A Probabilistic Proof of the Girth-Chromatic Number Theorem

Lars Noschinski

May 28, 2015

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

1 Auxiliary lemmas and setup 2
  1.1 Numbers . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 2
  1.2 Lists and Sets . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 5
  1.3 Limits and eventually . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 5
2 Undirected Simple Graphs 6
  2.1 Basic Properties . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 7
  2.2 Girth, Independence and Vertex Colorings . . . . . . . . . . . . . . . . . . . . . 9
3 Probability Space on Sets of Edges 12
  3.1 Graph Probabilities outside of Edge-Space locale . . . . . . . . . . . . . . . . . . 15
4 Short cycles 15
5 The Chromatic-Girth Theorem 18
theory Girth-Chromatic-Misc
imports
  Main
  ~/src/HOL/Library/Extended-Real
begin
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.

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
\[
evseq P \equiv \text{eventually } P \text{ sequentially}
\]

1.1 Numbers

**lemma** enat-in-Inf:

fixes \( S :: \) enat set

assumes \( \text{Inf } S \neq \text{top} \)

shows \( \text{Inf } S \in S \)

**proof** (rule contr)

assume \( A: \neg \text{thesis} \)

obtain \( n \) where \( \text{Inf-conv: } \text{Inf } S = \text{enat } n \) 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.\text{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 \( \text{Inf-conv} \) have \( \text{enat } (\text{Suc } n) \leq s \) by \( \langle \text{cases } s \rangle \) auto

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

with \( \text{Inf-conv} \) show \( \text{False} \) by auto

qed

**lemma** enat-in-INF:

fixes \( f :: 'a \Rightarrow \) enat

assumes \( \inf \text{f x: S. f x} \neq \text{top} \)

obtains \( x \) where \( x \in S \) and \( \inf \text{f x: S. f x} = f x \)

**proof**

from assms have \( \inf \text{f x: S. f x} \in f ' S \)

using enat-in-\text{Inf} [of f ' S] by auto

then obtain \( x \) where \( x \in S \) \( \inf \text{f x: S. f x} = f x \) by auto

then show \( \text{thesis} \). ..

qed

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

fixes \( f :: 'a \Rightarrow \) enat

assumes \( \not-\text{inf: } x \neq \infty \) and \( \text{less: } \forall y. \ y \in S \Rightarrow x < f y \)

shows \( x < (\inf \ y:S. f y) \)

**proof** (rule contr)

2
assume $A$: \neg \text{thesis}
then show False
proof (cases $x = (\INF y:S \ f \ y)$)
case True
  with assms have $(\INF y:S \ f \ y) \neq \text{top}$ by (simp add: top-enat-def)
  then obtain $z$ where $z \in S \ f \ z = x$
    by (rule enat-in-INF) (auto simp: True)
  then show False using less by auto
next
case False
  with $A$ have $(\INF y:S \ f \ y) < x$ by simp
  with less show False by (auto simp: INF-less-iff intro: order-less-asym)
qed

lemma enat-le-Sup-iff:
enat $k \leq \text{Sup } M \iff k = 0 \lor (\exists m \in M. \ enat k \leq m)$ (is $?L \longleftrightarrow ?R$)
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 $\lceil \text{enat } k \leq (\text{if finite } M \text{ then Max } M \text{ else } \infty); M \neq \{\} \Rightarrow \exists m \in M. \ enat k \leq m$.
  by (metis Max-in Sup-enat-def finite-enat-bounded linorder-linear)
  with $k \neq 0$ and $?L$ 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

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

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

lemma le-natceiling-iff:
  fixes $n :: \text{nat}$ and $r :: \text{real}$
  shows $n \leq r \Rightarrow n \leq \text{nat(ceiling } r)$
by linarith
lemma natceiling-le-iff:
  fixes \( n :: \text{nat} \) and \( r :: \text{real} \)
  shows \( r \leq n \implies \text{nat(ceiling r)} \leq n \)
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 \( (\text{n choose 2}) = (n * (n - 1)) \div 2 \)
proof
  show \(?thesis\)
proof (cases \( 2 \leq n \))
  case True
  then obtain \( m \) where \( n = \text{Suc (Suc m)} \)
  by (metis add-Suc le-Suc-ex numeral-2-eq-2)
  moreover have \( (\text{n choose 2}) = (\text{fact n div fact (n - 2)}) \div 2 \)
  using \( \langle 2 \leq n \rangle \) 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 \( x :: \text{real} \)
  assumes \( 1 < x \ y < 0 \)
shows \( x \text{ powr y < 1} \)
using assms less-log-iff by force

lemma powr-le-one-le:
  \( \forall x y :: \text{real} \ . \ 0 < x \implies x \leq 1 \implies 1 \leq y \implies x \text{ powr y} \leq x \)
proof
  fix \( x \ y :: \text{real} \)
  assume \( 0 < x \ x \leq 1 \ y \leq y \)
  have \( x \text{ powr y} = (1 / (1 / x)) \text{ powr y using} \langle 0 < x \rangle \) by (simp add: field-simps)
  also have \( \ldots = 1 / (1 / x) \text{ powr y using} \langle 0 < x \rangle \) by (simp add: powr-divide)
  also have \( \ldots \leq 1 / (1 / x) \text{ powr 1} \)
  proof
    have \( 1 / x \text{ using} \langle 0 < x \rangle \langle x \leq 1 \rangle \) by (auto simp: field-simps)
    then have \( (1 / x) \text{ powr 1} \leq (1 / x) \text{ powr y using} \langle 0 < x \rangle \)
    using \( \langle 1 \leq y \rangle \) by (simp only: powr-mono)
    then show \(?thesis\)
    by (metis \( \langle 1 \leq 1 / x \rangle \langle 1 \leq y \rangle \) neg-le-iff-le powr-minus-divide powr-mono)
  qed
  also have \( \ldots \leq x \) using \( \langle 0 < x \rangle \) 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 \( xs = [] \mid x \) where \( xs = [x] \) \( x \ y \ ys \) where \( xs = x \# y \# ys \)
by (metis list.exhaust)

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

lemma card-0-iff:
\( \text{card} \ A = 0 \longleftrightarrow A = \{\} \lor \neg \text{finite} \ A \)
by auto

1.3 Limits and eventually

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 (rule eventually-elim2) (rule order-trans)

lemma eventually-sequentially-lessI:
assumes \( \forall x. c < x \implies P x \)
shows eventually \( P \) sequentially
unfolding eventually-sequentially by (rule exI [where \( x = \text{Suc} \ c \)]) (auto intro: assms)

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

lemma LIMSEQ-inv-powr:
assumes \( 0 < c 0 < d \)
shows \( \text{lim} (%n. (c / n) \text{powr} d) \to 0 \)
proof (rule tendsto-zero-powrI)
from \( 0 < c \) have \( \forall x. 0 < x \implies 0 < c / x \) by simp
then show eventually (\( \lambda x. 0 < c / \text{real} x \)) sequentially
by (rule eventually-sequentiallyI[of 1]) simp

show \( \lambda x. c / \text{real} x \to 0 \)
proof (rule tendstoI)
fix \( e :: \text{real} \) assume \( 0 < e \)
then have \( \forall x. 0 < x \implies c / x < e \longleftrightarrow c / e < x \)
by (auto simp: field-simps)
then show eventually (\( \lambda x. \text{dist} (c / \text{real} x) 0 < c \)) sequentially
using (0 < c :0 < e)
by (intro eventually-sequentially-lessI[of nat(ceiling (c/e))])
(auto simp: dist-real-def nateiling-lessD)
qed
show 0 < d by (rule assms)
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 × uedge set
type-synonym uwalk = uvert list

abbreviation uedges :: ugraph ⇒ uedge set
where
uedges G ≡ snd G

abbreviation uverts :: ugraph ⇒ uvert set
where
uverts G ≡ fst G

fun mk-uedge :: uvert × uvert ⇒ uedge
where
mk-uedge (u, v) = {u, v}

All edges over a set of vertexes S:

definition all-edges S ≡ mk-uedge ' {uv ∈ S × S. fst uv ≠ snd uv}

definition uwellformed :: ugraph ⇒ bool
where
uwellformed G ≡ (∀ e∈uedges G. card e = 2 ∧ (∀ u ∈ e. u ∈ uverts G))

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

definition uwalk-length :: uwalk ⇒ nat
where
uwalk-length p ≡ length (uwalk-edges p)
definition uwalks :: ugraph ⇒ uwalk set where
uwalks G ≡ {p. set p ⊆ uverts G ∧ set (uwalk-edges p) ⊆ uedges G ∧ p ≠ []}

definition ucycles :: ugraph ⇒ uwalk set where
ucycles G ≡ {p. uwalk-length p ≥ 3 ∧ p ∈ uwalks G ∧ distinct (tl p) ∧ hd p = last p}

definition remove-vertex :: ugraph ⇒ nat ⇒ ugraph where
remove-vertex G u ≡ (uverts G − {u}, uedges G − {A ∈ uedges G. u ∈ 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 ⊆ ws ⇒ all-edges vs ⊆ all-edges ws
using assms unfolding all-edges-def by auto

lemma all-edges-subset-Pow: all-edges A ⊆ Pow A
by (auto simp: all-edges-def)

lemma in-mk-uedge-img: (a,b) ∈ A ∨ (b,a) ∈ A ⇒ {a,b} ∈ mk-uedge ' A
by (auto intro: rev-image-eqI)

lemma distinct-edgesI:
assumes distinct p shows distinct (uwalk-edges p)
proof –
from assms have ?thesis ∧ u. u /∈ set p ⇒ (∀v. u ≠ v ⇒ {u,v} /∈ 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 ⊆ {xs. set xs ⊆ uverts G ∧ length xs ≤ Suc (card (uverts G))}
proof (rule, simp)
fix p assume p ∈ ucycles G
then have distinct (tl p) and set p ⊆ uverts G
unfolding ucycles-def uwalks-def by auto
moreover
then have set (tl p) ⊆ uverts G
by (auto simp: list-set-tl)
with assms have card (set (tl p)) ≤ card (uverts G)
by (rule card_mono)

7
then have \( \text{length}(p) \leq 1 + \text{card}(\text{uverts} \ G) \)
using \text{distinct-card}[\text{OF} \ \text{distinct} \ (\text{tl} \ p)] \ by \ \text{auto}
ultimately show set \ p \subseteq \text{uverts} \ G \land \text{length} \ p \leq \text{Suc} \ (\text{card} \ (\text{uverts} \ G)) \ by 
\text{auto}
\text{qed}
moreover have finite \{xs. \ \text{set} \ xs \subseteq \text{uverts} \ G \land \text{length} \ xs \leq \text{Suc} \ (\text{card} \ (\text{uverts} \ G))\}
using \text{assms} by (\text{rule} \ \text{finite-lists-length-le})
ultimately show \ ?\text{thesis} \ by \ (\text{rule} \ \text{finite-subset})
\text{qed}

lemma \text{ucycles-distinct-edges}:
assumes \( c \in \text{ucycles} \ G \) shows \( \text{distinct} \ (\text{uwalk-edges} \ c) \)
proof –
from \text{assms} have \( c\text{-props}: \text{distinct} \ (\text{tl} \ c) \ \underline{4} \leq \text{length} \ c \ \text{hd} \ c = \text{last} \ c \)
by (\text{auto \ simp \ add: \ ucycles-def \ uwalk-length-conv})
then have \( \{\text{hd} \ c, \ \text{hd} \ (\text{tl} \ c)\} \notin \text{set} \ (\text{uwalk-edges} \ (\text{tl} \ c)) \)
proof (\text{induct} \ c \ \text{rule: \ uwalk-edges}. \ \text{induct})
case \( 3 \ x \ y \ ys \)
them have \( \text{hd} \ ys \neq \text{last} \ ys \) by (cases \( ys \)) \text{auto}
moreover from \( 3 \) have \( \text{uwalk-edges} \ (y \# \ y s) = \{y, \ \text{hd} \ y s\} \neq \text{uwalk-edges} \ y s \)
by (cases \( ys \)) \text{auto}
moreover \{fix \( xs \) have \( \text{set} \ (\text{uwalk-edges} \ xs) \subseteq \text{Pow} \ (\text{set} \ xs) \)
by (\text{induct} \ xs \ \text{rule: \ uwalk-edges}. \ \text{induct}) \ \text{auto} \}
ultimately show \( ?\text{case} \ \text{using} \ 3 \ \text{by} \ \text{auto} \)
\text{qed \ simp-all}
moreover from \text{assms} have \( \text{distinct} \ (\text{uwalk-edges} \ (\text{tl} \ c)) \)
by (\text{intro \ distinct-edgesI}) (\text{simp \ add: \ ucycles-def})
ultimately show \( ?\text{thesis} \ \text{by} \ (\text{cases} \ c \ \text{rule: \ list-exhaust3}) \ \text{auto} \)
\text{qed}

lemma \text{card-left-less-pair}:
fixes \( A :: \{a :: \text{linorder} \} \ \text{set} \)
assumes \text{finite} \( A \)
shows \( \text{card} \ \{(a,b). \ a \in A \land b \in A \land a < b\} = (\text{card} \ A \ast (\text{card} \ A - 1)) \ \text{div} \ 2 \)
using \text{assms}
proof (\text{induct} \ A) \ \text{case} \ (\text{insert} \ x \ A) \ \text{show} \ ?\text{case} \ \text{proof} \ (\text{cases} \ \text{card} \ A) \ \ \text{case} \ (\text{Suc} \ n) \ \text{auto}}
have \( \{(a,b).\ a \in \text{insert } x\ A \land b \in \text{insert } x\ A \land a < b\} \)
\(= \{(a,b).\ a \in A \land b \in A \land a < b\} \cup (\lambda a.\ \text{if } a < x \then (a,x) \else (x,a)) \)’
\(A\)
using \(x \notin A\) by (auto simp: order-less-le)
moreover have finite \(\{(a,b).\ a \in A \land b \in A \land a < b\}\)
using insert by (auto intro: finite-subset[of - A \times A])
moreover have \(\{(a,b).\ a \in A \land b \in A \land a < b\} \cap (\lambda a.\ \text{if } a < x \then (a,x) \else (x,a)) \)’
\(A\)
using \(x \notin A\) by auto
moreover have \(\{(a,b).\ a \in A \land b \in A \land a < b\} \cap (\lambda a.\ \text{if } a < x \then (a,x) \else (x,a)) \)’
\(A\)
= \{\} using \(x \notin A\) by auto
moreover have inj-on \(\lambda a.\ \text{if } a < x \then (a,x) \else (x,a)\) \(A\)
by (auto intro: inj-onI split: split-if-asm)
ultimately show ?thesis using insert Suc
qed (simp add: card-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 \(A = \text{mk-uedge } \{(a,b).\ a \in A \land b \in A \land a < b\}\) (is \(\{L = \{R\}\))
by (auto simp: all-edges-def intro: inj-onI split: split-if-asm)
then have “card \(L = \text{card } R\)” by simp
also have “… = card \(\{(a,b).\ a \in A \land b \in A \land a < b\}\)
using inj-on-mk-uedge by (blast intro: card-image inj-on)
also have “… = (card A \ast (card A - 1)) \div 2
using card-left-less-pair using assms by simp
also have “… = (card A \choose 2)
by (simp add: n-choose-2-nat)
finally show ?thesis .
qed

lemma verts-Gv: uverts \((G -- u)\) = uverts \(G\) \{-u\}
unfolding remove-vertex-def by simp

lemma edges-Gv: uedges \((G -- u)\) \(\subseteq\) uedges \(G\)
unfolding remove-vertex-def by auto

2.2 Girth, Independence and Vertex Colorings

definition girth :: ugraph \(\Rightarrow\) enat where
  “girth \(G = \inf p: \text{ucycles } G.\ \text{enat } (\text{uwalk-length } p)\)”
definition independent-sets :: ugraph \(\Rightarrow\) uvert set set where
  “independent-sets \(Gr = \{\text{vs. } \text{vs} \subseteq \text{uverts } Gr \land \text{all-edges vs } \cap \text{uedges } Gr = \{\}\)”
definition α :: ugraph ⇒ enat where
  α G ≡ SUP vs: independent-sets G. enat (card vs)

definition vertex-colorings :: ugraph ⇒ uvert set set set where
  vertex-colorings G ≡ { C. ∪ C = werts G ∧ (∀ c1∈C. ∃ c2∈C. c1 ≠ c2 → c1 ∩ c2 = {}) ∧ (∀ c∈C. c ≠ {} ∧ (∀ u ∈ c. ∀ v ∈ c. {u,v} ∉ uedges G))}

The chromatic number χ:

definition chromatic-number :: ugraph ⇒ enat where
  chromatic-number G ≡ INF c: (vertex-colorings G). enat (card c)

lemma independent-sets-mono:
  vs ∈ independent-sets G ⇒ us ⊆ vs ⇒ us ∈ 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-α-iff:
  assumes 0 < k
  shows k ≤ α Gr ←→ k ∈ card ' independent-sets Gr (is {?L ←→ ?R)
  proof
    assume ?L
    then obtain vs where vs ∈ independent-sets Gr and k ≤ card vs
      using assms unfolding α-def SUP-def enat-le-Sup-iff by auto
    moreover
    then obtain us where us ⊆ vs and k = card us
      using card-Ex-subset by auto
    ultimately
    have us ∈ independent-sets Gr by (auto intro: independent-sets-mono)
    then show ?R using ⟨k = card us⟩ by auto
  qed (auto intro: SUP-upper simp: α-def)

lemma zero-less-α:
  assumes werts G ≠ {}
  shows 0 < α G
  proof
    from assms obtain a where a ∈ werts G by auto
    then have 0 < enat (card {a}) {a} ∈ independent-sets G
      by (auto simp: independent-sets-def all-edges-def)
    then show ?thesis unfolding α-def less-SUP-iff ..
  qed

lemma α-le-card:
  assumes finite (werts G)
  shows α G ≤ card(werts G)
  proof
    { fix x assume x ∈ independent-sets G
      then have x ⊆ werts G by (auto simp: independent-sets-def) }

10
with assms show ?thesis unfolding α-def
  by (intro SUP-least) (auto intro: card-mono)
qed

lemma α-fin: finite (uverts G) |⇒ α G ≠ ∞
  using α-le-card[of G] by (cases α G) auto

lemma α-remove-le:
  shows α (G -- u) ≤ α G
proof –
  have independent-sets (G -- u) ⊆ independent-sets G (is ?L ⊆ ?R)
    using all-edges-subset-Pow by (simp add: independent-sets-def remove-vertex-def)
    blast
  then show ?thesis unfolding α-def
    by (rule SUP-subset-mono)
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 (uverts G)
  and neG: uverts G ≠ {}
  shows card (uverts G) / α G ≤ chromatic-number G
proof –
  from wf-G have (λv. {v}) ' uverts G ∈ vertex-colorings G
    by (auto simp: vertex-colorings-def uwellformed-def)
  then have chromatic-number G ≠ top
    by (simp add: chromatic-number-def) (auto simp: top-enat-def)
  then obtain vc where vc-vc: vc ∈ vertex-colorings G
    and vc-size:chromatic-number G = card vc
    unfolding chromatic-number-def by (rule enat-in-INF)
  have fin-vc-elems: ∀c. c ∈ vc |⇒ finite c
    using vc-vc by (intro finite-subset[OF - fin-G]) (auto simp: vertex-colorings-def)
  { have vc ⊆ Pow (uverts G) finite (Pow (uverts G))
    using assms vc-vc by (auto simp: vertex-colorings-def)
    then have finite vc by (rule finite-subset)
    with fin-vc-elems have (∑ c ∈ vc. card c) = card (uverts G)
      using vc-vc unfolding vertex-colorings-def
      by (simp add: card-Union-disjoint[symmetric])
  } note sum-vc-card = this
  have ∀c. c ∈ vc |⇒ c ∈ independent-sets G
    using vc-vc by (auto simp: vertex-colorings-def independent-sets-def all-edges-def)
  then have ∀c. c ∈ vc |⇒ card c ≤ α G
    using vc-vc fin-vc-elems by (subst le-α-iff) (auto simp add: vertex-colorings-def)
  then have (∑ c∈vc. card c) ≤ card vc * α G
using setsum-bounded[of vc card α G]
then have ereal-of-enat (card (uverts G)) ≤ ereal-of-enat (α G) * ereal-of-enat (card vc)
by (simp add: sum-vc-card ereal-of-enat-pushout ac-simps del: ereal-of-enat-simps)
by (simp add: ereal-divide-le-pos)
qed

end

theory Girth-Chromatic
imports Ugraphs Girth-Chromatic-Misc
begin

3 Probability Space on Sets of Edges

definition cylinder :: 'a set ⇒ 'a set ⇒ 'a set ⇒ 'a set set where
cylinder S A B = {T ∈ Pow S. A ⊆ T ∧ B ∩ T = {}}

lemma full-sum:
fixes p :: real
assumes finite S
shows (∑A∈Pow S. pˆcard A * (1 − p)ˆcard (S − A)) = 1
using assms
proof (induct)
case (insert s S)
have inj-on (insert s) (Pow S)
  and ∀x. S − insert s x = S − x
  and Pow S ∩ insert s ' Pow S = {}
  and (∀x. x ∈ Pow S ⇒ card (insert s x) = Suc (card (S − x)))
  using insert(1−2) by (auto simp: insert-Diff-if intro!: inj-on1)
moreover have (∀x. x ⊆ S ⇒ card (insert s x) = Suc (card x))
  using insert(1−2) by (subst card.insert) (auto dest: finite-subset)
ultimately show ?case
by (simp add: setsum.reindex setsum-right-distrib[symmetric] ac-simps
insert.hyps setsum.union-disjoint Pow-insert)
qed simp

Definition of the probability space on edges:

locale edge-space =
fixes n :: nat and p :: real
assumes p-prob: 0 ≤ p p ≤ 1
begin

definition S-verts :: nat set where
\[ S\text{-verts} \equiv \{1..n\} \]

**Definition S-edges :: uedge set where**
\[ S\text{-edges} = \text{all-edges } S\text{-verts} \]

**Definition edge-ugraph :: uedge set ⇒ ugraph where**
\[ \text{edge-ugraph } es \equiv (S\text{-verts, es } \cap S\text{-edges}) \]

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

**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 \( P \) = \( \text{Pow } S\text{-edges} \) by (simp add: P-def space-point-measure)

**Lemma sets-eq:** sets \( P \) = \( \text{Pow } (\text{Pow } S\text{-edges}) \) by (simp add: P-def sets-point-measure)

**Lemma emeasure-eq:**
\[ \text{emeasure } P \ A = (\text{if } A \subseteq \text{Pow } S\text{-edges} \text{ then } (\sum_{\text{edges} \in A} p ^\text{card } \text{edges } * (1 - p) ^\text{card } (S\text{-edges } - \text{edges})) \text{ else } 0) \]
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::- ⇒ real)
using finite-edges by (simp add: integrable-point-measure-finite P-def)

**Lemma borel-measurable-P[measurable]:** f ∈ borel-measurable \( P \)
unfolding P-def by simp

**Lemma prob-space-P:** prob-space \( P \)
**Proof**
\[ \text{show } \text{emeasure } P \ (\text{space } P) = 1 — \text{Sum of probabilities equals 1} \]
using finite-edges by (simp add: emeasure-eq full-sum one-ereal-def space-eq)

**Qed**

13
sublocale edge-space ⊆ prob-space P
  by (rule prob-space-P)

context edge-space
begin

lemma prob-eq:
  prob A = (if A ⊆ Pow S-edges then (∑ edges∈A. pˆcard edges * (1 − p)ˆcard (S-edges − edges)) else 0)
  using emeasure-eq[of A] unfolding emeasure-eq-measure by simp

lemma integral-finite-singleton: integralL P f = (∑ x∈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!: setsum.cong)

Probability of cylinder sets:

lemma cylinder-prob:
  assumes A ⊆ S-edges B ⊆ S-edges A ∩ B = {}
  shows prob (cylinder S-edges A B) = p ˆ (card A) * (1 − p) ˆ (card B) (is = ?pp A B)
proof −
  have Pow S-edges ∩ cylinder S-edges A B = cylinder S-edges A B
    ∨ x. x ∈ cylinder S-edges A B ⇒ A ∪ x = x
    ∨ x. x ∈ cylinder S-edges A B ⇒ finite x
    ∨ x. x ∈ cylinder S-edges A B ⇒ B ∩ (S-edges − B − x) = {}
    ∨ x. x ∈ cylinder S-edges A B ⇒ B ∪ (S-edges − B − x) = S-edges − x
    ∨ finite A finite B
    using assms by (auto simp add: cylinder-def intro: finite-subset)
then have (∑ T ∈ cylinder S-edges A B. ?pp T (S-edges − T)) = (∑ T ∈ cylinder S-edges A B. pˆ( (card A + card (T − A)) ∗ (1 − p)ˆ(card B + card ((S-edges − B) − T))))
  using finite-edges by (simp add: card-Un-Int)
also have ... = ?pp A B ∗ (∑ T ∈ cylinder S-edges A B. ?pp (T − A) (S-edges − B − T))
  by (simp add: power-add setsum-right-distrib ac-simps)
also have ... = ?pp A B
proof −
  have (∨ T. T ∈ cylinder S-edges A B ⇒ S-edges − B − T = (S-edges − A) − B − (T − A))
    Pow (S-edges − A − B) = (λx. x − A) ' cylinder S-edges A B
    inj-on (λx. x − A) (cylinder S-edges A B)
    finite (S-edges − A − B)
    using assms by (auto simp: cylinder-def intro!: inj-onI)
  with full-sum[of S-edges − A − B] show ?thesis by (simp add: setsum.reindex) qed
finally show ?thesis by (auto simp add: prob-eq cylinder-def)
Markov-inequality:

\textbf{fixes} \(a::\text{real}\) \textbf{and} \(X::\text{uedge set} \Rightarrow \text{real}\)
\textbf{assumes} \(0 < c \wedge x. 0 \leq f x\)
\textbf{shows} \(\text{prob}\ \{x \in \text{space } P. c \leq f x\} \leq \left(\int x. f x \frac{\partial}{\partial P}\right) / c\)

\textbf{proof} –
\textbf{from} \text{assms} \textbf{have} \(\int x. \text{ereal} (f x) \partial P = \left(\int x. f x \partial P\right)\)
\textbf{by} (intro \text{nn-integral-eq-integral} \text{auto})
\textbf{with} \text{assms} \textbf{show} \(?\text{thesis}\)
\textbf{using} \text{nn-integral-Markov-inequality[of } f P \text{ space } P \text{ 1 / } c]\)
\textbf{by} (simp cong: \text{nn-integral-cong add: emeasure-\text{eq-measure one-ereal-def})}

\textbf{qed}

end

3.1 Graph Probabilities outside of \textit{Edge-Space} locale

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

\textbf{abbreviation} \(\text{MGn}::(\text{nat} \Rightarrow \text{real}) \Rightarrow \text{nat} \Rightarrow (\text{uedge set}) \Rightarrow \text{measure}\)
\textbf{where} \(\text{MGn} p n \equiv (\text{edge-space}. \text{P n} (p n))\)

\textbf{abbreviation} \(\text{probGn}::(\text{nat} \Rightarrow \text{real}) \Rightarrow \text{nat} \Rightarrow (\text{uedge set} \Rightarrow \text{bool}) \Rightarrow \text{real}\)
\textbf{where} \(\text{probGn} p n P \equiv \text{measure} (\text{MGn} p n)\ \{\text{es} \in \text{space} (\text{MGn} p n). P \text{ es}\}\)

\textbf{lemma} \(\text{probGn-le}\):
\textbf{assumes} \(p-\text{prob}: 0 < p n p n < 1\)
\textbf{assumes} \(\text{sub}: \forall n \text{ es}. \text{es} \in \text{space} (\text{MGn} p n) \Rightarrow P n \text{ es} \Rightarrow Q n \text{ es}\)
\textbf{shows} \(\text{probGn} p n (P n) \leq \text{probGn} p n (Q n)\)

\textbf{proof} –
\textbf{from} \(p-\text{prob}\) \textbf{interpret} \(E: \text{edge-space n p n}\) \textbf{by} unfold-locales \text{auto}
\textbf{show} \(?\text{thesis}\)
\textbf{by} (auto intro!: \(E.\text{finite-measure-mono sub simp: E.space-\text{eq E.sets-\text{eq})}\))

\textbf{qed}

4 Short cycles

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

obtains a vertex in a short cycle:

\textbf{definition} \(\text{choose-v}::\text{ugraph} \Rightarrow \text{nat} \Rightarrow \text{uvert}\)
\textbf{where} \(\text{choose-v} G k \equiv \text{SOME } u. \exists p. p \in \text{short-cycles } G k \land u \in \text{set } p\)

\textbf{partial-function} (\text{tailrec}) \(\text{kill-short}::\text{ugraph} \Rightarrow \text{nat} \Rightarrow \text{ugraph}\)
\textbf{where} \(\text{kill-short} G k = (\text{if } \text{short-cycles } G k = \{\} \text{ then } G \text{ else } (\text{kill-short} (\text{G }\leftarrow (\text{choose-v } G k)) \ k))\)
lemma \textit{ksc-simps}[simp]:
short-cycles \ G k = \{\} \implies \text{kill-short} \ G k = \ G
short-cycles \ G k \neq \{\} \implies \text{kill-short} \ (\ G \shorttext{-- (choose-v} \ G k)) \ k
by \ (\text{auto simp: \textit{kill-short.simps}})

lemma
assumes \shorttext{short-cycles} \ G k \neq \{\}
shows \ \shorttext{choose-v--in-uverts}: \text{choose-v} \ G k \in \text{werts} \ G \ (\text{is \ ?t1})
and \ \shorttext{choose-v--in-short}: \exists \ p. \ p \in \shorttext{short-cycles} \ G k \ \wedge \ \text{choose-v} \ G k \in \text{set} \ p \ (\text{is \ ?t2})
proof –
from \textit{assms} obtain \ p \ where \ p \in \text{ucycles} \ G \ \text{uw-length} \ p \leq \ k

unfolding \ \shorttext{short-cycles-def} \ by \ \text{auto}
moreover
then obtain \ u \ where \ u \in \text{set} \ p \ unfolding \ \shorttext{ucycles-def}
by \ (\cases \ p) \ (\text{auto simp: \textit{uw-length-conv}})
ultimately have \ \exists \ u. \ p \in \shorttext{short-cycles} \ G k \ \wedge \ u \in \text{set} \ p
by \ (\text{auto simp: \shorttext{short-cycles-def}})
then show \ ?t2 \ by \ (\text{auto simp: \shorttext{choose-v-def intro: someI-ex}})
then show \ ?t1 \ by \ (\text{auto simp: \shorttext{short-cycles-def \ ucycles-def \ uwalks-def}})
qed

lemma \textit{kill-step-smaller}:
assumes \shorttext{short-cycles} \ G k \neq \{\}
shows \shorttext{short-cycles} \ (\ G \shorttext{-- \ choose-v} \ G k) \ k \subset \shorttext{short-cycles} \ G k
proof –
let \ ?cv = \text{choose-v} \ G k
from \textit{assms} obtain \ p \ where \ p \in \shorttext{short-cycles} \ G k \ ?cv \in \text{set} \ p
by \ \text{atomize-elim \ (rule \ \textit{choose-v--in-short})}

have \shorttext{short-cycles} \ (\ G \shorttext{-- ?cv}) \ k \subset \shorttext{short-cycles} \ G k
proof
fix \ p \ assume \ p \in \shorttext{short-cycles} \ (\ G \shorttext{-- ?cv}) \ k
then show \ p \in \shorttext{short-cycles} \ G k

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

Induction rule for \textit{kill-short}:

lemma \textit{kill-short-induct}[consumes 1, case-names empty kill-vert]:
assumes \ \textit{fin}: \text{finite} \ (\text{werts} \ G)
assumes \ a-empty: \ \bigwedge \ G. \shorttext{short-cycles} \ G k = \{\} \implies P \ G k
assumes \ a-kill: \ \bigwedge \ G. \text{finite} \ (\shorttext{short-cycles} \ G k) \implies \shorttext{short-cycles} \ G k \neq \{\}

16
 shows $P G k$

proof
  have finite (short-cycles $G k$)
    using finite-u-cycles[OF fin] by (auto simp: short-cycles-def)
  then show ?thesis
    by (induct short-cycles $G k$ arbitrary: $G$ rule: finite-psubset-induct)

qed

Large Girth (after kill-short):

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

Order of graph (after kill-short):

lemma kill-short-order-of-graph:
  assumes finite (uverts $G$)
  shows $\text{card (uverts } G) - \text{card (short-cycles } G k) \leq \text{card (uverts (kill-short } G k))$
  using assms assms
  proof (induct $G k$ rule: kill-short-induct)
    case (kill-vert $G$)
    let $?oG = G -- (\text{choose-v } G k)$
    have finite (uverts $?oG$)
      using kill-vert by (auto simp: remove-vertex-def)
    moreover
    have uverts (kill-short $G k) = uverts (kill-short $?oG k)$
      using kill-vert by simp
    moreover
    have card (uverts $G) = \text{Suc (card (uverts } ?oG))$
      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-α:
assumes finite (uverts G)  
shows α (kill-short G k) ≤ α G  
using assms  
proof (induct G k rule: kill-short-induct)  
case (kill-vert G)  
ote kill-vert(3)  
also have α (G −− (choose-v G k)) ≤ α G by (rule α-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 ≥ k k ≥ 2  
shows prob {es ∈ space P. k ≤ α (edge-ugraph es)}  
   ≤ (n choose k) * (1 − p)^k (k choose 2)  
proof —  
let ℱk-sets = {vs. vs ⊆ S-verts ∧ card vs = k}

{ fix vs assume A: vs ∈ ℱk-sets  
then have B: all-edges vs ⊆ S-edges  
   unfolding all-edges-def S-edges-def by blast  
   have {es ∈ space P. vs ∈ 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)^(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 ∈ space P. k ∈ card ` independent-sets (edge-ugraph es)}  
   = (∪ vs ∈ ℱk-sets. {es ∈ space P. vs ∈ independent-sets (edge-ugraph es)}) (is ℱL = ℱR)
unfolding image-def space-eq independent-sets-def by auto
then have prob \( L \leq (\sum_{es \in \?k-sets. \text{prob} \{ es \in \text{space} \ P, \text{vs} \in \text{independent-sets} (\text{edge-ugraph \ es)}\})
  by (auto intro!: finite-measure-subadditive-finite simp: space-eq sets-eq)
also have \( \ldots = (n \choose k) \cdot ((1 - p) \text{^} (k \choose 2)) \)
  by (simp add: prob-k-indep real-eq-of-nat S-verts-def n-subsets)
finally show \( \alpha \text{thesis using } k \geq 2 \) by (simp add: le-a-iff)
qed

Almost never many independent edges:

lemma almost-never-le-\( \alpha \):
fixes \( k \) :: nat
and \( p \) :: nat \Rightarrow real
assumes p-prob: \( \forall \infty. \ n \cdot 0 < p \cdot n \land p \cdot n < 1 \)
assumes [arith]: \( k > 0 \)
assumes N-prop: \( \forall \infty. \ (6 \cdot k \star \ln n) / n \leq p \cdot n \)
shows \((\lambda n. \text{probG} \ n \ p \ n (\lambda e. \ 1 / 2 \star n / k \leq \alpha (\text{edge-space. edge-ugraph \ n \ es}))\)
  \(\ldots\rightarrow 0 \)
(is \((\lambda n. \text{prob-fun-raw \ n}) \ldots\rightarrow 0 \))
proof -
let \( \text{prob-fun-raw \ n = probG} \ n \ p \ n (\lambda e. \ 1 / 2 \star n / k \leq \alpha (\text{edge-space. edge-ugraph \ n \ es})) \)
def \( r \equiv \lambda n :: \text{nat}. \ (1 / 2 \star n / k) \)
let \( \text{nr} \ n = \text{nat}(\text{ceiling} \ (r \ n)) \)
have \( r \text{-pos: } \land n. \ 0 < n \Longrightarrow 0 < r \cdot n \) by (auto simp: r-def field-simps)
have \( \text{nr-bounds: } \forall \infty. \ 2 \leq \text{nr} \ n \land \text{nr} \ n \leq n \)
  by (intro eventually-sequentiallyI[of 4 * k])
  (simp add: r-def nat-ceiling-le-eq le-natceiling-iff field-simps)
from \( \text{nr-bounds \ p-prob} \) have ev-prob-fun-raw-le:
\( \forall \infty. \ \text{probG} \ n \ p \ n (\lambda e. \ 1 / 2 \star n / k \leq \alpha (\text{edge-space. edge-ugraph \ n \ es})) \)
\( \leq (n \star \exp (-p \star n \star (\text{real} \ (\text{nr} \ n) - 1) / 2)) \ \text{pwr} \ ?nr \ n \)
(is \( \forall \infty. \ ?\text{prob-fun-raw-le \ n} \))
proof (rule eventually-elim2)
fix \( n :: \text{nat} \) assume \( A: \ 2 \leq \text{nr} \ n \land \text{nr} \ n \leq n \)
  \(0 < p \cdot n \land p \cdot n < 1 \)
then interpret \( pG: \text{edge-space \ n \ p \ n} \) by unfold-locales auto
have \( r: \ \text{real} \ (\text{nr} \ n - \text{Suc} \ 0) = \text{real} \ (\text{nr} \ n) - \text{Suc} \ 0 \) using \( A \) by auto
have \( \text{probG} \ n \ p \ n (\lambda e. \ 1 / 2 \star n / k \leq \alpha (\text{edge-space. edge-ugraph \ n \ es})) \)
  \(\leq (n \choose \text{nr} \ n) \star (1 - p \ n) \cdot ((\text{nr} \ n \ choose \ 2)) \)
using \( A \) by (auto intro: pG.random-prob-independent)
also have \( \ldots \leq n \ \text{pwr} \ ?\text{nr} \ n \star (1 - p \ n) \ \text{pwr} \ (\text{nr} \ n \ choose \ 2) \)
using \( A \)
by (simp add: powr-realpow power-real-of-nat binomial-le-pow del: real-of-nat-power)
also have \( \ldots = n \ \text{pwr} \ ?\text{nr} \ n \star (1 - p \ n) \ \text{pwr} \ (\text{nr} \ n \star (\text{nr} \ n - 1) / 2) \)
by (cases even (?nr n - 1))
(auto simp add: n-choose-2-nat real-of-nat-div)
also have \( \ldots = n \, \text{pwr} \, (?nr n \ast ((1 - p \, n) \, \text{pwr} \, ((?nr n - 1) / 2)) \, \text{pwr} \, ?nr n) \)
by (auto simp add: powr-powr r ac-simps)
also have \( \ldots \leq (n \ast \exp (- p \, n \ast (?nr n - 1) / 2)) \, \text{pwr} \, ?nr n \)
proof
have \((1 - p \, n) \, \text{pwr} \, ((?nr n - 1) / 2) \leq \exp (- p \, n) \, \text{pwr} \, ((?nr n - 1) / 2)) \)
using \( A \) by (auto simp: powr-mono2 diff-conv-add-uminus simp del: add-uminus-conv-diff)
also have \( \ldots = \exp (- p \, n \ast (?nr n - 1) / 2) \) by (auto simp: powr-def)
finally show \( \alpha \)
using \( A \) by (auto simp: powr-mono2 powr-mult)
qed
finally show \( \text{probGn} \, p \, n \) (fix \( A \) \( r \) by simp)
proof
have \( n \ast \exp (- p \, n \ast (\real ((?nr n - 1) / 2)) \, \text{pwr} \, ?nr n) \)
also have \( \ldots \leq (3 / 2) \ast \ln n \) (fix \( A \) \( r \) by simp)
finally show \( \alpha \)
using \( A \) \( r \) by (auto simp: field-simps)
qed

from \( p \)-prob \( N \)-prop

have cv-expr-bound: \( \forall \infty \, n \ast \exp (- p \, n \ast (\real ((?nr n) - 1) / 2) \leq (\exp 1 / n) \, \text{pwr} \, (1 / 2) \)
proof (elim eventually-rev-mp, intro eventually-sequentiallyI conjI impI)
fix \( n \) assume \( n \)-bound[arith]: \( 2 \leq n \)
and \( p \)-bound: \( 0 < p \, n \wedge p \, n < 1 \, (6 * k \ast \ln n) / \leq n \)
have \( r \)-bound: \( r \, n \leq (?nr n) \) by (rule real-nat-ceiling-ge)
also have \( n \ast \exp (- 3 / 2 \ast \ln n + p \, n / 2) \)
proof
have \( 0 < \ln n \) using \( n \)-bound by auto
then have \((3 / 2) \ast \ln n \leq ((6 * k \ast \ln n) / n) \ast (?nr n / 2) \)
using \( r \)-bound real-of-int-ceiling-ge[of n / 2 * k]
by (simp add: r-def field-simps del: real-of-int-ceiling-ge)
also have \( \ldots \leq p \, n \ast (?nr n / 2) \)
using \( n \)-bound \( r \)-bound \( r \)-pos[of n] by (auto simp: field-simps)
finally show \( \alpha \) using \( r \)-bound by (auto simp: field-simps)
qed
also have \( \ldots \leq n \ast \text{powr} (- 3 / 2) \ast \exp 1 \, \text{powr} (1 / 2) \)
using \( p \)-bound by (simp add: powr-def exp-add [symmetric])
also have \( \ldots \leq n \ast \text{powr} (-1 / 2) \ast \exp 1 \, \text{powr} (1 / 2) \) by (simp add: powr-mult-base)
also have \( \ldots = (\exp 1 / n) \, \text{powr} (1/2) \)
by (simp add: powr-divide powr-minus-divide)
finally show \( n \ast \exp (- p \, n \ast (\real (?nr n) - 1) / 2) \leq (\exp 1 / n) \, \text{powr} (1 / 2) \).
qed
have ceil-bound: \( G n. 1/2 \times n/k \leq \alpha G \mapsto \text{nat(ceiling } (1/2 \times n/k) \leq \alpha G \)

by (case-tac \( \alpha \) G) (auto simp: nat-ceiling-le-eq)

show \( \text{thesis} \)

proof (unfold ceil-bound, rule real-tendsto-sandwich)

  show \((\lambda n. \exp 1/n) \text{ powr } (1/2)\) ----> 0

  \(\forall \infty \cdot 0 \leq \text{prob-fun-raw } n\)

  using p-prob by (auto intro: measure-nonneg LIMSEQ-inv-powr elim: eventually-elim1)

next

from nr-bounds ev-expr-bound ev-prob-fun-raw-le

show \(\forall \infty \cdot \text{prob-fun-raw } n \leq (\exp 1/n) \text{ powr } (1/2)\)

proof (elim eventually-rev-mp, intro eventually-sequentiallyI impI conjI)

fix \( n \) assume \( A: 3 \leq n \)

and nr-bounds: \( 2 \leq ?nr n \land ?nr n \leq n \)

and prob-fun-raw-le: \( \text{prob-fun-raw-}le \)

and expr-bound: \( n \times \exp (-p n \times (\text{real } (\text{nat(ceiling } (r n)) - 1)/2) \leq (\exp 1/n) \text{ powr } (1/2)\)

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 \( \text{prob-fun-raw } n \leq (n \times \exp (-p n \times (\text{real } (?nr n) - 1)/2)) \text{ powr (?)nr n} \)

using prob-fun-raw-le by (simp add: r-def)

also have \( \ldots \leq ((\exp 1/n) \text{ powr } (1/2)) \text{ powr } ?nr n \)

using expr-bound \( A \) by (auto simp: powr-mono2)

also have \( \ldots \leq ((\exp 1/n) \text{ powr } (1/2)) \)

using nr-bounds ep-bound by (auto simp: powr-le-one-le)

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

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 \( \{ \text{es. card } \{ c \in \text{ucycles } (\text{edge-ugraph es}) \text{. uwalk-length } c = k \} \text{ \( \cap \) P} \)

\( = \text{of-nat } (\text{fact } n \text{ div fact } (n - k)) \times p \cdot k \)

proof

let \( \text{\#k-cycle } = \lambda \text{es. } c \cdot (c \in \text{ucycles } (\text{edge-ugraph es}) \land \text{uwalk-length } c = k \)

def \( C \equiv \lambda k. \{ c. \text{\#k-cycle } S\text{-edges } c \cdot k \}

\( C k \) is the set of all possible cycles of size \( k \) in \( \text{edge-ugraph } S\text{-edges} \)

def \( XG \equiv \lambda \text{es. } c \cdot \text{\#k-cycle } es \cdot c \cdot k \)

\( XG es \) is the set of cycles contained in a \( \text{edge-ugraph } es \)

def \( XC \equiv \lambda c. \{ es \in \text{space } P. \text{\#k-cycle es } c \cdot k \} \)
— "XC c is the set of graphs (edge sets) containing a cycle c"

then have \( \forall c. \text{XC} c \in \text{sets} \ P \)
and \( \text{XC-cyl} : \forall c. c \in C \implies \text{XC} c = \text{cylinder-edges (set (uwalk-edges c))} \)

\[
\begin{array}{l}
\text{by (auto simp: ucycles-def space-eq uwalks-def C-def cylinder-def sets-eq)}
\end{array}
\]

have \( \{ \sum \in \text{space } P. \text{card (XG x) * prob } \{x\}\} \)
by (simp add: XG-def integral-fin-sing space-eq)
also have \( \ldots = (\sum \in \text{space } P. \text{card (XG x) * prob } \{x\}\) \{x\})

\[
\begin{array}{l}
\text{proof –}
\end{array}
\]

have \( \text{XG-Int-C} : \forall s. s \in \text{space } P \implies C k \cap \text{XG} s = \text{XG} s \)

unfolding \( \text{XG-def C-def ucycles-def uwalks-def edge-ugraph-def by auto} \)

have \( \text{fin-XC} : \forall k. \text{finite (XG k)} \) and \( \text{finite (C k)} \)

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

have \( \ldots = (\sum \in \text{space } P. \text{card (XG x) * prob } \{x\}\) \{x\})
by (simp add: real-ev-of-nat)
also have \( \ldots = (\sum \in \text{space } P. (\sum \in \text{C k. if c \in XG x then prob } \{x\}\) else \(0\))\)

using \( \text{fin-C by (simp add: setsum.If-cases) (simp add: XG-Int-C)} \)
also have \( \ldots = (\sum \in \text{C k. (\sum x \in \text{space } P \cap \text{XC c. prob } \{x\}\) \{x\})} \)
using \( \text{finite-edges by (subst setsum.commute) (simp add: setsum.inter-restrict}} \)

\( \text{XG-def XC-def space-eq)} \)
also have \( \ldots = (\sum \in \text{C k. prob (XC c)}\)
using \( \text{fin-XC XC-in-sets} \)
by (auto simp add: prob-ev sets-ev space-ev intro!: sets-ev_cong)
finally show \( \text{?thesis by (simp add: XC-cyl)} \)

\( \text{qed} \)
also have \( \ldots = (\sum \in \text{C k. p * k)}\)

\( \text{proof –} \)

have \( \forall x. x \in C k \implies \text{card (set (uwalk-edges x)) = uwalk-length x} \)
by (auto simp: uwalk-length-def C-def ucycles-distinct-edges intro: distinct-card)
then show \( \text{?thesis by (auto simp: C-def ucycles-def uwalks-def cylinder-prob)} \)

\( \text{qed} \)
also have \( \ldots = \text{of-nat (fact n div fact (n - k)) * p * k} \)

\( \text{proof –} \)

have \( \text{inj-last-Cons} : \forall A. \text{inj-on (\lambda x. last xs \# es) A by (rule inj-on1) simp} \)
\{ fix xs A assume 3 \leq length xs - Suc 0 hd xs = last xs \then have \( \in (\lambda x. \text{last xs \# es) A \rightleftarrows \text{tl xs \in A} \)
by (cases xs) (auto simp: inj-image-mem-iff[OF inj-last-Cons] split: split-if-asm)
\}
\note {image-mem-iff-inst = this}

\{ fix xs have \( \in \text{uwalks (edge-ugraph S-edges) \rightleftarrows set (tl xs) \subseteq S-verts} \)
unfolding \( \text{uwalks-def by (induct xs) auto} \)

moreover \{ fix xs assume \( \text{set xs \subseteq S-verts 2 \leq length xs distinct xs} \)

\}
then have \((\text{last } xs \neq xs) \in \text{uwalks} (\text{edge-ugraph } S\text{-edges})\)

proof (induct \(xs\) rule: \(\text{uwalk-edges\_induct}\))

- case \((x \ y \ ys)\)

  - have \(S\text{-edges\_memI}: \forall x \ y. x \in S\text{-verts} \Rightarrow y \in S\text{-verts} \Rightarrow x \neq y \Rightarrow \{x, y\} \in S\text{-edges}\)

    unfolding \(S\text{-edges\_def}\) \(\text{all-edges\_def}\) \(\text{image\_def}\) by \(\text{auto}\)

  - have \(ys \neq [] \Rightarrow \text{set } ys \subseteq S\text{-verts} \Rightarrow \text{last } ys \in S\text{-verts} \Rightarrow \text{by } \text{auto}\)

- with \(3\) show \(?\text{case}\)

  - by (\(\text{auto simp add: uwalks\_def Suc\_le\_eq intro: } S\text{-edges\_memI}\))

qed \(\text{simp\_all}\)
by (auto simp: p-def)

qed

then

have \prob\short\count\leq; \forall n. \prob\G n p n (\lambda e. (\real n/2) \leq \short\count (~?ug n es))
\leq 2 \ast (k - 2) \ast n \text{ pour} (\varepsilon \ast k - 1) \text{ (is} \forall n. ?P n)

proof (elim eventually-rev-mp, intro eventually-sequentiallyI impI)

fix n :: nat

assume A : Suc k \leq n 0 < p n \land p n < 1

then interpret pG: edge-space n p n by unfold-locales auto

have 1 \leq n using A by auto

def mean\short\count \equiv \int es. \short\count (~?ug n es) \partial pG.P

have mean\short\count\leq: mean\short\count \leq (k - 2) \ast n \text{ pour} (\varepsilon \ast k)

proof

have small-empty: \forall k. k \leq 2 \implies \short\cycles (\edge-space.\edge-ugraph n es) k = {} by (auto simp add: \short\cycles-def ucycles-def)

have short\count\conv: \forall es. short\count (~?ug n es) = (\sum i=3..k. \real (\card {c \in ucycles (?ug n es). uwalk-length c = i}))

proof (unfold short\count-def, induct k)

  case 0 with small-empty show ?case by auto

next

  case (Suc k)

  show ?case proof (cases Suc k \leq 2)

  case True with small-empty show ?thesis by auto

next

  case False

  have \{c \in ucycles (?ug n es). uwalk-length c \leq Suc k\}
  = \{c \in ucycles (?ug n es). uwalk-length c \leq k\} \cup \{c \in ucycles (?ug n es). uwalk-length c = Suc k\}

  by auto

  moreover

  have finite (\uverts (\edge-space.\edge-ugraph n es)) by auto

  ultimately

  have \card {c \in ucycles (?ug n es). uwalk-length c \leq Suc k}
  = \card {c \in ucycles (?ug n es). uwalk-length c \leq k} + \card {c \in ucycles (?ug n es). uwalk-length c = 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 es. \card {c \in ucycles (?ug n es). uwalk-length c = i}) \partial pG.P

unfolding mean\short\count-def short\count\conv

by (subst integral-setsum) (auto intro: pG.integral-finite-singleton)

also have \ldots = (\sum i\in\{3..k\}. of-nat (\fact n \div \fact (n - i)) \ast p n \ast i)
using \( A \) by (simp add: \( pG.\text{mean-k-cycles} \))
also have \( \ldots \leq \left( \sum_{i \in \{3..k\}} n^i \right) \)
using \( \text{fact-div-fact-le-pow} \)
by (auto intro: setsum-mono simp: real-of-nat-def)
also have \( \ldots \leq \left( \sum_{i \in \{3..k\}} n \text{ powr } (\varepsilon \times k) \right) \)
using (\( 1 \leq n \) \( \theta < \varepsilon \) \( A \))
by (intro setsum-mono) (auto simp: p-def field-simps powr-mult-base)
\[
\text{powr-powr[\text{symmetric}] powr-mult[\text{symmetric}] powr-add[\text{symmetric}]}\]
finally show \(?\text{thesis} by (simp add: real-eq-of-nat)\)
qed

have \( pG.\text{prob} \{ es \in \text{space pG.P.n/2} \leq \text{short-count} (\varepsilon \text{ug} n \text{es}) \} \leq \text{mean-short-count / (n/2)} \)
unfolding \( \text{mean-short-count-def using (1 \leq n)} \)
by (intro \( pG.\text{Markov-inequality} \)) (auto simp: \( \text{short-count-def} \))
also have \( \ldots \leq 2 * (k - 2) * n \text{ powr } (\varepsilon \times k - 1) \)
proof –
have \( \text{mean-short-count / (n / 2)} \leq 2 * (k - 2) * (1 / n \text{ powr 1}) * n \text{ powr } (\varepsilon \times k) \)
using \( \text{mean-short-count-le (1 \leq n) by (simp add: field-simps)} \)
then show \(?\text{thesis} by (simp add: powr-divide2[\text{symmetric}] algebra-simps)\)
qed
finally show \(?\text{P n.} \)
qed

def \( \text{pf-short-count} \equiv \lambda n. \text{probGn p n (\lambda es. n/2} \leq \text{short-count (\varepsilon \text{ug} n \text{es})} \)
and \( \text{pf-\alpha} \equiv \lambda n. \text{probGn p n (\lambda es. 1/2} * n/k \leq \alpha (\text{edge-space.\text{edge-ugraph n es})} \)

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

have \( \text{lim-\alpha: pf-\alpha ----> 0} \)
proof –
have \( 0 < k \) using (\( 3 \leq k \) \) by simp

have \( \forall \infty n. (6\times k) * \ln n / n \leq p n \leftarrow (6\times k) * \ln n * n \text{ powr } \varepsilon \leq 1 \)
proof (rule eventually-sequentiallyI)
fix \( n :: \textbf{nat} \) assume \( 1 \leq n \)

then have \((6 \cdot k) \cdot \ln n / n \leq p \cdot n \iff (6\cdot k) \cdot \ln n * (n \text{ powr } - 1) \leq n \text{ powr } (\varepsilon - 1)\) 

by (subst \text{ powr-minus}) \(\text{simp add: divide-inverse \( p\text{-def} \)}\)

also have \( \ldots \iff (6\cdot k) \cdot \ln n * ((n \text{ powr } - 1) / (n \text{ powr } (\varepsilon - 1))) \leq n \text{ powr } (\varepsilon - 1) / (n \text{ powr } (\varepsilon - 1))\)

using \( \{1 \leq n \} \) by (auto \text{ simp: field-simps})

also have \( \ldots \iff (6\cdot k) \cdot \ln n * n \text{ powr } - \varepsilon \leq 1 \)

apply (simp add: \text{ powr-divide2})

using \( \{1 \leq n \} \) apply simp

done

finally show \((6\cdot k) \cdot \ln n / n \leq p \cdot n \iff (6\cdot k) \cdot \ln n * n \text{ powr } - \varepsilon \leq 1\) .

qed

then have \( (\forall \infty n. (6 \cdot k) \cdot \ln n / \text{ real } n \leq p \cdot n) \)

\( \iff (\forall \infty n. (6\cdot k) \cdot \ln n * n \text{ powr } - \varepsilon \leq 1) \)

by (rule eventually-subst)

also have \( \forall \infty n. (6\cdot k) \cdot \ln n * n \text{ powr } - \varepsilon \leq 1 \)

proof –

\{ fix \( n :: \textbf{nat} \) assume \( 0 < n \)

have \( \ln (\text{ real } n) \leq n \text{ powr } (\varepsilon / 2) / (\varepsilon / 2) \)

using \( \{0 < n\} \) \(\{0 < \varepsilon\} \) by (intro \text{ ln-power-bound}) \text{ auto}

also have \( \ldots \leq 2 / \varepsilon * n \text{ powr } (\varepsilon / 2) \) by (auto \text{ simp: field-simps})

finally have \((6\cdot k) \cdot \ln n * (n \text{ powr } - \varepsilon) \leq (6\cdot k) * (2 / \varepsilon * n \text{ powr } (\varepsilon / 2)) \)

* \((n \text{ powr } - \varepsilon)\)

using \( \{0 < n\} \) \(\{0 < k\} \) by (intro \text{ mult-right-mono} \text{ mult-left-mono}) \text{ auto}

also have \( \ldots = 12 * k / \varepsilon * n \text{ powr } (-\varepsilon / 2) \)

unfolding \text{ divide-inverse}

by (auto \text{ simp: field-simps powr-minus[symmetric] powr-add[symmetric]})

finally have \((6\cdot k) \cdot \ln n * (n \text{ powr } - \varepsilon) \leq 12 * k / \varepsilon * n \text{ powr } (-\varepsilon / 2)\) .

\}

then have \( \forall \infty n. (6\cdot k) \cdot \ln n * (n \text{ powr } - \varepsilon) \leq 12 * k / \varepsilon * n \text{ powr } (-\varepsilon / 2) \)

by (intro eventually-sequentially\{of \( I\}) \text{ 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 \( \{0 < \varepsilon\} \) by (intro \text{ tendsto-mult-right-zero} \text{ LIMSEQ-neg-powr}) \text{ auto}

then show \( ?\text{thesis} \)

using \( \{0 < \varepsilon\} \) by (auto \text{ elim: eventually-elim} \text{ simp: dist-real-def} \text{ dest!: tendstoD where \( e=1\})\)

qed

finally \( (\text{eventually-le-le}) \) show \( ?\text{thesis} \).

qed

finally have \( \forall \infty n. \text{ real } (6 \cdot k) \cdot \ln (\text{ real } n) / \text{ real } n \leq p \cdot n \).

with \( \text{ev-p} \) \(0 < k\); show \( ?\text{thesis}\)

unfolding \text{ pf-\( \alpha\)-def} \text{ by (rule almost-never-le-\( \alpha\)}

qed

from \text{ ev-short-count-le lim-\( \alpha\}[THEN tendstoD, of \( 1/2\]) \text{ ev-p}

have \( \forall \infty n. 0 < p \cdot n \land p \cdot n < 1 \land \text{ pf-short-count } n < 1/2 \land \text{ pf-\( \alpha\) } n < 1/2 \)

by \text{ simp (elim eventually-rev-mp, auto simp: eventually-sequentially dist-real-def)}
then obtain \( n \) where \( 0 < p n p n < 1 \) and \( \{ \text{arith} \}: 0 < n \\
and \text{probs}: \text{pf-short-count } n < 1/2 \text{ pf-} \alpha \ n < 1/2 \\
by (\text{auto simp: eventually-sequentially}) \\
then interpret \( E S: \text{edge-space } n (p n) \) by unfold-locales auto

have \( \text{rest-compl: } \bigwedge A \ P. A - \{ x \in A. P x \} = \{ x \in A. \neg P x \} \) by blast

from \text{probs have } \text{ES.prob } \{ \{ es \in \text{space } E S. P \ - \{ \{ es \in \text{space } E S. P \ - \{ \{ es \in \text{space } E S. P \ - \{ \{ es \in \text{space } E S. P \ - \{ \{ es \in \text{space } E S. P \} \} \} \right\} \text{is } 0 < \text{ES.prob } ?S \} \\
\text{by (subst ES.prob-compl) auto \\
also have } ?S = \{ \{ es \in \text{space } E S. P \ - \{ \{ es \in \text{space } E S. P \ - \{ \{ es \in \text{space } E S. P \} \} \} \} \text{is } \ldots = ?C \} \\
\text{by (auto simp: not-less rest-compl) \\
finally have } ?C \neq \{ \} \) by (intro notI) (simp only:, auto)

then obtain \( es \) where \( \text{es-props: } es \in \text{space } E S. P \)  \\
short-count \( \{ \{ es \in \text{space } E S. P \} \) < \( n \) \( \alpha \) \( \{ \{ es \in \text{space } E S. P \} \) < \( 1/2 * \ n/k \) \\
by auto \\
— now we obtained a high colored graph (few independent nodes) with almost no short cycles

\begin{align*}
\text{def } G &= \{ \{ es \} \} \text{finite } (\text{uverts } G) \text{short-count } G < \( n/2 \) \alpha \ G < \\
& \ 1/2 * \ n/k \\
\text{unfolding } G-\text{def using es-props by (auto simp: ES.S-verts-def)}
\end{align*}

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

have \( \text{enat } l \leq \text{enat } k \) by simp \\
also have \( \ldots < \text{girth } H \) using \( G-\text{props} \) by (auto simp: kill-short-large-girth H-def)

finally have \( T2: l < \text{girth } H \).

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

then have \( \text{uverts-}H; \text{uverts } H \neq \{ \} \) \( 0 < \text{card } (\text{uverts } H) \) by auto
then have \( 0 < \alpha \ H \) using zero-less-\( \alpha \) \text{uverts-}H \) by auto
have $\alpha$-HG: $\alpha H \leq \alpha G$

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

have $\text{enat } l \leq \text{ereal } k$ using $\langle l \leq k \rangle$ by auto
also have $\ldots < \langle n/2 \rangle / \alpha G$ using $G$-props $\langle 3 \leq k \rangle$
  by (cases $\alpha G$) (auto simp: real-of-nat-def[symmetric] field-simps)
also have $\ldots \leq \langle n/2 \rangle / \alpha H$ using $\alpha$-HG $\langle 0 < \alpha H \rangle$
  by (auto simp: real-of-enat-pushout intro!: real-divide-left-mono)
also have $\ldots \leq \text{chromatic-number } H$ using $\text{werts- } H T1$ by (intro chromatic-lb)
  auto
finally have $\text{T3: } l < \text{chromatic-number } H$
  by (simp del: real-of-enat-simps)

from $T1$ $T2$ $T3$ show $\text{?thesis by fast}$
qed

end

References