

# A Probabilistic Proof of the Girth-Chromatic Number Theorem

Lars Noschinski

September 13, 2023

## Abstract

This work presents a formalization of the Girth-Chromatic number theorem in graph theory, stating that graphs with arbitrarily large girth and chromatic number exist. The proof uses the theory of Random Graphs to prove the existence with probabilistic arguments and is based on [1].

## Contents

<b>1</b>	<b>Auxiliary lemmas and setup</b>	<b>2</b>
1.1	Numbers . . . . .	2
1.2	Lists and Sets . . . . .	4
1.3	Limits and eventually . . . . .	4
<b>2</b>	<b>Undirected Simple Graphs</b>	<b>5</b>
2.1	Basic Properties . . . . .	6
2.2	Girth, Independence and Vertex Colorings . . . . .	9
<b>3</b>	<b>Probability Space on Sets of Edges</b>	<b>11</b>
3.1	Graph Probabilities outside of <i>Edge-Space</i> locale . . . . .	14
<b>4</b>	<b>Short cycles</b>	<b>15</b>
<b>5</b>	<b>The Chromatic-Girth Theorem</b>	<b>17</b>
	<code>theory Girth-Chromatic-Misc</code>	
	<code>imports</code>	
	<code>Main</code>	
	<code>HOL-Library.Extended-Real</code>	
	<code>begin</code>	

# 1 Auxilliary lemmas and setup

This section contains facts about general concepts which are not directly connected to the proof of the Chromatic-Girth theorem. At some point in time, most of them could be moved to the Isabelle base library.

Also, a little bit of setup happens.

## 1.1 Numbers

**lemma** *enat-in-Inf*:

**fixes**  $S :: \text{enat set}$

**assumes**  $\text{Inf } S \neq \text{top}$

**shows**  $\text{Inf } S \in S$

**proof** (*rule ccontr*)

**assume**  $A: \sim ?thesis$

**obtain**  $n$  **where** *Inf-conv*:  $\text{Inf } S = \text{enat } n$  **using** *assms* **by** (*auto simp: top-enat-def*)

**{ fix**  $s$  **assume**  $s \in S$

**then have**  $\text{Inf } S \leq s$  **by** (*rule complete-lattice-class.Inf-lower*)

**moreover have**  $\text{Inf } S \neq s$  **using**  $A \langle s \in S \rangle$  **by** *auto*

**ultimately have**  $\text{Inf } S < s$  **by** *simp*

**with** *Inf-conv* **have**  $\text{enat } (\text{Suc } n) \leq s$  **by** (*cases s*) *auto*

**}**

**then have**  $\text{enat } (\text{Suc } n) \leq \text{Inf } S$  **by** (*simp add: le-Inf-iff*)

**with** *Inf-conv* **show** *False* **by** *auto*

**qed**

**lemma** *enat-in-INF*:

**fixes**  $f :: 'a \Rightarrow \text{enat}$

**assumes**  $(\text{INF } x \in S. f x) \neq \text{top}$

**obtains**  $x$  **where**  $x \in S$  **and**  $(\text{INF } x \in S. f x) = f x$

**proof** –

**from** *assms* **have**  $(\text{INF } x \in S. f x) \in f ' S$

**using** *enat-in-Inf* [*of f ' S*] **by** *auto*

**then obtain**  $x$  **where**  $x \in S$   $(\text{INF } x \in S. f x) = f x$  **by** *auto*

**then show** *?thesis* ..

**qed**

**lemma** *enat-less-INF-I*:

**fixes**  $f :: 'a \Rightarrow \text{enat}$

**assumes** *not-inf*:  $x \neq \infty$  **and** *less*:  $\bigwedge y. y \in S \implies x < f y$

**shows**  $x < (\text{INF } y \in S. f y)$

**using** *assms* **by** (*auto simp: Suc-ile-eq[symmetric] INF-greatest*)

**lemma** *enat-le-Sup-iff*:

$\text{enat } k \leq \text{Sup } M \iff k = 0 \vee (\exists m \in M. \text{enat } k \leq m)$  (**is**  $?L \iff ?R$ )

**proof** *cases*

**assume**  $k = 0$  **then show** *?thesis* **by** (*auto simp: enat-0*)

```

next
  assume  $k \neq 0$ 
  show ?thesis
  proof
    assume ?L
    then have  $\llbracket \text{enat } k \leq (\text{if finite } M \text{ then Max } M \text{ else } \infty); M \neq \{\} \rrbracket \implies \exists m \in M.$ 
  enat  $k \leq m$ 
    by (metis Max-in Sup-enat-def finite-enat-bounded linorder-linear)
  with  $\langle k \neq 0 \rangle$  and  $\langle ?L \rangle$  show ?R
    unfolding Sup-enat-def
    by (cases M={}) (auto simp add: enat-0[symmetric])
  next
    assume ?R then show ?L
      by (auto simp: enat-0 intro: complete-lattice-class.Sup-upper2)
  qed
qed

```

```

lemma enat-neq-zero-cancel-iff[simp]:
   $0 \neq \text{enat } n \longleftrightarrow 0 \neq n$ 
   $\text{enat } n \neq 0 \longleftrightarrow n \neq 0$ 
  by (auto simp: enat-0[symmetric])

```

```

lemma natceiling-lessD:  $\text{nat}(\text{ceiling } x) < n \implies x < \text{real } n$ 
  by linarith

```

```

lemma le-natceiling-iff:
  fixes  $n :: \text{nat}$  and  $r :: \text{real}$ 
  shows  $n \leq r \implies n \leq \text{nat}(\text{ceiling } r)$ 
  by linarith

```

```

lemma natceiling-le-iff:
  fixes  $n :: \text{nat}$  and  $r :: \text{real}$ 
  shows  $r \leq n \implies \text{nat}(\text{ceiling } r) \leq n$ 
  by linarith

```

```

lemma dist-real-noabs-less:
  fixes  $a b c :: \text{real}$  assumes  $\text{dist } a b < c$  shows  $a - b < c$ 
  using assms by (simp add: dist-real-def)

```

```

lemma n-choose-2-nat:
  fixes  $n :: \text{nat}$  shows  $(n \text{ choose } 2) = (n * (n - 1)) \text{ div } 2$ 
  proof -
    show ?thesis
    proof (cases 2 ≤ n)
      case True
        then obtain  $m$  where  $n = \text{Suc } (\text{Suc } m)$ 
          by (metis add-Suc le-Suc-ex numeral-2-eq-2)
        moreover have  $(n \text{ choose } 2) = (\text{fact } n \text{ div fact } (n - 2)) \text{ div } 2$ 

```

```

    using ⟨2 ≤ n⟩ by (simp add: binomial-altdef-nat
      div-mult2-eq[symmetric] mult commute numeral-2-eq-2)
    ultimately show ?thesis by (simp add: algebra-simps)
  qed (auto simp: binomial-eq-0)
qed

```

```

lemma powr-less-one:
  fixes x::real
  assumes 1 < x y < 0
  shows x powr y < 1
using assms less-log-iff by force

```

```

lemma powr-le-one-le:  $\bigwedge x y :: \text{real}. 0 < x \implies x \leq 1 \implies 1 \leq y \implies x \text{ powr } y \leq x$ 
proof -
  fix x y :: real
  assume 0 < x x ≤ 1 1 ≤ y
  have x powr y = (1 / (1 / x)) powr y using ⟨0 < x⟩ by (simp add: field-simps)
  also have ... = 1 / (1 / x) powr y using ⟨0 < x⟩ by (simp add: powr-divide)
  also have ... ≤ 1 / (1 / x) powr 1 proof -
    have 1 ≤ 1 / x using ⟨0 < x⟩ ⟨x ≤ 1⟩ by (auto simp: field-simps)
    then have (1 / x) powr 1 ≤ (1 / x) powr y using ⟨0 < x⟩
      using ⟨1 ≤ y⟩ by (simp only: powr-mono)
    then show ?thesis
      by (metis ⟨1 ≤ 1 / x⟩ ⟨1 ≤ y⟩ neg-le-iff-le powr-minus-divide powr-mono)
  qed
  also have ... ≤ x using ⟨0 < x⟩ by (auto simp: field-simps)
  finally show ?thesis x y .
qed

```

## 1.2 Lists and Sets

```

lemma list-set-tl:  $x \in \text{set } (\text{tl } xs) \implies x \in \text{set } xs$ 
by (cases xs) auto

```

```

lemma list-exhaust3:
  obtains  $xs = [] \mid x \text{ where } xs = [x] \mid x y \text{ ys where } xs = x \# y \# ys$ 
by (metis list.exhaust)

```

```

lemma card-Ex-subset:
   $k \leq \text{card } M \implies \exists N. N \subseteq M \wedge \text{card } N = k$ 
by (induct rule: inc-induct) (auto simp: card-Suc-eq)

```

## 1.3 Limits and eventually

We employ filters and the *eventually* predicate to deal with the  $\exists N. \forall n \geq N. P n$  cases. To make this more convenient, introduce a shorter syntax.

```

abbreviation evseq :: (nat  $\Rightarrow$  bool)  $\Rightarrow$  bool (binder  $\forall^\infty$  10) where
  evseq P  $\equiv$  eventually P sequentially

```

```

lemma eventually-le-le:
  fixes P :: 'a => ('b :: preorder)
  assumes eventually ( $\lambda x. P x \leq Q x$ ) net
  assumes eventually ( $\lambda x. Q x \leq R x$ ) net
  shows eventually ( $\lambda x. P x \leq R x$ ) net
using assms by eventually-elim (rule order-trans)

lemma LIMSEQ-neg-powr:
  assumes s:  $s < 0$ 
  shows ( $\%x. (\text{real } x) \text{ powr } s$ )  $\longrightarrow 0$ 
by (rule tendsto-neg-powr[OF assms filterlim-real-sequentially])

lemma LIMSEQ-inv-powr:
  assumes  $0 < c < d$ 
  shows ( $\lambda n :: \text{nat}. (c / n) \text{ powr } d$ )  $\longrightarrow 0$ 
proof (rule tendsto-zero-powrI)
  from  $\langle 0 < c \rangle$  have  $\bigwedge x. 0 < x \implies 0 < c / x$  by simp
  then show  $\forall^\infty n. 0 \leq c / \text{real } n$ 
    using assms(1) by auto
  show ( $\lambda x. c / \text{real } x$ )  $\longrightarrow 0$ 
    by (intro tendsto-divide-0[OF tendsto-const] filterlim-at-top-imp-at-infinity
      filterlim-real-sequentially tendsto-divide-0)
  show  $0 < d$  by (rule assms)
  show ( $\lambda x. d$ )  $\longrightarrow d$  by auto
qed

end
theory Ugraphs
imports
  Girth-Chromatic-Misc
begin

```

## 2 Undirected Simple Graphs

In this section, we define some basics of graph theory needed to formalize the Chromatic-Girth theorem.

For readability, we introduce synonyms for the types of vertexes, edges, graphs and walks.

```

type-synonym uvert = nat
type-synonym uedge = nat set
type-synonym ugraph = uvert set  $\times$  uedge set
type-synonym uwalk = uvert list

```

```

abbreviation uedges :: ugraph  $\Rightarrow$  uedge set where
  uedges G  $\equiv$  snd G

```

**abbreviation**  $uverts :: ugraph \Rightarrow uvert\ set$  **where**  
 $uverts\ G \equiv fst\ G$

**fun**  $mk-uedge :: uvert \times uvert \Rightarrow uedge$  **where**  
 $mk-uedge\ (u,v) = \{u,v\}$

All edges over a set of vertexes  $S$ :

**definition**  $all-edges\ S \equiv mk-uedge\ \{uv \in S \times S.\ fst\ uv \neq\ snd\ uv\}$

**definition**  $uwellformed :: ugraph \Rightarrow bool$  **where**  
 $uwellformed\ G \equiv (\forall e \in uedges\ G.\ card\ e = 2 \wedge (\forall u \in e.\ u \in uverts\ G))$

**fun**  $uwalk-edges :: uwalk \Rightarrow uedge\ list$  **where**  
 $uwalk-edges\ [] = []$   
 $| uwalk-edges\ [x] = []$   
 $| uwalk-edges\ (x \# y \# ys) = \{x,y\} \# uwalk-edges\ (y \# ys)$

**definition**  $uwalk-length :: uwalk \Rightarrow nat$  **where**  
 $uwalk-length\ p \equiv length\ (uwalk-edges\ p)$

**definition**  $uwalks :: ugraph \Rightarrow uwalk\ set$  **where**  
 $uwalks\ G \equiv \{p.\ set\ p \subseteq uverts\ G \wedge set\ (uwalk-edges\ p) \subseteq uedges\ G \wedge p \neq []\}$

**definition**  $ucycles :: ugraph \Rightarrow uwalk\ set$  **where**  
 $ucycles\ G \equiv \{p.\ uwalk-length\ p \geq 3 \wedge p \in uwalks\ G \wedge distinct\ (tl\ p) \wedge hd\ p = last\ p\}$

**definition**  $remove-vertex :: ugraph \Rightarrow nat \Rightarrow ugraph\ (-\ \dots\ -\ [60,60]\ 60)$  **where**  
 $remove-vertex\ G\ u \equiv (uverts\ G - \{u\}, uedges\ G - \{A \in uedges\ G.\ u \in A\})$

## 2.1 Basic Properties

**lemma**  $uwalk-length-conv: uwalk-length\ p = length\ p - 1$   
**by** ( $induct\ p\ rule: uwalk-edges.induct$ ) ( $auto\ simp: uwalk-length-def$ )

**lemma**  $all-edges-mono:$   
 $vs \subseteq ws \implies all-edges\ vs \subseteq all-edges\ ws$   
**unfolding**  $all-edges-def$  **by**  $auto$

**lemma**  $all-edges-subset-Pow: all-edges\ A \subseteq Pow\ A$   
**by** ( $auto\ simp: all-edges-def$ )

**lemma**  $in-mk-uedge-img: (a,b) \in A \vee (b,a) \in A \implies \{a,b\} \in mk-uedge\ \{A$   
**by** ( $auto\ intro: rev-image-eqI$ )

**lemma**  $in-mk-uedge-img-iff: \{a,b\} \in mk-uedge\ \{A \longleftrightarrow (a,b) \in A \vee (b,a) \in A$   
**by** ( $auto\ simp: doubleton-eq-iff\ intro: rev-image-eqI$ )

**lemma**  $distinct-edgesI:$

**assumes** *distinct p* **shows** *distinct (uwalk-edges p)*  
**proof** –  
**from** *assms* **have** *?thesis  $\wedge u. u \notin \text{set } p \implies (\wedge v. u \neq v \implies \{u,v\} \notin \text{set (uwalk-edges p)}$*   
**by** (*induct p rule: uwalk-edges.induct*) *auto*  
**then show** *?thesis* **by** *simp*  
**qed**

**lemma** *finite-ucycles:*

**assumes** *finite (uverts G)*  
**shows** *finite (ucycles G)*  
**proof** –  
**have** *ucycles G  $\subseteq$  {xs. set xs  $\subseteq$  uverts G  $\wedge$  length xs  $\leq$  Suc (card (uverts G))}*  
**proof** (*rule, simp*)  
**fix** *p* **assume** *p  $\in$  ucycles G*  
**then have** *distinct (tl p)* **and** *set p  $\subseteq$  uverts G*  
**unfolding** *ucycles-def uwalks-def* **by** *auto*  
**moreover**  
**then have** *set (tl p)  $\subseteq$  uverts G*  
**by** (*auto simp: list-set-tl*)  
**with** *assms* **have** *card (set (tl p))  $\leq$  card (uverts G)*  
**by** (*rule card-mono*)  
**then have** *length (p)  $\leq$  1 + card (uverts G)*  
**using** *distinct-card[OF  $\langle$ distinct (tl p) $\rangle$ ]* **by** *auto*  
**ultimately show** *set p  $\subseteq$  uverts G  $\wedge$  length p  $\leq$  Suc (card (uverts G))* **by** *auto*  
**qed**  
**moreover**  
**have** *finite {xs. set xs  $\subseteq$  uverts G  $\wedge$  length xs  $\leq$  Suc (card (uverts G))}*  
**using** *assms* **by** (*rule finite-lists-length-le*)  
**ultimately**  
**show** *?thesis* **by** (*rule finite-subset*)  
**qed**

**lemma** *ucycles-distinct-edges:*

**assumes** *c  $\in$  ucycles G* **shows** *distinct (uwalk-edges c)*  
**proof** –  
**from** *assms* **have** *c-props: distinct (tl c)  $4 \leq$  length c  $\wedge$  hd c = last c*  
**by** (*auto simp add: ucycles-def uwalk-length-conv*)  
**then have** *{hd c, hd (tl c)}  $\notin$  set (uwalk-edges (tl c))*  
**proof** (*induct c rule: uwalk-edges.induct*)  
**case** ( $\exists x y ys$ )  
**then have** *hd ys  $\neq$  last ys* **by** (*cases ys*) *auto*  
**moreover**  
**from**  $\exists$  **have** *uwalk-edges (y # ys) = {y, hd ys} # uwalk-edges ys*  
**by** (*cases ys*) *auto*  
**moreover**  
**{ fix** *xs* **have** *set (uwalk-edges xs)  $\subseteq$  Pow (set xs)*  
**by** (*induct xs rule: uwalk-edges.induct*) *auto* **}**  
**ultimately**

```

    show ?case using 3 by auto
qed simp-all
moreover
from assms have distinct (uwalk-edges (tl c))
  by (intro distinct-edgesI) (simp add: ucycles-def)
ultimately
show ?thesis by (cases c rule: list-exhaust3) auto
qed

```

**lemma** *card-left-less-pair*:

```

fixes A :: ('a :: linorder) set
assumes finite A
shows card {(a,b). a ∈ A ∧ b ∈ A ∧ a < b}
  = (card A * (card A - 1)) div 2

```

using *assms*

**proof** (*induct A*)

case (*insert x A*)

show ?case

**proof** (*cases card A*)

case (*Suc n*)

have {(*a,b*). *a* ∈ *insert x A* ∧ *b* ∈ *insert x A* ∧ *a* < *b*}

= {(*a,b*). *a* ∈ *A* ∧ *b* ∈ *A* ∧ *a* < *b*} ∪ (λ*a*. if *a* < *x* then (*a,x*) else (*x,a*)) ‘ *A*

using ⟨*x* ∉ *A*⟩ by (*auto simp: order-less-le*)

moreover

have *finite* {(*a,b*). *a* ∈ *A* ∧ *b* ∈ *A* ∧ *a* < *b*}

using *insert* by (*auto intro: finite-subset[of - A × A]*)

moreover

have {(*a,b*). *a* ∈ *A* ∧ *b* ∈ *A* ∧ *a* < *b*} ∩ (λ*a*. if *a* < *x* then (*a,x*) else (*x,a*)) ‘

*A* = {}

using ⟨*x* ∉ *A*⟩ by *auto*

moreover have *inj-on* (λ*a*. if *a* < *x* then (*a, x*) else (*x, a*)) *A*

by (*auto intro: inj-onI split: if-split-asm*)

ultimately show ?thesis using *insert Suc*

by (*simp add: card-Un-disjoint card-image del: if-image-distrib*)

qed (*simp add: card-eq-0-iff insert*)

qed *simp*

**lemma** *card-all-edges*:

assumes *finite A*

shows *card* (*all-edges A*) = *card A* choose 2

**proof** –

have *inj-on-mk-uedge: inj-on mk-uedge* {(*a,b*). *a* < *b*}

by (*rule inj-onI*) (*auto simp: doubleton-eq-iff*)

have *all-edges A* = *mk-uedge* ‘ {(*a,b*). *a* ∈ *A* ∧ *b* ∈ *A* ∧ *a* < *b*} (**is** ?*L* = ?*R*)

by (*auto simp: all-edges-def intro!: in-mk-uedge-img*)

then have *card* ?*L* = *card* ?*R* by *simp*

also have ... = *card* {(*a,b*). *a* ∈ *A* ∧ *b* ∈ *A* ∧ *a* < *b*}

using *inj-on-mk-uedge* by (*blast intro: card-image subset-inj-on*)



**also have**  $\dots = (\text{card } A * (\text{card } A - 1)) \text{ div } 2$   
**using** *card-left-less-pair* **using** *assms* **by** *simp*  
**also have**  $\dots = (\text{card } A \text{ choose } 2)$   
**by** (*simp add: n-choose-2-nat*)  
**finally show** *?thesis* .  
**qed**

**lemma** *verts-Gu*:  $\text{uverts } (G -- u) = \text{uverts } G - \{u\}$   
**unfolding** *remove-vertex-def* **by** *simp*

**lemma** *edges-Gu*:  $\text{uedges } (G -- u) \subseteq \text{uedges } G$   
**unfolding** *remove-vertex-def* **by** *auto*

## 2.2 Girth, Independence and Vertex Colorings

**definition** *girth* :: *ugraph*  $\Rightarrow$  *enat* **where**  
*girth*  $G \equiv \text{INF } p \in \text{ucycles } G. \text{enat } (\text{uwalk-length } p)$

**definition** *independent-sets* :: *ugraph*  $\Rightarrow$  *uvert set set* **where**  
*independent-sets*  $Gr \equiv \{vs. vs \subseteq \text{uverts } Gr \wedge \text{all-edges } vs \cap \text{uedges } Gr = \{\}\}$

**definition**  $\alpha$  :: *ugraph*  $\Rightarrow$  *enat* **where**  
 $\alpha$   $G \equiv \text{SUP } vs \in \text{independent-sets } G. \text{enat } (\text{card } vs)$

**definition** *vertex-colorings* :: *ugraph*  $\Rightarrow$  *uvert set set set* **where**  
*vertex-colorings*  $G \equiv \{C. \bigcup C = \text{uverts } G \wedge (\forall c1 \in C. \forall c2 \in C. c1 \neq c2 \longrightarrow c1 \cap c2 = \{\}) \wedge$   
 $(\forall c \in C. c \neq \{\}) \wedge (\forall u \in c. \forall v \in c. \{u, v\} \notin \text{uedges } G)\}$

The chromatic number  $\chi$ :

**definition** *chromatic-number* :: *ugraph*  $\Rightarrow$  *enat* **where**  
*chromatic-number*  $G \equiv \text{INF } c \in (\text{vertex-colorings } G). \text{enat } (\text{card } c)$

**lemma** *independent-sets-mono*:  
 $vs \in \text{independent-sets } G \Longrightarrow us \subseteq vs \Longrightarrow us \in \text{independent-sets } G$   
**using** *Int-mono[OF all-edges-mono, of us vs uedges G uedges G]*  
**unfolding** *independent-sets-def* **by** *auto*

**lemma** *le- $\alpha$ -iff*:  
**assumes**  $0 < k$   
**shows**  $k \leq \alpha$   $Gr \iff k \in \text{card } \text{'independent-sets } Gr$  (**is** *?L*  $\iff$  *?R*)  
**proof**

**assume** *?L*  
**then obtain** *vs* **where**  $vs \in \text{independent-sets } Gr$  **and**  $k \leq \text{card } vs$   
**using** *assms* **unfolding**  $\alpha$ -*def* *enat-le-Sup-iff* **by** *auto*  
**moreover**  
**then obtain** *us* **where**  $us \subseteq vs$  **and**  $k = \text{card } us$   
**using** *card-Ex-subset* **by** *auto*  
**ultimately**

**have**  $us \in \text{independent-sets } Gr$  **by** (*auto intro: independent-sets-mono*)  
**then show**  $?R$  **using**  $\langle k = \text{card } us \rangle$  **by** *auto*  
**qed** (*auto intro: SUP-upper simp:  $\alpha$ -def*)

**lemma** *zero-less- $\alpha$* :

**assumes**  $\text{uverts } G \neq \{\}$   
**shows**  $0 < \alpha G$

**proof** –

**from** *assms* **obtain**  $a$  **where**  $a \in \text{uverts } G$  **by** *auto*  
**then have**  $0 < \text{enat } (\text{card } \{a\})$   $\{a\} \in \text{independent-sets } G$   
**by** (*auto simp: independent-sets-def all-edges-def*)  
**then show** *?thesis unfolding  $\alpha$ -def less-SUP-iff ..*

**qed**

**lemma**  *$\alpha$ -le-card*:

**assumes** *finite* ( $\text{uverts } G$ )  
**shows**  $\alpha G \leq \text{card}(\text{uverts } G)$

**proof** –

{ **fix**  $x$  **assume**  $x \in \text{independent-sets } G$   
**then have**  $x \subseteq \text{uverts } G$  **by** (*auto simp: independent-sets-def*) }  
**with** *assms* **show** *?thesis unfolding  $\alpha$ -def*  
**by** (*intro SUP-least*) (*auto intro: card-mono*)

**qed**

**lemma**  *$\alpha$ -fin*:  $\text{finite } (\text{uverts } G) \implies \alpha G \neq \infty$

**using**  *$\alpha$ -le-card*[of  $G$ ] **by** (*cases  $\alpha G$* ) *auto*

**lemma**  *$\alpha$ -remove-le*:

**shows**  $\alpha (G -- u) \leq \alpha G$

**proof** –

**have**  $\text{independent-sets } (G -- u) \subseteq \text{independent-sets } G$  (**is**  $?L \subseteq ?R$ )  
**using** *all-edges-subset-Pow* **by** (*simp add: independent-sets-def remove-vertex-def*)

*blast*

**then show** *?thesis unfolding  $\alpha$ -def*  
**by** (*rule SUP-subset-mono*) *simp*

**qed**

A lower bound for the chromatic number of a graph can be given in terms of the independence number

**lemma** *chromatic-lb*:

**assumes** *wf-G*: *uwellformed*  $G$   
**and** *fin-G*: *finite* ( $\text{uverts } G$ )  
**and** *neG*:  $\text{uverts } G \neq \{\}$   
**shows**  $\text{card } (\text{uverts } G) / \alpha G \leq \text{chromatic-number } G$

**proof** –

**from** *wf-G* **have**  $(\lambda v. \{v\}) \text{ ` } \text{uverts } G \in \text{vertex-colorings } G$   
**by** (*auto simp: vertex-colorings-def uwellformed-def*)

**then have**  $\text{chromatic-number } G \neq \text{top}$   
**by** (*simp add: chromatic-number-def*) (*auto simp: top-enat-def*)

```

then obtain vc where vc-vc:  $vc \in \text{vertex-colorings } G$ 
  and vc-size:  $\text{chromatic-number } G = \text{card } vc$ 
  unfolding chromatic-number-def by (rule enat-in-INF)

have fin-vc-elems:  $\bigwedge c. c \in vc \implies \text{finite } c$ 
  using vc-vc by (intro finite-subset[OF - fin-G]) (auto simp: vertex-colorings-def)

have sum-vc-card:  $(\sum c \in vc. \text{card } c) = \text{card } (\text{uverts } G)$ 
  using fin-vc-elems vc-vc unfolding vertex-colorings-def
  by (simp add: card-Union-disjoint[symmetric] pairwise-def disjnt-def)

have  $\bigwedge c. c \in vc \implies c \in \text{independent-sets } G$ 
  using vc-vc by (auto simp: vertex-colorings-def independent-sets-def all-edges-def)
then have  $\bigwedge c. c \in vc \implies \text{card } c \leq \alpha G$ 
  using vc-vc fin-vc-elems by (subst le- $\alpha$ -iff) (auto simp add: vertex-colorings-def)
then have  $(\sum c \in vc. \text{card } c) \leq \text{card } vc * \alpha G$ 
  using sum-bounded-above[of vc card  $\alpha G$ ]
  by (simp add: of-nat-eq-enat[symmetric] of-nat-sum)
then have  $\text{ereal-of-enat } (\text{card } (\text{uverts } G)) \leq \text{ereal-of-enat } (\alpha G) * \text{ereal-of-enat } (\text{card } vc)$ 
  by (simp add: sum-vc-card eréal-of-enat-pushout ac-simps del: eréal-of-enat-simps)
with zero-less- $\alpha$ [OF neG]  $\alpha$ -fin[OF fin-G] vc-size show ?thesis
  by (simp add: eréal-divide-le-pos)
qed

end
theory Girth-Chromatic
imports
  Ugraphs
  Girth-Chromatic-Misc
  HOL-Probability.Probability
  HOL-Decision-Procs.Approximation
begin

```

### 3 Probability Space on Sets of Edges

**definition** *cylinder* ::  $'a \text{ set} \Rightarrow 'a \text{ set} \Rightarrow 'a \text{ set} \Rightarrow 'a \text{ set set}$  **where**  
*cylinder*  $S A B = \{T \in \text{Pow } S. A \subseteq T \wedge B \cap T = \{\}\}$

**lemma** *full-sum*:

```

fixes p :: real
assumes finite S
shows  $(\sum A \in \text{Pow } S. p^{\text{card } A} * (1 - p)^{\text{card } (S - A)}) = 1$ 
using assms
proof induct
  case (insert s S)
  have inj-on (insert s) (Pow S)
    and  $\bigwedge x. S - \text{insert } s x = S - x$ 
    and  $\text{Pow } S \cap \text{insert } s ` \text{Pow } S = \{\}$ 

```

**and**  $\bigwedge x. x \in \text{Pow } S \implies \text{card } (\text{insert } s \ S - x) = \text{Suc } (\text{card } (S - x))$   
**using** *insert(1-2)* **by** (*auto simp: insert-Diff-if intro!: inj-onI*)  
**moreover have**  $\bigwedge x. x \subseteq S \implies \text{card } (\text{insert } s \ x) = \text{Suc } (\text{card } x)$   
**using** *insert(1-2)* **by** (*subst card.insert*) (*auto dest: finite-subset*)  
**ultimately show** *?case*  
**by** (*simp add: sum.reindex sum-distrib-left[symmetric] ac-simps*  
*insert.hyps sum.union-disjoint Pow-insert*)

**qed** *simp*

Definition of the probability space on edges:

**locale** *edge-space* =  
**fixes**  $n :: \text{nat}$  **and**  $p :: \text{real}$   
**assumes** *p-prob*:  $0 \leq p \leq 1$   
**begin**

**definition** *S-verts* :: *nat set* **where**  
*S-verts*  $\equiv \{1..n\}$

**definition** *S-edges* :: *uedge set* **where**  
*S-edges* = *all-edges S-verts*

**definition** *edge-ugraph* :: *uedge set*  $\Rightarrow$  *ugraph* **where**  
*edge-ugraph es*  $\equiv (S\text{-verts}, es \cap S\text{-edges})$

**definition** *P* = *point-measure (Pow S-edges)* ( $\lambda s. p^{\text{card } s} * (1 - p)^{\text{card } (S\text{-edges} - s)}$ )

**lemma** *finite-verts[intro!]*: *finite S-verts*  
**by** (*auto simp: S-verts-def*)

**lemma** *finite-edges[intro!]*: *finite S-edges*  
**by** (*auto simp: S-edges-def all-edges-def finite-verts*)

**lemma** *finite-graph[intro!]*: *finite (uverts (edge-ugraph es))*  
**unfolding** *edge-ugraph-def* **by** *auto*

**lemma** *uverts-edge-ugraph[simp]*: *uverts (edge-ugraph es) = S-verts*  
**by** (*simp add: edge-ugraph-def*)

**lemma** *uedges-edge-ugraph[simp]*: *uedges (edge-ugraph es) = es  $\cap$  S-edges*  
**unfolding** *edge-ugraph-def* **by** *simp*

**lemma** *space-eq*: *space P = Pow S-edges* **by** (*simp add: P-def space-point-measure*)

**lemma** *sets-eq*: *sets P = Pow (Pow S-edges)* **by** (*simp add: P-def sets-point-measure*)

**lemma** *emeasure-eq*:  
*emeasure P A = (if A  $\subseteq$  Pow S-edges then  $(\sum \text{edges} \in A. p^{\text{card } \text{edges}} * (1 - p)^{\text{card } (S\text{-edges} - \text{edges})})$  else 0)*

```

using finite-edges p-prob
by (simp add: P-def space-point-measure emeasure-point-measure-finite
sets-point-measure emeasure-notin-sets)

lemma integrable-P[intro, simp]: integrable P (f::-  $\Rightarrow$  real)
using finite-edges by (simp add: integrable-point-measure-finite P-def)

lemma borel-measurable-P[measurable]: f  $\in$  borel-measurable P
unfolding P-def by simp

lemma prob-space-P: prob-space P
proof
show emeasure P (space P) = 1 — Sum of probabilities equals 1
using finite-edges by (simp add: emeasure-eq full-sum one-ereal-def space-eq)
qed

end

sublocale edge-space  $\subseteq$  prob-space P
by (rule prob-space-P)

context edge-space
begin

lemma prob-eq:
prob A = (if A  $\subseteq$  Pow S-edges then ( $\sum$  edges $\in$ A. pcard edges * (1 - p)card
(S-edges - edges)) else 0)
using emeasure-eq[of A] p-prob unfolding emeasure-eq-measure by (simp add:
sum-nonneg)

lemma integral-finite-singleton: integralL P f = ( $\sum$  x $\in$  Pow S-edges. f x * measure
P {x})
using p-prob prob-eq unfolding P-def
by (subst lebesgue-integral-point-measure-finite) (auto intro!: sum.cong)

Probability of cylinder sets:

lemma cylinder-prob:
assumes A  $\subseteq$  S-edges B  $\subseteq$  S-edges A  $\cap$  B = {}
shows prob (cylinder S-edges A B) = pcard A * (1 - p)card B (is - =
?pp A B)
proof -
have Pow S-edges  $\cap$  cylinder S-edges A B = cylinder S-edges A B
 $\wedge$  x. x  $\in$  cylinder S-edges A B  $\implies$  A  $\cup$  x = x
 $\wedge$  x. x  $\in$  cylinder S-edges A B  $\implies$  finite x
 $\wedge$  x. x  $\in$  cylinder S-edges A B  $\implies$  B  $\cap$  (S-edges - B - x) = {}
 $\wedge$  x. x  $\in$  cylinder S-edges A B  $\implies$  B  $\cup$  (S-edges - B - x) = S-edges - x
finite A finite B
using assms by (auto simp add: cylinder-def intro: finite-subset)
then have ( $\sum$  T $\in$  cylinder S-edges A B. ?pp T (S-edges - T))

```

$= (\sum T \in \text{cylinder } S\text{-edges } A \ B. p \wedge (\text{card } A + \text{card } (T - A)) * (1 - p) \wedge (\text{card } B + \text{card } ((S\text{-edges} - B) - T)))$   
**using** *finite-edges* **by** (*simp add: card-Un-Int*)  
**also have**  $\dots = ?pp \ A \ B * (\sum T \in \text{cylinder } S\text{-edges } A \ B. ?pp \ (T - A) \ (S\text{-edges} - B - T))$   
**by** (*simp add: power-add sum-distrib-left ac-simps*)  
**also have**  $\dots = ?pp \ A \ B$   
**proof** –  
**have**  $\bigwedge T. T \in \text{cylinder } S\text{-edges } A \ B \implies S\text{-edges} - B - T = (S\text{-edges} - A) - B - (T - A)$   
 $Pow \ (S\text{-edges} - A - B) = (\lambda x. x - A) \text{ `cylinder } S\text{-edges } A \ B$   
 $\text{inj-on } (\lambda x. x - A) \ (\text{cylinder } S\text{-edges } A \ B)$   
 $\text{finite } (S\text{-edges} - A - B)$   
**using** *assms* **by** (*auto simp: cylinder-def intro!: inj-onI*)  
**with** *full-sum*[of  $S\text{-edges} - A - B$ ] **show** *?thesis* **by** (*simp add: sum.reindex*)  
**qed**  
**finally show** *?thesis* **by** (*auto simp add: prob-eq cylinder-def*)  
**qed**

**lemma** *Markov-inequality*:

**fixes**  $a :: \text{real}$  **and**  $X :: \text{uedge set} \Rightarrow \text{real}$   
**assumes**  $0 < c \wedge x. 0 \leq f \ x$   
**shows**  $\text{prob } \{x \in \text{space } P. c \leq f \ x\} \leq (\int x. f \ x \ \partial P) / c$   
**proof** –  
**from** *assms* **have**  $(\int^+ x. \text{ennreal } (f \ x) \ \partial P) = (\int x. f \ x \ \partial P)$   
**by** (*intro nn-integral-eq-integral auto*)  
**with** *assms* **show** *?thesis*  
**using** *nn-integral-Markov-inequality*[of  $f \ \text{space } P \ P \ 1 / c$ ]  
**by** (*simp cong: nn-integral-cong add: emeasure-eq-measure ennreal-mult[symmetric]*)  
**qed**

**end**

### 3.1 Graph Probabilities outside of *Edge-Space* locale

These abbreviations allow a compact expression of probabilities about random graphs outside of the *Edge-Space* locale. We also transfer a few of the lemmas we need from the locale into the toplevel theory.

**abbreviation**  $MGn :: (\text{nat} \Rightarrow \text{real}) \Rightarrow \text{nat} \Rightarrow (\text{uedge set}) \text{ measure}$  **where**

$MGn \ p \ n \equiv (\text{edge-space}.P \ n \ (p \ n))$

**abbreviation**  $\text{probGn} :: (\text{nat} \Rightarrow \text{real}) \Rightarrow \text{nat} \Rightarrow (\text{uedge set} \Rightarrow \text{bool}) \Rightarrow \text{real}$  **where**

$\text{probGn} \ p \ n \ P \equiv \text{measure } (MGn \ p \ n) \ \{es \in \text{space } (MGn \ p \ n). P \ es\}$

**lemma** *probGn-le*:

**assumes** *p-prob*:  $0 < p \ n \ p \ n < 1$

**assumes** *sub*:  $\bigwedge n \ es. es \in \text{space } (MGn \ p \ n) \implies P \ n \ es \implies Q \ n \ es$

**shows**  $\text{probGn} \ p \ n \ (P \ n) \leq \text{probGn} \ p \ n \ (Q \ n)$

**proof** –

**from** *p-prob* **interpret**  $E: \text{edge-space } n \ p \ n$  **by** *unfold-locales auto*

**show** *?thesis*  
**by** (*auto intro!*: *E.finite-measure-mono sub simp: E.space-eq E.sets-eq*)  
**qed**

## 4 Short cycles

**definition** *short-cycles* :: *ugraph*  $\Rightarrow$  *nat*  $\Rightarrow$  *uwalk set* **where**  
*short-cycles* *G k*  $\equiv$   $\{p \in \text{ucycles } G. \text{uwalk-length } p \leq k\}$

obtains a vertex in a short cycle:

**definition** *choose-v* :: *ugraph*  $\Rightarrow$  *nat*  $\Rightarrow$  *uvert* **where**  
*choose-v* *G k*  $\equiv$  *SOME* *u. \exists p. p \in short-cycles G k \wedge u \in set p*

**partial-function** (*tailrec*) *kill-short* :: *ugraph*  $\Rightarrow$  *nat*  $\Rightarrow$  *ugraph* **where**  
*kill-short* *G k* = (*if short-cycles G k = {} then G else (kill-short (G -- (choose-v G k)) k)*)

**lemma** *ksc-simps[simp]*:  
*short-cycles G k = {}  $\implies$  kill-short G k = G*  
*short-cycles G k  $\neq$  {}  $\implies$  kill-short G k = kill-short (G -- (choose-v G k)) k*  
**by** (*auto simp: kill-short.simps*)

**lemma**  
**assumes** *short-cycles G k  $\neq$  {}*  
**shows** *choose-v--in-uverts: choose-v G k  $\in$  uverts G (is ?t1)*  
**and** *choose-v--in-short: \exists p. p \in short-cycles G k \wedge choose-v G k \in set p (is ?t2)*

**proof** –

**from** *assms obtain p where p  $\in$  ucycles G uwalk-length p  $\leq$  k*  
**unfolding** *short-cycles-def* **by** *auto*

**moreover**

**then obtain u where u  $\in$  set p unfolding ucycles-def**  
**by** (*cases p*) (*auto simp: uwalk-length-conv*)

**ultimately have**  $\exists u p. p \in \text{short-cycles } G k \wedge u \in \text{set } p$   
**by** (*auto simp: short-cycles-def*)

**then show** *?t2* **by** (*auto simp: choose-v-def intro!: someI-ex*)

**then show** *?t1* **by** (*auto simp: short-cycles-def ucycles-def uwalks-def*)

**qed**

**lemma** *kill-step-smaller*:  
**assumes** *short-cycles G k  $\neq$  {}*  
**shows** *short-cycles (G -- (choose-v G k)) k  $\subset$  short-cycles G k*

**proof** –

**let** *?cv = choose-v G k*

**from** *assms obtain p where p  $\in$  short-cycles G k ?cv  $\in$  set p*  
**by** *atomize-elim (rule choose-v--in-short)*

**have** *short-cycles (G -- ?cv) k  $\subseteq$  short-cycles G k*

**proof**

```

fix  $p$  assume  $p \in \text{short-cycles } (G \dashv\vdash ?cv) k$ 
then show  $p \in \text{short-cycles } G k$ 
  unfolding  $\text{short-cycles-def } \text{ucycles-def } \text{uwalks-def}$ 
  using  $\text{edges-Gu}[of\ G\ ?cv]$  by  $(\text{auto simp: } \text{verts-Gu})$ 
qed
moreover have  $p \notin \text{short-cycles } (G \dashv\vdash ?cv) k$ 
  using  $\langle ?cv \in \text{set } p \rangle$  by  $(\text{auto simp: } \text{short-cycles-def } \text{ucycles-def } \text{uwalks-def } \text{verts-Gu})$ 
ultimately show  $?thesis$  using  $\langle p \in \text{short-cycles } G k \rangle$  by  $\text{auto}$ 
qed

```

Induction rule for *kill-short*:

```

lemma  $\text{kill-short-induct}[\text{consumes } 1, \text{case-names empty kill-vert}]$ :
  assumes  $\text{fin: finite } (uverts\ G)$ 
  assumes  $a\text{-empty: } \bigwedge G. \text{short-cycles } G\ k = \{\} \implies P\ G\ k$ 
  assumes  $a\text{-kill: } \bigwedge G. \text{finite } (\text{short-cycles } G\ k) \implies \text{short-cycles } G\ k \neq \{\}$ 
     $\implies P\ (G \dashv\vdash (\text{choose-v } G\ k))\ k \implies P\ G\ k$ 
  shows  $P\ G\ k$ 
proof  $-$ 
  have  $\text{finite } (\text{short-cycles } G\ k)$ 
    using  $\text{finite-ucycles}[OF\ \text{fin}]$  by  $(\text{auto simp: } \text{short-cycles-def})$ 
  then show  $?thesis$ 
    by  $(\text{induct } \text{short-cycles } G\ k\ \text{arbitrary: } G\ \text{rule: } \text{finite-psubset-induct})$ 
     $(\text{metis } \text{kill-step-smaller } a\text{-kill } a\text{-empty})$ 
qed

```

Large Girth (after *kill-short*):

```

lemma  $\text{kill-short-large-girth}$ :
  assumes  $\text{finite } (uverts\ G)$ 
  shows  $k < \text{girth } (\text{kill-short } G\ k)$ 
using  $\text{assms}$ 
proof  $(\text{induct } G\ k\ \text{rule: } \text{kill-short-induct})$ 
  case  $(\text{empty } G)$ 
  then have  $\bigwedge p. p \in \text{ucycles } G \implies k < \text{enat } (uwalk\text{-length } p)$ 
    by  $(\text{auto simp: } \text{short-cycles-def})$ 
  with empty show  $?case$  by  $(\text{auto simp: } \text{girth-def intro: } \text{enat-less-INF-I})$ 
qed  $\text{simp}$ 

```

Order of graph (after *kill-short*):

```

lemma  $\text{kill-short-order-of-graph}$ :
  assumes  $\text{finite } (uverts\ G)$ 
  shows  $\text{card } (uverts\ G) - \text{card } (\text{short-cycles } G\ k) \leq \text{card } (uverts\ (\text{kill-short } G\ k))$ 
using  $\text{assms } \text{assms}$ 
proof  $(\text{induct } G\ k\ \text{rule: } \text{kill-short-induct})$ 
  case  $(\text{kill-vert } G)$ 
  let  $?oG = G \dashv\vdash (\text{choose-v } G\ k)$ 
  have  $\text{finite } (uverts\ ?oG)$ 
    using  $\text{kill-vert}$  by  $(\text{auto simp: } \text{remove-vertex-def})$ 

```



```

moreover
have  $uverts (kill-short\ G\ k) = uverts (kill-short\ ?oG\ k)$ 
  using kill-vert by simp
moreover
have  $card (uverts\ G) = Suc (card (uverts\ ?oG))$ 
  using choose-v-in-uverts kill-vert
  by (simp add: remove-vertex-def card-Suc-Diff1 del: card-Diff-insert)
moreover
have  $card (short-cycles\ ?oG\ k) < card (short-cycles\ G\ k)$ 
  by (intro psubset-card-mono kill-vert.hyps kill-step-smaller)
ultimately show ?case using kill-vert.hyps by presburger
qed simp

```

Independence number (after *kill-short*):

```

lemma kill-short- $\alpha$ :
  assumes finite (uverts G)
  shows  $\alpha (kill-short\ G\ k) \leq \alpha\ G$ 
using assms
proof (induct G k rule: kill-short-induct)
  case (kill-vert G)
  note kill-vert(3)
  also have  $\alpha (G \text{ -- } (choose-v\ G\ k)) \leq \alpha\ G$  by (rule  $\alpha$ -remove-le)
  finally show ?case using kill-vert by simp
qed simp

```

Wellformedness (after *kill-short*):

```

lemma kill-short-uwellformed:
  assumes finite (uverts G) uwellformed G
  shows uwellformed (kill-short G k)
using assms
proof (induct G k rule: kill-short-induct)
  case (kill-vert G)
  from kill-vert.prems have uwellformed (G -- (choose-v G k))
    by (auto simp: uwellformed-def remove-vertex-def)
  with kill-vert.hyps show ?case by simp
qed simp

```

## 5 The Chromatic-Girth Theorem

Probability of Independent Edges:

```

lemma (in edge-space) random-prob-independent:
  assumes  $n \geq k\ k \geq 2$ 
  shows  $prob \{es \in space\ P. k \leq \alpha (edge-ugraph\ es)\}$ 
     $\leq (n\ choose\ k) * (1-p) \wedge (k\ choose\ 2)$ 
proof –
  let ?k-sets =  $\{vs. vs \subseteq S\text{-verts} \wedge card\ vs = k\}$ 

  { fix vs assume A: vs ∈ ?k-sets

```

```

then have B: all-edges vs  $\subseteq$  S-edges
  unfolding all-edges-def S-edges-def by blast

have {es  $\in$  space P. vs  $\in$  independent-sets (edge-ugraph es)}
  = cylinder S-edges {} (all-edges vs) (is ?L = -)
  using A by (auto simp: independent-sets-def edge-ugraph-def space-eq cylinder-def)
then have prob ?L =  $(1-p) \wedge^k$  choose 2)
  using A B finite by (auto simp: cylinder-prob card-all-edges dest: finite-subset)
}
note prob-k-indep = this
  — probability that a fixed set of k vertices is independent in a random graph

have {es  $\in$  space P. k  $\in$  card 'independent-sets (edge-ugraph es)}
  = ( $\bigcup$  vs  $\in$  ?k-sets. {es  $\in$  space P. vs  $\in$  independent-sets (edge-ugraph es)}) (is
  ?L = ?R)
  unfolding image-def space-eq independent-sets-def by auto
then have prob ?L  $\leq$  ( $\sum$  vs  $\in$  ?k-sets. prob {es  $\in$  space P. vs  $\in$  independent-sets
  (edge-ugraph es})})
  by (auto intro!: finite-measure-subadditive-finite simp: space-eq sets-eq)
also have ... =  $(n \text{ choose } k) * ((1 - p) \wedge^k)$  choose 2)
  by (simp add: prob-k-indep S-verts-def n-subsets)
finally show ?thesis using  $\langle k \geq 2 \rangle$  by (simp add: le- $\alpha$ -iff)
qed

```

Almost never many independent edges:

**lemma** *almost-never-le- $\alpha$* :

```

fixes k :: nat
  and p :: nat  $\Rightarrow$  real
assumes p-prob:  $\forall^\infty n. 0 < p \ n \wedge p \ n < 1$ 
assumes [arith]: k > 0
assumes N-prop:  $\forall^\infty n. (6 * k * \ln n) / n \leq p \ n$ 
shows ( $\lambda n. \text{prob} \ G \ n \ p \ n \ (\lambda es. 1/2 * n / k \leq \alpha \ (\text{edge-space. edge-ugraph } n \ es))$ )
   $\longrightarrow 0$ 
  (is ( $\lambda n. \text{?prob-fun } n$ )  $\longrightarrow 0$ )

```

**proof** –

```

let ?prob-fun-raw n = prob G n p n ( $\lambda es. \text{nat}(\text{ceiling } (1/2 * n / k)) \leq \alpha \ (\text{edge-space. edge-ugraph } n \ es)$ )

```

```

define r where r n =  $1 / 2 * n / k$  for n :: nat

```

```

let ?nr =  $\lambda n. \text{nat}(\text{ceiling } (r \ n))$ 

```

```

have r-pos:  $\bigwedge n. 0 < n \implies 0 < r \ n$  by (auto simp: r-def field-simps)

```

```

have nr-bounds:  $\forall^\infty n. 2 \leq ?nr \ n \wedge ?nr \ n \leq n$ 

```

```

  by (intro eventually-sequentiallyI [of 4 * k])
  (simp add: r-def nat-ceiling-le-eq le-natceiling-iff field-simps)

```

```

from nr-bounds p-prob have ev-prob-fun-raw-le:

```

$\forall^\infty n. \text{probGn } p \ n \ (\lambda es. ?nr \ n \leq \alpha \ (\text{edge-space.edge-ugraph } n \ es))$   
 $\leq (n * \exp(-p \ n * (\text{real } (?nr \ n) - 1) / 2)) \ \text{powr } ?nr \ n$   
(is  $\forall^\infty n. ?\text{prob-fun-raw-le } n$ )

**proof** (rule eventually-elim2)  
**fix**  $n :: \text{nat}$  **assume**  $A: 2 \leq ?nr \ n \wedge ?nr \ n \leq n \ 0 < p \ n \wedge p \ n < 1$   
**then interpret**  $pG: \text{edge-space } n \ p \ n$  **by** *unfold-locales auto*

**have**  $r: \text{real } (?nr \ n - \text{Suc } 0) = \text{real } (?nr \ n) - \text{Suc } 0$  **using**  $A$  **by** *auto*

**have** [*simp*]:  $n > 0$  **using**  $A$  **by** *linarith*  
**have**  $\text{probGn } p \ n \ (\lambda es. ?nr \ n \leq \alpha \ (\text{edge-space.edge-ugraph } n \ es))$   
 $\leq (n \ \text{choose } ?nr \ n) * (1 - p \ n)^{(?nr \ n \ \text{choose } 2)}$   
**using**  $A$  **by** (*auto intro: pG.random-prob-independent*)  
**also have**  $\dots \leq n \ \text{powr } ?nr \ n * (1 - p \ n) \ \text{powr } (?nr \ n \ \text{choose } 2)$   
**using**  $A$  **by** (*simp add: powr-realpow of-nat-power [symmetric] binomial-le-pow*  
*del: of-nat-power*)  
**also have**  $\dots = n \ \text{powr } ?nr \ n * (1 - p \ n) \ \text{powr } (?nr \ n * (?nr \ n - 1) / 2)$   
**by** (*cases even (?nr \ n - 1)*)  
(*auto simp add: n-choose-2-nat real-of-nat-div*)  
**also have**  $\dots = n \ \text{powr } ?nr \ n * ((1 - p \ n) \ \text{powr } ((?nr \ n - 1) / 2)) \ \text{powr } ?nr \ n$   
 $n$   
**by** (*auto simp add: powr-powr r ac-simps*)  
**also have**  $\dots \leq (n * \exp(-p \ n * (?nr \ n - 1) / 2)) \ \text{powr } ?nr \ n$   
**proof** -  
**have**  $(1 - p \ n) \ \text{powr } ((?nr \ n - 1) / 2) \leq \exp(-p \ n) \ \text{powr } ((?nr \ n - 1) / 2)$   
 $2)$   
**using**  $A$  **by** (*auto simp: powr-mono2 diff-conv-add-uminus simp del:*  
*add-uminus-conv-diff*)  
**also have**  $\dots = \exp(-p \ n * (?nr \ n - 1) / 2)$  **by** (*auto simp: powr-def*)  
**finally show** *?thesis*  
**using**  $A$  **by** (*auto simp: powr-mono2 powr-mult*)  
**qed**  
**finally show**  $\text{probGn } p \ n \ (\lambda es. ?nr \ n \leq \alpha \ (\text{edge-space.edge-ugraph } n \ es))$   
 $\leq (n * \exp(-p \ n * (\text{real } (?nr \ n) - 1) / 2)) \ \text{powr } ?nr \ n$   
**using**  $A \ r$  **by** *simp*

**qed**

**from** *p-prob N-prop*  
**have** *ev-expr-bound*:  $\forall^\infty n. n * \exp(-p \ n * (\text{real } (?nr \ n) - 1) / 2) \leq (\exp 1 / n) \ \text{powr } (1 / 2)$

**proof** (*elim eventually-rev-mp, intro eventually-sequentiallyI conjI impI*)  
**fix**  $n$  **assume** *n-bound[arith]*:  $2 \leq n$   
**and** *p-bound*:  $0 < p \ n \wedge p \ n < 1 \ (6 * k * \ln n) / n \leq p \ n$   
**have** *r-bound*:  $r \ n \leq ?nr \ n$  **by** (*rule real-nat-ceiling-ge*)

**have**  $n * \exp(-p \ n * (\text{real } (?nr \ n) - 1) / 2) \leq n * \exp(-3 / 2 * \ln n + p \ n / 2)$

**proof** -  
**have**  $0 < \ln n$  **using** *n-bound* **by** *auto*

**then have**  $(3 / 2) * \ln n \leq ((6 * k * \ln n) / n) * (?nr n / 2)$   
**using** *r-bound le-of-int-ceiling[of n/2\*k]*  
**by** (*simp add: r-def field-simps del: le-of-int-ceiling*)  
**also have**  $\dots \leq p n * (?nr n / 2)$   
**using** *n-bound p-bound r-bound r-pos[of n]* **by** (*auto simp: field-simps*)  
**finally show** *?thesis using r-bound by (auto simp: field-simps)*  
**qed**  
**also have**  $\dots \leq n * n \text{ powr } (- 3 / 2) * \exp 1 \text{ powr } (1 / 2)$   
**using** *p-bound by (simp add: powr-def exp-add [symmetric])*  
**also have**  $\dots \leq n \text{ powr } (-1 / 2) * \exp 1 \text{ powr } (1 / 2)$  **by** (*simp add:*  
*powr-mult-base*)  
**also have**  $\dots = (\exp 1 / n) \text{ powr } (1/2)$   
**by** (*simp add: powr-divide powr-minus-divide*)  
**finally show**  $n * \exp (- p n * (\text{real } (?nr n) - 1) / 2) \leq (\exp 1 / n) \text{ powr } (1$   
 $/ 2)$  .  
**qed**

**have** *ceil-bound:  $\bigwedge G n. 1/2*n/k \leq \alpha G \longleftrightarrow \text{nat}(\text{ceiling } (1/2*n/k)) \leq \alpha G$*   
**by** (*case-tac  $\alpha G$  (auto simp: nat-ceiling-le-eq)*)

**show** *?thesis*  
**proof** (*unfold ceil-bound, rule real-tendsto-sandwich*)  
**show**  $(\lambda n. 0) \longrightarrow 0$   
 $(\lambda n. (\exp 1 / n) \text{ powr } (1 / 2)) \longrightarrow 0$   
 $\forall^\infty n. 0 \leq ?\text{prob-fun-raw } n$   
**using** *p-prob by (auto intro: measure-nonneg LIMSEQ-inv-powr elim: eventually-mono)*  
**next**  
**from** *nr-bounds ev-expr-bound ev-prob-fun-raw-le*  
**show**  $\forall^\infty n. ?\text{prob-fun-raw } n \leq (\exp 1 / n) \text{ powr } (1 / 2)$   
**proof** (*elim eventually-rev-mp, intro eventually-sequentiallyI impI conjI*)  
**fix** *n assume*  $A: 3 \leq n$   
**and** *nr-bounds:  $2 \leq ?nr n \wedge ?nr n \leq n$*   
**and** *prob-fun-raw-le:  $?\text{prob-fun-raw-le } n$*   
**and** *expr-bound:  $n * \exp (- p n * (\text{real } (\text{nat}(\text{ceiling } (r n))) - 1) / 2) \leq$*   
 $(\exp 1 / n) \text{ powr } (1 / 2)$

**have**  $\exp 1 < (3 :: \text{real})$  **by** (*approximation 6*)  
**then have**  $(\exp 1 / n) \text{ powr } (1 / 2) \leq 1 \text{ powr } (1 / 2)$   
**using** *A by (intro powr-mono2) (auto simp: field-simps)*  
**then have** *ep-bound:  $(\exp 1 / n) \text{ powr } (1 / 2) \leq 1$  by simp*

**have**  $?\text{prob-fun-raw } n \leq (n * \exp (- p n * (\text{real } (?nr n) - 1) / 2)) \text{ powr } (?nr n)$   
**using** *prob-fun-raw-le by (simp add: r-def)*  
**also have**  $\dots \leq ((\exp 1 / n) \text{ powr } (1 / 2)) \text{ powr } ?nr n$   
**using** *expr-bound A by (auto simp: powr-mono2)*  
**also have**  $\dots \leq ((\exp 1 / n) \text{ powr } (1 / 2))$   
**using** *nr-bounds ep-bound A by (auto simp: powr-le-one-le)*

**finally show**  $?prob\text{-}fun\text{-}raw\ n \leq (exp\ 1 / n)\ powr\ (1 / 2)$  .  
**qed**  
**qed**  
**qed**

Mean number of  $k$ -cycles in a graph. (Or rather of paths describing a circle of length  $k$ ):

**lemma** (in *edge-space*) *mean-k-cycles*:

**assumes**  $3 \leq k < n$

**shows**  $(\int es.\ card\ \{c \in\ ucycles\ (edge\text{-}ugraph\ es).\ uwalk\text{-}length\ c = k\}\ \partial\ P)$   
 $=\ of\text{-}nat\ (fact\ n\ div\ fact\ (n - k)) * p^k$

**proof** –

**let**  $?k\text{-}cycle = \lambda es\ c\ k.\ c \in\ ucycles\ (edge\text{-}ugraph\ es) \wedge uwalk\text{-}length\ c = k$

**define**  $C$  **where**  $C\ k = \{c.\ ?k\text{-}cycle\ S\text{-}edges\ c\ k\}$  **for**  $k$

–  $C\ k$  is the set of all possible cycles of size  $k$  in *edge-ugraph* *S-edges*

**define**  $XG$  **where**  $XG\ es = \{c.\ ?k\text{-}cycle\ es\ c\ k\}$  **for**  $es$

–  $XG\ es$  is the set of cycles contained in a *edge-ugraph* *es*

**define**  $XC$  **where**  $XC\ c = \{es \in\ space\ P.\ ?k\text{-}cycle\ es\ c\ k\}$  **for**  $c$

– " $XC\ c$  is the set of graphs (edge sets) containing a cycle  $c$ "

**then have**  $XC\text{-}in\text{-}sets: \bigwedge c.\ XC\ c \in\ sets\ P$

**and**  $XC\text{-}cyl: \bigwedge c.\ c \in\ C\ k \implies XC\ c = cylinder\ S\text{-}edges\ (set\ (uwalk\text{-}edges\ c))$

**{}**

**by** (*auto simp: ucycles-def space-eq uwalks-def C-def cylinder-def sets-eq*)

**have**  $(\int es.\ card\ \{c \in\ ucycles\ (edge\text{-}ugraph\ es).\ uwalk\text{-}length\ c = k\}\ \partial\ P)$   
 $= (\sum x \in\ space\ P.\ card\ (XG\ x) * prob\ \{x\})$

**by** (*simp add: XG-def integral-finite-singleton space-eq*)

**also have**  $\dots = (\sum c \in\ C\ k.\ prob\ (cylinder\ S\text{-}edges\ (set\ (uwalk\text{-}edges\ c))\ \{\}))$

**proof** –

**have**  $XG\text{-}Int\text{-}C: \bigwedge s.\ s \in\ space\ P \implies C\ k \cap XG\ s = XG\ s$

**unfolding**  $XG\text{-}def\ C\text{-}def\ ucycles\text{-}def\ uwalks\text{-}def\ edge\text{-}ugraph\text{-}def$  **by** *auto*

**have**  $fin\text{-}XC: \bigwedge k.\ finite\ (XC\ k)$  **and**  $fin\text{-}C: finite\ (C\ k)$

**unfolding**  $C\text{-}def\ XC\text{-}def$  **by** (*auto simp: finite-edges space-eq intro!: finite-ucycles*)

**have**  $(\sum x \in\ space\ P.\ card\ (XG\ x) * prob\ \{x\}) = (\sum x \in\ space\ P.\ (\sum c \in\ XG\ x.\ prob\ \{x\}))$

**by** *simp*

**also have**  $\dots = (\sum x \in\ space\ P.\ (\sum c \in\ C\ k.\ if\ c \in\ XG\ x\ then\ prob\ \{x\}\ else\ 0))$

**using**  $fin\text{-}C$  **by** (*simp add: sum.If-cases*) (*simp add: XG-Int-C*)

**also have**  $\dots = (\sum c \in\ C\ k.\ (\sum x \in\ space\ P \cap XC\ c.\ prob\ \{x\}))$

**using** *finite-edges* **by** (*subst sum.swap*) (*simp add: sum.inter-restrict XG-def XC-def space-eq*)

**also have**  $\dots = (\sum c \in\ C\ k.\ prob\ (XC\ c))$

**using**  $fin\text{-}XC\ XC\text{-}in\text{-}sets$

**by** (*auto simp add: prob-eq sets-eq space-eq intro!: sum.cong*)

**finally show**  $?thesis$  **by** (*simp add: XC-cyl*)

**qed**

```

also have ... =  $(\sum_{c \in C} k. p \wedge k)$ 
proof -
  have  $\bigwedge x. x \in C \ k \implies \text{card} (\text{set} (\text{uwalk-edges } x)) = \text{uwalk-length } x$ 
  by (auto simp: uwalk-length-def C-def ucycles-distinct-edges intro: distinct-card)
  then show ?thesis by (auto simp: C-def ucycles-def uwalks-def cylinder-prob)
qed
also have ... = of-nat (fact n div fact (n - k)) * p ^ k
proof -
  have inj-last-Cons:  $\bigwedge A. \text{inj-on } (\lambda es. \text{last } es \# es) \ A$  by (rule inj-onI simp)
  { fix xs A assume  $\exists \leq \text{length } xs - \text{Suc } 0 \ \text{hd } xs = \text{last } xs$ 
    then have  $xs \in (\lambda xs. \text{last } xs \# xs) \ 'A \longleftrightarrow \text{tl } xs \in A$ 
    by (cases xs (auto simp: inj-image-mem-iff[OF inj-last-Cons] split: if-split-asm))
  }
  note image-mem-iff-inst = this

  { fix xs have  $xs \in \text{uwalks} (\text{edge-ugraph } S\text{-edges}) \implies \text{set} (\text{tl } xs) \subseteq S\text{-verts}$ 
    unfolding uwalks-def by (induct xs auto) }
  moreover
  { fix xs assume  $\text{set } xs \subseteq S\text{-verts} \ \exists \leq \text{length } xs \ \text{distinct } xs$ 
    then have  $(\text{last } xs \# xs) \in \text{uwalks} (\text{edge-ugraph } S\text{-edges})$ 
    proof (induct xs rule: uwalk-edges.induct)
      case ( $\exists \ x \ y \ ys$ )
      have S-edges-memI:  $\bigwedge x \ y. x \in S\text{-verts} \implies y \in S\text{-verts} \implies x \neq y \implies \{x, y\} \in S\text{-edges}$ 
      unfolding S-edges-def all-edges-def image-def by auto

      have  $ys \neq [] \implies \text{set } ys \subseteq S\text{-verts} \implies \text{last } ys \in S\text{-verts}$  by auto
      with  $\exists$  show ?case
      by (auto simp add: uwalks-def Suc-le-eq intro: S-edges-memI)
    }
  qed simp-all}
  moreover note  $\langle \exists \leq k \rangle$ 
  ultimately
  have  $C \ k = (\lambda xs. \text{last } xs \# xs) \ ' \{xs. \text{length } xs = k \wedge \text{distinct } xs \wedge \text{set } xs \subseteq S\text{-verts}\}$ 
  by (auto simp: C-def ucycles-def uwalk-length-conv image-mem-iff-inst)
  moreover have  $\text{card } S\text{-verts} = n$  by (simp add: S-verts-def)
  ultimately have  $\text{card} (C \ k) = \text{fact } n \ \text{div} \ \text{fact} (n - k)$ 
  using  $\langle k < n \rangle$ 
  by (simp add: card-image[OF inj-last-Cons] card-lists-distinct-length-eq' fact-div-fact)
  then show ?thesis by simp
qed
finally show ?thesis by simp
qed

```

Girth-Chromatic number theorem:

**theorem** *girth-chromatic*:

**fixes**  $l :: \text{nat}$

**shows**  $\exists G. \text{uwellformed } G \wedge l < \text{girth } G \wedge l < \text{chromatic-number } G$

**proof** -

```

define  $k$  where  $k = \max 3 l$ 
define  $\varepsilon$  where  $\varepsilon = 1 / (2 * k)$ 
define  $p$  where  $p n = \text{real } n \text{ powr } (\varepsilon - 1)$  for  $n :: \text{nat}$ 

let  $?ug = \text{edge-space.edge-ugraph}$ 

define  $\text{short-count}$  where  $\text{short-count } g = \text{card } (\text{short-cycles } g k)$  for  $g$ 
— This random variable differs from the one used in the proof of theorem 11.2.2,
as we count the number of paths describing a circle, not the circles themselves

from  $k\text{-def}$  have  $3 \leq k l \leq k$  by  $\text{auto}$ 
from  $\langle 3 \leq k \rangle$  have  $\varepsilon\text{-props: } 0 < \varepsilon \varepsilon < 1 / k \varepsilon < 1$  by  $(\text{auto simp: } \varepsilon\text{-def field-simps})$ 

have  $\text{ev-p: } \forall^\infty n. 0 < p n \wedge p n < 1$ 
proof  $(\text{rule eventually-sequentiallyI})$ 
  fix  $n :: \text{nat}$  assume  $2 \leq n$ 
  with  $\langle \varepsilon < 1 \rangle$  have  $n \text{ powr } (\varepsilon - 1) < 1$  by  $(\text{auto intro!: powr-less-one})$ 
  then show  $0 < p n \wedge p n < 1$  using  $\langle 2 \leq n \rangle$ 
  by  $(\text{auto simp: } p\text{-def})$ 
qed
then
have  $\text{prob-short-count-le: } \forall^\infty n. \text{probGn } p n (\lambda \text{es. } (\text{real } n / 2) \leq \text{short-count } (?ug n \text{ es}))$ 
   $\leq 2 * (k - 2) * n \text{ powr } (\varepsilon * k - 1)$  (is  $\forall^\infty n. ?P n)$ 
proof  $(\text{elim eventually-rev-mp, intro eventually-sequentiallyI impI})$ 
  fix  $n :: \text{nat}$  assume  $A: \text{Suc } k \leq n \ 0 < p n \wedge p n < 1$ 
  then interpret  $pG: \text{edge-space } n \ p \ n$  by  $\text{unfold-locales auto}$ 
  have  $1 \leq n$  using  $A$  by  $\text{auto}$ 

define  $\text{mean-short-count}$  where  $\text{mean-short-count} = (\int \text{es. short-count } (?ug n \text{ es}) \partial \ pG.P)$ 

have  $\text{mean-short-count-le: } \text{mean-short-count} \leq (k - 2) * n \text{ powr } (\varepsilon * k)$ 
proof —
  have  $\text{small-empty: } \bigwedge \text{es } k. k \leq 2 \implies \text{short-cycles } (\text{edge-space.edge-ugraph } n \text{ es}) k = \{\}$ 
  by  $(\text{auto simp add: short-cycles-def ucycles-def})$ 
  have  $\text{short-count-conv: } \bigwedge \text{es. short-count } (?ug n \text{ es}) = (\sum_{i=3..k} \text{real } (\text{card } \{c \in \text{ucycles } (?ug n \text{ es}). \text{uwalk-length } c = i\}))$ 
  proof  $(\text{unfold short-count-def, induct } k)$ 
  case  $0$  with  $\text{small-empty}$  show  $?case$  by  $\text{auto}$ 
next
  case  $(\text{Suc } k)$ 
  show  $?case$  proof  $(\text{cases } \text{Suc } k \leq 2)$ 
  case  $\text{True}$  with  $\text{small-empty}$  show  $?thesis$  by  $\text{auto}$ 
next
  case  $\text{False}$ 
  have  $\{c \in \text{ucycles } (?ug n \text{ es}). \text{uwalk-length } c \leq \text{Suc } k\}$ 

```

$= \{c \in \text{ucycles } (?ug \ n \ es). \ \text{uwalk-length } c \leq k\} \cup \{c \in \text{ucycles } (?ug \ n \ es). \ \text{uwalk-length } c = \text{Suc } k\}$   
**by** *auto*  
**moreover**  
**have** *finite* (*uverts* (*edge-space.edge-ugraph* *n es*)) **by** *auto*  
**ultimately**  
**have** *card*  $\{c \in \text{ucycles } (?ug \ n \ es). \ \text{uwalk-length } c \leq \text{Suc } k\}$   
 $= \text{card } \{c \in \text{ucycles } (?ug \ n \ es). \ \text{uwalk-length } c \leq k\} + \text{card } \{c \in \text{ucycles } (?ug \ n \ es). \ \text{uwalk-length } c = \text{Suc } k\}$   
**using** *finite-ucycles* **by** (*subst card-Un-disjoint[symmetric]*) *auto*  
**then show** *?thesis*  
**using** *Suc False unfolding short-cycles-def* **by** (*auto simp: not-le*)  
**qed**  
**qed**

**have** *mean-short-count*  $= (\sum_{i=3..k} \int \text{es}. \ \text{card } \{c \in \text{ucycles } (?ug \ n \ es). \ \text{uwalk-length } c = i\} \ \partial \ pG.P)$   
**unfolding** *mean-short-count-def short-count-conv*  
**by** (*subst Bochner-Integration.integral-sum*) (*auto intro: pG.integral-finite-singleton*)  
**also have**  $\dots = (\sum_{i \in \{3..k\}} \text{of-nat } (\text{fact } n \ \text{div } \text{fact } (n - i)) * p \ n \ ^i)$   
**using** *A* **by** (*simp add: pG.mean-k-cycles*)  
**also have**  $\dots \leq (\sum_{i \in \{3..k\}} n^i * p \ n^i)$   
**apply** (*rule sum-mono*)  
**by** (*meson A fact-div-fact-le-pow Suc-leD atLeastAtMost-iff of-nat-le-iff order-trans mult-le-cancel-iff1 zero-less-power*)  
**also have**  $\dots \leq (\sum_{i \in \{3..k\}} n \ \text{powr } (\varepsilon * k))$   
**using**  $\langle 1 \leq n \rangle \langle 0 < \varepsilon \rangle A$   
**by** (*intro sum-mono*) (*auto simp: p-def field-simps powr-mult-base powr-powr powr-realpow[symmetric] powr-mult[symmetric] powr-add[symmetric]*)  
**finally show** *?thesis* **by** *simp*  
**qed**

**have** *pG.prob*  $\{es \in \text{space } pG.P. \ n/2 \leq \text{short-count } (?ug \ n \ es)\} \leq \text{mean-short-count} / (n/2)$   
**unfolding** *mean-short-count-def* **using**  $\langle 1 \leq n \rangle$   
**by** (*intro pG.Markov-inequality*) (*auto simp: short-count-def*)  
**also have**  $\dots \leq 2 * (k - 2) * n \ \text{powr } (\varepsilon * k - 1)$   
**proof** –  
**have** *mean-short-count*  $/ (n / 2) \leq 2 * (k - 2) * (1 / n \ \text{powr } 1) * n \ \text{powr } (\varepsilon * k)$   
**using** *mean-short-count-le*  $\langle 1 \leq n \rangle$  **by** (*simp add: field-simps*)  
**then show** *?thesis* **by** (*simp add: powr-diff algebra-simps*)  
**qed**  
**finally show** *?P n* .  
**qed**

**define** *pf-short-count pf- $\alpha$*   
**where** *pf-short-count*  $n = \text{probGn } p \ n \ (\lambda es. \ n/2 \leq \text{short-count } (?ug \ n \ es))$   
**and** *pf- $\alpha$*   $n = \text{probGn } p \ n \ (\lambda es. \ 1/2 * n/k \leq \alpha \ (\text{edge-space.edge-ugraph } n \ es))$



```

for n

have ev-short-count-le:  $\forall^\infty n. \text{pf-short-count } n < 1 / 2$ 
proof -
  have  $\varepsilon * k - 1 < 0$ 
  using  $\varepsilon\text{-props } \langle 3 \leq k \rangle$  by (auto simp: field-simps)
  then have  $(\lambda n. 2 * (k - 2) * n \text{ powr } (\varepsilon * k - 1)) \longrightarrow 0$  (is ?bound  $\longrightarrow$ 
0)
  by (intro tendsto-mult-right-zero LIMSEQ-neg-powr)
  then have  $\forall^\infty n. \text{dist } (?bound \ n) \ 0 < 1 / 2$ 
  by (rule tendstoD) simp
  with prob-short-count-le show ?thesis
  by (rule eventually-elim2) (auto simp: dist-real-def pf-short-count-def)
qed

have lim- $\alpha$ :  $\text{pf-}\alpha \longrightarrow 0$ 
proof -
  have  $0 < k$  using  $\langle 3 \leq k \rangle$  by simp

  have  $\forall^\infty n. (6*k) * \ln n / n \leq p \ n \iff (6*k) * \ln n * n \text{ powr } - \varepsilon \leq 1$ 
  proof (rule eventually-sequentiallyI)
    fix n :: nat assume  $1 \leq n$ 
    then have  $(6 * k) * \ln n / n \leq p \ n \iff (6*k) * \ln n * (n \text{ powr } - 1) \leq n$ 
    powr  $(\varepsilon - 1)$ 
    by (subst powr-minus) (simp add: divide-inverse p-def)
    also have  $\dots \iff (6*k) * \ln n * ((n \text{ powr } - 1) / (n \text{ powr } (\varepsilon - 1))) \leq n$ 
    powr  $(\varepsilon - 1) / (n \text{ powr } (\varepsilon - 1))$ 
    using  $\langle 1 \leq n \rangle$  by (auto simp: field-simps)
    also have  $\dots \iff (6*k) * \ln n * n \text{ powr } - \varepsilon \leq 1$ 
    by (simp add: powr-diff [symmetric])
    finally show  $(6*k) * \ln n / n \leq p \ n \iff (6*k) * \ln n * n \text{ powr } - \varepsilon \leq 1$  .
  qed
  then have  $(\forall^\infty n. (6 * k) * \ln n / \text{real } n \leq p \ n)$ 
   $\iff (\forall^\infty n. (6*k) * \ln n * n \text{ powr } - \varepsilon \leq 1)$ 
  by (rule eventually-subst)
  also have  $\forall^\infty n. (6*k) * \ln n * n \text{ powr } - \varepsilon \leq 1$ 
  proof -
    { fix n :: nat assume  $0 < n$ 
      have  $\ln (\text{real } n) \leq n \text{ powr } (\varepsilon/2) / (\varepsilon/2)$ 
      using  $\langle 0 < n \rangle \langle 0 < \varepsilon \rangle$  by (intro ln-powr-bound) auto
      also have  $\dots \leq 2/\varepsilon * n \text{ powr } (\varepsilon/2)$  by (auto simp: field-simps)
      finally have  $(6*k) * \ln n * (n \text{ powr } - \varepsilon) \leq (6*k) * (2/\varepsilon * n \text{ powr } (\varepsilon/2))$ 
      *  $(n \text{ powr } - \varepsilon)$ 
      using  $\langle 0 < n \rangle \langle 0 < k \rangle$  by (intro mult-right-mono mult-left-mono) auto
      also have  $\dots = 12*k/\varepsilon * n \text{ powr } (-\varepsilon/2)$ 
      unfolding divide-inverse
      by (auto simp: field-simps powr-minus[symmetric] powr-add[symmetric])
      finally have  $(6*k) * \ln n * (n \text{ powr } - \varepsilon) \leq 12*k/\varepsilon * n \text{ powr } (-\varepsilon/2)$  .
    }
  }

```

**then have**  $\forall^\infty n. (6*k) * \ln n * (n \text{ powr } - \varepsilon) \leq 12*k/\varepsilon * n \text{ powr } (-\varepsilon/2)$   
**by** (*intro eventually-sequentiallyI[of 1]*) *auto*  
**also have**  $\forall^\infty n. 12*k/\varepsilon * n \text{ powr } (-\varepsilon/2) \leq 1$   
**proof** –  
**have**  $(\lambda n. 12*k/\varepsilon * n \text{ powr } (-\varepsilon/2)) \longrightarrow 0$   
**using**  $\langle 0 < \varepsilon \rangle$  **by** (*intro tendsto-mult-right-zero LIMSEQ-neg-powr*) *auto*  
**then show** *?thesis*  
**using**  $\langle 0 < \varepsilon \rangle$  **by** – (*drule tendstoD[where e=1], auto elim: eventually-mono*)  
**qed**  
**finally** (*eventually-le-le*) **show** *?thesis* .  
**qed**  
**finally have**  $\forall^\infty n. \text{real } (6 * k) * \ln (\text{real } n) / \text{real } n \leq p * n$  .  
**with** *ev-p*  $\langle 0 < k \rangle$  **show** *?thesis unfolding pf- $\alpha$ -def* **by** (*rule almost-never-le- $\alpha$* )  
**qed**

**from** *ev-short-count-le lim- $\alpha$ [THEN tendstoD, of 1/2]* *ev-p*  
**have**  $\forall^\infty n. 0 < p * n \wedge p * n < 1 \wedge \text{pf-short-count } n < 1/2 \wedge \text{pf-}\alpha \text{ } n < 1/2$   
**by** *simp (elim eventually-rev-mp, auto simp: eventually-sequentially dist-real-def)*  
**then obtain** *n* **where**  $0 < p * n \wedge p * n < 1$  **and** [*arith*]:  $0 < n$   
**and** *probs*:  $\text{pf-short-count } n < 1/2 \wedge \text{pf-}\alpha \text{ } n < 1/2$   
**by** (*auto simp: eventually-sequentially*)  
**then interpret** *ES*: *edge-space* *n* (*p n*) **by** *unfold-locales auto*

**have** *rest-compl*:  $\bigwedge A P. A - \{x \in A. P x\} = \{x \in A. \neg P x\}$  **by** *blast*

**from** *probs* **have** *ES.prob* ( $\{es \in \text{space } ES.P. n/2 \leq \text{short-count } (?ug \text{ } n \text{ } es)\} \cup \{es \in \text{space } ES.P. 1/2 * n/k \leq \alpha (?ug \text{ } n \text{ } es)\}$ )  $\leq \text{pf-short-count } n + \text{pf-}\alpha \text{ } n$   
**unfolding** *pf-short-count-def pf- $\alpha$ -def* **by** (*subst ES.finite-measure-subadditive*)  
*auto*  
**also have**  $\dots < 1$  **using** *probs* **by** *auto*  
**finally have**  $0 < ES.\text{prob } (\text{space } ES.P - (\{es \in \text{space } ES.P. n/2 \leq \text{short-count } (?ug \text{ } n \text{ } es)\} \cup \{es \in \text{space } ES.P. 1/2 * n/k \leq \alpha (?ug \text{ } n \text{ } es)\}))$  (**is**  $0 < ES.\text{prob } ?S$ )  
**by** (*subst ES.prob-compl*) *auto*  
**also have**  $?S = \{es \in \text{space } ES.P. \text{short-count } (?ug \text{ } n \text{ } es) < n/2 \wedge \alpha (?ug \text{ } n \text{ } es) < 1/2 * n/k\}$  (**is**  $\dots = ?C$ )  
**by** (*auto simp: not-less rest-compl*)  
**finally have**  $?C \neq \{\}$  **by** (*intro notI*) (*simp only:., auto*)  
**then obtain** *es* **where** *es-props*:  $es \in \text{space } ES.P$   
 $\text{short-count } (?ug \text{ } n \text{ } es) < n/2 \wedge \alpha (?ug \text{ } n \text{ } es) < 1/2 * n/k$   
**by** *auto*  
— now we obtained a high colored graph (few independent nodes) with almost no short cycles

**define** *G* **where**  $G = ?ug \text{ } n \text{ } es$   
**define** *H* **where**  $H = \text{kill-short } G \text{ } k$

**have** *G-props*:  $uverts \text{ } G = \{1..n\}$  *finite* (*uverts* *G*)  $\text{short-count } G < n/2 \wedge G <$

$1/2 * n/k$   
**unfolding**  $G$ -def **using**  $es$ -props **by** ( $auto$  simp:  $ES.S$ -verts-def)

**have**  $uwellformed$   $G$  **by** ( $auto$  simp:  $G$ -def  $uwellformed$ -def  $all$ -edges-def  $ES.S$ -edges-def)  
**with**  $G$ -props **have**  $T1$ :  $uwellformed$   $H$  **unfolding**  $H$ -def **by** ( $intro$   $kill$ -short- $uwellformed$ )

**have**  $enat$   $l \leq enat$   $k$  **using**  $\langle l \leq k \rangle$  **by** simp  
**also have**  $\dots < girth$   $H$  **using**  $G$ -props **by** ( $auto$  simp:  $kill$ -short- $large$ - $girth$   $H$ -def)

**finally have**  $T2$ :  $l < girth$   $H$  .

**have**  $card$ - $H$ :  $n/2 \leq card$  ( $uverts$   $H$ )  
**using**  $G$ -props  $es$ -props  $kill$ -short- $order$ -of- $graph$ [of  $G$   $k$ ] **by** ( $simp$  add:  $short$ -count-def  $H$ -def)

**then have**  $uverts$ - $H$ :  $uverts$   $H \neq \{\}$   $0 < card$  ( $uverts$   $H$ ) **by**  $auto$   
**then have**  $0 < \alpha$   $H$  **using**  $zero$ -less- $\alpha$   $uverts$ - $H$  **by**  $auto$

**have**  $\alpha$ - $HG$ :  $\alpha$   $H \leq \alpha$   $G$   
**unfolding**  $H$ -def  $G$ -def **by** ( $auto$   $intro$ :  $kill$ -short- $\alpha$ )

**have**  $enat$   $l \leq ereal$   $k$  **using**  $\langle l \leq k \rangle$  **by**  $auto$   
**also have**  $\dots < (n/2) / \alpha$   $G$  **using**  $G$ -props  $\langle 3 \leq k \rangle$   
**by** ( $cases$   $\alpha$   $G$ ) ( $auto$  simp:  $field$ -simps)  
**also have**  $\dots \leq (n/2) / \alpha$   $H$  **using**  $\alpha$ - $HG$   $\langle 0 < \alpha$   $H \rangle$   
**by** ( $auto$  simp:  $ereal$ -of- $enat$ - $pushout$   $intro$ !:  $ereal$ - $divide$ - $left$ - $mono$ )  
**also have**  $\dots \leq card$  ( $uverts$   $H$ ) /  $\alpha$   $H$  **using**  $card$ - $H$   $\langle 0 < \alpha$   $H \rangle$   
**by** ( $auto$   $intro$ !:  $ereal$ - $divide$ - $right$ - $mono$ )  
**also have**  $\dots \leq chromatic$ -number  $H$  **using**  $uverts$ - $H$   $T1$  **by** ( $intro$   $chromatic$ -lb)

$auto$   
**finally have**  $T3$ :  $l < chromatic$ -number  $H$   
**by** ( $simp$  del:  $ereal$ -of- $enat$ -simps)

**from**  $T1$   $T2$   $T3$  **show**  $?thesis$  **by** fast

**qed**

**end**

## References

- [1] R. Diestel. *Graph Theory*, volume 173 of *Graduate Texts in Mathematics*. Springer, 4 edition, 2010. <http://diestel-graph-theory.com>.