Properties of Random Graphs – Subgraph Containment

Lars Hupel

March 17, 2025

Abstract

Random graphs are graphs with a fixed number of vertices, where each edge is present with a fixed probability. We are interested in the probability that a random graph contains a certain pattern, for example a cycle or a clique. A very high edge probability gives rise to perhaps too many edges (which degrades performance for many algorithms), whereas a low edge probability might result in a disconnected graph. We prove a theorem about a threshold probability such that a higher edge probability will asymptotically almost surely produce a random graph with the desired subgraph.

Contents

1 Introduction

Random graphs have been introduced by Erdős and Rényi in [2]. They describe a probability space where, for a fixed number of vertices, each possible edge is present with a certain probability independent from other edges, but with the same probability for each edge. They study what properties emerge when increasing the number of vertices, or as they call it, “the evolution of such a random graph”. The theorem which we will prove here is a slightly different version from that in the first section of that paper.

Here, we are interested in the probability that a random graph contains a certain pattern, for example a cycle or a clique. A very high edge probability gives rise to perhaps too many edges, which is usually undesired since it degrades the performance of many algorithms, whereas a low edge probability might result in a disconnected graph. The central theorem determines a threshold probability such that a higher edge probability will asymptotically almost surely produce a random graph with the desired subgraph.

The proof is outlined in [1, § 11.4] and [3, § 3]. The work is based on the comprehensive formalization of probability theory in Isabelle/HOL and on a previous definition of graphs in a work by Noschinski [4]. There, Noschinski formalized the proof that graphs with arbitrarily large girth and chromatic number exist. While the proof in this paper uses a different approach, the definition of a probability space on edges turned out to be quite useful.

2 Miscellaneous and contributed lemmas

theory *Ugraph-Misc*
imports
  HOL−Probability.Probability
  Girth-Chromatic.Girth-Chromatic-Misc
begin

lemma *sum-square*:
  fixes $a :: {'}i \Rightarrow {'}a :: \{monoid\text{-}mult,\ semiring\text{-}0\}$
  shows $(\sum i \in I.\ a\ i)\hat{}2 = (\sum i \in I.\ \sum j \in I.\ a\ i * a\ j)$
  by (*simp only*: *sum-product power2-eq-square*)

lemma *sum-split*:
  $finite\ I \Longrightarrow$
  $(\sum i \in I.\ if\ p\ i\ then\ f\ i\ else\ g\ i) = (\sum i \mid i \in I \wedge p\ i.\ f\ i) + (\sum i \mid i \in I \wedge \neg p\ i.\ g\ i)$
  by (*simp add*: *sum.If-cases Int-def*)

lemma *sum-split2*:
  assumes *finite I*
  shows $(\sum i \mid i \in I \wedge P\ i.\ if\ Q\ i\ then\ f\ i\ else\ g\ i) = (\sum i \mid i \in I \wedge P\ i \wedge Q\ i.\ f\ i) + (\sum i \mid i \in I \wedge P\ i \wedge \neg Q\ i.\ g\ i)$

proof (*subst sum.If-cases*)
show *finite* $\{i \in I.\ P\ i\}$
using *assms* **by** *simp*

have $\{i \in I.\ P\ i\} \cap Collect\ Q = \{i \in I.\ P\ i \wedge Q\ i\}$ $\{i \in I.\ P\ i\} \cap -\ Collect\ Q = \{i \in I.\ P\ i \wedge \neg\ Q\ i\}$
by *auto*
thus $sum\ f\ (\{i \in I.\ P\ i\} \cap Collect\ Q) + sum\ g\ (\{i \in I.\ P\ i\} \cap -\ Collect\ Q) = sum\ f\ \{i \in I.\ P\ i \wedge Q\ i\} + sum\ g\ \{i \in I.\ P\ i \wedge \neg\ Q\ i\}$
by *presburger*
qed

lemma *sum-upper*:
fixes $f :: 'i \Rightarrow 'a :: ordered\text{-}comm\text{-}monoid\text{-}add$
assumes $finite\ I$ $\bigwedge i.\ i \in I \Longrightarrow 0 \leq f\ i$
shows $(\sum i \mid i \in I \wedge P\ i.\ f\ i) \leq sum\ f\ I$
proof −
have $sum\ f\ I = (\sum i \in I.\ if\ P\ i\ then\ f\ i\ else\ f\ i)$
by *simp*
hence $sum\ f\ I = (\sum i \mid i \in I \wedge P\ i.\ f\ i) + (\sum i \mid i \in I \wedge \neg P\ i.\ f\ i)$
by (*simp only*: *sum-split*[*OF* ‹*finite I*›])
moreover have $0 \leq (\sum i \mid i \in I \wedge \neg P\ i.\ f\ i)$
by (*rule sum-nonneg*) (*simp add*: *assms*)
ultimately show *?thesis*
by (*metis* (*full-types*) *add.comm-neutral add-left-mono*)
qed

lemma *sum-lower*:
fixes $f :: 'i \Rightarrow 'a :: ordered\text{-}comm\text{-}monoid\text{-}add$
assumes $finite\ I$ $i \in I$ $\bigwedge i.\ i \in I \Longrightarrow 0 \leq f\ i$ $x < f\ i$
shows $x < sum\ f\ I$
proof −
have $x < f\ i$ **by** *fact*
also have $\ldots \leq sum\ f\ I$
using *sum-mono2*[*OF* ‹*finite I*›, *of* $\{i\}$ *f*] *assms* **by** *auto*
finally show *?thesis* **.**
qed

lemma *sum-lower-or-eq*:
fixes $f :: 'i \Rightarrow 'a :: ordered\text{-}comm\text{-}monoid\text{-}add$
assumes $finite\ I$ $i \in I$ $\bigwedge i.\ i \in I \Longrightarrow 0 \leq f\ i$ $x \leq f\ i$
shows $x \leq sum\ f\ I$
proof −
have $x \leq f\ i$ **by** *fact*
also have $\ldots \leq sum\ f\ I$
using *sum-mono2*[*OF* ‹*finite I*›, *of* $\{i\}$ *f*] *assms* **by** *auto*
finally show *?thesis* **.**
qed

lemma *sum-left-div-distrib*:
  fixes $f :: 'i \Rightarrow real$
  shows $(\sum i \in I.\ f\ i\ /\ x) = sum\ f\ I\ /\ x$
proof −
  have $(\sum i \in I.\ f\ i\ /\ x) = (\sum i \in I.\ f\ i * (1\ /\ x))$
    by *simp*
  also have $\ldots = sum\ f\ I * (1\ /\ x)$
    by (*rule sum-distrib-right*[*symmetric*])
  also have $\ldots = sum\ f\ I\ /\ x$
    by *simp*
  finally show *?thesis*
    .
qed

lemma *powr-mono3*:
  fixes *x*::*real*
  assumes $0 < x$ $x < 1$ $b \le a$
  shows $x\ powr\ a \le x\ powr\ b$
proof −
  have $x\ powr\ a = 1\ /\ x\ powr\ -a$
    by (*simp add*: *powr-minus-divide*)
  also have $\ldots = (1\ /\ x)\ powr\ -a$
    using *assms* **by** (*simp add*: *powr-divide*)
  also have $\ldots \le (1\ /\ x)\ powr\ -b$
    using *assms* **by** (*simp add*: *powr-mono*)
  also have $\ldots = 1\ /\ x\ powr\ -b$
    using *assms* **by** (*simp add*: *powr-divide*)
  also have $\ldots = x\ powr\ b$
    by (*simp add*: *powr-minus-divide*)
  finally show *?thesis*
    .
qed

lemma *card-union*: $finite\ A \Longrightarrow finite\ B \Longrightarrow card\ (A \cup B) = card\ A + card\ B - card\ (A \cap B)$
by (*metis card-Un-Int*[*symmetric*] *diff-add-inverse2*)

lemma *card-1-element*:
  assumes $card\ E = 1$
  shows $\exists a.\ E = \{a\}$
proof −
  from *assms* **obtain** *a* **where** $a \in E$
    by *force*
  let $?E' = E - \{a\}$

  have $finite\ ?E'$
    using *assms card-ge-0-finite* **by** *force*
  hence $card\ (insert\ a\ ?E') = 1 + card\ ?E'$
    using *card.insert-remove* **by** *fastforce*

```
  moreover have E = insert a ?E'
    using ‹a ∈ E› by blast
  ultimately have card E = 1 + card ?E'
    by simp
  hence card ?E' = 0
    using assms by simp
  hence ?E' = {}
    using ‹finite ?E'› by simp
  thus ?thesis
    using ‹a ∈ E› by blast
qed

lemma card-2-elements:
  assumes card E = 2
  shows ∃ a b. E = {a, b} ∧ a ≠ b
proof −
  from assms obtain a where a ∈ E
    by force
  let ?E' = E − {a}

  have finite ?E'
    using assms card-ge-0-finite by force
  hence card (insert a ?E') = 1 + card ?E'
    using card.insert-remove by fastforce
  moreover have E = insert a ?E'
    using ‹a ∈ E› by blast
  ultimately have card E = 1 + card ?E'
    by simp
  hence card ?E' = 1
    using assms by simp
  then obtain b where ?E' = {b}
    using card-1-element by blast
  hence E = {a, b}
    using ‹a ∈ E› by blast
  moreover have a ≠ b
    using ‹?E' = {b}› by blast
  ultimately show ?thesis
    by blast
qed

lemma bij-lift:
  assumes bij-betw f A B
  shows bij-betw (λe. f ‘ e) (Pow A) (Pow B)
proof −
  have f: inj-on f A f ‘ A = B
    using assms unfolding bij-betw-def by simp-all
  have inj-on (λe. f ‘ e) (Pow A)
    unfolding inj-on-def by clarify (metis f(1) inv-into-image-cancel)
  moreover have (λe. f ‘ e) ‘ (Pow A) = (Pow B)
```

by (*metis* $f(2)$ *image-Pow-surj*)
ultimately show *?thesis*
unfolding *bij-betw-def* **by** *simp*
qed

lemma *card-inj-subs*: *inj-on* f $A \Longrightarrow B \subseteq A \Longrightarrow$ *card* $(f$ ` $B)$ = *card* B
by (*metis card-image subset-inj-on*)

lemma *image-comp-cong*: $(\bigwedge a.\ a \in A \Longrightarrow f\ a = f\ (g\ a)) \Longrightarrow f$ ` $A = f$ ` $(g$ ` $A)$
by *auto*

abbreviation *less-fun* :: $(nat \Rightarrow real) \Rightarrow (nat \Rightarrow real) \Rightarrow bool$ (**infix** ‹≪› *50*)
where
$f \ll g \equiv (\lambda n.\ f\ n\ /\ g\ n) \longrightarrow 0$

context
fixes f :: $nat \Rightarrow real$
begin

lemma *LIMSEQ-power-zero*: $f \longrightarrow 0 \Longrightarrow 0 < n \Longrightarrow (\lambda x.\ f\ x\ \hat{}\ n :: real) \longrightarrow 0$
by (*metis power-eq-0-iff tendsto-power*)

lemma *LIMSEQ-cong*:
assumes $f \longrightarrow x$ $\forall^\infty n.\ f\ n = g\ n$
shows $g \longrightarrow x$
by (*rule real-tendsto-sandwich*[**where** $f = f$ **and** $h = f$, *OF eventually-mono*[*OF assms*(*2*)] *eventually-mono*[*OF assms*(*2*)]]) (*auto simp*: *assms*(*1*))
print-statement *Lim-transform-eventually*

lemma *LIMSEQ-le-zero*:
assumes $g \longrightarrow 0$ $\forall^\infty n.\ 0 \le f\ n$ $\forall^\infty n.\ f\ n \le g\ n$
shows $f \longrightarrow 0$
by (*rule real-tendsto-sandwich*[*OF assms*(*2*) *assms*(*3*) *tendsto-const assms*(*1*)])

lemma *LIMSEQ-const-mult*:
assumes $f \longrightarrow a$
shows $(\lambda x.\ c * f\ x) \longrightarrow c * a$
by (*rule tendsto-mult*[*OF tendsto-const*[**where** $k = c$] *assms*])

lemma *LIMSEQ-const-div*:
assumes $f \longrightarrow a$ $c \neq 0$
shows $(\lambda x.\ f\ x\ /\ c) \longrightarrow a\ /\ c$
using *LIMSEQ-const-mult*[**where** $c = 1/c$] *assms* **by** *simp*

end

lemma *quot-bounds*:

fixes x :: $'a$ :: *linordered-field*
assumes $x \leq x'$ $y' \leq y$ $0 < y$ $0 \leq x$ $0 < y'$
shows $x \ / \ y \leq x' \ / \ y'$
proof (*rule order-trans*)
have $0 \leq y$
using *assms* **by** *simp*
thus $x \ / \ y \leq x' \ / \ y$
using *assms* **by** (*simp add*: *divide-right-mono*)
next
have $0 \leq x'$
using *assms* **by** *simp*
moreover have $0 < y * y'$
using *assms* **by** *simp*
ultimately show $x' \ / \ y \leq x' \ / \ y'$
using *assms* **by** (*simp add*: *divide-left-mono*)
qed

lemma *less-fun-bounds*:
assumes $f' \ll g'$ $\forall^\infty n.\ f\ n \leq f'\ n$ $\forall^\infty n.\ g'\ n \leq g\ n$ $\forall^\infty n.\ 0 \leq f\ n$ $\forall^\infty n.\ 0 < g\ n$ $\forall^\infty n.\ 0 < g'\ n$
shows $f \ll g$
proof (*rule real-tendsto-sandwich*)
show $\forall^\infty n.\ 0 \leq f\ n \ / \ g\ n$
using *assms*(*4*,*5*) **by** *eventually-elim simp*
next
show $\forall^\infty n.\ f\ n \ / \ g\ n \leq f'\ n \ / \ g'\ n$
using *assms*(*2*−) **by** *eventually-elim* (*simp only*: *quot-bounds*)
qed (*auto intro*: *assms*(*1*))

lemma *less-fun-const-quot*:
assumes $f \ll g$ $c \neq 0$
shows $(\lambda n.\ b * f\ n) \ll (\lambda n.\ c * g\ n)$
proof −
have $(\lambda n.\ (b * (f\ n \ / \ g\ n)) \ / \ c) \longrightarrow (b * 0) \ / \ c$
using *assms* **by** (*rule LIMSEQ-const-div*[*OF LIMSEQ-const-mult*])
hence $(\lambda n.\ (b * (f\ n \ / \ g\ n)) \ / \ c) \longrightarrow 0$
by *simp*
with *eventually-sequentiallyI* **show** *?thesis*
by (*fastforce intro*: *Lim-transform-eventually*)
qed

lemma *partition-set-of-intersecting-sets-by-card*:
assumes *finite A*
shows $\{B.\ A \cap B \neq \{\}\} = (\bigcup n \in \{1..card\ A\}.\ \{B.\ card\ (A \cap B) = n\})$
proof (*rule set-eqI*, *rule iffI*)
fix B
assume $B \in \{B.\ A \cap B \neq \{\}\}$
hence $0 < card\ (A \cap B)$
using *assms* **by** *auto*

moreover have *card* $(A \cap B) \leq$ *card* A
  using *assms* **by** (*simp add*: *card-mono*)
ultimately have *card* $(A \cap B) \in \{1..card\ A\}$
  by *simp*
thus $B \in (\bigcup n \in \{1..card\ A\}.\ \{B.\ card\ (A \cap B) = n\})$
  by *blast*
qed *force*

lemma *card-set-of-intersecting-sets-by-card*:
  assumes $A \subseteq I$ *finite* I $k \leq n$ $n \leq$ *card* I $k \leq$ *card* A
  shows *card* $\{B.\ B \subseteq I \wedge card\ B = n \wedge card\ (A \cap B) = k\} = (card\ A\ choose\ k) * ((card\ I - card\ A)\ choose\ (n - k))$
proof $-$
  note *finite-A* = *finite-subset*[*OF assms*(*1*,*2*)]

  have *card* $\{B.\ B \subseteq I \wedge card\ B = n \wedge card\ (A \cap B) = k\} = card\ (\{K.\ K \subseteq A \wedge card\ K = k\} \times \{B'.\ B' \subseteq I - A \wedge card\ B' = n - k\})$ (**is** *card ?lhs* = *card ?rhs*)
    proof (*rule bij-betw-same-card*[*symmetric*])
      let *?f* = $\lambda(K, B').\ K \cup B'$
      have *inj-on ?f ?rhs*
        by (*blast intro*: *inj-onI*)
      moreover have *?f* ‘ *?rhs* = *?lhs*
        proof (*rule set-eqI*, *rule iffI*)
          fix B
          assume $B \in$ *?f* ‘ *?rhs*
          then obtain K B' **where** K: $K \subseteq A$ *card* $K = k$ $B' \subseteq I - A$ *card* $B' = n - k$ $K \cup B' = B$
            by *blast*
          show $B \in$ *?lhs*
            proof *safe*
              fix x **assume** $x \in B$ **thus** $x \in I$
                using K ‹$A \subseteq I$› **by** *blast*
            next
              have *card* B = *card* K + *card* B' − *card* $(K \cap B')$
                using K *assms* **by** (*metis card-union finite-A finite-subset finite-Diff*)
              moreover have $K \cap B' = \{\}$
                using K *assms* **by** *blast*
              ultimately show *card* $B = n$
                using K *assms* **by** *simp*
            next
              have $A \cap B = K$
                using K *assms*(*1*) **by** *blast*
              thus *card* $(A \cap B) = k$
                using K **by** *simp*
            qed
        next
          fix B
          assume $B \in$ *?lhs*
          hence B: $B \subseteq I$ *card* $B = n$ *card* $(A \cap B) = k$

by *auto*
let *?K* = *A* ∩ *B*
let *?B′* = *B* − *A*
have *?K* ⊆ *A* *card ?K* = *k* *?B′* ⊆ *I* − *A*
using *B* **by** *auto*
moreover have *card ?B′* = *n* − *k*
using *B finite-A assms*(*1*) **by** (*metis Int-commute card-Diff-subset-Int finite-Un inf.left-idem le-iff-inf sup-absorb2*)
ultimately have (*?K*, *?B′*) ∈ *?rhs*
by *blast*
moreover have *B* = *?f* (*?K*, *?B′*)
by *auto*
ultimately show *B* ∈ *?f* ‘ *?rhs*
by *blast*
qed
ultimately show *bij-betw ?f ?rhs ?lhs*
unfolding *bij-betw-def* **..**
qed
also have ... = ($\sum$ *K* | *K* ⊆ *A* ∧ *card K* = *k*. *card* {*B′*. *B′* ⊆ *I* − *A* ∧ *card B′* = *n* − *k*})
proof (*rule card-SigmaI*, *safe*)
show *finite* {*K*. *K* ⊆ *A* ∧ *card K* = *k*}
by (*blast intro*: *finite-subset*[**where** *B* = *Pow A*] *finite-A*)
next
fix *K*
assume *K* ⊆ *A*
thus *finite* {*B′*. *B′* ⊆ *I* − *A* ∧ *card B′* = *n* − *card K*}
using *assms* **by** *auto*
qed
also have ... = *card* {*K*. *K* ⊆ *A* ∧ *card K* = *k*} ∗ *card* {*B′*. *B′* ⊆ *I* − *A* ∧ *card B′* = *n* − *k*}
by *simp*
also have ... = (*card A choose k*) ∗ (*card* (*I* − *A*) *choose* (*n* − *k*))
by (*simp only*: *n-subsets*[*OF finite-A*] *n-subsets*[*OF finite-Diff*[*OF assms*(*2*)]])
also have ... = (*card A choose k*) ∗ ((*card I* − *card A*) *choose* (*n* − *k*))
by (*simp only*: *card-Diff-subset*[*OF finite-A assms*(*1*)])
finally show *?thesis*
.
qed

lemma *card-dep-pair-set*:
assumes *finite A* ⋀*a*. *a* ⊆ *A* ⟹ *finite* (*f a*)
shows *card* {(*a*, *b*). *a* ⊆ *A* ∧ *card a* = *n* ∧ *b* ⊆ *f a* ∧ *card b* = *g a*} = ($\sum$ *a* | *a* ⊆ *A* ∧ *card a* = *n*. *card* (*f a*) *choose g a*) (**is** *card ?S* = *?C*)
proof −
have *S*: *?S* = *Sigma* {*a*. *a* ⊆ *A* ∧ *card a* = *n*} (λ*a*. {*b*. *b* ⊆ *f a* ∧ *card b* = *g a*}) (**is** - = *Sigma ?A ?B*)
by *auto*

have *card* (*Sigma ?A ?B*) = ($\sum$ *a* ∈ {*a*. *a* ⊆ *A* ∧ *card a* = *n*}. *card* (*?B a*))
proof (*rule card-SigmaI*, *safe*)
show *finite ?A*
by (*rule finite-subset*[*OF - finite-Collect-subsets*[*OF assms*(*1*)]]) *blast*
next
fix *a*
assume *a* ⊆ *A*
hence *finite* (*f a*)
by (*fact assms*(*2*))
thus *finite* (*?B a*)
by (*rule finite-subset*[*rotated*, *OF finite-Collect-subsets*]) *blast*
qed
also have . . . = *?C*
proof (*rule sum.cong*)
fix *a*
assume *a* ∈ {*a*. *a* ⊆ *A* ∧ *card a* = *n*}
hence *finite* (*f a*)
using *assms*(*2*) **by** *blast*
thus *card* (*?B a*) = *card* (*f a*) *choose g a*
by (*fact n-subsets*)
qed *simp*
finally have *card* (*Sigma ?A ?B*) = *?C*
.

thus *?thesis*
by (*subst S*)
qed

lemma *prod-cancel-nat*:
— Contributed by Manuel Eberl
fixes *f*::′*a* ⇒ *nat*
assumes *B* ⊆ *A* **and** *finite A* **and** ∀ *x*∈*B*. *f x* ≠ *0*
shows *prod f A* / *prod f B* = *prod f* (*A* − *B*) (**is** *?A* / *?B* = *?C*)
proof−
from *prod.subset-diff*[*OF assms*(*1*,*2*)] **have** *?A* = *?C* ∗ *?B* **by** *auto*
moreover have *?B* ≠ *0* **using** *assms* **by** (*simp add*: *finite-subset*)
ultimately show *?thesis* **by** *simp*
qed

lemma *prod-id-cancel-nat*:
— Contributed by Manuel Eberl
fixes *A*::*nat set*
assumes *B* ⊆ *A* **and** *finite A* **and** *0* ∉ *B*
shows $\prod$ *A* / $\prod$ *B* = $\prod$ (*A*−*B*)
using *assms*(*1*−*2*) **by** (*rule prod-cancel-nat*) (*metis assms*(*3*))

lemma (**in** *prob-space*) *integrable-squareD*:
— Contributed by Johannes Hölzl
fixes *X* :: - ⇒ *real*

assumes *integrable M* ($\lambda x.$ (X x)^2) $X \in$ *borel-measurable M*
shows *integrable M X*
proof –
have *integrable M* ($\lambda x.$ *max 1* ((X x)^2))
using *assms* **by** *auto*
then show *integrable M X*
proof (*rule Bochner-Integration.integrable-bound*[*OF - - always-eventually*[*OF allI*]])
fix *x* **show** *norm* (X x) $\leq$ *norm* (*max 1* ((X x)^2))
using *abs-le-square-iff*[*of 1 X x*] *power-increasing*[*of 1 2 abs* (X x)]
by (*auto split*: *split-max*)
qed *fact*
qed

end
theory *Prob-Lemmas*
imports
HOL−Probability.Probability
Girth-Chromatic.Girth-Chromatic
Ugraph-Misc
begin

3 Lemmas about probabilities

In this section, auxiliary lemmas for computing bounds on expectation and probabilites of random variables are set up.

3.1 Indicator variables and valid probability values

abbreviation *rind* :: $'a$ *set* $\Rightarrow$ $'a$ $\Rightarrow$ *real* **where**
rind $\equiv$ *indicator*

lemma *product-indicator*:
rind A x $*$ *rind B x* = *rind* ($A \cap B$) *x*
unfolding *indicator-def*
by *auto*

We call a real number 'valid' iff it is in the range 0 to 1, inclusively, and additionally 'nonzero' iff it is neither 0 nor 1.

abbreviation *valid-prob* (p :: *real*) $\equiv$ $0 \leq p \wedge p \leq 1$
abbreviation *nonzero-prob* (p :: *real*) $\equiv$ $0 < p \wedge p < 1$

A function $'a \Rightarrow$ *real* is a 'valid probability function' iff each value in the image is valid, and similarly for 'nonzero'.

abbreviation *valid-prob-fun f* $\equiv$ ($\forall n.$ *valid-prob* (f n))
abbreviation *nonzero-prob-fun f* $\equiv$ ($\forall n.$ *nonzero-prob* (f n))

lemma *nonzero-fun-is-valid-fun*: *nonzero-prob-fun f* $\Longrightarrow$ *valid-prob-fun f*
by (*simp add*: *less-imp-le*)

3.2 Expectation and variance

context *prob-space*
begin

Note that there is already a notion of independent sets (see *indep-set*), but we use the following – simpler – definition:

definition *indep A B* $\longleftrightarrow$ *prob* $(A \cap B)$ = *prob A* $*$ *prob B*

The probability of an indicator variable is equal to its expectation:

lemma *expectation-indicator*:
  $A \in$ *events* $\Longrightarrow$ *expectation* (*rind A*) = *prob A*
  by *simp*

For a non-negative random variable X, the Markov inequality gives the following upper bound:

$$\Pr[X \geq a] \leq \frac{\mathrm{E}[X]}{a}$$

lemma *markov-inequality*:
  assumes $\bigwedge a.\ 0 \leq X\ a$ **and** *integrable M X* $0 < t$
  shows *prob* $\{a \in$ *space* $M.\ t \leq X\ a\} \leq$ *expectation* $X\ /\ t$
proof –
  — proof adapted from *edge-space.Markov-inequality*, but generalized to arbitrary *prob-spaces*
  have $(\int^{+} x.$ *ennreal* $(X\ x)\ \partial M) = (\int x.\ X\ x\ \partial M)$
    using *assms* **by** (*intro nn-integral-eq-integral*) *auto*
  thus *?thesis*
    using *assms nn-integral-Markov-inequality*[*of X space M M 1 / t*]
   by (*auto cong*: *nn-integral-cong simp*: *emeasure-eq-measure ennreal-mult*[*symmetric*])
qed

$\mathrm{Var}[X] = \mathrm{E}[X^2] - \mathrm{E}[X]^2$

lemma *variance-expectation*:
  fixes X :: $'a \Rightarrow$ *real*
  assumes *integrable M* $(\lambda x.\ (X\ x)\hat{}2)$ **and** $X \in$ *borel-measurable M*
  shows
    integrable M $(\lambda x.\ (X\ x -$ *expectation* $X)\hat{}2)$ (**is** *?integrable*)
    variance X = *expectation* $(\lambda x.\ (X\ x)\hat{}2) - ($*expectation* $X)\hat{}2$ (**is** *?variance*)
proof –
  have *int*: *integrable M X*
    using *integrable-squareD*[*OF assms*] **by** *simp*

  have $(\lambda x.\ (X\ x -$ *expectation* $X)\hat{}2) = (\lambda x.\ (X\ x)\hat{}2 + ($*expectation* $X)\hat{}2 - (2 * X\ x *$ *expectation* $X))$
    by (*simp only*: *power2-diff*)

hence
variance X = expectation (λx. (X x)^2) + (expectation X)^2 + expectation (λx. − (2 ∗ X x ∗ expectation X))
?integrable
using *integral-add* **by** (*simp add*: *int assms prob-space*)+

thus *?variance ?integrable*
by (*simp add*: *int power2-eq-square*)+
qed

A corollary from the Markov inequality is Chebyshev's inequality, which gives an upper bound for the deviation of a random variable from its expectation:

$$\Pr[|Y - \mathrm{E}[Y]| \geq s] \leq \frac{\mathrm{Var}[X]}{a^2}$$

lemma *chebyshev-inequality*:
fixes *Y* :: *'a ⇒ real*
assumes *Y-int*: *integrable M (λy. (Y y)^2)*
assumes *Y-borel*: *Y ∈ borel-measurable M*
fixes *s* :: *real*
assumes *s-pos*: *0 < s*
shows *prob {a ∈ space M. s ≤ |Y a − expectation Y|} ≤ variance Y / s^2*
proof −
let *?X = λa. (Y a − expectation Y)^2*
let *?t = s^2*

have *0 < ?t*
using *s-pos* **by** *simp*
hence *prob {a ∈ space M. ?t ≤ ?X a} ≤ variance Y / s^2*
using *markov-inequality variance-expectation[OF Y-int Y-borel]* **by** (*simp add*: *field-simps*)
moreover have *{a ∈ space M. ?t ≤ ?X a} = {a ∈ space M. s ≤ |Y a − expectation Y|}*
using *abs-le-square-iff s-pos* **by** *force*
ultimately show *?thesis*
by *simp*
qed

Hence, we can derive an upper bound for the probability that a random variable is 0.

corollary *chebyshev-prob-zero*:
fixes *Y* :: *'a ⇒ real*
assumes *Y-int*: *integrable M (λy. (Y y)^2)*
assumes *Y-borel*: *Y ∈ borel-measurable M*
assumes *μ-pos*: *expectation Y > 0*
shows *prob {a ∈ space M. Y a = 0} ≤ expectation (λy. (Y y)^2) / (expectation Y)^2 − 1*
proof −

let $?s$ = *expectation* Y

have *prob* $\{a \in$ *space* $M.\ Y\ a = 0\} \leq$ *prob* $\{a \in$ *space* $M.\ ?s \leq |Y\ a - ?s|\}$
using *Y-borel* **by** (*auto intro*!: *finite-measure-mono borel-measurable-diff borel-measurable-abs borel-measurable-le*)
also have $\ldots \leq$ *variance* $Y\ /\ ?s\hat{}2$
using *assms* **by** (*fact chebyshev-inequality*)
also have $\ldots = ($*expectation* $(\lambda y.\ (Y\ y)\hat{}2) - ?s\hat{}2)\ /\ ?s\hat{}2$
using *Y-int Y-borel* **by** (*simp add*: *variance-expectation*)
also have $\ldots =$ *expectation* $(\lambda y.\ (Y\ y)\hat{}2)\ /\ ?s\hat{}2 - 1$
using *μ-pos* **by** (*simp add*: *field-simps*)
finally show *?thesis* **.**
qed

end

3.3 Sets of indicator variables

This section introduces some inequalities about expectation and other values related to the sum of a set of random indicators.

locale *prob-space-with-indicators* = *prob-space* +
fixes I :: *'i set*
assumes *finite-I*: *finite* I

fixes A :: $'i \Rightarrow\ 'a$ *set*
assumes A: A ‘ $I \subseteq$ *events*

assumes *prob-non-zero*: $\exists\, i \in I.\ 0 <$ *prob* $(A\ i)$
begin

We call the underlying sets $A\ i$ for each $i \in I$, and the corresponding indicator variables $X\ i$. The sum is denoted by Y, and its expectation by μ.

definition $X\ i =$ *rind* $(A\ i)$
definition $Y\ x = (\sum i \in I.\ X\ i\ x)$

definition $\mu =$ *expectation* Y

In the lecture notes, the following two relations are called $\sim$ and $\nsim$, respectively. Note that they are not the opposite of each other.

abbreviation *ineq-indep* :: $'i \Rightarrow\ 'i \Rightarrow$ *bool* **where**
ineq-indep $i\ j \equiv (i \neq j \land$ *indep* $(A\ i)\ (A\ j))$

abbreviation *ineq-dep* :: $'i \Rightarrow\ 'i \Rightarrow$ *bool* **where**
ineq-dep $i\ j \equiv (i \neq j \land \neg$*indep* $(A\ i)\ (A\ j))$

definition $\Delta_a = (\sum i \in I.\ \sum j \mid j \in I \land i \neq j.$ *prob* $(A\ i \cap A\ j))$
definition $\Delta_d = (\sum i \in I.\ \sum j \mid j \in I \land$ *ineq-dep* $i\ j.$ *prob* $(A\ i \cap A\ j))$

lemma Δ-*zero*:
  assumes $\bigwedge i\ j.\ i \in I \Longrightarrow j \in I \Longrightarrow i \neq j \Longrightarrow indep\ (A\ i)\ (A\ j)$
  shows $\Delta_d = 0$
proof −
  {
    fix *i*
    assume $i \in I$
    hence $\{j.\ j \in I \wedge ineq\text{-}dep\ i\ j\} = \{\}$
      using *assms* **by** *auto*
    hence $(\sum j \mid j \in I \wedge ineq\text{-}dep\ i\ j.\ prob\ (A\ i \cap A\ j)) = 0$
      using *sum.empty* **by** *metis*
  }
  hence $\Delta_d = (0 :: real) * card\ I$
    unfolding Δ_d-*def* **by** *simp*
  thus *?thesis*
    by *simp*
qed

lemma *A-events*[*measurable*]: $i \in I \Longrightarrow A\ i \in events$
using *A* **by** *auto*

lemma *expectation-X-Y*: $\mu = (\sum i{\in}I.\ expectation\ (X\ i))$
unfolding μ-*def* *Y-def*[*abs-def*] *X-def*
by (*simp add*: *less-top*[*symmetric*])

lemma *expectation-X-non-zero*: $\exists\, i \in I.\ 0 < expectation\ (X\ i)$
unfolding *X-def* **using** *prob-non-zero expectation-indicator* **by** *simp*

corollary μ-*non-zero*[*simp*]: $0 < \mu$
unfolding *expectation-X-Y*
using *expectation-X-non-zero*
by (*auto intro*!: *sum-lower finite-I*
        simp add: *expectation-indicator X-def*)

lemma Δ_d-*nonneg*: $0 \leq \Delta_d$
unfolding Δ_d-*def*
by (*simp add*: *sum-nonneg*)

corollary μ-*sq-non-zero*[*simp*]: $0 < \mu\hat{\ }2$
by (*rule zero-less-power*) *simp*

lemma *Y-square-unfold*: $(\lambda x.\ (Y\ x)\hat{\ }2) = (\lambda x.\ \sum i \in I.\ \sum j \in I.\ rind\ (A\ i \cap A\ j)\ x)$
unfolding *fun-eq-iff Y-def X-def*
by (*auto simp*: *sum-square product-indicator*)

lemma *integrable-Y-sq*[*simp*]: *integrable M* $(\lambda y.\ (Y\ y)\hat{\ }2)$
unfolding *Y-square-unfold*

by (*simp add*: *sets.Int less-top*[*symmetric*])

lemma *measurable-Y*[*measurable*]: $Y \in$ *borel-measurable M*
unfolding *Y-def*[*abs-def*] *X-def* **by** *simp*

lemma *expectation-Y-Δ*: *expectation* $(\lambda x.\ (Y\ x)\hat{\ }2) = \mu + \Delta_a$
proof −
  let $?ei = \lambda i\ j.\ expectation\ (rind\ (A\ i \cap A\ j))$

  have $expectation\ (\lambda x.\ (Y\ x)\hat{\ }2) = (\sum i \in I.\ \sum j \in I.\ ?ei\ i\ j)$
    unfolding *Y-square-unfold* **by** (*simp add*: *less-top*[*symmetric*])
  also have $\ldots = (\sum i \in I.\ \sum j \in I.\ if\ i = j\ then\ ?ei\ i\ j\ else\ ?ei\ i\ j)$
    by *simp*
  also have $\ldots = (\sum i \in I.\ (\sum j \mid j \in I \wedge i = j.\ ?ei\ i\ j) + (\sum j \mid j \in I \wedge i \neq j.$
$?ei\ i\ j))$
    by (*simp only*: *sum-split*[*OF finite-I*])
  also have $\ldots = (\sum i \in I.\ \sum j \mid j \in I \wedge i = j.\ ?ei\ i\ j) + (\sum i \in I.\ \sum j \mid j \in I$
$\wedge\ i \neq j.\ ?ei\ i\ j)$ (**is** - = *?lhs* + *?rhs*)
    by (*fact sum.distrib*)
  also have $\ldots = \mu + \Delta_a$
    proof −
      have $?lhs = \mu$
        proof −
          {
            fix *i*
            assume *i*: $i \in I$
            have $(\sum j \mid j \in I \wedge i = j.\ ?ei\ i\ j) = (\sum j \mid j \in I \wedge i = j.\ ?ei\ i\ i)$
              by *simp*
            also have $\ldots = (\sum j \mid i = j.\ ?ei\ i\ i)$
              using *i* **by** *metis*
            also have $\ldots = expectation\ (rind\ (A\ i))$
              by *auto*
            finally have $(\sum j \mid j \in I \wedge i = j.\ ?ei\ i\ j) = \ldots$ **.**
          }
          hence $?lhs = (\sum i{\in}I.\ expectation\ (rind\ (A\ i)))$
            by *force*
          also have $\ldots = \mu$
            unfolding *expectation-X-Y X-def* **..**
          finally show $?lhs = \mu$ **.**
        qed
      moreover have $?rhs = \Delta_a$
        proof −
          {
            fix *i j*
            assume $i \in I\ j \in I$
            with *A* **have** $A\ i \cap A\ j \in events$ **by** *blast*
            hence $?ei\ i\ j = prob\ (A\ i \cap A\ j)$
              by (*fact expectation-indicator*)
          }

thus *?thesis*
unfolding Δ_a*-def* **by** *simp*
qed
ultimately show *?lhs* + *?rhs* = $\mu + \Delta_a$
by *simp*
qed
finally show *?thesis* **.**
qed

lemma Δ*-expectation-X*: $\Delta_a \leq \mu\hat{\ }2 + \Delta_d$
proof −
let $?p = \lambda i\, j.\ prob\ (A\ i \cap A\ j)$
let $?p' = \lambda i\, j.\ prob\ (A\ i) * prob\ (A\ j)$
let $?ie = \lambda i\, j.\ indep\ (A\ i)\ (A\ j)$

have $\Delta_a = (\sum i \in I.\ \sum j \mid j \in I \wedge i \neq j.\ if\ ?ie\ i\ j\ then\ ?p\ i\ j\ else\ ?p\ i\ j)$
unfolding Δ_a*-def* **by** *simp*
also have $\ldots = (\sum i \in I.\ (\sum j \mid j \in I \wedge ineq\text{-}indep\ i\ j.\ ?p\ i\ j) + (\sum j \mid j \in I \wedge ineq\text{-}dep\ i\ j.\ ?p\ i\ j))$
by (*simp only*: *sum-split2*[*OF finite-I*])
also have $\ldots = (\sum i \in I.\ \sum j \mid j \in I \wedge ineq\text{-}indep\ i\ j.\ ?p\ i\ j) + \Delta_d$ (**is** - = *?lhs* + -)
unfolding Δ_d*-def* **by** (*fact sum.distrib*)
also have $\ldots \leq \mu\hat{\ }2 + \Delta_d$
proof (*rule add-right-mono*)
have $(\sum i{\in}I.\ \sum j \mid j \in I \wedge ineq\text{-}indep\ i\ j.\ ?p\ i\ j) = (\sum i \in I.\ \sum j \mid j \in I \wedge ineq\text{-}indep\ i\ j.\ ?p'\ i\ j)$
unfolding *indep-def* **by** *simp*
also have $\ldots \leq (\sum i \in I.\ \sum j \in I.\ ?p'\ i\ j)$
proof (*rule sum-mono*)
fix i
assume $i \in I$
show $(\sum j \mid j \in I \wedge ineq\text{-}indep\ i\ j.\ ?p'\ i\ j) \leq (\sum j{\in}I.\ ?p'\ i\ j)$
by (*rule sum-upper*[*OF finite-I*]) (*simp add*: *zero-le-mult-iff*)
qed
also have $\ldots = (\sum i \in I.\ prob\ (A\ i))\hat{\ }2$
by (*fact sum-square*[*symmetric*])
also have $\ldots = (\sum i \in I.\ expectation\ (X\ i))\hat{\ }2$
unfolding *X-def* **using** *expectation-indicator A* **by** *simp*
also have $\ldots = \mu\hat{\ }2$
using *expectation-X-Y*[*symmetric*] **by** *simp*
finally show $?lhs \leq \mu\hat{\ }2$ **.**
qed
finally show *?thesis* **.**
qed

lemma *prob-*μ*-*Δ_a: $prob\ \{a \in space\ M.\ Y\ a = 0\} \leq 1\ /\ \mu + \Delta_a\ /\ \mu\hat{\ }2 - 1$
proof −
have $prob\ \{a \in space\ M.\ Y\ a = 0\} \leq expectation\ (\lambda y.\ (Y\ y)\hat{\ }2)\ /\ \mu\hat{\ }2 - 1$

unfolding *μ-def* **by** (*rule chebyshev-prob-zero*) (*simp add*: *μ-def*[*symmetric*])+
also have ... $= (\mu + \Delta_a) \ / \ \mu\hat{\ }2 - 1$
using *expectation-Y-Δ* **by** *simp*
also have ... $= 1 \ / \ \mu + \Delta_a \ / \ \mu\hat{\ }2 - 1$
unfolding *power2-eq-square* **by** (*simp add*: *field-simps add-divide-distrib*)
finally show *?thesis* **.**
qed

lemma *prob-μ-*Δ_d: *prob* $\{a \in space\ M.\ Y\ a = 0\} \leq 1/\mu + \Delta_d/\mu\hat{\ }2$
proof −
have *prob* $\{a \in space\ M.\ Y\ a = 0\} \leq 1/\mu + \Delta_a/\mu\hat{\ }2 - 1$
by (*fact prob-μ-*Δ_a)
also have ... $= (1/\mu - 1) + \Delta_a/\mu\hat{\ }2$
by *simp*
also have ... $\leq (1/\mu - 1) + (\mu\hat{\ }2 + \Delta_d)/\mu\hat{\ }2$
using *divide-right-mono*[*OF Δ-expectation-X*] **by** *simp*
also have ... $= 1/\mu + \Delta_d/\mu\hat{\ }2$
using *μ-sq-non-zero* **by** (*simp add*: *field-simps*)
finally show *?thesis* **.**
qed

end

end

4 Lemmas about undirected graphs

theory *Ugraph-Lemmas*
imports
Prob-Lemmas
Girth-Chromatic.Girth-Chromatic
begin

The complete graph is a graph where all possible edges are present. It is wellformed by definition.

definition *complete* :: *nat set* ⇒ *ugraph* **where**
complete V = (*V*, *all-edges V*)

lemma *complete-wellformed*: *uwellformed* (*complete V*)
unfolding *complete-def uwellformed-def all-edges-def*
by *simp*

If the set of vertices is finite, the set of edges in the complete graph is finite.

lemma *all-edges-finite*: *finite V* ⟹ *finite* (*all-edges V*)
unfolding *all-edges-def*
by *simp*

corollary *complete-finite-edges*: *finite V* ⟹ *finite* (*uedges* (*complete V*))

unfolding *complete-def* **using** *all-edges-finite*
by *simp*

The sets of possible edges of disjoint sets of vertices are disjoint.

lemma *all-edges-disjoint*: $S \cap T = \{\} \Longrightarrow$ *all-edges* $S \cap$ *all-edges* $T = \{\}$
unfolding *all-edges-def*
by *force*

A graph is called 'finite' if its set of edges and its set of vertices are finite.

definition *finite-graph* $G \equiv$ *finite* (*uverts* G) $\wedge$ *finite* (*uedges* G)

The complete graph is finite.

corollary *complete-finite*: *finite* $V \Longrightarrow$ *finite-graph* (*complete* V)
using *complete-finite-edges* **unfolding** *finite-graph-def complete-def*
by *simp*

A graph is called 'nonempty' if it contains at least one vertex and at least one edge.

definition *nonempty-graph* $G \equiv$ *uverts* $G \neq \{\} \wedge$ *uedges* $G \neq \{\}$

A random graph is both wellformed and finite.

lemma (**in** *edge-space*) *wellformed-and-finite*:
assumes $E \in$ *Pow S-edges*
shows *finite-graph* (*edge-ugraph* E) *uwellformed* (*edge-ugraph* E)
unfolding *finite-graph-def*
proof
show *finite* (*uverts* (*edge-ugraph* E))
unfolding *edge-ugraph-def S-verts-def* **by** *simp*
next
show *finite* (*uedges* (*edge-ugraph* E))
using *assms* **unfolding** *edge-ugraph-def S-edges-def* **by** (*auto intro*: *all-edges-finite*)
next
show *uwellformed* (*edge-ugraph* E)
using *complete-wellformed* **unfolding** *edge-ugraph-def S-edges-def complete-def uwellformed-def* **by** *force*
qed

The probability for a random graph to have e edges is p^e.

lemma (**in** *edge-space*) *cylinder-empty-prob*:
$A \subseteq$ *S-edges* $\Longrightarrow$ *prob* (*cylinder S-edges* A $\{\}$) $= p$ ^ (*card* A)
using *cylinder-prob* **by** *auto*

4.1 Subgraphs

definition *subgraph* :: *ugraph* $\Rightarrow$ *ugraph* $\Rightarrow$ *bool* **where**
subgraph G' $G \equiv$ *uverts* $G' \subseteq$ *uverts* $G \wedge$ *uedges* $G' \subseteq$ *uedges* G

lemma *subgraph-refl*: *subgraph* G G

unfolding *subgraph-def*
by *simp*

lemma *subgraph-trans*: *subgraph* G'' G' $\Longrightarrow$ *subgraph* G' G $\Longrightarrow$ *subgraph* G'' G
unfolding *subgraph-def*
by *auto*

lemma *subgraph-antisym*: *subgraph* G G' $\Longrightarrow$ *subgraph* G' G $\Longrightarrow$ $G = G'$
unfolding *subgraph-def*
by (*auto simp add*: *Product-Type.prod-eqI*)

lemma *subgraph-complete*:
assumes *uwellformed* G
shows *subgraph* G (*complete* (*uverts* G))
proof –
{
fix e
assume $e \in$ *uedges* G
with *assms* **have** *card* $e = 2$ **and** u: $\bigwedge u.\ u \in e \Longrightarrow u \in$ *uverts* G
unfolding *uwellformed-def* **by** *auto*
moreover then obtain u v **where** $e = \{u, v\}$ $u \neq v$
by (*metis card-2-elements*)
ultimately have $e =$ *mk-uedge* (u, v) $u \in$ *uverts* G $v \in$ *uverts* G
by *auto*
hence $e \in$ *all-edges* (*uverts* G)
unfolding *all-edges-def* **using** ‹$u \neq v$› **by** *fastforce*
}
thus *?thesis*
unfolding *complete-def subgraph-def* **by** *auto*
qed

corollary *wellformed-all-edges*: *uwellformed* G $\Longrightarrow$ *uedges* $G \subseteq$ *all-edges* (*uverts* G)
using *subgraph-complete subgraph-def complete-def* **by** *simp*

corollary *max-edges-graph*:
assumes *uwellformed* G *finite* (*uverts* G)
shows *card* (*uedges* G) $\leq$ (*card* (*uverts* G))^*2*
proof –
have *card* (*uedges* G) $\leq$ *card* (*uverts* G) *choose 2*
by (*metis all-edges-finite assms card-all-edges card-mono wellformed-all-edges*)
thus *?thesis*
by (*metis binomial-le-pow le0 neq0-conv order.trans zero-less-binomial-iff*)
qed

lemma *subgraph-finite*: ⟦ *finite-graph* G; *subgraph* G' G ⟧ $\Longrightarrow$ *finite-graph* G'
unfolding *finite-graph-def subgraph-def*
by (*metis rev-finite-subset*)

corollary *wellformed-finite*:
  assumes *finite* (*uverts* G) **and** *uwellformed* G
  shows *finite-graph* G
proof (*rule subgraph-finite*[**where** G = *complete* (*uverts* G)])
  show *subgraph* G (*complete* (*uverts* G))
    using *assms* **by** (*simp add*: *subgraph-complete*)
next
  have *finite* (*uedges* (*complete* (*uverts* G)))
    using *complete-finite-edges*[*OF assms*(*1*)] **.**
  thus *finite-graph* (*complete* (*uverts* G))
    unfolding *finite-graph-def complete-def* **using** *assms*(*1*) **by** *auto*
qed

definition *subgraphs* :: *ugraph* ⇒ *ugraph set* **where**
subgraphs $G = \{G'.$ *subgraph* G' $G\}$

definition *nonempty-subgraphs* :: *ugraph* ⇒ *ugraph set* **where**
nonempty-subgraphs $G = \{G'.$ *uwellformed* G' ∧ *subgraph* G' G ∧ *nonempty-graph* $G'\}$

lemma *subgraphs-finite*:
  assumes *finite-graph* G
  shows *finite* (*subgraphs* G)
proof −
  have *subgraphs* $G = \{(V', E').\ V' \subseteq$ *uverts* $G \wedge E' \subseteq$ *uedges* $G\}$
    unfolding *subgraphs-def subgraph-def* **by** *force*
  moreover have *finite* (*uverts* G) *finite* (*uedges* G)
    using *assms* **unfolding** *finite-graph-def* **by** *auto*
  ultimately show *?thesis*
    by *simp*
qed

corollary *nonempty-subgraphs-finite*: *finite-graph* $G \Longrightarrow$ *finite* (*nonempty-subgraphs* G)
using *subgraphs-finite*
unfolding *nonempty-subgraphs-def subgraphs-def*
by *auto*

4.2 Induced subgraphs

definition *induced-subgraph* :: *uvert set* ⇒ *ugraph* ⇒ *ugraph* **where**
induced-subgraph V G = (V, *uedges* G ∩ *all-edges* V)

lemma *induced-is-subgraph*:
  $V \subseteq$ *uverts* $G \Longrightarrow$ *subgraph* (*induced-subgraph* V G) G
  $V \subseteq$ *uverts* $G \Longrightarrow$ *subgraph* (*induced-subgraph* V G) (*complete* V)
unfolding *subgraph-def induced-subgraph-def complete-def*
by *simp+*

lemma *induced-wellformed*: *uwellformed* $G \Longrightarrow V \subseteq$ *uverts* $G \Longrightarrow$ *uwellformed* (*induced-subgraph* V G)
unfolding *uwellformed-def induced-subgraph-def all-edges-def*
by *force*

lemma *subgraph-union-induced*:
  assumes *uverts* $H_1 \subseteq S$ **and** *uverts* $H_2 \subseteq T$
  assumes *uwellformed* H_1 **and** *uwellformed* H_2
  shows *subgraph* H_1 (*induced-subgraph* S G) $\wedge$ *subgraph* H_2 (*induced-subgraph* T G) $\longleftrightarrow$
    subgraph (*uverts* $H_1 \cup$ *uverts* H_2, *uedges* $H_1 \cup$ *uedges* H_2) (*induced-subgraph* $(S \cup T)$ G)
unfolding *induced-subgraph-def subgraph-def*
apply *auto*
using *all-edges-mono* **apply** *blast*
using *all-edges-mono* **apply** *blast*
using *assms*(*1*,*2*) *wellformed-all-edges*[*OF assms*(*3*)] *wellformed-all-edges*[*OF assms*(*4*)] *all-edges-mono*[*OF assms*(*1*)] *all-edges-mono*[*OF assms*(*2*)]
apply *auto*
done

lemma (**in** *edge-space*) *induced-subgraph-prob*:
  assumes *uverts* $H \subseteq V$ **and** *uwellformed* H **and** $V \subseteq$ *S-verts*
  shows *prob* $\{$*es* $\in$ *space* P. *subgraph* H (*induced-subgraph* V (*edge-ugraph es*))$\}$ $= p$ ^ *card* (*uedges* H) (**is** *prob ?A* = -)
proof −
  have *prob ?A* = *prob* (*cylinder S-edges* (*uedges* H) $\{\}$)
   unfolding *cylinder-def space-eq subgraph-def induced-subgraph-def edge-ugraph-def S-edges-def*
   by (*rule arg-cong*[*OF Collect-cong*]) (*metis* (*no-types*) *assms*(*1*,*2*) *Pow-iff all-edges-mono fst-conv inf-absorb1 inf-bot-left le-inf-iff snd-conv wellformed-all-edges*)
  also have ... $= p$ ^ *card* (*uedges* H)
    proof (*rule cylinder-empty-prob*)
      have *uedges* $H \subseteq$ *all-edges* (*uverts* H)
        by (*rule wellformed-all-edges*[*OF assms*(*2*)])
      also have *all-edges* (*uverts* H) $\subseteq$ *all-edges S-verts*
        using *assms* **by** (*auto simp*: *all-edges-mono*[*OF subset-trans*])
      finally show *uedges* $H \subseteq$ *S-edges*
        unfolding *S-edges-def* **.**
    qed
  finally show *?thesis*
    .
qed

4.3 Graph isomorphism

We define graph isomorphism slightly different than in the literature. The usual definition is that two graphs are isomorphic iff there exists a bijection between the vertex sets which preserves the adjacency. However, this complicates many proofs.

Instead, we define the intuitive mapping operation on graphs. An isomorphism between two graphs arises if there is a suitable mapping function from the first to the second graph. Later, we show that this operation can be inverted.

fun *map-ugraph* :: $(nat \Rightarrow nat) \Rightarrow ugraph \Rightarrow ugraph$ **where**
map-ugraph $f\ (V,\ E) = (f\ `\ V,\ (\lambda e.\ f\ `\ e)\ `\ E)$

definition *isomorphism* :: $ugraph \Rightarrow ugraph \Rightarrow (nat \Rightarrow nat) \Rightarrow bool$ **where**
isomorphism $G_1\ G_2\ f \equiv$ *bij-betw* f (*uverts* G_1) (*uverts* G_2) $\wedge\ G_2 =$ *map-ugraph* f G_1

abbreviation *isomorphic* :: $ugraph \Rightarrow ugraph \Rightarrow bool$ (‹- $\simeq$ -›) **where**
$G_1 \simeq G_2 \equiv$ *uwellformed* $G_1 \wedge$ *uwellformed* $G_2 \wedge (\exists f.$ *isomorphism* $G_1\ G_2\ f)$

lemma *map-ugraph-id*: *map-ugraph id* = *id*
unfolding *fun-eq-iff*
by *simp*

lemma *map-ugraph-trans*: *map-ugraph* $(g \circ f)$ = (*map-ugraph* g) $\circ$ (*map-ugraph* f)
  by (*simp add*: *fun-eq-iff image-image*)

lemma *map-ugraph-wellformed*:
  assumes *uwellformed* G **and** *inj-on* f (*uverts* G)
  shows *uwellformed* (*map-ugraph* f G)
unfolding *uwellformed-def*
proof *safe*
  fix e'
  assume $e' \in$ *uedges* (*map-ugraph* f G)
  hence $e' \in (\lambda e.\ f\ `\ e)\ `$ (*uedges* G)
    by (*metis map-ugraph.simps snd-conv surjective-pairing*)
  then obtain e **where** e: $e' = f\ `\ e$ $e \in$ *uedges* G
    by *blast*
  hence *card* $e = 2$ $e \subseteq$ *uverts* G
    using *assms*(*1*) **unfolding** *uwellformed-def* **by** *blast+*
  thus *card* $e' = 2$
    using e(*1*) **by** (*simp add*: *card-inj-subs*[*OF assms*(*2*)])

  fix u'
  assume $u' \in e'$
  hence $u' \in f\ `\ e$
    using e **by** *force*
  then obtain u **where** u: $u' = f\ u$ $u \in e$
    by *blast*
  hence $u \in$ *uverts* G
    using *assms*(*1*) e(*2*) **unfolding** *uwellformed-def* **by** *blast*
  hence $u' \in f\ `$ *uverts* G

using $u(1)$ **by** *simp*
thus $u' \in$ *uverts* (*map-ugraph f G*)
by (*metis map-ugraph.simps fst-conv surjective-pairing*)
qed

lemma *map-ugraph-finite*: *finite-graph G* $\Longrightarrow$ *finite-graph* (*map-ugraph f G*)
unfolding *finite-graph-def*
by (*metis finite-imageI fst-conv map-ugraph.simps snd-conv surjective-pairing*)

lemma *map-ugraph-preserves-sub*:
assumes *subgraph* G_1 G_2
shows *subgraph* (*map-ugraph f* G_1) (*map-ugraph f* G_2)
proof −
have f ' *uverts* $G_1 \subseteq f$ ' *uverts* G_2 ($\lambda e.\ f$ ' e) ' *uedges* $G_1 \subseteq$ ($\lambda e.\ f$ ' e) ' *uedges* G_2
using *assms*(*1*) **unfolding** *subgraph-def* **by** *auto*
thus *?thesis*
unfolding *subgraph-def* **by** (*metis map-ugraph.simps fst-conv snd-conv surjective-pairing*)
qed

lemma *isomorphic-refl*: *uwellformed G* $\Longrightarrow$ $G \simeq G$
unfolding *isomorphism-def*
by (*metis bij-betw-id id-def map-ugraph-id*)

lemma *isomorphic-trans*:
assumes $G_1 \simeq G_2$ **and** $G_2 \simeq G_3$
shows $G_1 \simeq G_3$
proof −
from *assms* **obtain** f_1 f_2 **where**
bij: *bij-betw* f_1 (*uverts* G_1) (*uverts* G_2) *bij-betw* f_2 (*uverts* G_2) (*uverts* G_3) **and**
map: G_2 = *map-ugraph* f_1 G_1 G_3 = *map-ugraph* f_2 G_2
unfolding *isomorphism-def* **by** *blast*

let *?f* = $f_2 \circ f_1$
have *bij-betw ?f* (*uverts* G_1) (*uverts* G_3)
using *bij* **by** (*simp add*: *bij-betw-comp-iff*)
moreover have G_3 = *map-ugraph ?f* G_1
using *map* **by** (*simp add*: *map-ugraph-trans*)
moreover have *uwellformed* G_1 *uwellformed* G_3
using *assms* **unfolding** *isomorphism-def* **by** *simp+*
ultimately show $G_1 \simeq G_3$
unfolding *isomorphism-def* **by** *blast*
qed

lemma *isomorphic-sym*:
assumes $G_1 \simeq G_2$
shows $G_2 \simeq G_1$
proof *safe*

from *assms* **obtain** f **where** *isomorphism* G_1 G_2 f
  by *blast*
  hence *bij*: *bij-betw* f (*uverts* G_1) (*uverts* G_2) **and** *map*: G_2 = *map-ugraph* f G_1
  unfolding *isomorphism-def* **by** *auto*

  let *?f'* = *inv-into* (*uverts* G_1) f
  have *bij'*: *bij-betw* *?f'* (*uverts* G_2) (*uverts* G_1)
  by (*rule* *bij-betw-inv-into*) *fact*
  moreover have *uverts* G_1 = *?f'* ‘ *uverts* G_2
  using *bij'* **unfolding** *bij-betw-def* **by** *force*
  moreover have *uedges* G_1 = (λe. *?f'* ‘ e) ‘ *uedges* G_2
  proof −
  have *uedges* G_1 = *id* ‘ *uedges* G_1
  by *simp*
  also have ... = (λe. *?f'* ‘ (f ‘ e)) ‘ *uedges* G_1
  proof (*rule* *image-cong*)
  fix a
  assume $a \in$ *uedges* G_1
  hence $a \subseteq$ *uverts* G_1
  using *assms* **unfolding** *isomorphism-def* *uwellformed-def* **by** *blast*
  thus *id* a = *inv-into* (*uverts* G_1) f ‘ f ‘ a
  by (*metis* (*full-types*) *id-def* *bij* *bij-betw-imp-inj-on* *inv-into-image-cancel*)
  qed *simp*
  also have ... = (λe. *?f'* ‘ e) ‘ ((λe. f ‘ e) ‘ *uedges* G_1)
  by (*rule* *image-image*[*symmetric*])
  also have ... = (λe. *?f'* ‘ e) ‘ *uedges* G_2
  using *bij* *map* **by** (*metis* *map-ugraph.simps* *prod.collapse* *snd-eqD*)
  finally show *?thesis*
  .
  qed
  ultimately have *isomorphism* G_2 G_1 *?f'*
  unfolding *isomorphism-def* **by** (*metis* *map-ugraph.simps* *split-pairs*)
  thus $\exists f$. *isomorphism* G_2 G_1 f
  by *blast*
qed (*auto* *simp*: *assms*)

lemma *isomorphic-cards*:
  assumes $G_1 \simeq G_2$
  shows
  card (*uverts* G_1) = *card* (*uverts* G_2) (**is** *?V*)
  card (*uedges* G_1) = *card* (*uedges* G_2) (**is** *?E*)
proof −
  from *assms* **obtain** f **where**
  bij: *bij-betw* f (*uverts* G_1) (*uverts* G_2) **and**
  map: G_2 = *map-ugraph* f G_1
  unfolding *isomorphism-def* **by** *blast*
  from *assms* **have** *wellformed*: *uwellformed* G_1 *uwellformed* G_2
  by *simp+*

show *?V*
by (*rule bij-betw-same-card*[*OF bij*])

let *?g* $= \lambda e.\ f$ ‘ *e*
have *bij-betw ?g* (*Pow* (*uverts* G_1)) (*Pow* (*uverts* G_2))
by (*rule bij-lift*[*OF bij*])
moreover have *uedges* $G_1 \subseteq$ *Pow* (*uverts* G_1)
using *wellformed*(*1*) **unfolding** *uwellformed-def* **by** *blast*
ultimately have *card* (*?g* ‘ *uedges* G_1) = *card* (*uedges* G_1)
unfolding *bij-betw-def* **by** (*metis card-inj-subs*)
thus *?E*
by (*metis map map-ugraph.simps snd-conv surjective-pairing*)
qed

4.4 Isomorphic subgraphs

The somewhat sloppy term ‘isomorphic subgraph’ denotes a subgraph which is isomorphic to a fixed other graph. For example, saying that a graph contains a triangle usually means that it contains *any* triangle, not the specific triangle with the nodes 1, 2 and 3. Hence, such a graph would have a triangle as an isomorphic subgraph.

definition *subgraph-isomorphic* :: *ugraph* $\Rightarrow$ *ugraph* $\Rightarrow$ *bool* (‹- $\sqsubseteq$ -›) **where**
$G' \sqsubseteq G \equiv$ *uwellformed* $G \wedge (\exists\, G''.\ G' \simeq G'' \wedge$ *subgraph* $G''\ G)$

lemma *subgraph-is-subgraph-isomorphic*: $\llbracket$ *uwellformed* G'; *uwellformed* G; *subgraph* $G'\ G$ $\rrbracket \Longrightarrow G' \sqsubseteq G$
unfolding *subgraph-isomorphic-def*
by (*metis isomorphic-refl*)

lemma *isomorphic-is-subgraph-isomorphic*: $G_1 \simeq G_2 \Longrightarrow G_1 \sqsubseteq G_2$
unfolding *subgraph-isomorphic-def*
by (*metis subgraph-refl*)

lemma *subgraph-isomorphic-refl*: *uwellformed* $G \Longrightarrow G \sqsubseteq G$
unfolding *subgraph-isomorphic-def*
by (*metis isomorphic-refl subgraph-refl*)

lemma *subgraph-isomorphic-pre-iso-closed*:
assumes $G_1 \simeq G_2$ **and** $G_2 \sqsubseteq G_3$
shows $G_1 \sqsubseteq G_3$
unfolding *subgraph-isomorphic-def*
proof
show *uwellformed* G_3
using *assms* **unfolding** *subgraph-isomorphic-def* **by** *blast*
next
from *assms*(*2*) **obtain** $G_2{}'$ **where** $G_2 \simeq G_2{}'$ *subgraph* $G_2{}'\ G_3$
unfolding *subgraph-isomorphic-def* **by** *blast*
moreover with *assms*(*1*) **have** $G_1 \simeq G_2{}'$

by (*metis isomorphic-trans*)
ultimately show $\exists\, G''.\ G_1 \simeq G'' \wedge$ *subgraph* $G''\ G_3$
by *blast*
qed

lemma *subgraph-isomorphic-pre-subgraph-closed*:
assumes *uwellformed* G_1 **and** *subgraph* $G_1\ G_2$ **and** $G_2 \sqsubseteq G_3$
shows $G_1 \sqsubseteq G_3$
unfolding *subgraph-isomorphic-def*
proof
show *uwellformed* G_3
using *assms* **unfolding** *subgraph-isomorphic-def* **by** *blast*
next
from *assms*(*3*) **obtain** G_2' **where** $G_2 \simeq G_2'$ *subgraph* $G_2'\ G_3$
unfolding *subgraph-isomorphic-def* **by** *blast*
then obtain *f* **where** *bij*: *bij-betw f* (*uverts* G_2) (*uverts* G_2') $G_2' =$ *map-ugraph f* G_2
unfolding *isomorphism-def* **by** *blast*
let *?*$G_1' =$ *map-ugraph f* G_1

have *bij-betw f* (*uverts* G_1) (*f* ‘ *uverts* G_1)
using *bij*(*1*) *assms*(*2*) **unfolding** *subgraph-def* **by** (*auto intro*: *bij-betw-subset*)
moreover hence *uwellformed ?*G_1'
using *map-ugraph-wellformed*[*OF assms*(*1*)] **unfolding** *bij-betw-def* **..**
ultimately have $G_1 \simeq$ *?*G_1'
using *assms*(*1*) **unfolding** *isomorphism-def* **by** (*metis map-ugraph.simps fst-conv surjective-pairing*)
moreover have *subgraph ?*$G_1'\ G_3$
using *subgraph-trans*[*OF map-ugraph-preserves-sub*[*OF assms*(*2*)]] *bij*(*2*) ‹*subgraph* $G_2'\ G_3$› **by** *simp*
ultimately show $\exists\, G''.\ G_1 \simeq G'' \wedge$ *subgraph* $G''\ G_3$
by *blast*
qed

lemmas *subgraph-isomorphic-pre-closed* = *subgraph-isomorphic-pre-subgraph-closed subgraph-isomorphic-pre-iso-closed*

lemma *subgraph-isomorphic-trans*[*trans*]:
assumes $G_1 \sqsubseteq G_2$ **and** $G_2 \sqsubseteq G_3$
shows $G_1 \sqsubseteq G_3$
proof −
from *assms*(*1*) **obtain** G **where** $G_1 \simeq G$ *subgraph* $G\ G_2$
unfolding *subgraph-isomorphic-def* **by** *blast*
thus *?thesis*
using *assms*(*2*) **by** (*metis subgraph-isomorphic-pre-closed*)
qed

lemma *subgraph-isomorphic-post-iso-closed*: $[\![\ H \sqsubseteq G;\ G \simeq G'\ ]\!] \Longrightarrow H \sqsubseteq G'$
using *isomorphic-is-subgraph-isomorphic subgraph-isomorphic-trans*

by *blast*

lemmas *subgraph-isomorphic-post-closed* = *subgraph-isomorphic-post-iso-closed*

lemmas *subgraph-isomorphic-closed* = *subgraph-isomorphic-pre-closed subgraph-isomorphic-post-closed*

4.5 Density

The density of a graph is the quotient of the number of edges and the number of vertices of a graph.

definition *density* :: *ugraph* ⇒ *real* **where**
density G = *card* (*uedges G*) / *card* (*uverts G*)

The maximum density of a graph is the density of its densest nonempty subgraph.

definition *max-density* :: *ugraph* ⇒ *real* **where**
max-density G = *Lattices-Big.Max* (*density* ‘ *nonempty-subgraphs G*)

We prove some obvious results about the maximum density, such as that there is a subgraph which has the maximum density and that the (maximum) density is preserved by isomorphisms. The proofs are a bit complicated by the fact that most facts about *Max* require non-emptiness of the target set, but we need that anyway to get a value out of it.

lemma *subgraph-has-max-density*:
  assumes *finite-graph G* **and** *nonempty-graph G* **and** *uwellformed G*
  shows ∃ *G'*. *density G'* = *max-density G* ∧ *subgraph G' G* ∧ *nonempty-graph G'* ∧ *finite-graph G'* ∧ *uwellformed G'*
proof −
  have *G* ∈ *nonempty-subgraphs G*
    unfolding *nonempty-subgraphs-def* **using** *subgraph-refl assms* **by** *simp*
  hence *density G* ∈ *density* ‘ *nonempty-subgraphs G*
    by *simp*
  hence (*density* ‘ *nonempty-subgraphs G*) ≠ {}
    by *fast*
  hence *max-density G* ∈ (*density* ‘ *nonempty-subgraphs G*)
    unfolding *max-density-def* **by** (*auto simp add*: *nonempty-subgraphs-finite*[*OF assms*(*1*)] *Max.closed*)
  thus *?thesis*
    unfolding *nonempty-subgraphs-def* **using** *subgraph-finite*[*OF assms*(*1*)] **by** *force*
qed

lemma *max-density-is-max*:
  assumes *finite-graph G* **and** *finite-graph G'* **and** *nonempty-graph G'* **and** *uwellformed G'* **and** *subgraph G' G*
  shows *density G'* ≤ *max-density G*
unfolding *max-density-def*

proof (*rule Max-ge*)
  show *finite* (*density* ‘ *nonempty-subgraphs* *G*)
    using *assms*(*1*) **by** (*simp add*: *nonempty-subgraphs-finite*)
next
  show *density* $G' \in$ *density* ‘ *nonempty-subgraphs* *G*
    unfolding *nonempty-subgraphs-def* **using** *assms* **by** *blast*
qed

lemma *max-density-gr-zero*:
  assumes *finite-graph* *G* **and** *nonempty-graph* *G* **and** *uwellformed* *G*
  shows *0* < *max-density* *G*
proof −
  have *0* < *card* (*uverts* *G*) *0* < *card* (*uedges* *G*)
    using *assms* **unfolding** *finite-graph-def* *nonempty-graph-def* **by** *auto*
  hence *0* < *density* *G*
    unfolding *density-def* **by** *simp*
  also have *density* *G* ≤ *max-density* *G*
    using *assms* **by** (*simp add*: *max-density-is-max* *subgraph-refl*)
  finally show *?thesis*
    .
qed

lemma *isomorphic-density*:
  assumes $G_1 \simeq G_2$
  shows *density* G_1 = *density* G_2
unfolding *density-def*
using *isomorphic-cards*[*OF assms*]
by *simp*

lemma *isomorphic-max-density*:
  assumes $G_1 \simeq G_2$ **and** *nonempty-graph* G_1 **and** *nonempty-graph* G_2 **and** *finite-graph* G_1 **and** *finite-graph* G_2
  shows *max-density* G_1 = *max-density* G_2
proof −
  — The proof strategy is not completely straightforward. We first show that if two graphs are isomorphic, the maximum density of one graph is less or equal than the maximum density of the other graph. The reason is that this proof is quite long and the desired result directly follows from the symmetry of the isomorphism relation.[1]
  {
    fix *A B*
    assume *A*: *nonempty-graph* *A* *finite-graph* *A*
    assume *iso*: $A \simeq B$

    then obtain *f* **where** *f*: *B* = *map-ugraph f A* *bij-betw f* (*uverts* *A*) (*uverts* *B*)
      unfolding *isomorphism-def* **by** *blast*

[1]Some famous mathematician once said that if you prove that $a \leq b$ and $b \leq a$, you know *that* these numbers are equal, but not *why*. Since many proofs in this work are mostly opaque to me, I can live with that.

have *wellformed*: *uwellformed A*
using *iso* **unfolding** *isomorphism-def* **by** *simp*
— We observe that the set of densities of the subgraphs does not change if we map the subgraphs first.
have *density* ‘ *nonempty-subgraphs A* = *density* ‘ (*map-ugraph f* ‘ *nonempty-subgraphs A*)
proof (*rule image-comp-cong*)
fix *G*
assume $G \in$ *nonempty-subgraphs A*
hence *uverts G* $\subseteq$ *uverts A uwellformed G*
unfolding *nonempty-subgraphs-def subgraph-def* **by** *simp*+
hence *inj-on f* (*uverts G*)
using *f*(*2*) **unfolding** *bij-betw-def* **by** (*metis subset-inj-on*)
hence $G \simeq$ *map-ugraph f G*
unfolding *isomorphism-def bij-betw-def*
by (*metis map-ugraph.simps fst-conv surjective-pairing map-ugraph-wellformed* ‹*uwellformed G*›)
thus *density G* = *density* (*map-ugraph f G*)
by (*fact isomorphic-density*)
qed
— Additionally, we show that the operations *nonempty-subgraphs* and *map-ugraph* can be swapped without changing the densities. This is an obvious result, because *map-ugraph* does not change the structure of a graph. Still, the proof is a bit hairy, which is why we only show inclusion in one direction and use symmetry of isomorphism later.
also have ... $\subseteq$ *density* ‘ *nonempty-subgraphs* (*map-ugraph f A*)
proof (*rule image-mono*, *rule subsetI*)
fix G''
assume $G'' \in$ *map-ugraph f* ‘ *nonempty-subgraphs A*
then obtain G' **where** *G-subst*: G'' = *map-ugraph f* G' $G' \in$ *nonempty-subgraphs A*
by *blast*
hence G': *subgraph* G' *A nonempty-graph* G' *uwellformed* G'
unfolding *nonempty-subgraphs-def* **by** *auto*
hence *inj-on f* (*uverts* G')
using *f* **unfolding** *bij-betw-def subgraph-def* **by** (*metis subset-inj-on*)
hence *uwellformed* G''
using *map-ugraph-wellformed* G' *G-subst* **by** *simp*
moreover have *nonempty-graph* G''
using G' *G-subst* **unfolding** *nonempty-graph-def* **by** (*metis map-ugraph.simps fst-conv snd-conv surjective-pairing empty-is-image*)
moreover have *subgraph* G'' (*map-ugraph f A*)
using *map-ugraph-preserves-sub* G' *G-subst* **by** *simp*
ultimately show $G'' \in$ *nonempty-subgraphs* (*map-ugraph f A*)
unfolding *nonempty-subgraphs-def* **by** *simp*
qed
finally have *density* ‘ *nonempty-subgraphs A* $\subseteq$ *density* ‘ *nonempty-subgraphs* (*map-ugraph f A*)
.

hence *max-density* $A \leq$ *max-density* (*map-ugraph f A*)
unfolding *max-density-def*
proof (*rule Max-mono*)
have $A \in$ *nonempty-subgraphs A*
using *A iso* **unfolding** *nonempty-subgraphs-def* **by** (*simp add*: *subgraph-refl*)
thus *density* ‘ *nonempty-subgraphs A* $\neq \{\}$
by *blast*
next
have *finite* (*nonempty-subgraphs* (*map-ugraph f A*))
by (*rule nonempty-subgraphs-finite*[*OF map-ugraph-finite*[*OF A*(*2*)]])
thus *finite* (*density* ‘ *nonempty-subgraphs* (*map-ugraph f A*))
by *blast*
qed
hence *max-density* $A \leq$ *max-density B*
by (*subst f*)
}
then show *?thesis*
by (*meson assms isomorphic-sym order-antisym-conv*)
qed

4.6 Fixed selectors

In the proof of the main theorem in the lecture notes, the concept of a “fixed copy” of a graph is fundamental.

Let H be a fixed graph. A ‘fixed selector’ is basically a function mapping a set with the same size as the vertex set of H to a new graph which is isomorphic to H and its vertex set is the same as the input set.[2]

definition *is-fixed-selector H f* = $(\forall V.$ *finite* $V \wedge$ *card* (*uverts H*) = *card* $V \longrightarrow H \simeq f\ V \wedge$ *uverts* $(f\ V) = V)$

Obviously, there may be many possible fixed selectors for a given graph. First, we show that there is always at least one. This is sufficient, because we can always obtain that one and use its properties without knowing exactly which one we chose.

lemma *ex-fixed-selector*:
assumes *uwellformed H* **and** *finite-graph H*
obtains *f* **where** *is-fixed-selector H f*
proof
— I guess this is the only place in the whole work where we make use of a nifty little HOL feature called *SOME*, which is basically Hilbert’s choice operator. The reason is that any bijection between the the vertex set of H and the input set gives rise to a fixed selector function. In the lecture notes, a specific bijection was defined, but this is shorter and more elegant.
let *?bij* = $\lambda V.$ *SOME g. bij-betw g* (*uverts H*) *V*
let *?f* = $\lambda V.$ *map-ugraph* (*?bij V*) *H*
{

[2] We call such a selector *fixed* because its result is deterministic.

fix V :: *uvert set*
assume *finite V card* (*uverts H*) = *card V*
moreover have *finite* (*uverts H*)
using *assms* **unfolding** *finite-graph-def* **by** *simp*
ultimately have *bij-betw* (*?bij V*) (*uverts H*) *V*
by (*metis finite-same-card-bij someI-ex*)
moreover hence ∗: *uverts* (*?f V*) = *V* ∧ *uwellformed* (*?f V*)
using *map-ugraph-wellformed*[*OF assms*(*1*)]
by (*metis bij-betw-def map-ugraph.simps fst-conv surjective-pairing*)
ultimately have ∗∗: *H* ≃ *?f V*
unfolding *isomorphism-def* **using** *assms*(*1*) **by** *auto*
note ∗ ∗∗
}
thus *is-fixed-selector H ?f*
unfolding *is-fixed-selector-def* **by** *blast*
qed

lemma *fixed-selector-induced-subgraph*:
assumes *is-fixed-selector H f* **and** *card* (*uverts H*) = *card V* **and** *finite V*
assumes *sub*: *subgraph* (*f V*) (*induced-subgraph V G*) **and** *V*: *V* ⊆ *uverts G* **and** *G*: *uwellformed G*
shows *H* ⊑ *G*
by (*meson G V assms induced-is-subgraph*(*1*) *is-fixed-selector-def sub subgraph-isomorphic-def subgraph-trans*)

end

5 Classes and properties of graphs

theory *Ugraph-Properties*
imports
Ugraph-Lemmas
Girth-Chromatic.Girth-Chromatic
begin

A "graph property" is a set of graphs which is closed under isomorphism.

type-synonym *ugraph-class* = *ugraph set*

definition *ugraph-property* :: *ugraph-class* ⇒ *bool* **where**
ugraph-property C ≡ ∀ *G* ∈ *C*. ∀ *G'*. *G* ≃ *G'* ⟶ *G'* ∈ *C*

abbreviation *prob-in-class* :: (*nat* ⇒ *real*) ⇒ *ugraph-class* ⇒ *nat* ⇒ *real* **where**
prob-in-class p c n ≡ *probGn p n* (λ*es. edge-space.edge-ugraph n es* ∈ *c*)

From now on, we consider random graphs not with fixed edge probabilities but rather with a probability function depending on the number of vertices. Such a function is called a "threshold" for a graph property iff

- for asymptotically *larger* probability functions, the probability that a random graph is an element of that class tends to 1 (“1-statement”), and

- for asymptotically *smaller* probability functions, the probability that a random graph is an element of that class tends to 0 (“0-statement”).

definition *is-threshold* :: *ugraph-class* ⇒ (*nat* ⇒ *real*) ⇒ *bool* **where**
is-threshold c t ≡ *ugraph-property c* ∧ (∀ *p*. *nonzero-prob-fun p* ⟶
  (*p* ≪ *t* ⟶ *prob-in-class p c* ⟶ *0*) ∧
  (*t* ≪ *p* ⟶ *prob-in-class p c* ⟶ *1*))

lemma *is-thresholdI*[*intro*]:
  assumes *ugraph-property c*
  assumes ⋀*p*. ⟦ *nonzero-prob-fun p*; *p* ≪ *t* ⟧ ⟹ *prob-in-class p c* ⟶ *0*
  assumes ⋀*p*. ⟦ *nonzero-prob-fun p*; *t* ≪ *p* ⟧ ⟹ *prob-in-class p c* ⟶ *1*
  shows *is-threshold c t*
using *assms* **unfolding** *is-threshold-def* **by** *blast*

end

6 The subgraph threshold theorem

theory *Subgraph-Threshold*
imports
  Ugraph-Properties
begin

lemma (**in** *edge-space*) *measurable-pred*[*measurable*]: *Measurable.pred P Q*
  by (*simp add*: *P-def sets-point-measure space-point-measure subset-eq*)

This section contains the main theorem. For a fixed nonempty graph H, we consider the graph property of ‘containing an isomorphic subgraph of H’. This is obviously a valid property, since it is closed under isomorphism.

The corresponding threshold function is

$$t(n) = n^{-\frac{1}{\rho'(H)}},$$

where ρ' denotes *max-density*.

definition *subgraph-threshold* :: *ugraph* ⇒ *nat* ⇒ *real* **where**
subgraph-threshold H n = *n powr* (−(*1* / *max-density H*))

theorem
  assumes *nonempty*: *nonempty-graph H* **and** *finite*: *finite-graph H* **and** *well-formed*: *uwellformed H*
  shows *is-threshold* {*G*. *H* ⊑ *G*} (*subgraph-threshold H*)
proof
  show *ugraph-property* {*G*. *H* ⊑ *G*}

unfolding *ugraph-property-def* **using** *subgraph-isomorphic-closed* **by** *blast*
next

— To prove the 0-statement, we introduce the subgraph with the maximum density as H_0. Note that $\rho(H_0) = \rho'(H)$.

fix p :: *nat* $\Rightarrow$ *real*

obtain H_0 **where** H_0: *density* H_0 = *max-density* H *subgraph* H_0 H *nonempty-graph* H_0 *finite-graph* H_0 *uwellformed* H_0
using *subgraph-has-max-density assms* **by** *blast*
hence *card*: *0* < *card* (*uverts* H_0) *0* < *card* (*uedges* H_0)
unfolding *nonempty-graph-def finite-graph-def* **by** *auto*

let *?v* = *card* (*uverts* H_0)
let *?e* = *card* (*uedges* H_0)

assume *p-nz*: *nonzero-prob-fun p*
hence *p*: *valid-prob-fun p*
by (*fact nonzero-fun-is-valid-fun*)

— Firstly, we follow from the assumption that p is asympotically less than the threshold function that the product

$$p(n)^{|E(H_0)|} \cdot n^{|V(H_0)|}$$

tends to 0.

assume $p \ll$ *subgraph-threshold H*
moreover
{
fix n
have *p n* / *n powr* (−(*1* / *max-density H*)) = *p n* ∗ *n powr* (*1* / *max-density H*)
by (*simp add*: *powr-minus-divide*)
also have . . . = *p n* ∗ *n powr* (*1* / *density* H_0)
using H_0 **by** *simp*
also have . . . = *p n* ∗ *n powr* (*?v* / *?e*)
using *card* **unfolding** *density-def* **by** *simp*
finally have *p n* / *n powr* (−(*1* / *max-density H*)) = . . .
.
}
ultimately have (λ*n*. *p n* ∗ *n powr* (*?v* / *?e*)) ⟶ *0*
unfolding *subgraph-threshold-def* **by** *simp*
moreover have ⋀*n*. *1* ≤ *n* ⟹ *0* < *p n* ∗ *n powr* (*?v* / *?e*)
by (*auto simp*: *p-nz*)
ultimately have (λ*n*. (*p n* ∗ *n powr* (*?v* / *?e*)) *powr ?e*) ⟶ *0*
using *card*(*2*) *p* **by** (*force intro*: *tendsto-zero-powrI*)
hence *limit*: (λ*n*. *p n powr ?e* ∗ *n powr ?v*) ⟶ *0*
by (*rule LIMSEQ-cong*[*OF* - *eventually-sequentiallyI*[**where** *c* = *1*]])

(*auto simp*: *p card p-nz powr-powr powr-mult*)

{
fix n
assume n: *?v* $\leq n$

interpret *ES*: *edge-space n* (*p n*)
by *unfold-locales* (*auto simp*: *p*)

let *?graph-of* = *ES.edge-ugraph*

— After fixing an n, we define a family of random variables X indexed by a set of vertices v and a set of edges e. Each X is an indicator for the event that (v, e) is isomorphic to H_0 and a subgraph of a random graph. The sum of all these variables is denoted by Y and counts the total number of copies of H_0 in a random graph.

let $?X = \lambda H_0{}'.$ *rind* $\{es \in$ *space ES.P. subgraph* $H_0{}'$ (*?graph-of es*) $\wedge H_0 \simeq H_0{}'\}$
let $?I = \{(v,\ e).\ v \subseteq \{1..n\} \wedge$ *card* $v = ?v \wedge e \subseteq$ *all-edges* $v \wedge$ *card* $e = ?e\}$
let $?Y = \lambda es.\ \sum H_0{}' \in\ ?I.\ ?X\ H_0{}'\ es$

— Now we prove an upper bound for the probability that a random graph contains a copy of H. Observe that in that case, Y takes a value greater or equal than 1.

have *prob-in-class p* $\{G.\ H \sqsubseteq G\}$ n = *probGn p n* $(\lambda es.\ H \sqsubseteq$ *?graph-of es*)
by *simp*
also have ... $\leq$ *probGn p n* $(\lambda es.\ 1 \leq\ ?Y\ es)$
proof (*rule ES.finite-measure-mono*, *safe*)
fix *es*
assume *es*: $es \in$ *space* (*MGn p n*)

assume $H \sqsubseteq$ *?graph-of es*
hence $H_0 \sqsubseteq$ *?graph-of es* — since H_0 is a subgraph of H
using H_0 **by** (*fast intro*: *subgraph-isomorphic-pre-subgraph-closed*)
then obtain $H_0{}'$ **where** $H_0{}'$: *subgraph* $H_0{}'$ (*?graph-of es*) $H_0 \simeq H_0{}'$
unfolding *subgraph-isomorphic-def*
by *blast*

show $1 \leq\ ?Y\ es$
proof (*rule sum-lower-or-eq*)
— The only relevant step here is to provide the specific instance of (v, e) such that $X_{(v,e)}$ takes a value greater or equal than 1. This is trivial, as we already obtained that one above (i.e. $H_0{}'$). The remainder of the proof is just bookkeeping.
show $1 \leq\ ?X\ H_0{}'\ es$ — by definition of X
using $H_0{}'$ *es* **by** *simp*
next
have *uverts* $H_0{}' \subseteq \{1..n\}$ *uedges* $H_0{}' \subseteq es$

using $H_0{}'(1)$ **unfolding** *subgraph-def ES.edge-ugraph-def ES.S-verts-def ES.S-edges-def* **by** *simp+*
moreover have *card* (*uverts* $H_0{}'$) = *?v* *card* (*uedges* $H_0{}'$) = *?e*
by (*simp add*: *isomorphic-cards*[*OF* ‹$H_0 \simeq H_0{}'$›])+
moreover have *uedges* $H_0{}' \subseteq$ *all-edges* (*uverts* $H_0{}'$)
using $H_0{}'$ **by** (*simp add*: *wellformed-all-edges*)
ultimately show $H_0{}' \in$ *?I*
by *auto*
next
have *?I* $\subseteq$ *subgraphs* (*complete* {*1..n*})
unfolding *complete-def subgraphs-def subgraph-def* **using** *all-edges-mono* **by** *auto blast*
moreover have *finite* (*subgraphs* (*complete* {*1..n*}))
by (*simp add*: *complete-finite subgraphs-finite*)
ultimately show *finite ?I*
by (*fact finite-subset*)
qed *simp*
qed *simp*

— Applying Markov's inequality leaves us with estimating the expectation of Y, which is the sum of the individual X.
also have ... $\leq$ *ES.expectation ?Y* / *1*
by (*rule prob-space.markov-inequality*) (*auto simp*: *ES.prob-space-P sum-nonneg*)
also have ... = *ES.expectation ?Y*
by *simp*
also have ... = ($\sum$ $H_0{}' \in$ *?I*. *ES.expectation* (*?X* $H_0{}'$))
by (*rule Bochner-Integration.integral-sum*(*1*)) *simp*

— Each expectation is bound by $p(n)^{|E(H_0)|}$. For the proof, we ignore the fact that the corresponding graph has to be isomorphic to H_0, which only increases the probability and thus the expectation. This only leaves us to compute the probability that all edges are present, which is given by *edge-space.cylinder-prob*.
also have ... $\leq$ ($\sum$ $H_0{}' \in$ *?I*. *p n* ^ *?e*)
proof (*rule sum-mono*)
fix $H_0{}'$
assume $H_0{}'$: $H_0{}' \in$ *?I*
have *ES.expectation* (*?X* $H_0{}'$) = *ES.prob* {*es* $\in$ *space ES.P*. *subgraph* $H_0{}'$ (*?graph-of es*) $\wedge$ $H_0 \simeq H_0{}'$}
by (*rule ES.expectation-indicator*) (*auto simp*: *ES.sets-eq ES.space-eq*)
also have ... $\leq$ *ES.prob* {*es* $\in$ *space ES.P*. *uedges* $H_0{}' \subseteq$ *es*}
unfolding *subgraph-def* **by** (*rule ES.finite-measure-mono*) (*auto simp*: *ES.sets-eq ES.space-eq*)
also have ... = *ES.prob* (*cylinder ES.S-edges* (*uedges* $H_0{}'$) {})
unfolding *cylinder-def ES.space-eq* **by** *simp*
also have ... = *p n* ^ *card* (*uedges* $H_0{}'$)
proof (*rule ES.cylinder-empty-prob*)
have *uverts* $H_0{}' \subseteq$ {*1..n*} *uedges* $H_0{}' \subseteq$ *all-edges* (*uverts* $H_0{}'$)
using $H_0{}'$ **by** *auto*
hence *uedges* $H_0{}' \subseteq$ *all-edges* {*1..n*}

using *all-edges-mono* **by** *blast*
thus *uedges* $H_0{}'$ $\subseteq$ *ES.S-edges*
unfolding *ES.S-edges-def ES.S-verts-def* **by** *simp*
qed
also have ... = *p n* ^ *?e*
using $H_0{}'$ **by** *fastforce*
finally show *ES.expectation* (*?X* $H_0{}'$) $\leq$ ...
.
qed

— Since we have a sum of constant summands, we can rewrite it as a product.
also have ... = *card ?I* ∗ *p n* ^ *?e*
by (*rule sum-constant*)

— We have to count the number of possible pairs (v, e). From the definition of the index set, note that we first choose $|V(H_0)|$ elements out of a set of n vertices and then $|E(H_0)|$ elements out of all possible edges over these vertices.
also have ... = ((*n choose ?v*) ∗ ((*?v choose 2*) *choose ?e*)) ∗ *p n* ^ *?e*
proof (*rule arg-cong*[**where** *x* = *card ?I*])
have *card ?I* = ($\sum$ *v* | *v* $\subseteq$ {*1..n*} ∧ *card v* = *?v*. *card* (*all-edges v*) *choose ?e*)
by (*rule card-dep-pair-set*[**where** *A* = {*1..n*} **and** *n* = *?v* **and** *f* = *all-edges*])
(*auto simp*: *finite-subset all-edges-finite*)
also have ... = ($\sum$ *v* | *v* $\subseteq$ {*1..n*} ∧ *card v* = *?v*. (*?v choose 2*) *choose ?e*)
proof (*rule sum.cong*)
fix *v*
assume *v* ∈ {*v*. *v* $\subseteq$ {*1..n*} ∧ *card v* = *?v*}
hence *v* $\subseteq$ {*1..n*} *card v* = *?v*
by *auto*
thus *card* (*all-edges v*) *choose ?e* = (*?v choose 2*) *choose ?e*
by (*simp add*: *card-all-edges finite-subset*)
qed *rule*
also have ... = *card* ({*v*. *v* $\subseteq$ {*1..n*} ∧ *card v* = *?v*}) ∗ ((*?v choose 2*) *choose ?e*)
by *simp*
also have ... = (*n choose ?v*) ∗ ((*?v choose 2*) *choose ?e*)
by (*simp add*: *n-subsets*)
finally show *card ?I* = ...
.
qed
also have ... = (*n choose ?v*) ∗ (((*?v choose 2*) *choose ?e*) ∗ *p n* ^ *?e*)
by *simp*

— Here, we use n^k as an upper bound for $\binom{n}{k}$.
also have ... $\leq$ (*n* ^ *?v*) ∗ (((*?v choose 2*) *choose ?e*) ∗ *p n* ^ *?e*) (**is** - $\leq$ - ∗ *?r*)
proof (*rule mult-right-mono*)
have *n choose ?v* $\leq$ *n* ^ *?v*

by (*rule binomial-le-pow*) (*rule n*)
thus *real* (*n choose ?v*) $\leq$ *real* (*n* ^ *?v*)
by (*metis of-nat-le-iff*)
next
show *0* $\leq$ *?r* **using** *p* **by** *simp*
qed
also have ... $\leq$ ((*?v choose 2*) *choose ?e*) $*$ (*p n* ^ *?e* $*$ *n* ^ *?v*) (**is** - $\leq$ *?factor* $*$ -)
by *simp*
also have ... = *?factor* $*$ (*p n powr ?e* $*$ *n powr ?v*)
using *n card*(*1*) ‹*nonzero-prob-fun p*› **by** (*simp add*: *powr-realpow*)

finally have *prob-in-class p* {*G*. *H* $\sqsubseteq$ *G*} *n* $\leq$ *?factor* $*$ (*p n powr ?e* $*$ *n powr ?v*)
.
}

— The final upper bound is a multiple of the expression which we have proven to tend to 0 in the beginning.
thus *prob-in-class p* {*G*. *H* $\sqsubseteq$ *G*} $\longrightarrow$ *0*
by (*rule LIMSEQ-le-zero*[*OF tendsto-mult-right-zero*[*OF limit*] *eventually-sequentiallyI*[*OF measure-nonneg*] *eventually-sequentiallyI*])
next
fix *p* :: *nat* $\Rightarrow$ *real*
assume *p-threshold*: *subgraph-threshold H* $\ll$ *p*

— To prove the 1-statement, we obtain a fixed selector *f* as defined in section 4.6.
from *assms* **obtain** *f* **where** *f*: *is-fixed-selector H f*
using *ex-fixed-selector* **by** *blast*

let *?v* = *card* (*uverts H*)
let *?e* = *card* (*uedges H*)

— We observe that several terms involving $|V(H)|$ are positive.
have *v-e-nz*: *0* < *real ?v 0* < *real ?e*
using *nonempty finite* **unfolding** *nonempty-graph-def finite-graph-def* **by** *auto*
hence *0* < *real ?v* ^ *?v* **by** *simp*
hence *vpowv-inv-gr-z*: *0* < *1* / *?v* ^ *?v* **by** *simp*

— For a given n, let A be a family of events indexed by a set S. Each A contains the graphs whose induced subgraphs over S contain the selected copy of H by f over S.
let *?A* = λ*n*. λ*S*. {*es* $\in$ *space* (*edge-space.P n* (*p n*)). *subgraph* (*f S*) (*induced-subgraph S* (*edge-space.edge-ugraph n es*))}
let *?I* = λ*n*. {*S*. *S* $\subseteq$ {*1*..*n*} $\wedge$ *card S* = *?v*}

assume *p-nz*: *nonzero-prob-fun p*
hence *p*: *valid-prob-fun p*
by (*fact nonzero-fun-is-valid-fun*)

{
  fix *n*
  — At this point, we can assume almost anything about *n*: We only have to show that a function converges, hence the necessary properties are allowed to be violated for small values of *n*.
  assume *n-2v*: $2 * ?v \le n$
  hence *n*: $?v \le n$
    by *simp*

  have *is-es*: *edge-space* (*p n*)
    by *unfold-locales* (*auto simp*: *p*)

  then interpret *edge-space n p n*
    .

  let $?A = ?A\ n$
  let $?I = ?I\ n$

  — A nice potpourri with some technical facts about *S*.
  {
    fix *S*
    assume $S \in ?I$
    hence *0*: $S \subseteq \{1..n\}$ $?v = card\ S$ *finite S*
      by (*auto intro*: *finite-subset*)
    hence *1*: $H \simeq f\ S$ *uverts* $(f\ S) = S$
      using *f wellformed-finite* **unfolding** *finite-graph-def is-fixed-selector-def* **by** *auto*
    have *2*: *finite-graph* (*f S*)
      using *0*(*3*) *1*(*1*,*2*) **by** (*metis wellformed-finite*)
    have *3*: *nonempty-graph* (*f S*)
      using *0*(*2*) *1*(*1*,*2*) **by** (*metis card-eq-0-iff finite finite-graph-def isomorphic-cards*(*2*) *nonempty nonempty-graph-def prod.collapse snd-conv*)
    note *0 1 2 3*
  }
  note *I* = *this*

  — In the following two blocks, we prove the probabilities of the events *A* and the probability of the intersection of two events *A*. For both cases, we employ the auxiliary lemma *edge-space.induced-subgraph-prob* which is not very interesting. For the latter however, the tricky part is to argue that such an intersection is equivalent to the *union* of the desired copies of *H* to be contained in the *union* of the induced subgraphs.
  {
    fix *S*
    assume *S*: $S \in ?I$
    note $S' = I[OF\ S]$
    have *prob* $(?A\ S) = p\ n\ \hat{}\ ?e$
      using *isomorphic-cards*(*2*)[*OF* S'(*4*)] S' **by** (*simp add*: *S-verts-def induced-subgraph-prob*)

}
note *prob-A* = *this*

{
fix *S T*
assume $S \in$ *?I* **note** $S = I$[*OF this*]
assume $T \in$ *?I* **note** $T = I$[*OF this*]
— Note that we do not restrict S and T to be disjoint, since we need the general case later to determine when two events are independent. Additionally, it would be unneeded at this point.

have *prob* (*?A S* ∩ *?A T*) = *prob* {*es* ∈ *space P*. *subgraph* ($S \cup T$, *uedges* (*f S*) ∪ *uedges* (*f T*)) (*induced-subgraph* ($S \cup T$) (*edge-ugraph es*))} (**is** - = *prob ?M*)
proof (*rule arg-cong*[**where** *f* = *prob*])
have *?A S* ∩ *?A T* = {*es* ∈ *space P*. *subgraph* (*f S*) (*induced-subgraph S* (*edge-ugraph es*)) ∧ *subgraph* (*f T*) (*induced-subgraph T* (*edge-ugraph es*))}
by *blast*
also have ... = *?M*
using *S T* **by** (*auto simp*: *subgraph-union-induced*)
finally show *?A S* ∩ *?A T* = ...
.
qed
also have ... = *p n* ^ *card* (*uedges* ($S \cup T$, *uedges* (*f S*) ∪ *uedges* (*f T*)))
proof (*rule induced-subgraph-prob*)
show *uwellformed* ($S \cup T$, *uedges* (*f S*) ∪ *uedges* (*f T*))
using *S*(*4*,*5*) *T*(*4*,*5*) **unfolding** *uwellformed-def* **by** *auto*
next
show $S \cup T \subseteq$ *S-verts*
using *S*(*1*) *T*(*1*) **unfolding** *S-verts-def* **by** *simp*
qed *simp*
also have ... = *p n* ^ *card* (*uedges* (*f S*) ∪ *uedges* (*f T*))
by *simp*

finally have *prob* (*?A S* ∩ *?A T*) = *p n* ^ *card* (*uedges* (*f S*) ∪ *uedges* (*f T*))
.
}
note *prob-A-intersect* = *this*

— Another technical detail is that our family of events A are a valid instantiation for the "Δ lemmas" from section 3.3.
have *is-psi*: *prob-space-with-indicators P ?I ?A*
proof
show *finite ?I*
by (*rule finite-subset*[**where** *B* = *Pow* {*1*..*n*}]) *auto*
next
show *?A* ‘ *?I* ⊆ *sets P*
unfolding *sets-eq space-eq* **by** *blast*
next

let *?V* = {*1..?v*}
have *0* < *prob* (*?A ?V*)
by (*simp add*: *prob-A n p-nz*)
moreover have *?V* ∈ *?I*
using *n* **by** *force*
ultimately show ∃ *i* ∈ *?I*. *0* < *prob* (*?A i*)
by *blast*
qed

then interpret *prob-space-with-indicators P ?I ?A*
.

— We proceed by reducing the claim of the 1-statement that the probability tends to 1 to showing that the expectation that the sum of all indicators of the respective events A tends to 0. (The actual reduction is done at the end of the proof, we merely collect the facts here.)
have *compl-prob*: *1* − *prob* {*es* ∈ *space P*. ¬ *H* ⊑ *edge-ugraph es*} = *prob-in-class p* {*G*. *H* ⊑ *G*} *n*
by (*subst prob-compl*[*symmetric*]) (*auto simp*: *space-eq sets-eq intro*: *arg-cong*[**where** *f* = *prob*])

have *prob* {*es* ∈ *space P*. ¬ *H* ⊑ *edge-ugraph es*} ≤ *prob* {*es* ∈ *space P*. *Y es* = *0*} (**is** *?compl* ≤ -)
proof (*rule finite-measure-mono*, *safe*)
fix *es*
assume *es* ∈ *space P*
hence *es*: *uwellformed* (*edge-ugraph es*)
unfolding *space-eq* **by** (*rule wellformed-and-finite*(*2*))
assume *H*: ¬ *H* ⊑ *edge-ugraph es*
{
fix *S*
assume *S* ⊆ {*1..n*} *card S* = *?v*
moreover hence *finite S S* ⊆ *uverts* (*edge-ugraph es*)
unfolding *uverts-edge-ugraph S-verts-def* **by** (*auto intro*: *finite-subset*)
ultimately have ¬ *subgraph* (*f S*) (*induced-subgraph S* (*edge-ugraph es*))
using *H es* **by** (*metis fixed-selector-induced-subgraph*[*OF f*])
hence *X S es* = *0*
unfolding *X-def* **by** *simp*
}
thus *Y es* = *0*
unfolding *Y-def* **by** *simp*
qed *simp*

— By applying the Δ lemma, we obtain our central inequality. The rest of the proof gives bounds for μ, Δ_d and quotients which occur on the right hand side.
hence *compl-upper*: *?compl* ≤ *1* / μ + Δ_d / μ^*2*
by (*rule order-trans*) (*fact prob-*μ*-*Δ_d)

— Lower bound for the expectation. We use $\left(\frac{n}{k}\right)^k$ as lower bound for $\binom{n}{k}$.

have $1 \ / \ ?v \ \hat{} \ ?v * (real\ n \ \hat{} \ ?v * p\ n \ \hat{} \ ?e) = (n \ / \ ?v) \ \hat{} \ ?v * p\ n \ \hat{} \ ?e$
  by (*simp add*: *power-divide*)
also have ... $\leq (n\ choose\ ?v) * p\ n \ \hat{} \ ?e$
  proof (*rule mult-right-mono*, *rule binomial-ge-n-over-k-pow-k*)
    show $?v \leq n$
      using n **.**
    show $0 \leq p\ n \ \hat{} \ ?e$
      using p **by** *simp*
  qed
also have ... $= (\sum S \in ?I.\ p\ n \ \hat{} \ ?e)$
  by (*simp add*: *n-subsets*)
also have ... $= (\sum S \in ?I.\ prob\ (?A\ S))$
  by (*simp add*: *prob-A*)
also have ... $= \mu$
  unfolding *expectation-X-Y X-def* **using** *expectation-indicator* **by** *force*
finally have *ex-lower*: $1 \ / \ (?v \ \hat{} \ ?v) * (real\ n \ \hat{} \ ?v * p\ n \ \hat{} \ ?e) \leq \mu$
  .

— Upper bound for the inverse expectation. Follows trivially from above.
have *ex-lower-pos*: $0 < 1 \ / \ ?v \ \hat{} \ ?v * (real\ n \ \hat{} \ ?v * p\ n \ \hat{} \ ?e)$
  proof (*rule mult-pos-pos*[*OF vpowv-inv-gr-z mult-pos-pos*])
    have $0 < real\ n$
      using n *nonempty finite* **unfolding** *nonempty-graph-def finite-graph-def*
by *auto*
    thus $0 < real\ n \ \hat{} \ ?v$
      by *simp*
  next
    show $0 < p\ n \ \hat{}\ card\ (uedges\ H)$
      using *p-nz* **by** *simp*
  qed
hence $1 \ / \ \mu \leq 1 \ / \ (1 \ / \ ?v \ \hat{} \ ?v * (real\ n \ \hat{} \ ?v * p\ n \ \hat{} \ ?e))$
by (*rule divide-left-mono*[*OF ex-lower zero-le-one mult-pos-pos*[*OF* μ*-non-zero*]])
hence *inv-ex-upper*: $1 \ / \ \mu \leq ?v \ \hat{} \ ?v * (1 \ / \ (real\ n \ \hat{} \ ?v * p\ n \ \hat{} \ ?e))$
  by *simp*

— Recall the definition of Δ_d:

$$\Delta_d = \sum_{\substack{S \in I,\, T \in I \\ S \neq T \\ A_S,\, A_T \text{ not independent}}} \Pr[A_S \cap A_T]$$

We are going to prove an upper bound for that sum, so we can safely augment the index set by replacing it with a neccessary condition.
The idea is that if the two sets S and T are not independent, their intersection is not empty. We prove that by contraposition, i.e. if the intersection is empty, then they are independent. This in turn can be shown using some basic properties of *f*.
{
  fix $S\ T$
  assume $S \in ?I\ T \in ?I$
  hence $*$: $prob\ (?A\ S) * prob\ (?A\ T) = p\ n \ \hat{} (2 * ?e)$

using *prob-A* **by** (*simp add*: *power-even-eq power2-eq-square*)

note $S = I[OF \langle S \in ?I\rangle]$
note $T = I[OF \langle T \in ?I\rangle]$
assume *disj*: $S \cap T = \{\}$

have *prob* (*?A S* ∩ *?A T*) = *p n* ^ *card* (*uedges* (*f S*) ∪ *uedges* (*f T*))
using ‹*S* ∈ *?I*› ‹*T* ∈ *?I*› **by** (*fact prob-A-intersect*)
also have ... = *p n* ^(*card* (*uedges* (*f S*)) + *card* (*uedges* (*f T*)))
proof (*rule arg-cong*[*OF card-Un-disjoint*])
have *finite-graph* (*f S*) *finite-graph* (*f T*)
using *S T* **by** (*auto simp*: *wellformed-finite*)
thus *finite* (*uedges* (*f S*)) *finite* (*uedges* (*f T*))
unfolding *finite-graph-def* **by** *auto*
next
have *uedges* (*f S*) ⊆ *all-edges S* *uedges* (*f T*) ⊆ *all-edges T*
using *S*(*4*,*5*) *T*(*4*,*5*) **by** (*metis wellformed-all-edges*)+
moreover have *all-edges S* ∩ *all-edges T* = {}
by (*fact all-edges-disjoint*[*OF disj*])
ultimately show *uedges* (*f S*) ∩ *uedges* (*f T*) = {}
by *blast*
qed
also have ... = *p n* ^(*2* ∗ *?e*)
using *isomorphic-cards*(*2*)[*OF isomorphic-sym*[*OF S*(*4*)]] *isomorphic-cards*(*2*)[*OF isomorphic-sym*[*OF T*(*4*)]] **by** (*simp add*: *mult-2*)
finally have ∗∗: *prob* (*?A S* ∩ *?A T*) = ...
.

from ∗ ∗∗ **have** *indep* (*?A S*) (*?A T*)
unfolding *indep-def* **by** *force*
}
note *indep* = *this*

— Now we prove an upper bound for Δ_d.
have Δ_d = (∑*S* ∈ *?I*. ∑*T* | *T* ∈ *?I* ∧ *ineq-dep S T*. *prob* (*?A S* ∩ *?A T*))
unfolding Δ_d*-def* **..**

— Augmenting the index set as described above.
also have ... ≤ (∑*S* ∈ *?I*. ∑*T* | *T* ∈ *?I* ∧ *S* ∩ *T* ≠ {}. *prob* (*?A S* ∩ *?A T*))
by (*rule sum-mono*[*OF sum-mono2*]) (*auto simp*: *indep measure-nonneg*)

— So far, we are adding the intersection probabilities over pairs of sets which have a nonempty intersection. Since we know that these intersections have at least one element (as they are nonempty) and at most $|V(H)|$ elements (by definition of I). In this step, we will partition this sum by cardinality of the intersections.
also have ... = (∑*S* ∈ *?I*. ∑*T* ∈ (⋃*k* ∈ {*1*..*?v*}. {*T* ∈ *?I*. *card* (*S* ∩ *T*) = *k*}). *prob* (*?A S* ∩ *?A T*))
proof (*rule sum.cong*, *rule refl*, *rule sum.cong*)

fix S
assume $S \in ?I$
note $I(2,3)[OF\ this]$
hence $\{T.\ S \cap T \neq \{\}\} = (\bigcup k \in \{1..?v\}.\ \{T.\ card\ (S \cap T) = k\})$
by $(simp\ add{:}\ partition\text{-}set\text{-}of\text{-}intersecting\text{-}sets\text{-}by\text{-}card)$
thus $\{T \in ?I.\ S \cap T \neq \{\}\} = (\bigcup k{\in}\{1..?v\}.\ \{T \in ?I.\ card\ (S \cap T) = k\})$
by *blast*
qed *simp*
also have $\ldots = (\sum S \in ?I.\ \sum k = 1..?v.\ \sum T \mid T \in ?I \wedge card\ (S \cap T) = k.$ $prob\ (?A\ S \cap ?A\ T))$
by $(rule\ sum.cong,\ rule\ refl,\ rule\ sum.UNION\text{-}disjoint)\ auto$
also have $\ldots = (\sum k = 1..?v.\ \sum S \in ?I.\ \sum T \mid T \in ?I \wedge card\ (S \cap T) = k.$ $prob\ (?A\ S \cap ?A\ T))$
by $(rule\ sum.swap)$

— In this step, we compute an upper bound for the intersection probability and argue that it only depends on the cardinality of the intersection.
also have $\ldots \leq (\sum k = 1..?v.\ \sum S \in ?I.\ \sum T \mid T \in ?I \wedge card\ (S \cap T) = k.$ $p\ n\ powr\ (2 * ?e - max\text{-}density\ H * k))$
proof $(rule\ sum\text{-}mono)+$
fix k
assume k: $k \in \{1..?v\}$
fix $S\ T$
assume $S \in ?I\ T \in \{T.\ T \in ?I \wedge card\ (S \cap T) = k\}$
hence $T \in ?I$ **and** $ST\text{-}k$: $card\ (S \cap T) = k$
by *auto*
note $S = I[OF\ \langle S \in ?I\rangle]$
note $T = I[OF\ \langle T \in ?I\rangle]$

let $?cST = card\ (uedges\ (f\ S) \cap uedges\ (f\ T))$

— We already know the intersection probability.
have $prob\ (?A\ S \cap ?A\ T) = p\ n\ \widehat{}\ card\ (uedges\ (f\ S) \cup uedges\ (f\ T))$
using $\langle S \in ?I\rangle\ \langle T \in ?I\rangle$ **by** $(fact\ prob\text{-}A\text{-}intersect)$

— Now, we consider the number of edges shared by the copies of H over S and T.
also have $\ldots = p\ n\ \widehat{}\,(card\ (uedges\ (f\ S)) + card\ (uedges\ (f\ T)) - ?cST)$
using $S\ T$ **unfolding** *finite-graph-def* **by** $(simp\ add{:}\ card\text{-}union)$
also have $\ldots = p\ n\ \widehat{}\,(?e + ?e - ?cST)$
by $(metis\ isomorphic\text{-}cards(2)[OF\ S(4)]\ isomorphic\text{-}cards(2)[OF\ T(4)])$
also have $\ldots = p\ n\ \widehat{}\,(2 * ?e - ?cST)$
by $(simp\ add{:}\ mult\text{-}2)$
also have $\ldots = p\ n\ powr\ (2 * ?e - ?cST)$
using $p\text{-}nz$ **by** $(simp\ add{:}\ powr\text{-}realpow)$
also have $\ldots = p\ n\ powr\ (real\ (2 * ?e) - real\ ?cST)$
using $isomorphic\text{-}cards[OF\ S(4)]\ S(6)$ **by** $(metis\ of\text{-}nat\text{-}diff\ card\text{-}mono$ $finite\text{-}graph\text{-}def\ inf\text{-}le1\ mult\text{-}le\text{-}mono\ mult\text{-}numeral\text{-}1\ numeral\text{-}One\ one\text{-}le\text{-}numeral)$

— Since the intersection graph is also an isomorphic subgraph of H, we know that its density has to be less than or equal to the maximum density of H. The proof is quite technical.

also have $\ldots \leq p\ n\ powr\ (2 * ?e - max\text{-}density\ H * k)$
proof $(rule\ powr\text{-}mono3)$
have $?cST = density\ (S \cap T,\ uedges\ (f\ S) \cap uedges\ (f\ T)) * k$
unfolding *density-def* **using** *k ST-k* **by** *simp*
also have $\ldots \leq max\text{-}density\ (f\ S) * k$
proof $(rule\ mult\text{-}right\text{-}mono,\ cases\ uedges\ (f\ S) \cap uedges\ (f\ T) = \{\})$
case *True*
hence $density\ (S \cap T,\ uedges\ (f\ S) \cap uedges\ (f\ T)) = 0$
unfolding *density-def* **by** *simp*
also have $0 \leq density\ (f\ S)$
unfolding *density-def* **by** *simp*
also have $density\ (f\ S) \leq max\text{-}density\ (f\ S)$
using *S* **by** (*simp add*: *max-density-is-max subgraph-refl*)
finally show $density\ (S \cap T,\ uedges\ (f\ S) \cap uedges\ (f\ T)) \leq max\text{-}density\ (f\ S)$
.
next
case *False*
show $density\ (S \cap T,\ uedges\ (f\ S) \cap uedges\ (f\ T)) \leq max\text{-}density\ (f\ S)$
proof $(rule\ max\text{-}density\text{-}is\text{-}max)$
show $finite\text{-}graph\ (S \cap T,\ uedges\ (f\ S) \cap uedges\ (f\ T))$
using $T(3,6)$ **by** (*metis finite-Int finite-graph-def fst-eqD snd-conv*)
show $nonempty\text{-}graph\ (S \cap T,\ uedges\ (f\ S) \cap uedges\ (f\ T))$
unfolding *nonempty-graph-def* **using** *k ST-k False* **by** *force*
show $uwellformed\ (S \cap T,\ uedges\ (f\ S) \cap uedges\ (f\ T))$
using $S(4,5)$ $T(4,5)$ **unfolding** *uwellformed-def* **by** (*metis Int-iff fst-eqD snd-eqD*)
show $subgraph\ (S \cap T,\ uedges\ (f\ S) \cap uedges\ (f\ T))\ (f\ S)$
using $S(5)$ **by** (*metis fst-eqD inf-sup-ord*(1) *snd-conv subgraph-def*)
qed (*simp add*: *S*)
qed *simp*
also have $\ldots = max\text{-}density\ H * k$
using *assms S* **by** (*simp add*: *isomorphic-max-density*[**where** $G_1 = H$ **and** $G_2 = f\ S$])
finally have $?cST \leq max\text{-}density\ H * k$
.
thus $2 * ?e - max\text{-}density\ H * k \leq 2 * ?e - real\ ?cST$
by *linarith*
qed (*auto simp*: *p-nz*)
finally show $prob\ (?A\ S \cap ?A\ T) \leq \ldots$
.
qed

— Further rewriting the index sets.
also have $\ldots = (\sum k = 1..?v.\ \sum (S,\ T) \in (SIGMA\ S : ?I.\ \{T \in ?I.\ card\ (S$

$\cap$ *T*) = *k*}). *p n powr* (*2* $*$ *?e* $-$ *max-density H* $*$ *k*))
by (*rule sum.cong*, *rule refl*, *rule sum.Sigma*) *auto*
also have ... = ($\sum$ *k* = *1..?v*. *card* (*SIGMA S* : *?I*. {*T* $\in$ *?I*. *card* (*S* $\cap$ *T*) = *k*}) $*$ *p n powr* (*2* $*$ *?e* $-$ *max-density H* $*$ *k*))
by (*rule sum.cong*) *auto*

— Here, we compute the cardinality of the index sets and use the same upper bounds for the binomial coefficients as for the 0-statement.
also have ... $\leq$ ($\sum$ *k* = *1..?v*. *?v* ^ *k* $*$ (*real n* ^ (*2* $*$ *?v* $-$ *k*) $*$ *p n powr* (*2* $*$ *?e* $-$ *max-density H* $*$ *k*)))
proof (*rule sum-mono*)
fix *k*
assume *k*: *k* $\in$ {*1..?v*}
let *?p* = *p n powr* (*2* $*$ *?e* $-$ *max-density H* $*$ *k*)

have *card* (*SIGMA S* : *?I*. {*T* $\in$ *?I*. *card* (*S* $\cap$ *T*) = *k*}) = ($\sum$ *S* $\in$ *?I*. *card* {*T* $\in$ *?I*. *card* (*S* $\cap$ *T*) = *k*}) (**is** *?lhs* = -)
by *simp*
also have ... = ($\sum$ *S* $\in$ *?I*. (*?v choose k*) $*$ ((*n* $-$ *?v*) *choose* (*?v* $-$ *k*)))
using *n k* **by** (*fastforce simp*: *card-set-of-intersecting-sets-by-card*)
also have ... = (*n choose ?v*) $*$ ((*?v choose k*) $*$ ((*n* $-$ *?v*) *choose* (*?v* $-$ *k*)))
by (*auto simp*: *n-subsets*)
also have ... $\leq$ *n* ^ *?v* $*$ ((*?v choose k*) $*$ ((*n* $-$ *?v*) *choose* (*?v* $-$ *k*)))
using *n* **by** (*simp add*: *binomial-le-pow*)
also have ... $\leq$ *n* ^ *?v* $*$ *?v* ^ *k* $*$ ((*n* $-$ *?v*) *choose* (*?v* $-$ *k*))
using *k* **by** (*simp add*: *binomial-le-pow*)
also have ... $\leq$ *n* ^ *?v* $*$ *?v* ^ *k* $*$ (*n* $-$ *?v*) ^ (*?v* $-$ *k*)
using *n-2v* **by** (*simp add*: *binomial-le-pow*)
also have ... $\leq$ *n* ^ *?v* $*$ *?v* ^ *k* $*$ *n* ^ (*?v* $-$ *k*)
by (*simp add*: *power-mono*)
also have ... = *?v* ^ *k* $*$ (*n* ^ (*?v* + (*?v* $-$ *k*)))
by (*simp add*: *power-add*)
also have ... = *?v* ^ *k* $*$ *n* ^ (*2* $*$ *?v* $-$ *k*) (**is** - = *?rhs*)
using *k* **by** (*simp add*: *mult-2*)
finally have *?lhs* $\leq$ *?rhs* **.**
hence *real ?lhs* $\leq$ *real ?rhs*
using *of-nat-le-iff* **by** *blast*
moreover have *0* $\leq$ *?p*
by *simp*
ultimately have *?lhs* $*$ *?p* $\leq$ *?rhs* $*$ *?p*
by (*rule mult-right-mono*)
also have ... = *?v* ^ *k* $*$ (*real n* ^ (*2* $*$ *?v* $-$ *k*) $*$ *?p*)
by *simp*
finally show *?lhs* $*$ *?p* $\leq$ ...
.
qed
finally have *delta-upper*: $\Delta_d \leq$ ($\sum$ *k* = *1..?v*. *?v* ^ *k* $*$ (*real n* ^ (*2* $*$ *?v* $-$ *k*) $*$ *p n powr* (*2* $*$ *?e* $-$ *max-density H* $*$ *k*)))

.

— At this point, we have established all neccessary bounds.
note *is-es is-psi compl-prob compl-upper ex-lower ex-lower-pos inv-ex-upper delta-upper*
}
note *facts* = *this*

— Recall our central inequality. We now prove that both summands tend to 0. This is mainly an exercise in bookkeeping and real arithmetics as no intelligent ideas are involved.
have (λn. *1* / *prob-space-with-indicators*.μ (*MGn p n*) (*?I n*) (*?A n*)) $\longrightarrow$ *0*
proof (*rule LIMSEQ-le-zero*)
have (λn. *1* / (*real n* ^ *?v* ∗ *p n* ^ *?e*)) $\longrightarrow$ *0*
proof (*rule LIMSEQ-le-zero*[*OF - eventually-sequentiallyI eventually-sequentiallyI*])
fix *n*
show *0* ≤ *1* / (*real n* ^ *?v* ∗ *p n* ^ *?e*)
using *p* **by** *simp*

assume *n*: *1* ≤ *n*
have *1* / (*real n* ^ *?v* ∗ *p n* ^ *?e*) = *1* / (*real n powr ?v* ∗ *p n powr ?e*)
using *n p-nz* **by** (*simp add*: *powr-realpow*[*symmetric*])
also have ... = *real n powr* −*real ?v* ∗ *p n powr* −*real ?e*
by (*simp add*: *powr-minus-divide*)
also have ... = (*real n powr* −(*?v* / *?e*)) *powr ?e* ∗ (*p n powr* −*1*) *powr ?e*
using *v-e-nz*
by (*metis mult-minus1 nonzero-eq-divide-eq powr-powr order.irrefl*)
also have ... = (*real n powr* −(*?v* / *?e*) ∗ *p n powr* −*1*) *powr ?e*
using *powr-mult* **by** *presburger*
also have ... = (*real n powr* −(*1* / (*?e* / *?v*)) ∗ *p n powr* −*1*) *powr ?e*
by *simp*
also have ... ≤ (*real n powr* −(*1* / *max-density H*) ∗ *p n powr* −*1*) *powr ?e*
apply (*rule powr-mono2*[*OF - - mult-right-mono*[*OF powr-mono*[*OF le-imp-neg-le*[*OF divide-left-mono*]]]])
using *n v-e-nz p p-nz*
by (*auto simp*:
max-density-is-max[*unfolded density-def*, *OF finite finite nonempty wellformed subgraph-refl*]
max-density-gr-zero[*OF finite nonempty wellformed*])
also have ... = (*real n powr* −(*1* / *max-density H*) ∗ (*1* / *p n powr 1*)) *powr ?e*
by (*simp add*: *powr-minus-divide*[*symmetric*])
also have ... = (*real n powr* −(*1* / *max-density H*) / *p n*) *powr ?e*
using *p p-nz* **by** *simp*
also have ... = (*subgraph-threshold H n* / *p n*) *powr ?e*
unfolding *subgraph-threshold-def* **..**
finally show *1* / (*real n* ^ *?v* ∗ *p n* ^ *?e*) ≤ (*subgraph-threshold H n* / *p*

n) *powr ?e* **.**
next
show (*λn*. (*subgraph-threshold H n* / *p n*) *powr real* (*card* (*uedges H*))) ⟶ *0*
using *p-threshold p-nz v-e-nz*
by (*auto simp*: *subgraph-threshold-def divide-nonneg-pos intro*!: *tendsto-zero-powrI*)
qed
hence (*λn*. *?v* ^ *?v* ∗ (*1* / (*real n* ^ *?v* ∗ *p n* ^ *?e*))) ⟶ *real* (*?v* ^ *?v*) ∗ *0*
by (*rule LIMSEQ-const-mult*)
thus (*λn*. *?v* ^ *?v* ∗ (*1* / (*real n* ^ *?v* ∗ *p n* ^ *?e*))) ⟶ *0*
by *simp*
next
show ∀∞*n*. *0* ≤ *1* / *prob-space-with-indicators.μ* (*MGn p n*) (*?I n*) (*?A n*)
by (*rule eventually-sequentiallyI*[*OF less-imp-le*[*OF divide-pos-pos*[*OF* - *prob-space-with-indicators.μ-non-zero*[*OF facts*(*2*)]]]]) *simp+*
next
show ∀∞*n*. *1* / *prob-space-with-indicators.μ* (*MGn p n*) (*?I n*) (*?A n*) ≤ *?v* ^ *?v* ∗ (*1* / (*real n* ^ *?v* ∗ *p n* ^ *?e*))
using *facts*(*7*) **by** (*rule eventually-sequentiallyI*)
qed
moreover have (*λn*. *prob-space-with-indicators.*Δ_d (*MGn p n*) (*?I n*) (*?A n*)) ≪ (*λn*. (*prob-space-with-indicators.μ* (*MGn p n*) (*?I n*) (*?A n*))^*2*)
proof (*rule less-fun-bounds*)
let *?num* = *λn k*. *?v* ^ *k* ∗ (*real n* ^ (*2* ∗ *?v* − *k*) ∗ *p n powr* (*2* ∗ *?e* − *max-density H* ∗ *k*))
let *?den* = *λn*. ((*1* / *?v* ^ *?v*) ∗ (*real n* ^ *?v* ∗ *p n* ^ *?e*))^*2*

— We have to show that a sum is asymptotically smaller than a constant term. We do that by showing that each summand is asymptotically smaller than the term.

{
fix *k*
assume *k*: *k* ∈ {*1..?v*}
let *?den′* = *λn*. (*1* / *?v* ^ *?v*)^*2* ∗ (*real n* ^(*2* ∗ *?v*) ∗ *p n* ^(*2* ∗ *?e*))
have *den′*: *?den′* = *?den*
by (*subst power-mult-distrib*) (*simp add*: *power-mult-distrib power-even-eq*)

have (*λn*. *?num n k*) ≪ *?den′*
proof (*rule less-fun-const-quot*)
have (*λn*. (*subgraph-threshold H n* / *p n*) *powr* (*max-density H* ∗ *k*)) ⟶ *0*
using *p-threshold mult-pos-pos*[*OF max-density-gr-zero*[*OF finite nonempty wellformed*]] *p-nz k*
by (*auto simp*: *subgraph-threshold-def divide-nonneg-pos intro*!: *tendsto-zero-powrI*)
thus (*λn*. (*real n* ^(*2* ∗ *?v* − *k*) ∗ *p n powr* (*2* ∗ *?e* − *max-density H* ∗ *k*)) / (*real n* ^(*2* ∗ *?v*) ∗ *p n* ^(*2* ∗ *?e*))) ⟶ *0*

proof (*rule LIMSEQ-cong*[*OF* - *eventually-sequentiallyI*])
fix *n* :: *nat*
assume *n*: *1* ≤ *n*
have (*real n* ^(*2* ∗ *?v* − *k*) ∗ *p n powr* (*2* ∗ *?e* − *max-density H* ∗ *k*)) / (*real n* ^(*2* ∗ *?v*) ∗ *p n* ^(*2* ∗ *?e*)) =
(*n powr* (*2* ∗ *?v* − *k*) ∗ *p n powr* (*2* ∗ *?e* − *max-density H* ∗ *k*)) / (*n powr* (*2* ∗ *?v*) ∗ *p n powr* (*2* ∗ *?e*)) (**is** *?lhs* = -)
using *n p-nz* **by** (*simp add*: *powr-realpow*[*symmetric*])
also have ... = (*n powr* (*2* ∗ *?v* − *k*) / *n powr* (*2* ∗ *?v*)) ∗ (*p n powr* (*2* ∗ *?e* − *max-density H* ∗ *k*) / (*p n powr* (*2* ∗ *?e*)))
by *simp*
also have ... = (*n powr* (*real* (*2* ∗ *?v* − *k*) − *2* ∗ *?v*)) ∗ *p n powr* ((*2* ∗ *?e* − *max-density H* ∗ *k*) − (*2* ∗ *?e*))
by (*simp add*: *powr-diff* [*symmetric*])
also have ... = *n powr* −*real k* ∗ *p n powr* ((*2* ∗ *?e* − *max-density H* ∗ *k*) − (*2* ∗ *?e*))
apply (*rule arg-cong*[**where** *y* = − *real k*])
using *k* **by** *fastforce*
also have ... = *n powr* −*real k* ∗ *p n powr* − (*max-density H* ∗ *k*)
by *simp*
also have ... = (*n powr* −(*1* / *max-density H*)) *powr* (*max-density H* ∗ *k*) ∗ *p n powr* − (*max-density H* ∗ *k*)
using *max-density-gr-zero*[*OF finite nonempty wellformed*] **by** (*simp add*: *powr-powr*)
also have ... = (*n powr* −(*1* / *max-density H*)) *powr* (*max-density H* ∗ *k*) ∗ (*p n powr* −*1*) *powr* (*max-density H* ∗ *k*)
by (*metis mult-minus1 powr-powr*)
also have ... = (*n powr* −(*1* / *max-density H*) ∗ *p n powr* −*1*) *powr* (*max-density H* ∗ *k*)
using *powr-mult* **by** *presburger*
also have ... = (*n powr* −(*1* / *max-density H*) ∗ (*1* / *p n powr 1*)) *powr* (*max-density H* ∗ *k*)
by (*simp add*: *powr-minus-divide*[*symmetric*])
also have ... = (*n powr* −(*1* / *max-density H*) / *p n*) *powr* (*max-density H* ∗ *k*)
by (*simp add*: *p p-nz*)
also have ... = (*subgraph-threshold H n* / *p n*) *powr* (*max-density H* ∗ *k*) (**is** - = *?rhs*)
unfolding *subgraph-threshold-def* **..**
finally have *?lhs* = *?rhs*
.
thus *?rhs* = *?lhs*
by *simp*
qed
next
show (*1* / *?v* ^ *?v*)^*2* ≠ *0*
using *vpowv-inv-gr-z* **by** *auto*
qed

hence $(\lambda n.\ ?num\ n\ k) \ll ?den$
by $(rule\ subst[OF\ den'])$
}
hence $(\lambda n.\ \sum k = 1..?v.\ ?num\ n\ k\ /\ ?den\ n) \longrightarrow (\sum k = 1..?v.\ 0)$
by $(rule\ tendsto\text{-}sum)$
hence $(\lambda n.\ \sum k = 1..?v.\ ?num\ n\ k\ /\ ?den\ n) \longrightarrow 0$
by *simp*
moreover have $(\lambda n.\ \sum k = 1..?v.\ ?num\ n\ k\ /\ ?den\ n) = (\lambda n.\ (\sum k = 1..?v.\ ?num\ n\ k)\ /\ ?den\ n)$
by $(simp\ add{:}\ sum\text{-}left\text{-}div\text{-}distrib)$
ultimately show $(\lambda n.\ \sum k = 1..?v.\ ?num\ n\ k) \ll ?den$
by *metis*

show $\forall^\infty n.\ prob\text{-}space\text{-}with\text{-}indicators.\Delta_d\ (MGn\ p\ n)\ (?I\ n)\ (?A\ n) \le (\sum k = 1..?v.\ ?num\ n\ k)$
using $facts(8)$ **by** $(rule\ eventually\text{-}sequentiallyI)$

show $\forall^\infty n.\ ?den\ n \le (prob\text{-}space\text{-}with\text{-}indicators.\mu\ (MGn\ p\ n)\ (?I\ n)\ (?A\ n))\hat{}2$
using $facts(5)\ facts(6)$ **by** $(rule\ eventually\text{-}sequentiallyI[OF\ power\text{-}mono[OF\ \text{-}\ less\text{-}imp\text{-}le]])$

show $\forall^\infty n.\ 0 \le prob\text{-}space\text{-}with\text{-}indicators.\Delta_d\ (MGn\ p\ n)\ (?I\ n)\ (?A\ n)$
using $facts(2)$ **by** $(rule\ eventually\text{-}sequentiallyI[OF\ prob\text{-}space\text{-}with\text{-}indicators.\Delta_d\text{-}nonneg])$

show $\forall^\infty n.\ 0 < (prob\text{-}space\text{-}with\text{-}indicators.\mu\ (MGn\ p\ n)\ (?I\ n)\ (?A\ n))\hat{}2$
using $facts(2)$ **by** $(rule\ eventually\text{-}sequentiallyI[OF\ prob\text{-}space\text{-}with\text{-}indicators.\mu\text{-}sq\text{-}non\text{-}zero])$

show $\forall^\infty n.\ 0 < ?den\ n$
using $facts(6)$ **by** $(rule\ eventually\text{-}sequentiallyI[OF\ zero\text{-}less\text{-}power])$
qed
ultimately have $(\lambda n.$
$1\ /\ prob\text{-}space\text{-}with\text{-}indicators.\mu\ (MGn\ p\ n)\ (?I\ n)\ (?A\ n) +$
$prob\text{-}space\text{-}with\text{-}indicators.\Delta_d\ (MGn\ p\ n)\ (?I\ n)\ (?A\ n)\ /\ (prob\text{-}space\text{-}with\text{-}indicators.\mu\ (MGn\ p\ n)\ (?I\ n)\ (?A\ n))\hat{}2$
$) \longrightarrow 0$
by $(subst\ add\text{-}0\text{-}left[$**where** $a = 0,\ symmetric])\ (rule\ tendsto\text{-}add)$

— By now, we can actually perform the reduction mentioned above.
hence $(\lambda n.\ probGn\ p\ n\ (\lambda es.\ \neg\ H \sqsubseteq edge\text{-}space.edge\text{-}ugraph\ n\ es)) \longrightarrow 0$
proof $(rule\ LIMSEQ\text{-}le\text{-}zero)$
show $\forall^\infty n.\ 0 \le probGn\ p\ n\ (\lambda es.\ \neg\ H \sqsubseteq edge\text{-}space.edge\text{-}ugraph\ n\ es)$
by $(rule\ eventually\text{-}sequentiallyI)\ (rule\ measure\text{-}nonneg)$
next
show $\forall^\infty n.$
$probGn\ p\ n\ (\lambda es.\ \neg\ H \sqsubseteq edge\text{-}space.edge\text{-}ugraph\ n\ es) \le$
$1\ /\ prob\text{-}space\text{-}with\text{-}indicators.\mu\ (MGn\ p\ n)\ (?I\ n)\ (?A\ n) +$
$prob\text{-}space\text{-}with\text{-}indicators.\Delta_d\ (MGn\ p\ n)\ (?I\ n)\ (?A\ n)\ /\ (prob\text{-}space\text{-}with\text{-}indicators.\mu\ (MGn\ p\ n)\ (?I\ n)\ (?A\ n))\hat{}2$

by (*rule eventually-sequentiallyI*[*OF facts*(*4*)])
qed
hence ($\lambda n.$ *1* $-$ *probGn p n* ($\lambda es.$ $\neg$ *H* $\sqsubseteq$ *edge-space.edge-ugraph n es*)) $\longrightarrow$ *1*
using *tendsto-diff*[*OF tendsto-const*] **by** *fastforce*
thus *prob-in-class p* {*G*. *H* $\sqsubseteq$ *G*} $\longrightarrow$ *1*
by (*rule LIMSEQ-cong*[*OF - eventually-sequentiallyI*[*OF facts*(*3*)]])
qed

end

References

[1] R. Diestel. *Graph Theory, 4th Edition*, volume 173 of *Graduate texts in mathematics*. Springer, 2012.

[2] P. Erdős and A. Rényi. On the evolution of random graphs. *Magyar Tud. Akad. Mat. Kutató Int. Közl*, 5:17–61, 1960.

[3] S. Janson, T. Luczak, and A. Rucinski. *Random graphs*. John Wiley & Sons, 2011.

[4] L. Noschinski. A probabilistic proof of the girth-chromatic number theorem. *Archive of Formal Proofs*, Feb. 2012. http://isa-afp.org/entries/Girth_Chromatic.shtml, Formal proof development.