the algorithm that colors all k -tuples of vertices in two given undirected input graphs and iteratively refines the color classes based on information of previously obtained colors.

For graphs G and G' , if k is the smallest number such that the k -dimensional WL-algorithm distinguishes G and G' , then $k + 1$ is the smallest number of variables of a sentence in *first-order logic with counting* that distinguishes G & G' .

There are pairs of graphs on n vertices that require a dimension of $\Omega(n)$ for the WL-algorithm to distinguish them. [Cai, Fürer, and Immerman 1992]

Which WL-dimension suffices for the class of planar graphs?

Themenverteilung

Themenverteilung

	Niveau
1. Monadic second-order logic on graphs	Bsc
2. Metatheorem für Knotenüberdeckung	Msc
3. 2SAT in Linearzeit	Bsc
4. Runtime lower bounds based on (S)ETH	Msc
5. Walking the dog takes time	Bsc/Msc
6. Quantifier elimination: general method	(Bsc)/Msc
7. Quantifier elimination: the case of the reals	(Bsc)/Msc
8. Quantifier elimination: application to algorithmic geometry	Bsc/Msc
9. The Weisfeiler–Leman dimension of planar graphs	Msc

maximal 8 Teams/Termine!

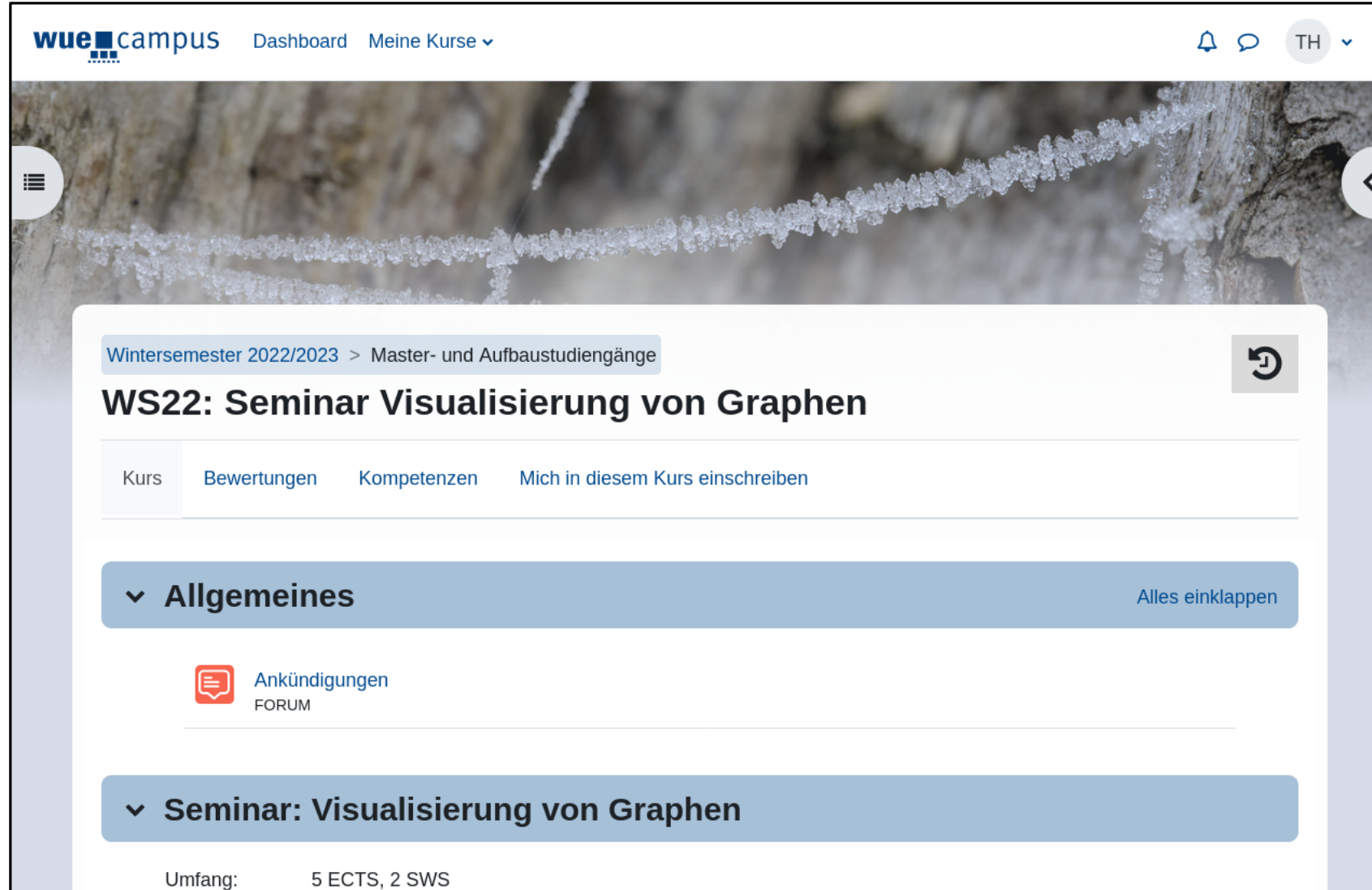
grau = diese Themen vergeben wir nur bei großem Andrang

Nächste Schritte

- In WueCampus anmelden

Nächste Schritte

- In WueCampus anmelden



The screenshot displays the WueCampus user interface. At the top, the 'wuecampus' logo is on the left, and navigation links for 'Dashboard' and 'Meine Kurse' are in the center. On the right, there are icons for notifications, chat, and a user profile dropdown showing 'TH'. Below the header is a large banner image of a tree trunk with white, crystalline structures. A hamburger menu icon is on the left, and a back arrow is on the right. The main content area features a breadcrumb trail: 'Wintersemester 2022/2023 > Master- und Aufbaustudiengänge'. Below this is the course title 'WS22: Seminar Visualisierung von Graphen' with a refresh icon. A horizontal menu contains 'Kurs' (selected), 'Bewertungen', 'Kompetenzen', and 'Mich in diesem Kurs einschreiben'. A blue bar with a dropdown arrow and the text 'Allgemeines' is followed by an 'Alles einklappen' link. Below this is a red speech bubble icon labeled 'Ankündigungen FORUM'. Another blue bar with a dropdown arrow and the text 'Seminar: Visualisierung von Graphen' is shown. At the bottom, the text 'Umfang: 5 ECTS, 2 SWS' is displayed.

wuecampus Dashboard Meine Kurse ▾

Wintersemester 2022/2023 > Master- und Aufbaustudiengänge

WS22: Seminar Visualisierung von Graphen

Kurs Bewertungen Kompetenzen Mich in diesem Kurs einschreiben

▼ Allgemeines Alles einklappen

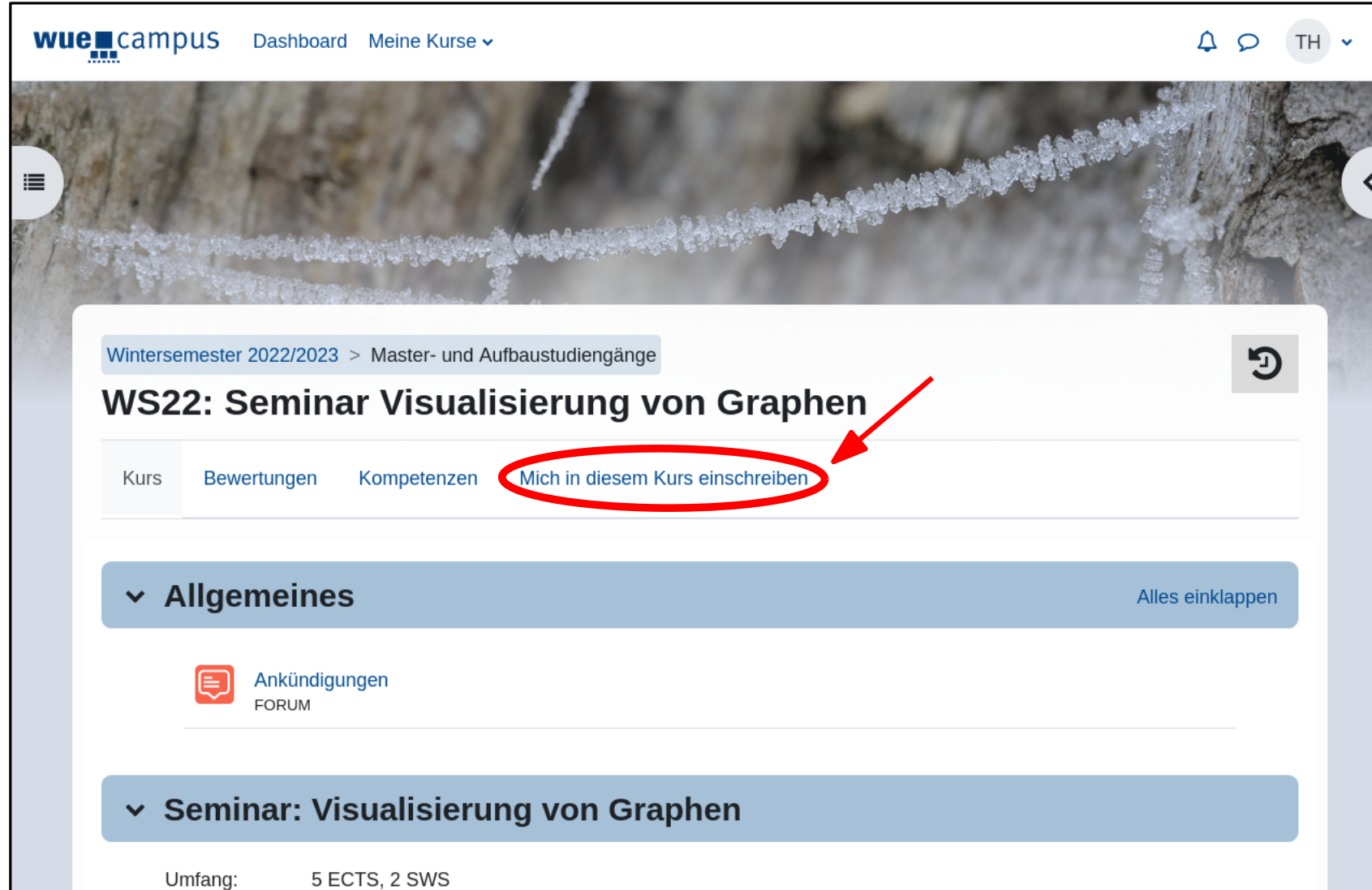
Ankündigungen FORUM

▼ Seminar: Visualisierung von Graphen

Umfang: 5 ECTS, 2 SWS

Nächste Schritte

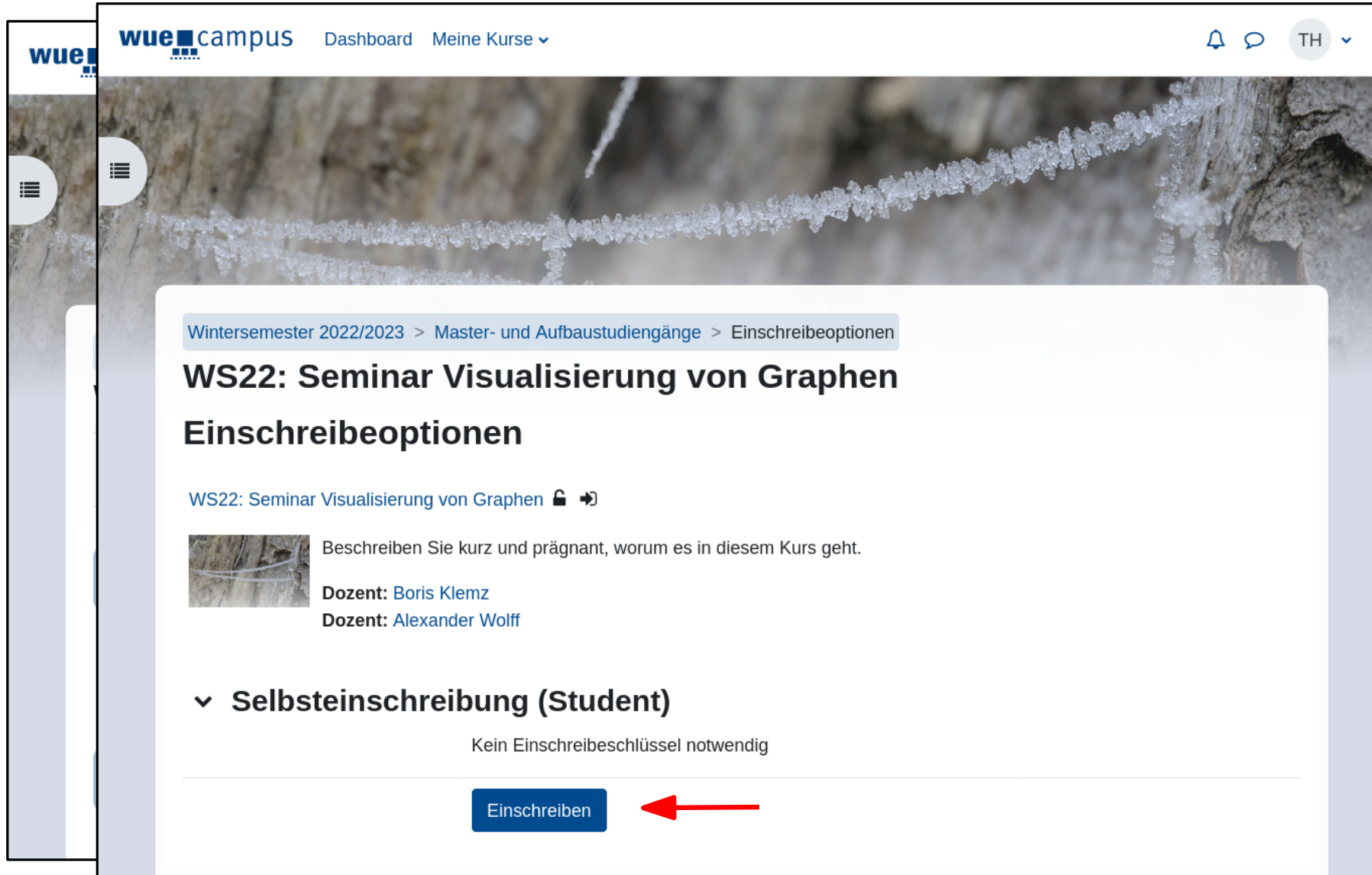
- In WueCampus anmelden



The screenshot displays the WueCampus user interface. At the top, the header includes the 'wuecampus' logo, navigation links for 'Dashboard' and 'Meine Kurse', and a user profile icon labeled 'TH'. Below the header is a banner image of a tree trunk with white chalk-like markings. The main content area shows the breadcrumb 'Wintersemester 2022/2023 > Master- und Aufbaustudiengänge' followed by the course title 'WS22: Seminar Visualisierung von Graphen'. A red circle highlights the button 'Mich in diesem Kurs einschreiben', with a red arrow pointing to it from the right. Below the course title are tabs for 'Kurs', 'Bewertungen', 'Kompetenzen', and the highlighted button. A blue bar with a dropdown arrow and the text 'Allgemeines' is followed by a section for 'Ankündigungen FORUM' with a red speech bubble icon. Another blue bar with a dropdown arrow and the text 'Seminar: Visualisierung von Graphen' is shown below. At the bottom, the text 'Umfang: 5 ECTS, 2 SWS' is visible.

Nächste Schritte

- In WueCampus anmelden



The screenshot displays the WueCampus user interface. At the top, the header includes the 'wuecampus' logo, navigation links for 'Dashboard' and 'Meine Kurse', and a user profile icon labeled 'TH'. The main content area features a breadcrumb trail: 'Wintersemester 2022/2023 > Master- und Aufbaustudiengänge > Einschreibeoptionen'. Below this, the course title 'WS22: Seminar Visualisierung von Graphen' is prominently displayed, followed by the section heading 'Einschreibeoptionen'. A small thumbnail image of a tree trunk is shown next to the course description: 'Beschreiben Sie kurz und prägnant, worum es in diesem Kurs geht.' The lecturers are listed as 'Dozent: Boris Klemz' and 'Dozent: Alexander Wolff'. A dropdown menu is open, showing the option 'Selbsteinschreibung (Student)'. Below this option, it states 'Kein Einschreibeschlüssel notwendig'. At the bottom of the page, a blue button labeled 'Einschreiben' is highlighted with a red arrow pointing to it from the right.

wuecampus Dashboard Meine Kurse ▾

Wintersemester 2022/2023 > Master- und Aufbaustudiengänge > Einschreibeoptionen

WS22: Seminar Visualisierung von Graphen

Einschreibeoptionen

WS22: Seminar Visualisierung von Graphen 🔒 ➔

 Beschreiben Sie kurz und prägnant, worum es in diesem Kurs geht.

Dozent: Boris Klemz
Dozent: Alexander Wolff

▼ **Selbsteinschreibung (Student)**

Kein Einschreibeschlüssel notwendig

Einschreiben ←

Nächste Schritte

- In WueCampus anmelden

Nächste Schritte

- In WueCampus anmelden
- In WueStudy anmelden

Nächste Schritte

- In WueCampus anmelden
- In WueStudy anmelden
- Überblick verschaffen und Kurzvortrag vorbereiten

Nächste Schritte

- In WueCampus anmelden
- In WueStudy anmelden
- Überblick verschaffen und Kurzvortrag vorbereiten
- Bei Fragen (oder *spätestens drei Wochen vor dem eigenen Vortrag*) an die BetreuerIn wenden

Nächste Schritte

- In WueCampus anmelden
- In WueStudy anmelden
- Überblick verschaffen und Kurzvortrag vorbereiten
- Bei Fragen (oder *spätestens drei Wochen vor dem eigenen Vortrag*) an die BetreuerIn wenden

Bei allgemeinen Fragen kann gerne das Diskussionsforum im WueCampus genutzt werden!

Nächste Schritte

- In WueCampus anmelden
- In WueStudy anmelden
- Überblick verschaffen und Kurzvortrag vorbereiten
- Bei Fragen (oder *spätestens drei Wochen vor dem eigenen Vortrag*) an die BetreuerIn wenden

Bei allgemeinen Fragen kann gerne das Diskussionsforum im WueCampus genutzt werden!

Zum Abschluss: Demonstration des Programms IPE
zum Erstellen von Abbildungen und Folien

<http://ipe.otfried.org/>

Nächste Schritte

- In WueCampus anmelden
- In WueStudy anmelden
- Überblick verschaffen und Kurzvortrag vorbereiten
- Bei Fragen (oder *spätestens drei Wochen vor dem eigenen Vortrag*) an die BetreuerIn wenden

Bei allgemeinen Fragen kann gerne das Diskussionsforum im WueCampus genutzt werden!

Zum Abschluss: Demonstration des Programms IPE
 zum Erstellen von Abbildungen und Folien

<http://ipe.otfried.org/>

Übrigens: ein gemeinsames git-Verzeichnis eignet sich hervorragend zum gemeinsamen Bearbeiten von .tex, aber auch .ipe Dateien!

Nächste Schritte

- In WueCampus anmelden
- In WueStudy anmelden
- Überblick verschaffen und Kurzvortrag vorbereiten
- Bei Fragen (oder *spätestens drei Wochen vor dem eigenen Vortrag*) an die BetreuerIn wenden

Bei allgemeinen Fragen kann gerne das Diskussionsforum im WueCampus genutzt werden!

Zum Abschluss: Demonstration des Programms IPE
zum Erstellen von Abbildungen und Folien

<http://ipe.otfried.org/>

Übrigens: ein gemeinsames git-Verzeichnis eignet sich hervorragend zum gemeinsamen Bearbeiten von .tex, aber auch .ipe Dateien!

<https://gitlab2.informatik.uni-wuerzburg.de/>