An Introduction to the Probabilistic Method in Isabelle/HOL

Chelsea Edmonds | cle47@cl.cam.ac.uk

Lawrence C. Paulson | lp15@cl.cam.ac.uk

Department of Computer Science and Technology | University of Cambridge

Women in EuroProofNet Workshop| ITP 2023

UNIVERSITY OF CAMBRIDGE

PhD Research supported by a joint Cambridge Australia Scholarship and Cambridge Department of Computer Science Qualcomm Studentship

Additionally supported by the ERC Advanced Grant ALEXANDRIA (Project GA 742178).

Overview

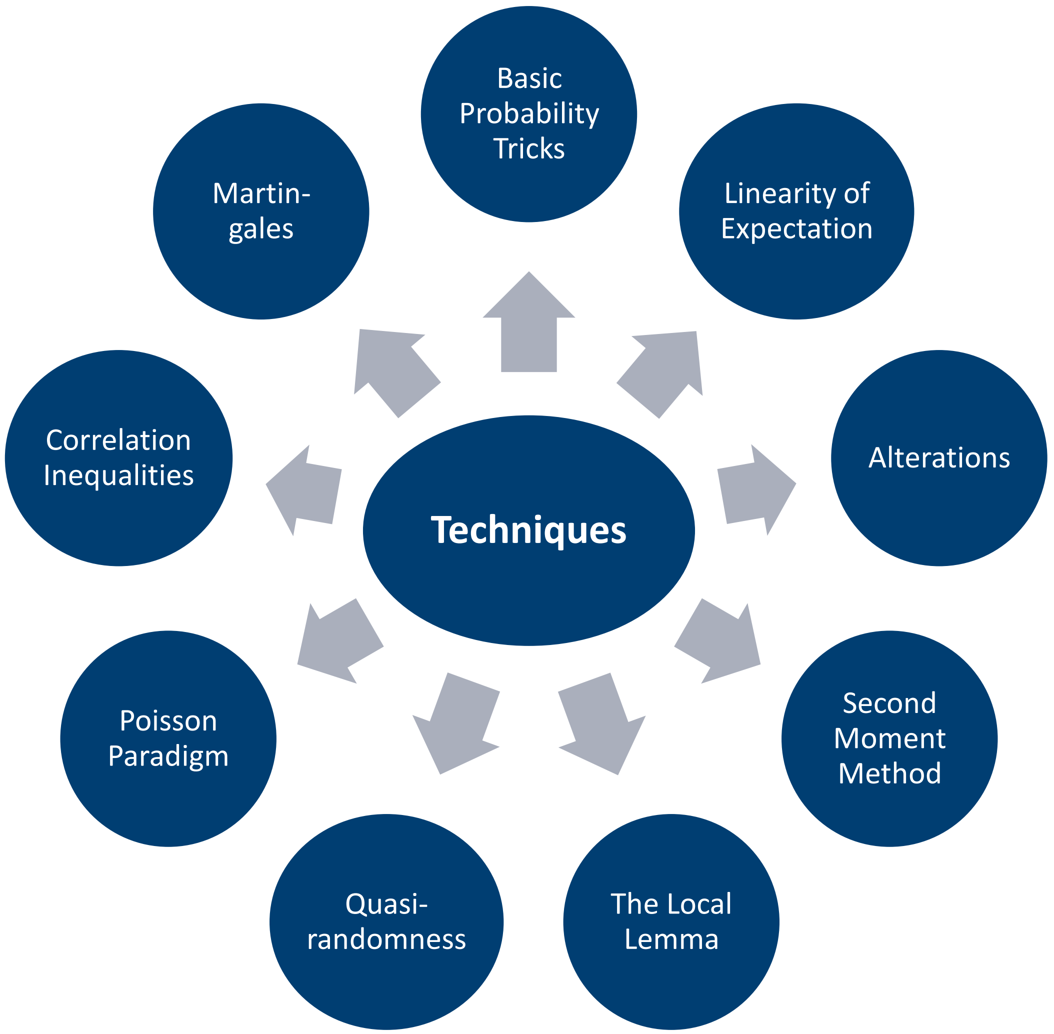
- The Motivating Problem:
  - What is the Probabilistic Method
  - Existence of Hypergraph Colourings
- Some Brief Isabelle/HOL Background
- The Probabilistic Method Framework
- Applying the Framework
- Extensions & Discussions

What is the Probabilistic Method?

Key Idea

Given a probability space over structures…

Show the structures have the desired properties with positive probability.

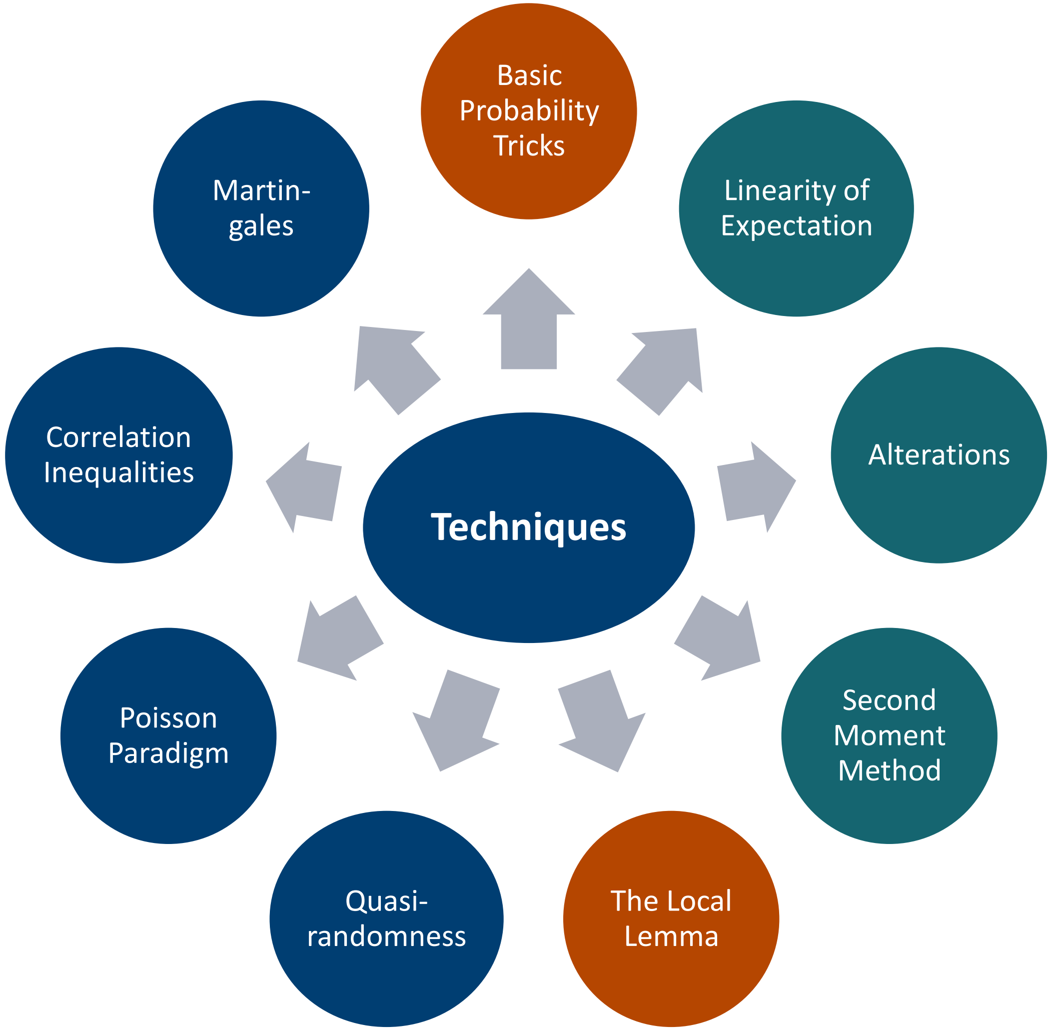


What is the Probabilistic Method?

Key Idea

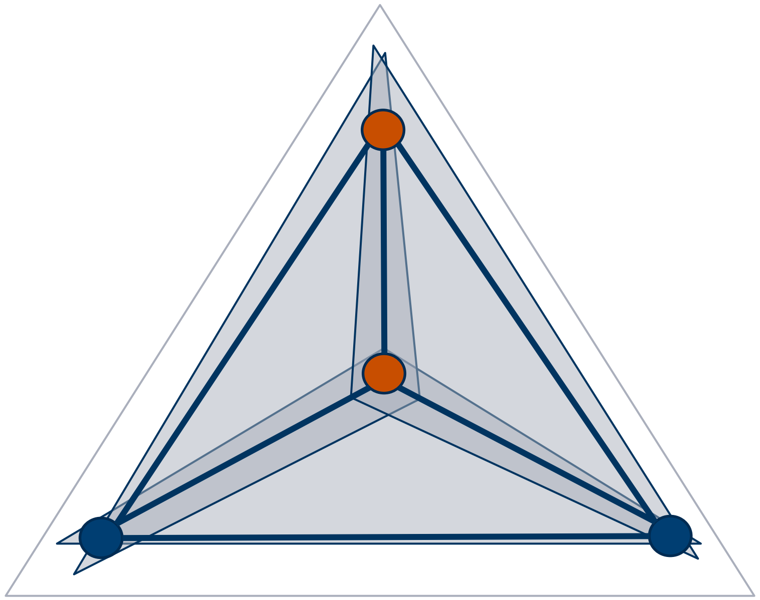
Given a probability space over structures…

Show the structures have the desired properties with positive probability.

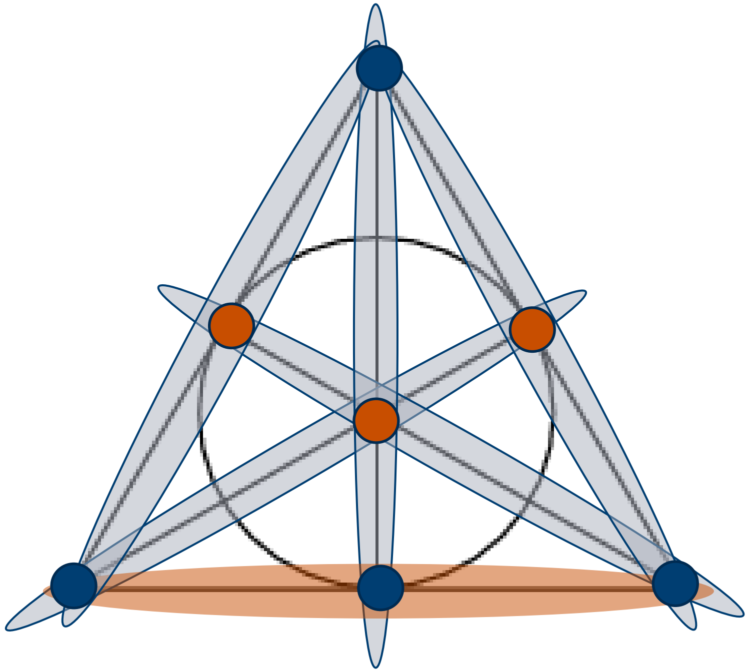


Hypergraph Colourings.

- A *hypergraph* (V, E), where E is a collection of subsets of V of any size, is “colourable” if there is a vertex colouring such that no edge is monochromatic.



2- colourable 3-uniform w/ 4 edges



Not 2- colourable 3-uniform w/ 7 edges

A Basic Proof

The Probabilistic Method:

Prove existence by showing a structure has a desired property with probability > 0

(or avoids bad properties with probability < 1)

Proposition 1.3.1 [Erdős (1963a)] *Every n-uniform hypergraph with less than 2^{n-1} edges has property B. Therefore $m(n) \geq 2^{n-1}$.*

Proof. Let $H = (V, E)$ be an n-uniform hypergraph with less than 2^{n-1} edges. Color V randomly by two colors. For each edge $e \in E$, let A_e be the event that e is monochromatic. Clearly $\Pr[A_e] = 2^{1-n}$. Therefore

$$\Pr\left[\bigvee_{e \in E} A_e\right] \leq \sum_{e \in E} \Pr[A_e] < 1$$

and there is a two-coloring without monochromatic edges. ■

The Probabilistic Method - Why Formalise?

The Probabilistic Method is one of the most powerful and widely used tools applied in combinatorics (Alon & Spencer, 2015).

- No prior formalisations on hypergraph colourings -> many applications.
- Interest in formalised maths has grown significantly -> particularly in combinatorics.
- Only three pre-existing formalisations which use the probabilistic method -> focused on theorems not general techniques.
- Predominance of this method in modern combinatorics research -> motivated by many applications -> how can we make this easy for people to formalise future work?

Identified Formalisation Challenges

- Reliance on human intuition
- Complex calculations
- Set up involved
- Definitions and Notation

General Techniques and Methods are needed!

A first attempt at formalising a proof written in 1 line on paper!

```
proof -
  fix e assume a: "e ∈ set_mset E"
  then have "{f ∈ C . edge_is_monochromatic2 f e} = (⋃ c∈ {0..<2} .{f ∈ C . ∀ v ∈ e . f v = c})"
    using edge_is_monochromatic_set_union[of e 2] C_def by simp
  also have "... = (⋃ c∈ {0::nat, 1} .{f ∈ C . ∀ v ∈ e . f v = c})"
    by fastforce
  finally have eq: "{f ∈ C . edge_is_monochromatic2 f e} = {f ∈ C . ∀ v ∈ e . f v = (0::nat)} ∪ {f
    by auto
  have prob_c: "⋀ c. c ∈ {0..<2} ⟹ P.prob {f ∈ C . ∀ v ∈ e . f v = c} = 1/(2 powi k)"
  proof -
    fix c :: colour assume cin: "c ∈ {0..<2}"
    have ess: "e ⊆ 𝒱" using a wellformed by auto
    then have lt: "card e ≤ card 𝒱"
      by (simp add: card_mono local.finite)
    then have scard: "card {f ∈ C . ∀ v ∈ e . f v = c} = (2 :: real) powi ((card 𝒱) - card e)"
      unfolding C_def using all_n_vertex_colourings_fun_alt[of 2] card_PiE_filter_range_set[of c 2
      using cin by fastforce
    have "P.prob {f ∈ C . ∀ v ∈ e . f v = c} = card {f ∈ C . ∀ v ∈ e . f v = c}/ (card C)"
      using measure_uniform_count_measure[of C "{f ∈ C . ∀ v ∈ e . f v = c} "] finC
      by fastforce
    also have "... = (2 powi ((card 𝒱) - card e))/(2 powi (card 𝒱))" using Ccard scard by simp
    also have "... = 2 powi (int (card 𝒱 - card e) - int (card 𝒱))" by (simp add: power_int_diff)
    also have "... = 2 powi (int (card 𝒱) - int (card e) - int (card 𝒱))" using int_ops lt by simp
    also have "... = 2 powi -(card e)" using assms(1) by (simp add: of_nat_diff)
    also have "... = inverse (2 powi (k))" using uniform a power_int_minus[of 2 "(int k)"] by simp
    finally show "P.prob {f ∈ C . ∀ v ∈ e . f v = c} = 1/(2 powi k)"
      by (simp add: inverse_eq_divide)
  qed
  have ss: "⋀ c .{f ∈ C. ∀v∈e. f v = c} ∈ P.events"
    by (simp add: sts)
  have "⋀ f . f ∈ C ⟹ ¬ ((∀ v ∈ e . f v = (0::nat)) ∧ (∀ v ∈ e . f v = (1::nat)))"
  proof (rule ccontr)
    fix f assume fin: "f ∈ C"
    assume "¬ ¬ ((∀v∈e. f v = 0) ∧ (∀v∈e. f v = 1))"
    then have con: "(∀v∈e. f v = 0) ∧ (∀v∈e. f v = 1)" by auto
    then obtain v where "v ∈ e" using blocks_nempty a by auto
    then show False using fin con by auto
  qed
  then have disj: "{f ∈ C . ∀ v ∈ e . f v = (0::nat)} ∩ {f ∈ C . ∀ v ∈ e . f v = (1::nat)} = {}" b
  then have "P.prob {f ∈ C . edge_is_monochromatic2 f e} = P.prob ({f ∈ C . ∀ v ∈ e . f v = (0::nat
    using eq by simp
  also have "... = P.prob {f ∈ C . ∀ v ∈ e . f v = (0::nat)} + P.prob {f ∈ C . ∀ v ∈ e . f v = (1:
    using P.finite_measure_Union[of "{f ∈ C . ∀ v ∈ e . f v = (0::nat)}" "{f ∈ C . ∀ v ∈ e . f v =
  also have "... = 2/(2 powi (int k))" using prob_c by simp
  also have "... = 2/(2* (2 powi ((int k) - 1)))" using assms(3)
    by (metis power_int_commutes power_int_minus_mult zero_neq_numeral)
  finally show "P.prob {f ∈ C . edge_is_monochromatic2 f e} = 2 powi (1 - int k)"
    by (simp add: power_int_diff)
qed
```

2. Isabelle Background

Isabelle /HOL

- Simple type theory
- Sledgehammar – automated proof search.
- Search tools: Query Search, Find Facts, SErAPIS
- The Isar structured proof language
- Interactive Development Environments
- Extensive existing libraries in Maths & Computer Science in the Archive of Formal Proofs (AFP)
- Additional features: Code generation, modularity, polymorphism, documentation generation …

Locales Basics

- Locales are Isabelle's module system. From a logical perspective, they are simply persistent contexts.

$$\bigwedge x_1 \dots x_n.\, [\![A_1; \dots; A_m]\!] \Rightarrow C.$$

- A simple example (taken from the Locales tutorial):

Notation

Parameters

```
locale partial_order =
  fixes le :: "'a ⇒ 'a ⇒ bool" (infixl "⊑" 50)
  assumes refl [intro, simp]: "x ⊑ x"
    and anti_sym [intro]: "⟦ x ⊑ y; y ⊑ x ⟧ ⟹ x = y"
    and trans [trans]: "⟦ x ⊑ y; y ⊑ z ⟧ ⟹ x ⊑ z"
```

Assumptions

Locales Basics – Inheritance and Interpretations

- We have direct inheritance

```
locale lattice = partial_order +
  assumes ex_inf: "∃inf. is_inf x y inf"
    and ex_sup: "∃sup. is_sup x y sup"
begin
```

- And indirect inheritance

```
sublocale total_order ⊆ lattice
```

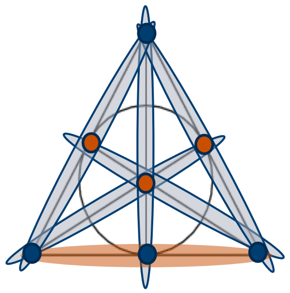
- Interpretations (global & local)

```
interpretation int: partial_order "(≤) :: [int, int] ⇒ bool"
  rewrites "int.less x y = (x < y)"
proof -
```

3. The Probabilistic Method

The Basic Method

1. Introduce randomness to the Problem Domain
2. Identify the desired properties/properties to avoid
3. Show object has desired properties with $P > 0$
4. In a finite space, there must then be an element of the space with the property!



Applying the Method

Goal: Prove that every k-uniform hypergraph with fewer than 2^k-1 edges is 2-colourable

1. Colour a graph with 2 colours randomly
2. Property: colouring results in no edges being monochromatic.
3. Show the complement: probability of all edges being monochromatic < 1
4. P(A) = 1 – (¬A). Positive probability, and exemplar colouring can be obtained.

Formalisation Framework - Summary

Formal Framework

1. **Define a probability space**
2. Define object properties
3. *Calculate probability bounds*
4. **Obtain exemplar object**

Traditional Framework

1. Introduce randomness to the Problem Domain
2. Identify the desired properties/properties to avoid
3. Show object has desired properties with $P > 0$
4. In a finite space, there must then be an element of the space with the property!

The Formalisation Framework – Step 1

To "introduce randomness" we must define a probability space $(\Omega, \mathcal{F}, P)$ formally

Define the measure

```
define C where  "C = (all_n_vertex_colourings_fun 2)"
let ?M = "uniform_count_measure C"
```

Define the prob space

```
interpret P: prob_space ?M
  using assms(1) by (intro prob_space_uniform_count_measure)(simp_all add: C_def vertex
```

Useful lemmas

```
have sp: "space ?M = C"
  by (simp add: space_uniform_count_measure)
have sts: "P.events = Pow C" by (simp add: sets_uniform_count_measure)
have finE: "finite (set_mset E)" by simp
have finC: "finite C" using vertex_colourings_fun_fin C_def by simp
have Ccard: "card C = 2 powi (card 𝒱)" using count_vertex_colourings_fun C_def by auto
```

The Formalisation Framework – Step 1 General!

```
locale vertex_fn_space = fin_hypersystem_vne +
  fixes F :: "'a set ⇒ 'b set"
  fixes p :: "'b ⇒ real"
  assumes ne: "F 𝒱 ≠ {}"
  assumes fin: "finite (F 𝒱)"
  assumes pgte0: "⋀ fv . fv ∈ F 𝒱 ⟹ p fv ≥ 0"
  assumes sump: "(∑x ∈ (F 𝒱) . p x) = 1"
begin
```

```
sublocale vertex_fn_space ⊆ prob_space M
  using prob_space_M .
```

```
definition "Ω ≡ F 𝒱" (* model space *)

lemma fin_Ω: "finite Ω"
  unfolding Ω_def using fin by auto

lemma ne_Ω: "Ω ≠ {}"
  unfolding Ω_def using ne by simp

definition "M = point_measure Ω p"
```

We use locales on incidence systems to create an abstract vertex space, which can be extended for different properties.

A Vertex Colouring Space

```
locale vertex_colour_space = fin_hypergraph_nt +
  fixes n :: nat (*Number of colours *)
  assumes n_lt_order: "n ≤ order"
  assumes n_not_zero: "n ≠ 0"

sublocale vertex_colour_space ⊆ vertex_prop_space 𝒱 E "{0..<n}"
  rewrites "ΩU = 𝒞ⁿ"
proof -
  have "{0..<n} ≠ {}" using n_not_zero by simp
  then interpret vertex_prop_space 𝒱 E "{0..<n}"
    by (unfold_locales) (simp_all)
  show "vertex_prop_space 𝒱 E {0..<n}" by (unfold_locales)
  show "ΩU = 𝒞ⁿ"
    using Ω_def all_n_vertex_colourings_alt by auto
qed
```

Context contains general lemmas on vertex colourings for any future applications of the probabilistic method to colourings!

The Formalisation Framework – Step 3

- The Union bound:

```
lemma Union_bound_avoid:
  assumes "finite A"
  assumes "(∑a ∈ A. prob a) < 1"
  assumes "A ⊆ events"
  shows "prob (space M - ⋃A) > 0"
```

- The Complete Independence Bound

```
lemma complete_indep_bound3:
  assumes "finite A"
  assumes "A ≠ {}"
  assumes "F ` A ⊆ events"
  assumes "indep_events F A"
  assumes "⋀ a . a ∈ A ⟹ prob (F a) < 1"
  shows "prob (⋂a ∈ A. space M - F a) > 0"
```

The Formalisation Framework – Step 4

- Obtaining an object from a probability!
- Some basic rules

```
lemma prob_lt_one_obtain:
  assumes "{e ∈ space M . Q e} ∈ events"
  assumes "prob {e ∈ space M . Q e} < 1"
  obtains e where "e ∈ space M" and "¬ Q e"
```

```
lemma prob_gt_zero_obtain:
  assumes "{e ∈ space M . Q e} ∈ events"
  assumes "prob {e ∈ space M . Q e} > 0"
  obtains e where "e ∈ space M" and "Q e"
```

- Combining steps 3 & 4!

```
lemma Union_bound_obtain_fun:
  assumes "finite A"
  assumes "(∑a ∈ A. prob (f a)) < 1"
  assumes "f ` A ⊆ events"
  obtains e where "e ∈ space M" and "e ∉ ⋃( f` A)"
```

4. The Framework … In Practice

The Proof - Formalised

Proposition 1.3.1 [Erdős (1963a)] *Every n-uniform hypergraph with less than 2^{n-1} edges has property B. Therefore $m(n) \geq 2^{n-1}$.*

Proof. Let $H = (V, E)$ be an n-uniform hypergraph with less than 2^{n-1} edges. Color V randomly by two colors. For each edge $e \in E$, let A_e be the event that e is monochromatic. Clearly $\Pr[A_e] = 2^{1-n}$. Therefore

$$\Pr\left[\bigvee_{e \in E} A_e\right] \leq \sum_{e \in E} \Pr[A_e] < 1$$

and there is a two-coloring without monochromatic edges. ■

```
context fin_kuniform_hypergraph_nt
begin
proposition erdos_propertyB:
  assumes "size E < (2^(k - 1))"
  assumes "k > 0"
  shows "has_property_B"
proof -
(* (1) Set up the probability space: "Colour V randomly with two colours" *)
  interpret P: vertex_colour_space 𝒱 E 2
    by unfold_locales (auto simp add: order_ge_two)
(* (2) define the event to avoid - monochromatic edges *)
  define A where "A ≡(λ e. {f ∈ 𝒞² . mono_edge f e})"
(* (3) Calculation 1: Clearly Pr[Ae] = 2^(1- n). *)
  have pe: "⋀ e. e ∈ set_mset E ⟹ P.prob {f ∈ 𝒞² . mono_edge f e} = 2 powi (1 - int k)"
    using P.prob_monochromatic_edge uniform assms(1) by fastforce
(* (3) Calculation 2: Have Pr (of Ae for any e) ≤ Sum over e (Pr (A e)) < 1 *)
  have "(∑e ∈ set_mset E. P.prob (A e)) < 1"
  proof -
    have "int k - 1 = int (k - 1)" using assms by linarith
    then have "card (set_mset E) < 2 powi (int k - 1)" using card_size_set_mset[of E] assms by simp
    then have "(∑e ∈ (set_mset E). P.prob (A e)) < 2 powi (int k - 1) * 2 powi (1 - int k)"
      unfolding A_def using pe by simp
    moreover have "((2 :: real) powi ((int k) - 1)) * (2 powi (1 - (int k))) = 1"
      using power_int_add[of 2 "int k - 1" "1- int k"] by force
    ultimately show ?thesis using power_int_add[of 2 "int k - 1" "1- int k"] by simp
  qed
  moreover have "A ` (set_mset E) ⊆ P.events" unfolding A_def P.sets_eq by blast
(* (4) obtain a colouring avoiding bad events *)
  ultimately obtain f where "f ∈ 𝒞²" and "f ∉ ⋃(A `(set_mset E))"
    using P.Union_bound_obtain_fun[of "set_mset E" A] finite_set_mset P.space_eq by auto
  thus ?thesis using event_is_proper_colouring A_def is_n_colourable_def by auto
qed
```

The Proof

Proposition 1.3.1 **[Erdős (1963a)]** *Every n-uniform hypergraph with less than 2^{n-1} edges has property B. Therefore $m(n) \geq 2^{n-1}$.*

Proof. Let $H = (V, E)$ be an n-uniform hypergraph with less than 2^{n-1} edges. Color V randomly by two colors. For each edge $e \in E$, let A_e be the event that e is monochromatic. Clearly $\Pr[A_e] = 2^{1-n}$. Therefore

$$\Pr\left[\bigvee_{e \in E} A_e\right] \leq \sum_{e \in E} \Pr[A_e] < 1$$

and there is a two-coloring without monochromatic edges. ■

```
context fin_kuniform_hypergraph_nt
begin
proposition erdos_propertyB:
  assumes "size E < (2^(k - 1))"
  assumes "k > 0"
  shows "has_property_B"
proof -
(* (1) Set up the probability space: "Colour V randomly with two colours" *)
  interpret P: vertex_colour_space 𝒱 E 2
    by unfold_locales (auto simp add: order_ge_two)
(* (2) define the event to avoid - monochromatic edges *)
  define A where "A ≡(λ e. {f ∈ 𝒞² . mono_edge f e})"
(* (3) Calculation 1: Clearly Pr[Ae] = 2^(1- n). *)
  have pe: "⋀ e. e ∈ set_mset E ⟹ P.prob {f ∈ 𝒞² . mono_edge f e} = 2 powi (1 - int k)"
    using P.prob_monochromatic_edge uniform assms(1) by fastforce
(* (3) Calculation 2: Have Pr (of Ae for any e) ≤ Sum over e (Pr (A e)) < 1 *)
  have "(∑e ∈ set_mset E. P.prob (A e)) < 1"
  proof -
    have "int k - 1 = int (k - 1)" using assms by linarith
    then have "card (set_mset E) < 2 powi (int k - 1)" using card_size_set_mset[of E] assms by simp
    then have "(∑e ∈ (set_mset E). P.prob (A e)) < 2 powi (int k - 1) * 2 powi (1 - int k)"
      unfolding A_def using pe by simp
    moreover have "((2 :: real) powi ((int k) - 1)) * (2 powi (1 - (int k))) = 1"
      using power_int_add[of 2 "int k - 1" "1- int k"] by force
    ultimately show ?thesis using power_int_add[of 2 "int k - 1" "1- int k"] by simp
  qed
  moreover have "A ` (set_mset E) ⊆ P.events" unfolding A_def P.sets_eq by blast
(* (4) obtain a colouring avoiding bad events *)
  ultimately obtain f where "f ∈ 𝒞²" and "f ∉ ⋃(A `(set_mset E))"
    using P.Union_bound_obtain_fun[of "set_mset E" A] finite_set_mset P.space_eq by auto
  thus ?thesis using event_is_proper_colouring A_def is_n_colourable_def by auto
qed
```

A Side Note on Independence & Intuition

Clearly $\mathbf{Pr}[A_e] = 2^{1-n}$

- i.e. Clearly vertex colouring events are independent, so we can just apply P(AB) = P(A)P(B) right?
- **BUT - This is circular reasoning!**
  - To establish independence, we must prove the multiplication rule holds.
  - Use a counting lemma instead on sets of functions

```
lemma prob_edge_colour:
  assumes "e ∈# E" "c ∈ {0..<n}"
  shows "prob {f ∈ 𝒞ⁿ . mono_edge_col f e c} = 1/(n powi (card e))"
proof -
  have "card {0..<n} = n" by simp
  moreover have "𝒞ⁿ = 𝒱 →E {0..<n}" using all_n_vertex_colourings_alt by blast
  moreover have "{0..<n} ≠ {}" using n_not_zero by simp
  ultimately show ?thesis using prob_uniform_ex_fun_space[of 𝒱 _ "{0..<n}" e] n_not_zero
      finite_sets wellformed assms by (simp add: MU_def V_nempty mono_edge_col_def)
qed
```

5. Extensions & Discussion

Extensions of Work

- Formalisation of the Lovasz Local Lemma & Variations as a much more advanced bounding technique
- Extensive additions to libraries on conditional probability and independence
- Further applications to hypergraph colouring existence problems
- Future work: more techniques and applications to different incidence systems!

Formal Maths: Challenges and Insights

Challenges

- Human intuition is not easy to translate
- Search tools are great, but struggle with equivalent concepts/notation. Further documentation/annotation tools could help here.
- Static vs dynamic library management
- Calculations such as summations/products continue to be tricky.

Insights

- Enabled significant more detail on proofs (or established proofs for "intuitive" facts).
- Locales can mirror hierarchies effectively, transfer facts, and are great for modularity
- Modularity and Proof Engineering is important!
- Search tool developments & automation beneficial
- Connecting communities

Concluding Thoughts

- Formalisation of Mathematics has come a long way
- This project easily combined libraries across different fields (probability and combinatorics).
- Use of probability in proofs relies heavily on intuition, which presents many more opportunities for both challenges and deeper proof insights!
- Paper with more advanced work to come!

Contact: cle47@cl.cam.ac.uk