Utrecht University

TU/e EINDHOVEN UNIVERSITY OF TECHNOLOGY

25 April, 2025

Tutte's theorem as an educational formalization project

Pim Otte
PhD Candidate

Outline

Introduction

Motivation

Tutte's theorem

Educational formalization project

Conclusion

Outline

Introduction

Motivation

Tutte’s theorem

Educational formalization project

Conclusion

About me

- Pim Otte
- PhD candidate at Utrecht University & Technical University of Eindhoven
- Topic "Type Theory for Education"
- Supervisors: Johan Commelin, Paige Randall North & Jim Portegies
- Webpage: `https://pim.otte.dev`

Topic

- Formalization of Tutte's theorem in Lean 4, almost merged to mathlib
- Framework for educational formalization projects

Outline

Introduction

Motivation

Tutte’s theorem

Educational formalization project

Conclusion

Motivation

- Why Tutte's theorem?

Motivation

- Why Tutte's theorem?
  - Characterizes perfect matchings in graphs

Motivation

- Why Tutte's theorem?
  - Characterizes perfect matchings in graphs
  - Formalization provides basis for graph algorithms

Motivation

- Why Tutte’s theorem?
  - Characterizes perfect matchings in graphs
  - Formalization provides basis for graph algorithms
- Framework for educational formalization projects

Motivation

- Why Tutte's theorem?
  - Characterizes perfect matchings in graphs
  - Formalization provides basis for graph algorithms
- Framework for educational formalization projects
  - Asymmetry in number of teachers vs students

Motivation

- Why Tutte's theorem?
  - Characterizes perfect matchings in graphs
  - Formalization provides basis for graph algorithms
- Framework for educational formalization projects
  - Asymmetry in number of teachers vs students
  - Bridging the gap between "TPIL" and "blueprint projects"

Motivation

- Why Tutte's theorem?
  - Characterizes perfect matchings in graphs
  - Formalization provides basis for graph algorithms
- Framework for educational formalization projects
  - Asymmetry in number of teachers vs students
  - Bridging the gap between "TPIL" and "blueprint projects"
- Shout-outs:
  - Mathlib reviewers; Yaël Dillies, Bhavik Mehta, Kyle Miller

Motivation

- Why Tutte's theorem?
  - Characterizes perfect matchings in graphs
  - Formalization provides basis for graph algorithms
- Framework for educational formalization projects
  - Asymmetry in number of teachers vs students
  - Bridging the gap between "TPIL" and "blueprint projects"
- Shout-outs:
  - Mathlib reviewers; Yaël Dillies, Bhavik Mehta, Kyle Miller
  - Related work: Formalization of graph algorithms in Isabelle/HOL by Abdulaziz [Abd24].

Outline

Introduction

Motivation

Tutte's theorem

Educational formalization project

Conclusion

Perfect matching

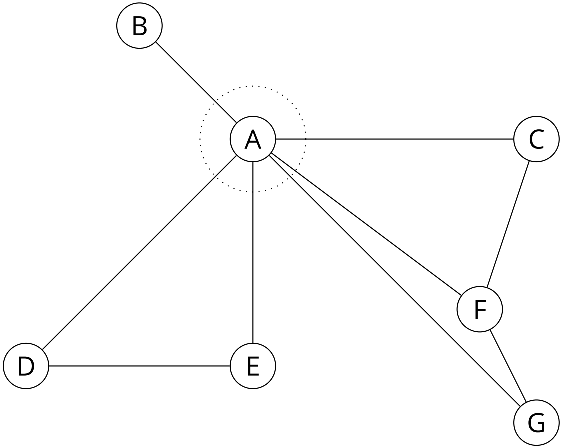
```
def IsMatching (M : Subgraph G) : Prop :=
  ∀ {|v|}, v ∈ M.verts → ∃! w, M.Adj v w

def IsSpanning (G' : Subgraph G) : Prop := ∀ v : V, v ∈
    G'.verts

def IsPerfectMatching (M : G.Subgraph) : Prop :=
  M.IsMatching ∧ M.IsSpanning
```

Tutte's theorem

Theorem (Tutte, 1947).

A graph G has a perfect matching if and only if for any subset $U \subseteq V$ the graph $G - U$ has at most $|U|$ components of odd size.

Tutte Violator



Tutte violator

Definition (Tutte violator).

A subset $|U|$ such that $G - U$ has more than $|U|$ components of odd size.

```
def IsTutteViolator (G: SimpleGraph V) (u : Set V) : Prop :=
  u.ncard < ((⊤ : G.Subgraph).deleteVerts
    u).coe.oddComponents.ncard
```

Formal proof of Tutte's theorem

```
theorem tutte [Fintype V] :
  (∃ (M : Subgraph G) , M.IsPerfectMatching) ↔
  (∀ (u : Set V), ¬ G.IsTutteViolator u) := by
classical
refine ⟨by rintro ⟨M, hM⟩; apply not_IsTutteViolator hM, ?_⟩
contrapose!
intro h
by_cases hvOdd : Odd (Fintype.card V)
· exact ⟨∅, isTutteViolator_empty hvOdd⟩
· exact exists_TutteViolator h (Nat.not_odd_iff_even.mp hvOdd)
```

Sub-proof structure

Lemma.

If no perfect matching exists, then a Tutte violator exists.

Proof structure.

- Work with an edge-maximal counterexample
- Split into two cases. The graph remaining after removing the “universal vertices” either:
  1. decomposes into cliques.
  2. does not decompose into cliques.

□

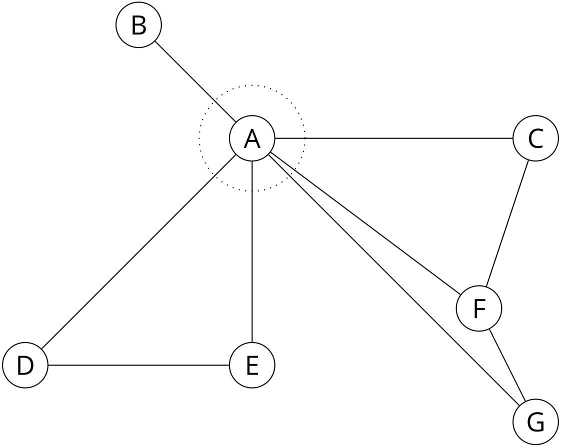
Universal vertices

Definition (Universal Vertices).

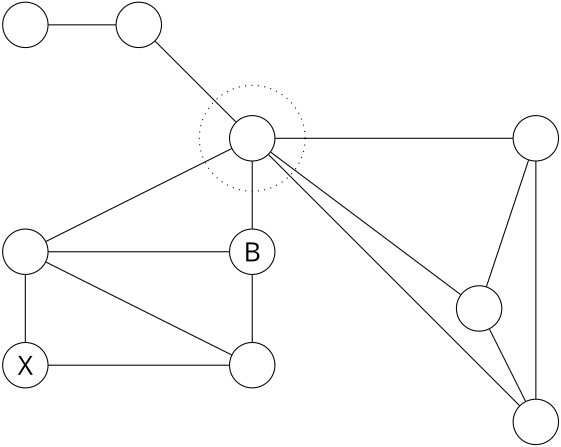
The set of vertices adjacent to all other vertices.

```
def universalVerts (G : SimpleGraph V) : Set V :=
  {v : V | ∀ ⦃w⦄, v ≠ w → G.Adj w v}
```

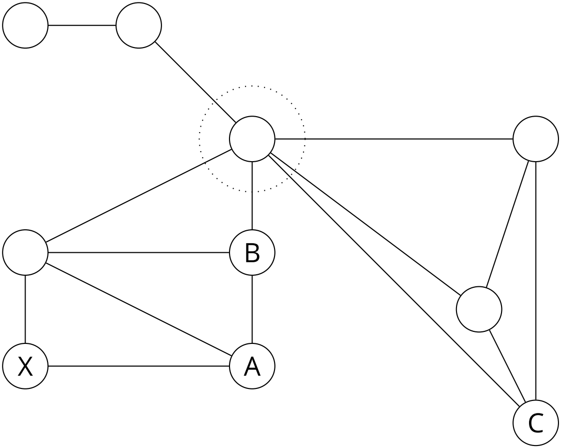
Universal vertices



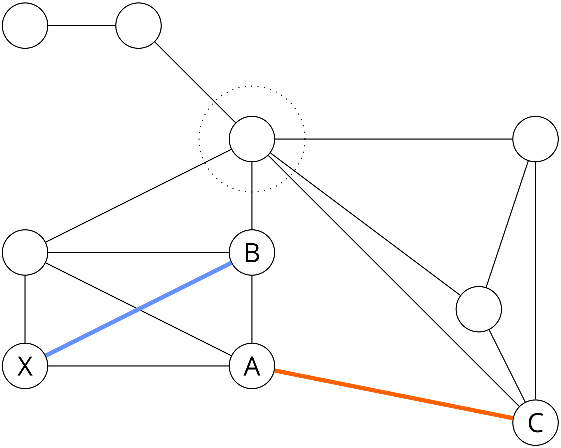
The interesting case



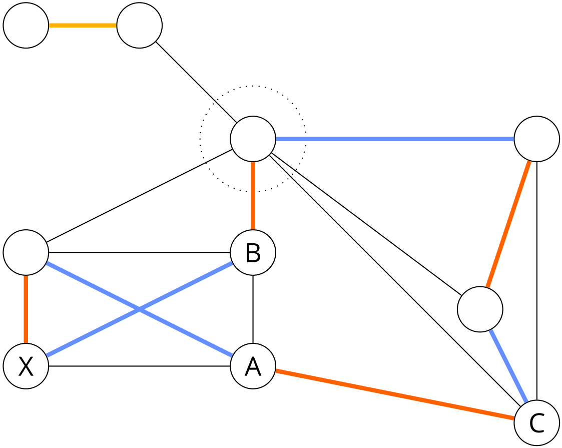
The interesting case



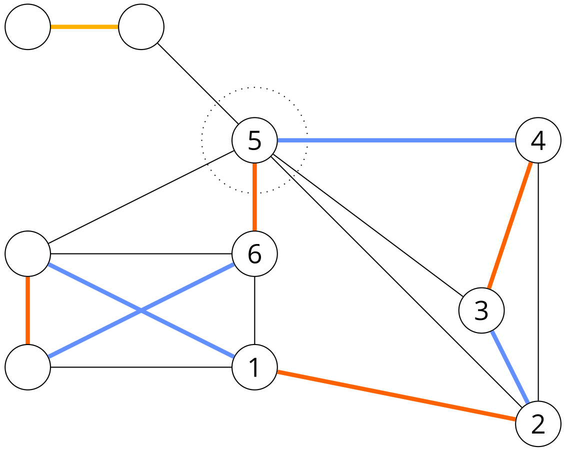
The interesting case



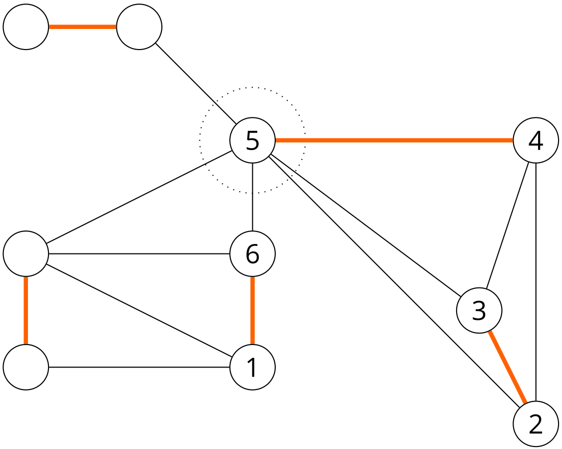
The interesting case



The interesting case



The interesting case



The interesting case

- Construct cycles as symmetric difference of perfect matchings
- Show that these cycles are alternating
- Show that symmetric difference with alternating graph preserves perfect matching
- Construct specific alternating cycle needed.

Symmetric difference: SimpleGraph vs Subgraph

- for `H H' : Subgraph G` it holds that
  `(H △ H').verts = (H \ H').verts ∪ (H' \ H).verts`

Symmetric difference: SimpleGraph vs Subgraph

- for `H H' : Subgraph G` it holds that
  `(H △ H').verts = (H \ H').verts ∪ (H' \ H).verts`
- So, for two perfect matchings `(H △ H').verts = ∅`
- This problem doesn't occur for the symmetric difference on SimpleGraphs

IsCycles and IsAlternating

```
def IsCycles (G : SimpleGraph V) :=
  ∀ ⦃v⦄, (G.neighborSet v).Nonempty →
          (G.neighborSet v).ncard = 2

def IsAlternating (G G' : SimpleGraph V) :=
  ∀ ⦃v w w': V⦄, w ≠ w' → G.Adj v w → G.Adj v w' →
    (G'.Adj v w ↔ ¬ G'.Adj v w')
```

IsCycles is really cycles

```
lemma
    IsCycles.exists_cycle_toSubgraph_verts_eq_connectedComponentSupp
    [Finite V] {c : G.ConnectedComponent} (h : G.IsCycles)
    (hv : v ∈ c.supp) (hn : (G.neighborSet v).Nonempty) :
  ∃ (p : G.Walk v v), p.IsCycle ∧ p.toSubgraph.verts =
    c.supp := by
classical
obtain ⟨w, hw⟩ := hn
obtain ⟨u, p, hp⟩ :=
    SimpleGraph.adj_and_reachable_delete_edges_iff_exists_cycle.mp
  ⟨hw, h.reachable_deleteEdges hw⟩
have hvp : v ∈ p.support :=
    SimpleGraph.Walk.fst_mem_support_of_mem_edges _ hp.2
have : p.toSubgraph.verts = c.supp := by sorry
use p.rotate hvp
rw [← this]
exact ⟨hp.1.rotate _, by simp_all⟩
```

Symmetric difference of matchings is alternating

```
lemma Subgraph.IsPerfectMatching.isAlternating_symmDiff_left
    {M' : Subgraph G'} (hM : M.IsPerfectMatching)
    (hM' : M'.IsPerfectMatching) :
    (M.spanningCoe ∆ M'.spanningCoe).IsAlternating
    M.spanningCoe := by
  intro v w w' hww' hvw hvw'
  obtain ⟨v1, hm1, hv1⟩ := hM.1 (hM.2 v)
  obtain ⟨v2, hm2, hv2⟩ := hM'.1 (hM'.2 v)
  simp only [symmDiff_def] at *
  aesop
```

Symmetric difference of matchings are cycles

```
lemma Subgraph.IsPerfectMatching.symmDiff_isCycles
  {M : Subgraph G} {M' : Subgraph G'} (hM :
    M.IsPerfectMatching) (hM' : M'.IsPerfectMatching) :
  (M.spanningCoe △ M'.spanningCoe).IsCycles := by
intro v
obtain ⟨w, hw⟩ := hM.1 (hM.2 v)
obtain ⟨w', hw'⟩ := hM'.1 (hM'.2 v)
simp only [symmDiff_def, Set.ncard_eq_two, ne_eq,
    imp_iff_not_or, Set.not_nonempty_iff_eq_empty,
  Set.eq_empty_iff_forall_not_mem,
    SimpleGraph.mem_neighborSet, SimpleGraph.sup_adj,
    sdiff_adj,
  spanningCoe_adj, not_or, not_and, not_not]
by_cases hww' : w = w'
· simp_all [← imp_iff_not_or, hww']
· right
  use w, w'
  aesop
```

Symmetric difference along alternating cycles preserves matching

```
lemma Subgraph.IsPerfectMatching.symmDiff_of_isAlternating
    (hM : M.IsPerfectMatching) (hG' : G'.IsAlternating
    M.spanningCoe) (hG'cyc : G'.IsCycles) : (⊤ : Subgraph
    (M.spanningCoe △ G')).IsPerfectMatching := by
rw [Subgraph.isPerfectMatching_iff]
intro v
simp only [toSubgraph_adj, symmDiff_def,
    SimpleGraph.sup_adj, sdiff_adj, Subgraph.spanningCoe_adj]
obtain ⟨w, hw⟩ := hM.1 (hM.2 v)
by_cases h : G'.Adj v w
· -- Inside cycle, so other edge is in matching
  obtain ⟨w', hw'⟩ := hG'cyc.other_adj_of_adj h
  sorry
· -- Outside cycle
  sorry
```

Lessons from formalization

- Work in the “correct” ambient type
- Develop the right abstraction

Outline

Introduction

Motivation

Tutte’s theorem

Educational formalization project

Conclusion

Context

- Reason for formalizing Tutte’s theorem

Context

- Reason for formalizing Tutte’s theorem
- Gap between available “teachers” and “students”

Context

- Reason for formalizing Tutte's theorem
- Gap between available "teachers" and "students"
- Gap between "TPIL", minor contributions and "blueprint projects"

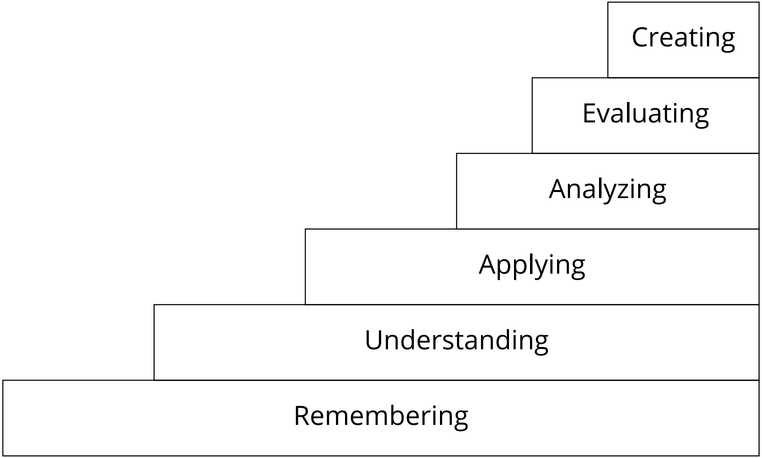
Context

- Reason for formalizing Tutte’s theorem
- Gap between available “teachers” and “students”
- Gap between “TPIL”, minor contributions and “blueprint projects”
- Goal: Enable educational formalization projects with minimal teacher input.

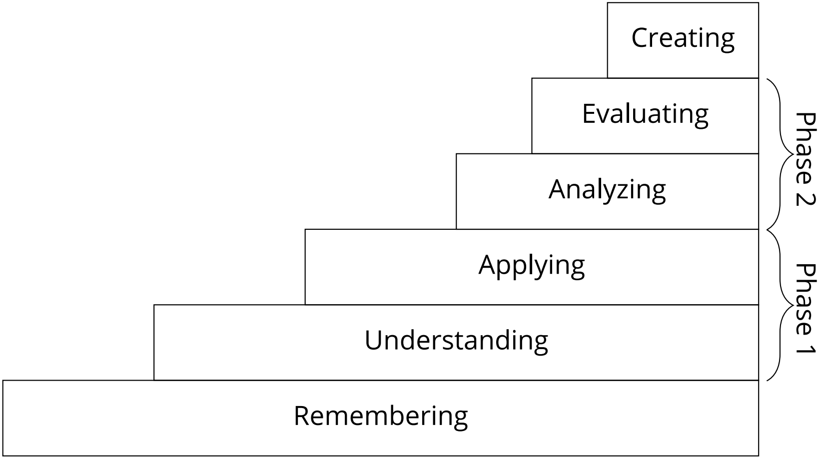
The framework

	Phase 1	Phase 2
Deliverable	Initial formalization	Polished formalization
Learning goals	Working with an ITP Proving goals Formulating intermediate goals	Refactoring formal proofs Architecting formal proofs
Teacher role	Provide goal statement Recommend resources	Review formalization
Student role	Focus on learning	Attention for details Attention for structure

Intermezzo: Bloom’s taxonomy



Intermezzo: Bloom’s taxonomy



Tutte’s theorem as an educational formalization project

	Phase 1	Phase 2
Deliverable	5000 lines of spaghetti	36 Pull Requests
Learning goals	Typeclass Synthesis have-tactic structure	Classical approach Golfing
Teacher role Lean Community	Provided goal statement Recommended resources	Reviewed formalization
Student role Pim Otte	Focus on getting a compiling proof	Refactored to mathlib-level

Proposed conditions

- Student: In the Goldilocks zone of Bloom's taxonomy

Proposed conditions

- Student: In the Goldilocks zone of Bloom's taxonomy
- Teacher:
  - Select suitable goal

Proposed conditions

- Student: In the Goldilocks zone of Bloom’s taxonomy
- Teacher:
  - Select suitable goal
  - Provide resources

Proposed conditions

- Student: In the Goldilocks zone of Bloom's taxonomy
- Teacher:
    - Select suitable goal
    - Provide resources
    - Review in the second phase

Proposed conditions

- Student: In the Goldilocks zone of Bloom's taxonomy
- Teacher:
  - Select suitable goal
  - Provide resources
  - Review in the second phase
  - Communicate framework to students

Outline

Introduction

Motivation

Tutte's theorem

Educational formalization project

Conclusion

Conclusion

- Formalized Tutte's theorem

Conclusion

- Formalized Tutte's theorem
- Proposed a framework for educational formalization projects

Conclusion

- Formalized Tutte’s theorem
- Proposed a framework for educational formalization projects
- Opportunities: Formalizing graph algorithms, testing the framework in traditional setting.

References

Mohammad Abdulaziz, *A formal correctness proof of edmonds' blossom shrinking algorithm*, 2024.

Utrecht University

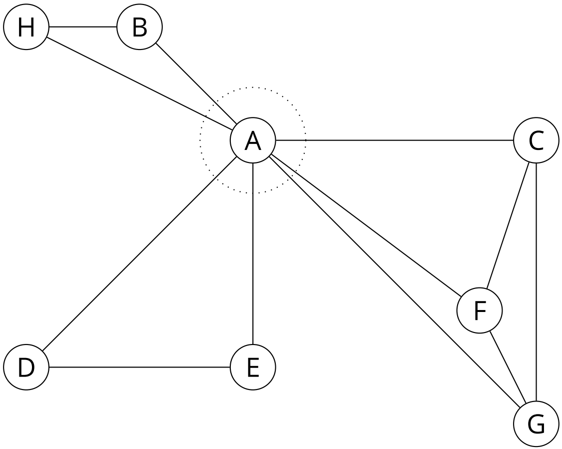
Sharing science,
shaping tomorrow

Thanks to Yaël Dillies, Bhavik Mehta, Kyle Miller and other mathlib reviewers.

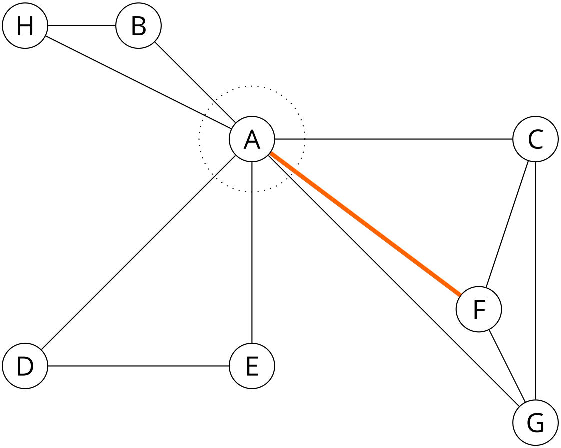
TU/e EINDHOVEN UNIVERSITY OF TECHNOLOGY

pim.otte.dev

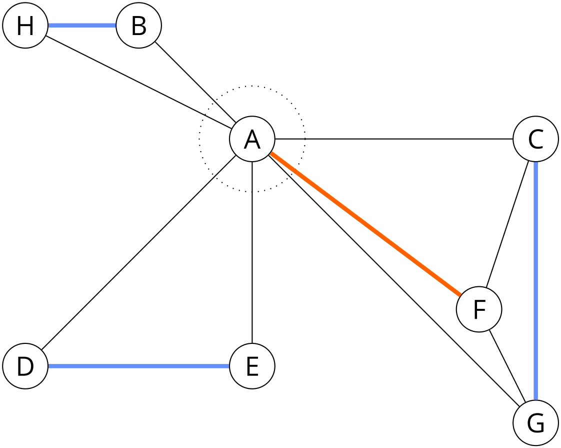
Oops! All cliques



Oops! All cliques



Oops! All cliques



Formalization: Oops! All cliques

```
theorem Subgraph.IsPerfectMatching.exists_of_isClique_supp (hveven :
    Even (Fintype.card V)) (h : ¬G.IsTutteViolator G.universalVerts) (h' : ∀
    (K : G.deleteUniversalVerts.coe.ConnectedComponent),
    G.deleteUniversalVerts.coe.IsClique K.supp) : ∃ (M : Subgraph G),
    M.IsPerfectMatching := by
classical
obtain ⟨M, ⟨hM, hsub⟩⟩ :=
    IsMatching.exists_verts_compl_subset_universalVerts h h'
obtain ⟨M', hM'⟩ := ((G.isClique_universalVerts.subset
    hsub).even_iff_exists_isMatching
(Set.toFinite _)).mp (by simpa [Set.even_ncard_compl_iff hveven,
    -Set.toFinset_card,← Set.ncard_eq_toFinset_card'] using
    hM.even_card)
use M ⊔ M'
have hspan : (M ⊔ M').IsSpanning := by
rw [Subgraph.isSpanning_iff, Subgraph.verts_sup, hM'.1]
exact M.verts.union_compl_self
exact ⟨hM.sup hM'.2 (by
simp only [hM.support_eq_verts, hM'.2.support_eq_verts, hM'.1,
    Subgraph.verts_sup]
exact (Set.disjoint_compl_left_iff_subset.mpr fun {|a|} a ↦ a).symm), hspan⟩
```

Utrecht University

Sharing science, *shaping tomorrow*

Thanks to Yaël Dillies, Bhavik Mehta, Kyle Miller and other mathlib reviewers.

TU/e EINDHOVEN UNIVERSITY OF TECHNOLOGY

TU/e EINDHOVEN UNIVERSITY OF TECHNOLOGY

pim.otte.dev