

Expander Graphs

Emin Karayel

February 6, 2026

Abstract

Expander Graphs are low-degree graphs that are highly connected. They have diverse applications, for example in derandomization and pseudo-randomness, error-correcting codes, as well as pure mathematical subjects such as metric embeddings. This entry formalizes the concept and derives main theorems about them such as Cheeger's inequality or tail bounds on distribution of random walks on them. It includes a strongly explicit construction for every size and spectral gap. The latter is based on the Margulis-Gabber-Galil graphs and several graph operations that preserve spectral properties. The proofs are based on the survey papers/monographs by Hoory et al. [4] and Vadhan [11], as well as results from Impagliazzo and Kabanets [5] and Murtagh et al. [9]

Contents

1	Introduction	2
2	Preliminary Results	2
2.1	Constructive Chernoff Bound	2
2.2	Congruence Method	4
2.3	Multisets	5
3	Definitions	7
4	Setup for Types to Sets	13
5	Algebra-only Theorems	15
6	Spectral Theory	20
7	Cheeger Inequality	27
8	Margulis Gabber Galil Construction	29
9	Random Walks	35
10	Graph Powers	38
11	Strongly Explicit Expander Graphs	41
12	Expander Walks as Pseudorandom Objects	45

1 Introduction

A good introduction into Expander Graphs can be found in the survey article by Hoory et al. [4]: An expander graph is an infinite family of undirected regular graphs¹ with increasing sizes, but constant degrees, all fulfilling a non-trivial expansion condition consistently. Most common are the following expansion conditions:

- One-sided spectral expansion – an upper-bound on the second largest eigenvalue λ_2 of the adjacency matrix,
- Two-sided spectral expansion – an upper-bound on the absolute value of both λ_2 and λ_n the smallest eigenvalue,
- Edge expansion – a lower-bound on the relative count of edges between any subset and its complement.

There are various implications between the three types of families, most notably the Cheeger inequality, which relates edge-expansion to (one-sided) spectral expansion. (Section 7)

This entry formalizes

- definitions for the expansion conditions, as well as proofs for the relations between them,
- a construction and proofs of spectral expansion of the Margulis-Gabber-Galil expander (Section 8), and
- proofs of how expansion-properties are affected by graph operations (Sections 10 and 11).

And concludes with a construction of strongly explicit expanders for every size and spectral gap with asymptotically optimal degree (Section 11).

It also includes a proof of the hitting property, i.e., tail-bounds for the probability that a random walk in an expander graph remains inside a given subset, as well as Chernoff-type bounds on the number of times a given subset will be hit by a random walk. (Section 9)

The basis for the graph theory relies on the formalization by Lars Noschinski [10]. Most of the algebraic development is carried out in the type-based formalization of linear algebra in “HOL-Analysis”. To achieve that I have transferred some results from the set based world into the type-based world - most notably unified diagonalization of commuting hermitian matrices by Echenim [2] (Section 6). The transfer happens using the pre-existing framework by Divasón et al. [1].

On the otherhand, results that are obtained using the stochastic matrix, but do not explicitly reference it are transferred back into purely graph-theoretic theorems using the Types-To-Sets mechanism by Kuncăr and Popescu [7] (Section 4), i.e., the stochastic matrix is defined using a local type (isomorphic to the vertex set.)

2 Preliminary Results

2.1 Constructive Chernoff Bound

This section formalizes Theorem 5 by Impagliazzo and Kabanets [5]. It is a general result with which Chernoff-type tail bounds for various kinds of weakly dependent random variables can be obtained. The results here are general and will be applied in Section 9 to random walks in expander graphs.

theory *Constructive-Chernoff-Bound*

imports

HOL-Probability.Probability-Measure

Universal-Hash-Families.Universal-Hash-Families-More-Product-PMF

Weighted-Arithmetic-Geometric-Mean.Weighted-Arithmetic-Geometric-Mean

Negative-Association.Negative-Association-Util

begin

lemma *powr-mono-rev*:

¹A graph is regular if every node has the same degree.

fixes $x :: \text{real}$
assumes $a \leq b$ **and** $x > 0$ $x \leq 1$
shows $x \text{ powr } b \leq x \text{ powr } a$
 $\langle \text{proof} \rangle$

lemma *exp-powr*: $(\text{exp } x) \text{ powr } y = \text{exp } (x*y)$ **for** $x :: \text{real}$
 $\langle \text{proof} \rangle$

lemma *integrable-pmf-iff-bounded*:
fixes $f :: 'a \Rightarrow \text{real}$
assumes $\bigwedge x. x \in \text{set-pmf } p \implies \text{abs } (f x) \leq C$
shows $\text{integrable } (\text{measure-pmf } p) f$
 $\langle \text{proof} \rangle$

lemma *split-pair-pmf*:
 $\text{measure-pmf.prob } (\text{pair-pmf } A B) S = \text{integral}^L A (\lambda a. \text{measure-pmf.prob } B \{b. (a,b) \in S\})$
(is ?L = ?R)
 $\langle \text{proof} \rangle$

lemma *split-pair-pmf-2*:
 $\text{measure}(\text{pair-pmf } A B) S = \text{integral}^L B (\lambda a. \text{measure-pmf.prob } A \{b. (b,a) \in S\})$
(is ?L = ?R)
 $\langle \text{proof} \rangle$

theorem *impagliazzo-kabanets-pmf*:
fixes $Y :: \text{nat} \Rightarrow 'a \Rightarrow \text{bool}$
fixes $p :: 'a \text{ pmf}$
assumes $n > 0$
assumes $\bigwedge i. i \in \{..\langle n \rangle\} \implies \delta i \in \{0..1\}$
assumes $\bigwedge S. S \subseteq \{..\langle n \rangle\} \implies \text{measure } p \{\omega. (\forall i \in S. Y i \omega)\} \leq (\prod i \in S. \delta i)$
defines $\delta\text{-avg} \equiv (\sum_{i \in \{..\langle n \rangle\}} \delta i) / n$
assumes $\gamma \in \{\delta\text{-avg}..1\}$
assumes $\delta\text{-avg} > 0$
shows $\text{measure } p \{\omega. \text{real } (\text{card } \{i \in \{..\langle n \rangle\}. Y i \omega\}) \geq \gamma * n\} \leq \text{exp } (-\text{real } n * \text{KL-div } \gamma \delta\text{-avg})$
(is ?L \leq ?R)
 $\langle \text{proof} \rangle$

The distribution of a random variable with a countable range is a discrete probability space, i.e., induces a PMF. Using this it is possible to generalize the previous result to arbitrary probability spaces.

lemma *(in prob-space) establish-pmf*:
fixes $f :: 'a \Rightarrow 'b$
assumes $rv: \text{random-variable discrete } f$
assumes $\text{countable } (f \text{ ' space } M)$
shows $\text{distr } M \text{ discrete } f \in \{M. \text{prob-space } M \wedge \text{sets } M = \text{UNIV} \wedge (\text{AE } x \text{ in } M. \text{measure } M \{x\} \neq 0)\}$
 $\langle \text{proof} \rangle$

lemma *singletons-image-eq*:
 $(\lambda x. \{x\}) \text{ ' } T \subseteq \text{Pow } T$
 $\langle \text{proof} \rangle$

theorem *(in prob-space) impagliazzo-kabanets*:
fixes $Y :: \text{nat} \Rightarrow 'a \Rightarrow \text{bool}$
assumes $n > 0$
assumes $\bigwedge i. i \in \{..\langle n \rangle\} \implies \text{random-variable discrete } (Y i)$
assumes $\bigwedge i. i \in \{..\langle n \rangle\} \implies \delta i \in \{0..1\}$

assumes $\bigwedge S. S \subseteq \{..<n\} \implies \mathcal{P}(\omega \text{ in } M. (\forall i \in S. Y i \omega)) \leq (\prod i \in S. \delta i)$
defines $\delta\text{-avg} \equiv (\sum i \in \{..<n\}. \delta i) / n$
assumes $\gamma \in \{\delta\text{-avg}..1\} \delta\text{-avg} > 0$
shows $\mathcal{P}(\omega \text{ in } M. \text{real} (\text{card} \{i \in \{..<n\}. Y i \omega\}) \geq \gamma * n) \leq \text{exp} (-\text{real } n * \text{KL-div } \gamma \delta\text{-avg})$
(is ?L ≤ ?R)
 <proof>

Bounds and properties of *KL-div*

lemma *KL-div-mono-right-aux-1:*

assumes $0 \leq p \ p \leq q \ q \leq q' \ q' < 1$
shows $\text{KL-div } p \ q - 2*(p-q)^{\wedge}2 \leq \text{KL-div } p \ q' - 2*(p-q')^{\wedge}2$
 <proof>

lemma *KL-div-mono-right-aux-2:*

assumes $0 < q' \ q' \leq q \ q \leq p \ p \leq 1$
shows $\text{KL-div } p \ q - 2*(p-q)^{\wedge}2 \leq \text{KL-div } p \ q' - 2*(p-q')^{\wedge}2$
 <proof>

lemma *KL-div-mono-right-aux:*

assumes $(0 \leq p \wedge p \leq q \wedge q \leq q' \wedge q' < 1) \vee (0 < q' \wedge q' \leq q \wedge q \leq p \wedge p \leq 1)$
shows $\text{KL-div } p \ q - 2*(p-q)^{\wedge}2 \leq \text{KL-div } p \ q' - 2*(p-q')^{\wedge}2$
 <proof>

lemma *KL-div-mono-right:*

assumes $(0 \leq p \wedge p \leq q \wedge q \leq q' \wedge q' < 1) \vee (0 < q' \wedge q' \leq q \wedge q \leq p \wedge p \leq 1)$
shows $\text{KL-div } p \ q \leq \text{KL-div } p \ q'$ **(is ?L ≤ ?R)**
 <proof>

lemma *KL-div-lower-bound:*

assumes $p \in \{0..1\} \ q \in \{0<..<1\}$
shows $2*(p-q)^{\wedge}2 \leq \text{KL-div } p \ q$
 <proof>

end

2.2 Congruence Method

The following is a method for proving equalities of large terms by checking the equivalence of subterms. It is possible to precisely control which operators to split by.

theory *Extra-Congruence-Method*

imports

Main

HOL-Eisbach.Eisbach

begin

datatype *cong-tag-type* = *CongTag*

definition *cong-tag-1* :: $('a \Rightarrow 'b) \Rightarrow \text{cong-tag-type}$

where *cong-tag-1* $x = \text{CongTag}$

definition *cong-tag-2* :: $('a \Rightarrow 'b \Rightarrow 'c) \Rightarrow \text{cong-tag-type}$

where *cong-tag-2* $x = \text{CongTag}$

definition *cong-tag-3* :: $('a \Rightarrow 'b \Rightarrow 'c \Rightarrow 'd) \Rightarrow \text{cong-tag-type}$

where *cong-tag-3* $x = \text{CongTag}$

lemma *arg-cong3:*

assumes $x1 = x2 \ y1 = y2 \ z1 = z2$

shows $f \ x1 \ y1 \ z1 = f \ x2 \ y2 \ z2$

<proof>

```
method intro-cong for  $A :: \text{cong-tag-type list}$  uses more =  
  (match  $A$ ) in  
    cong-tag-1  $f \# h$  (multi) for  $f :: 'a \Rightarrow 'b$  and  $h$   
       $\Rightarrow \langle \text{intro-cong } h \text{ more:more arg-cong[where } f=f \rangle$   
    | cong-tag-2  $f \# h$  (multi) for  $f :: 'a \Rightarrow 'b \Rightarrow 'c$  and  $h$   
       $\Rightarrow \langle \text{intro-cong } h \text{ more:more arg-cong2[where } f=f \rangle$   
    | cong-tag-3  $f \# h$  (multi) for  $f :: 'a \Rightarrow 'b \Rightarrow 'c \Rightarrow 'd$  and  $h$   
       $\Rightarrow \langle \text{intro-cong } h \text{ more:more arg-cong3[where } f=f \rangle$   
    | -  $\Rightarrow \langle \text{intro more refl} \rangle$ 
```

bundle *intro-cong-syntax*

begin

notation *cong-tag-1* ($\langle \sigma_1 \rangle$)

notation *cong-tag-2* ($\langle \sigma_2 \rangle$)

notation *cong-tag-3* ($\langle \sigma_3 \rangle$)

end

lemma *restr-Collect-cong*:

assumes $\bigwedge x. x \in A \implies P x = Q x$

shows $\{x \in A. P x\} = \{x \in A. Q x\}$

<proof>

end

2.3 Multisets

Some preliminary results about multisets.

theory *Expander-Graphs-Multiset-Extras*

imports

HOL-Library.Multiset

Extra-Congruence-Method

begin

unbundle *intro-cong-syntax*

This is an induction scheme over the distinct elements of a multisets: We can represent each multiset as a sum like: *replicate-mset* n_1 x_1 + *replicate-mset* n_2 x_2 + ... + *replicate-mset* n_k x_k where the x_i are distinct.

lemma *disj-induct-mset*:

assumes $P \{ \# \}$

assumes $\bigwedge^n M x. P M \implies \neg(x \in \# M) \implies n > 0 \implies P (M + \text{replicate-mset } n x)$

shows $P M$

<proof>

lemma *sum-mset-conv*:

fixes $f :: 'a \Rightarrow 'b :: \{ \text{semiring-1} \}$

shows $\text{sum-mset} (\text{image-mset } f A) = \text{sum} (\lambda x. \text{of-nat} (\text{count } A x) * f x) (\text{set-mset } A)$

<proof>

lemma *sum-mset-conv-2*:

fixes $f :: 'a \Rightarrow 'b :: \{ \text{semiring-1} \}$

assumes $\text{set-mset } A \subseteq B$ *finite* B

shows $\text{sum-mset} (\text{image-mset } f A) = \text{sum} (\lambda x. \text{of-nat} (\text{count } A x) * f x) B$ (**is** $?L = ?R$)

<proof>

lemma *count-mset-exp*: $\text{count } A \ x = \text{size } (\text{filter-mset } (\lambda y. y = x) \ A)$
<proof>

lemma *mset-repl*: $\text{mset } (\text{replicate } k \ x) = \text{replicate-mset } k \ x$
<proof>

lemma *count-image-mset-inj*:
assumes *inj f*
shows $\text{count } (\text{image-mset } f \ A) \ (f \ x) = \text{count } A \ x$
<proof>

lemma *count-image-mset-0-triv*:
assumes $x \notin \text{range } f$
shows $\text{count } (\text{image-mset } f \ A) \ x = 0$
<proof>

lemma *filter-mset-ex-predicates*:
assumes $\bigwedge x. \neg P \ x \vee \neg Q \ x$
shows $\text{filter-mset } P \ M + \text{filter-mset } Q \ M = \text{filter-mset } (\lambda x. P \ x \vee Q \ x) \ M$
<proof>

lemma *sum-count-2*:
assumes *finite F*
shows $\text{sum } (\text{count } M) \ F = \text{size } (\text{filter-mset } (\lambda x. x \in F) \ M)$
<proof>

definition *concat-mset* :: $(\text{'a multiset}) \ \text{multiset} \Rightarrow \text{'a multiset}$
where $\text{concat-mset } xss = \text{fold-mset } (\lambda xs \ ys. xs + ys) \ \{\#\} \ xss$

lemma *image-concat-mset*:
 $\text{image-mset } f \ (\text{concat-mset } xss) = \text{concat-mset } (\text{image-mset } (f) \ xss)$
<proof>

lemma *concat-add-mset*:
 $\text{concat-mset } (\text{image-mset } (\lambda x. f \ x + g \ x) \ xs) = \text{concat-mset } (\text{image-mset } f \ xs) + \text{concat-mset } (\text{image-mset } g \ xs)$
<proof>

lemma *concat-add-mset-2*:
 $\text{concat-mset } (xs + ys) = \text{concat-mset } xs + \text{concat-mset } ys$
<proof>

lemma *size-concat-mset*:
 $\text{size } (\text{concat-mset } xss) = \text{sum-mset } (\text{image-mset } \text{size } xss)$
<proof>

lemma *filter-concat-mset*:
 $\text{filter-mset } P \ (\text{concat-mset } xss) = \text{concat-mset } (\text{image-mset } (\text{filter-mset } P) \ xss)$
<proof>

lemma *count-concat-mset*:
 $\text{count } (\text{concat-mset } xss) \ xs = \text{sum-mset } (\text{image-mset } (\lambda x. \text{count } x \ xs) \ xss)$
<proof>

lemma *set-mset-concat-mset*:
 $\text{set-mset } (\text{concat-mset } xss) = \bigcup (\text{set-mset } \text{' } (\text{set-mset } xss))$
<proof>

lemma *concat-mset-empty*: $\text{concat-mset } \{\#\} = \{\#\}$
 ⟨proof⟩

lemma *concat-mset-single*: $\text{concat-mset } \{\#x\# \} = x$
 ⟨proof⟩

lemma *concat-disjoint-union-mset*:
 assumes *finite I*
 assumes $\bigwedge i. i \in I \implies \text{finite } (A i)$
 assumes $\bigwedge i j. i \in I \implies j \in I \implies i \neq j \implies A i \cap A j = \{\}$
 shows $\text{mset-set } (\bigcup (A ' I)) = \text{concat-mset } (\text{image-mset } (\text{mset-set } \circ A) (\text{mset-set } I))$
 ⟨proof⟩

lemma *size-filter-mset-conv*:
 $\text{size } (\text{filter-mset } f A) = \text{sum-mset } (\text{image-mset } (\lambda x. \text{of-bool } (f x) :: \text{nat}) A)$
 ⟨proof⟩

lemma *filter-mset-const*: $\text{filter-mset } (\lambda-. c) xs = (\text{if } c \text{ then } xs \text{ else } \{\#\})$
 ⟨proof⟩

lemma *repeat-image-concat-mset*:
 $\text{repeat-mset } n (\text{image-mset } f A) = \text{concat-mset } (\text{image-mset } (\lambda x. \text{replicate-mset } n (f x)) A)$
 ⟨proof⟩

lemma *mset-prod-eq*:
 assumes *finite A finite B*
 shows
 $\text{mset-set } (A \times B) = \text{concat-mset } \{\# \{\# (x,y). y \in \# \text{mset-set } B \#\} .x \in \# \text{mset-set } A \#\}$
 ⟨proof⟩

lemma *sum-mset-repeat*:
 fixes $f :: 'a \Rightarrow 'b :: \{\text{comm-monoid-add, semiring-1}\}$
 shows $\text{sum-mset } (\text{image-mset } f (\text{repeat-mset } n A)) = \text{of-nat } n * \text{sum-mset } (\text{image-mset } f A)$
 ⟨proof⟩

unbundle *no intro-cong-syntax*

end

3 Definitions

This section introduces regular graphs as a sublocale in the graph theory developed by Lars Noschinski [10] and introduces various expansion coefficients.

theory *Expander-Graphs-Definition*

imports

Graph-Theory.Digraph-Isomorphism

HOL-Analysis.L2-Norm

Extra-Congruence-Method

Expander-Graphs-Multiset-Extras

Jordan-Normal-Form.Conjugate

Interpolation-Polynomials-HOL-Algebra.Interpolation-Polynomial-Cardinalities

begin

unbundle *intro-cong-syntax*

definition *arcs-betw* **where** $\text{arcs-betw } G u v = \{a. a \in \text{arcs } G \wedge \text{head } G a = v \wedge \text{tail } G a = u\}$

The following is a stronger notion than the notion of symmetry defined in *Graph-Theory.Digraph*, it requires that the number of edges from v to w must be equal to the number of edges from w to v for any pair of vertices $v w \in \text{verts } G$.

definition *symmetric-multi-graph* **where** *symmetric-multi-graph* $G =$
 $(\text{fin-digraph } G \wedge (\forall v w. \{v, w\} \subseteq \text{verts } G \longrightarrow \text{card } (\text{arcs-betw } G w v) = \text{card } (\text{arcs-betw } G v w)))$

lemma *symmetric-multi-graphI*:

assumes *fin-digraph* G

assumes *bij-betw* f $(\text{arcs } G)$ $(\text{arcs } G)$

assumes $\bigwedge e. e \in \text{arcs } G \implies \text{head } G (f e) = \text{tail } G e \wedge \text{tail } G (f e) = \text{head } G e$

shows *symmetric-multi-graph* G

$\langle \text{proof} \rangle$

lemma *symmetric-multi-graphD2*:

assumes *symmetric-multi-graph* G

shows *fin-digraph* G

$\langle \text{proof} \rangle$

lemma *symmetric-multi-graphD*:

assumes *symmetric-multi-graph* G

shows $\text{card } \{e \in \text{arcs } G. \text{head } G e = v \wedge \text{tail } G e = w\} = \text{card } \{e \in \text{arcs } G. \text{head } G e = w \wedge \text{tail } G e = v\}$

(is $\text{card } ?L = \text{card } ?R$)

$\langle \text{proof} \rangle$

lemma *symmetric-multi-graphD3*:

assumes *symmetric-multi-graph* G

shows

$\text{card } \{e \in \text{arcs } G. \text{tail } G e = v \wedge \text{head } G e = w\} = \text{card } \{e \in \text{arcs } G. \text{tail } G e = w \wedge \text{head } G e = v\}$

$\langle \text{proof} \rangle$

lemma *symmetric-multi-graphD4*:

assumes *symmetric-multi-graph* G

shows $\text{card } (\text{arcs-betw } G v w) = \text{card } (\text{arcs-betw } G w v)$

$\langle \text{proof} \rangle$

lemma *symmetric-degree-eq*:

assumes *symmetric-multi-graph* G

assumes $v \in \text{verts } G$

shows $\text{out-degree } G v = \text{in-degree } G v$ **(is** $?L = ?R$)

$\langle \text{proof} \rangle$

definition *edges* **where** *edges* $G = \text{image-mset } (\text{arc-to-ends } G) (\text{mset-set } (\text{arcs } G))$

lemma **(in** *fin-digraph*) *count-edges*:

$\text{count } (\text{edges } G) (u, v) = \text{card } (\text{arcs-betw } G u v)$ **(is** $?L = ?R$)

$\langle \text{proof} \rangle$

lemma **(in** *fin-digraph*) *count-edges-sym*:

assumes *symmetric-multi-graph* G

shows $\text{count } (\text{edges } G) (v, w) = \text{count } (\text{edges } G) (w, v)$

$\langle \text{proof} \rangle$

lemma **(in** *fin-digraph*) *edges-sym*:

assumes *symmetric-multi-graph* G

shows $\{\# (y, x). (x, y) \in \# (\text{edges } G) \# \} = \text{edges } G$

<proof>

definition *vertices-from* $G v = \{\# \text{snd } e \mid e \in \# \text{edges } G. \text{fst } e = v \#\}$

definition *vertices-to* $G v = \{\# \text{fst } e \mid e \in \# \text{edges } G. \text{snd } e = v \#\}$

context *fin-digraph*

begin

lemma *edge-set*:

assumes $x \in \# \text{edges } G$

shows $\text{fst } x \in \text{verts } G \text{ snd } x \in \text{verts } G$

<proof>

lemma *verts-from-alt*:

$\text{vertices-from } G v = \text{image-mset } (\text{head } G) (\text{mset-set } (\text{out-arcs } G v))$

<proof>

lemma *verts-to-alt*:

$\text{vertices-to } G v = \text{image-mset } (\text{tail } G) (\text{mset-set } (\text{in-arcs } G v))$

<proof>

lemma *set-mset-vertices-from*:

$\text{set-mset } (\text{vertices-from } G x) \subseteq \text{verts } G$

<proof>

lemma *set-mset-vertices-to*:

$\text{set-mset } (\text{vertices-to } G x) \subseteq \text{verts } G$

<proof>

end

A symmetric multigraph is regular if every node has the same degree. This is the context in which the expansion conditions are introduced.

locale *regular-graph* = *fin-digraph* +

assumes *sym*: *symmetric-multi-graph* G

assumes *verts-non-empty*: $\text{verts } G \neq \{\}$

assumes *arcs-non-empty*: $\text{arcs } G \neq \{\}$

assumes *reg'*: $\bigwedge v w. v \in \text{verts } G \implies w \in \text{verts } G \implies \text{out-degree } G v = \text{out-degree } G w$

begin

definition d **where** $d = \text{out-degree } G$ (*SOME* $v. v \in \text{verts } G$)

lemmas *count-sym* = *count-edges-sym*[*OF sym*]

lemma *reg*:

assumes $v \in \text{verts } G$

shows $\text{out-degree } G v = d \text{ in-degree } G v = d$

<proof>

definition n **where** $n = \text{card } (\text{verts } G)$

lemma *n-gt-0*: $n > 0$

<proof>

lemma *d-gt-0*: $d > 0$

<proof>

definition *g-inner* :: $('a \Rightarrow ('c :: \text{conjugatable-field})) \Rightarrow ('a \Rightarrow 'c) \Rightarrow 'c$

where $g\text{-inner } f g = (\sum x \in \text{verts } G. (f x) * \text{conjugate } (g x))$

lemma *conjugate-divide[simp]*:

fixes $x y :: 'c :: \text{conjugatable-field}$

shows $\text{conjugate } (x / y) = \text{conjugate } x / \text{conjugate } y$

<proof>

lemma *g-inner-simps*:

$g\text{-inner } (\lambda x. 0) g = 0$

$g\text{-inner } f (\lambda x. 0) = 0$

$g\text{-inner } (\lambda x. c * f x) g = c * g\text{-inner } f g$

$g\text{-inner } f (\lambda x. c * g x) = \text{conjugate } c * g\text{-inner } f g$

$g\text{-inner } (\lambda x. f x - g x) h = g\text{-inner } f h - g\text{-inner } g h$

$g\text{-inner } (\lambda x. f x + g x) h = g\text{-inner } f h + g\text{-inner } g h$

$g\text{-inner } f (\lambda x. g x + h x) = g\text{-inner } f g + g\text{-inner } f h$

$g\text{-inner } f (\lambda x. g x / c) = g\text{-inner } f g / \text{conjugate } c$

$g\text{-inner } (\lambda x. f x / c) g = g\text{-inner } f g / c$

<proof>

definition $g\text{-norm } f = \text{sqrt } (g\text{-inner } f f)$

lemma *g-norm-eq*: $g\text{-norm } f = L2\text{-set } f (\text{verts } G)$

<proof>

lemma *g-inner-cauchy-schwartz*:

fixes $f g :: 'a \Rightarrow \text{real}$

shows $|g\text{-inner } f g| \leq g\text{-norm } f * g\text{-norm } g$

<proof>

lemma *g-inner-cong*:

assumes $\bigwedge x. x \in \text{verts } G \implies f1 x = f2 x$

assumes $\bigwedge x. x \in \text{verts } G \implies g1 x = g2 x$

shows $g\text{-inner } f1 g1 = g\text{-inner } f2 g2$

<proof>

lemma *g-norm-cong*:

assumes $\bigwedge x. x \in \text{verts } G \implies f x = g x$

shows $g\text{-norm } f = g\text{-norm } g$

<proof>

lemma *g-norm-nonneg*: $g\text{-norm } f \geq 0$

<proof>

lemma *g-norm-sq*:

$g\text{-norm } f^2 = g\text{-inner } f f$

<proof>

definition *g-step* :: $('a \Rightarrow \text{real}) \Rightarrow ('a \Rightarrow \text{real})$

where $g\text{-step } f v = (\sum x \in \text{in-arcs } G v. f (\text{tail } G x) / \text{real } d)$

lemma *g-step-simps*:

$g\text{-step } (\lambda x. f x + g x) y = g\text{-step } f y + g\text{-step } g y$

$g\text{-step } (\lambda x. f x / c) y = g\text{-step } f y / c$

<proof>

lemma *g-inner-step-eq*:

$g\text{-inner } f (g\text{-step } f) = (\sum a \in \text{arcs } G. f (\text{head } G a) * f (\text{tail } G a)) / d$ (**is** ?L = ?R)

<proof>

definition Λ -test

where Λ -test = $\{f. g\text{-norm } f^{\wedge}2 \neq 0 \wedge g\text{-inner } f (\lambda. 1) = 0\}$

lemma Λ -test-ne:

assumes $n > 1$

shows Λ -test $\neq \{\}$

\langle proof \rangle

lemma Λ -test-empty:

assumes $n = 1$

shows Λ -test = $\{\}$

\langle proof \rangle

The following are variational definitions for the maximum of the spectrum (resp. maximum modulus of the spectrum) of the stochastic matrix (excluding the Perron eigenvalue 1). Note that both values can still obtain the value one 1 (if the multiplicity of the eigenvalue 1 is larger than 1 in the stochastic matrix, or in the modulus case if -1 is an eigenvalue).

The definition relies on the supremum of the Rayleigh-Quotient for vectors orthogonal to the stationary distribution). In Section 6, the equivalence of this value with the algebraic definition will be shown. The definition here has the advantage that it is (obviously) independent of the matrix representation (ordering of the vertices) used.

definition $\Lambda_2 :: \text{real}$

where $\Lambda_2 = (\text{if } n > 1 \text{ then } (\text{SUP } f \in \Lambda\text{-test. } g\text{-inner } f (g\text{-step } f) / g\text{-inner } f f) \text{ else } 0)$

definition $\Lambda_a :: \text{real}$

where $\Lambda_a = (\text{if } n > 1 \text{ then } (\text{SUP } f \in \Lambda\text{-test. } |g\text{-inner } f (g\text{-step } f)| / g\text{-inner } f f) \text{ else } 0)$

lemma sum-arcs-tail:

fixes $f :: 'a \Rightarrow ('c :: \text{semiring-1})$

shows $(\sum a \in \text{arcs } G. f (\text{tail } G a)) = \text{of-nat } d * (\sum v \in \text{verts } G. f v)$ (is ?L = ?R)

\langle proof \rangle

lemma sum-arcs-head:

fixes $f :: 'a \Rightarrow ('c :: \text{semiring-1})$

shows $(\sum a \in \text{arcs } G. f (\text{head } G a)) = \text{of-nat } d * (\sum v \in \text{verts } G. f v)$ (is ?L = ?R)

\langle proof \rangle

lemma bdd-above-aux:

$|\sum a \in \text{arcs } G. f (\text{head } G a) * f (\text{tail } G a)| \leq d * g\text{-norm } f^{\wedge}2$ (is ?L \leq ?R)

\langle proof \rangle

lemma bdd-above-aux-2:

assumes $f \in \Lambda$ -test

shows $|g\text{-inner } f (g\text{-step } f)| / g\text{-inner } f f \leq 1$

\langle proof \rangle

lemma bdd-above-aux-3:

assumes $f \in \Lambda$ -test

shows $g\text{-inner } f (g\text{-step } f) / g\text{-inner } f f \leq 1$ (is ?L \leq ?R)

\langle proof \rangle

lemma bdd-above- Λ : bdd-above $((\lambda f. |g\text{-inner } f (g\text{-step } f)| / g\text{-inner } f f) \text{ ' } \Lambda\text{-test})$

\langle proof \rangle

lemma bdd-above- Λ_2 : bdd-above $((\lambda f. g\text{-inner } f (g\text{-step } f) / g\text{-inner } f f) \text{ ' } \Lambda\text{-test})$

<proof>

lemma Λ_{le-1} : $\Lambda_a \leq 1$

<proof>

lemma Λ_2 -le-1: $\Lambda_2 \leq 1$

<proof>

lemma Λ -ge-0: $\Lambda_a \geq 0$

<proof>

lemma *os-expanderI*:

assumes $n > 1$

assumes $\bigwedge f. g\text{-inner } f (\lambda-. 1)=0 \implies g\text{-inner } f (g\text{-step } f) \leq C * g\text{-norm } f^2$

shows $\Lambda_2 \leq C$

<proof>

lemma *os-expanderD*:

assumes $g\text{-inner } f (\lambda-. 1) = 0$

shows $g\text{-inner } f (g\text{-step } f) \leq \Lambda_2 * g\text{-norm } f^2$ (**is** ?L ≤ ?R)

<proof>

lemma *expander-intro-1*:

assumes $C \geq 0$

assumes $\bigwedge f. g\text{-inner } f (\lambda-. 1)=0 \implies |g\text{-inner } f (g\text{-step } f)| \leq C * g\text{-norm } f^2$

shows $\Lambda_a \leq C$

<proof>

lemma *expander-intro*:

assumes $C \geq 0$

assumes $\bigwedge f. g\text{-inner } f (\lambda-. 1)=0 \implies |\sum a \in \text{arcs } G. f(\text{head } G a) * f(\text{tail } G a)| \leq C * g\text{-norm } f^2$

shows $\Lambda_a \leq C/d$

<proof>

lemma *expansionD1*:

assumes $g\text{-inner } f (\lambda-. 1) = 0$

shows $|g\text{-inner } f (g\text{-step } f)| \leq \Lambda_a * g\text{-norm } f^2$ (**is** ?L ≤ ?R)

<proof>

lemma *expansionD*:

assumes $g\text{-inner } f (\lambda-. 1) = 0$

shows $|\sum a \in \text{arcs } G. f(\text{head } G a) * f(\text{tail } G a)| \leq d * \Lambda_a * g\text{-norm } f^2$ (**is** ?L ≤ ?R)

<proof>

definition *edges-betw* **where** $\text{edges-betw } S T = \{a \in \text{arcs } G. \text{tail } G a \in S \wedge \text{head } G a \in T\}$

This parameter is the edge expansion. It is usually denoted by the symbol h or $h(G)$ in text books. Contrary to the previous definitions it doesn't have a spectral theoretic counter part.

definition Λ_e **where** $\Lambda_e = (\text{if } n > 1 \text{ then}$

$(\text{MIN } S \in \{S. S \subseteq \text{verts } G \wedge 2 * \text{card } S \leq n \wedge S \neq \{\}\}. \text{real } (\text{card } (\text{edges-betw } S (-S))) / \text{card } S)$ else 0)

lemma *edge-expansionD*:

assumes $S \subseteq \text{verts } G \wedge 2 * \text{card } S \leq n$

shows $\Lambda_e * \text{card } S \leq \text{real } (\text{card } (\text{edges-betw } S (-S)))$

<proof>

```

lemma edge-expansionI:
  fixes  $\alpha :: \text{real}$ 
  assumes  $n > 1$ 
  assumes  $\bigwedge S. S \subseteq \text{verts } G \implies 2 * \text{card } S \leq n \implies S \neq \{\} \implies \text{card } (\text{edges-betw } S \ (-S)) \geq \alpha * \text{card } S$ 
  shows  $\Lambda_e \geq \alpha$ 
  <proof>

end

```

```

lemma regular-graphI:
  assumes symmetric-multi-graph  $G$ 
  assumes  $\text{verts } G \neq \{\} \ d > 0$ 
  assumes  $\bigwedge v. v \in \text{verts } G \implies \text{out-degree } G \ v = d$ 
  shows regular-graph  $G$ 
  <proof>

```

The following theorems verify that a graph isomorphisms preserve symmetry, regularity and all the expansion coefficients.

```

lemma (in fin-digraph) symmetric-graph-iso:
  assumes digraph-iso  $G \ H$ 
  assumes symmetric-multi-graph  $G$ 
  shows symmetric-multi-graph  $H$ 
  <proof>

```

```

lemma (in regular-graph)
  assumes digraph-iso  $G \ H$ 
  shows regular-graph-iso: regular-graph  $H$ 
  and regular-graph-iso-size: regular-graph.n  $H = n$ 
  and regular-graph-iso-degree: regular-graph.d  $H = d$ 
  and regular-graph-iso-expansion-le: regular-graph. $\Lambda_a$   $H \leq \Lambda_a$ 
  and regular-graph-iso-os-expansion-le: regular-graph. $\Lambda_2$   $H \leq \Lambda_2$ 
  and regular-graph-iso-edge-expansion-ge: regular-graph. $\Lambda_e$   $H \geq \Lambda_e$ 
  <proof>

```

```

lemma (in regular-graph)
  assumes digraph-iso  $G \ H$ 
  shows regular-graph-iso-expansion: regular-graph. $\Lambda_a$   $H = \Lambda_a$ 
  and regular-graph-iso-os-expansion: regular-graph. $\Lambda_2$   $H = \Lambda_2$ 
  and regular-graph-iso-edge-expansion: regular-graph. $\Lambda_e$   $H = \Lambda_e$ 
  <proof>

```

```

unbundle no intro-cong-syntax

```

```

end

```

4 Setup for Types to Sets

```

theory Expander-Graphs-TTS
  imports
    Expander-Graphs-Definition
    HOL-Analysis.Cartesian-Space
    HOL-Types-To-Sets.Types-To-Sets
  begin

```

This section sets up a sublocale with the assumption that there is a finite type with the same cardinality as the vertex set of a regular graph. This allows defining the adjacency

matrix for the graph using type-based linear algebra.

Theorems shown in the sublocale that do not refer to the local type are then lifted to the *regular-graph* locale using the Types-To-Sets mechanism.

locale *regular-graph-tts* = *regular-graph* +
fixes *n-itself* :: ('n :: finite) itself
assumes *td*: $\exists (f :: ('n \Rightarrow 'a))$ *g*. *type-definition* *f g* (*verts G*)
begin

definition *td-components* :: ('n \Rightarrow 'a) \times ('a \Rightarrow 'n)
where *td-components* = (SOME *q*. *type-definition* (*fst q*) (*snd q*) (*verts G*))

definition *enum-verts* **where** *enum-verts* = *fst td-components*

definition *enum-verts-inv* **where** *enum-verts-inv* = *snd td-components*

sublocale *type-definition* *enum-verts* *enum-verts-inv* *verts G*
 \langle *proof* \rangle

lemma *enum-verts: bij-betw* *enum-verts UNIV* (*verts G*)
 \langle *proof* \rangle

The stochastic matrix associated to the graph.

definition *A* :: ('c::field) $\hat{\sim}^n \hat{\sim}^n$ **where**
 $A = (\chi \ i \ j.$ *of-nat* (*count* (*edges G*) (*enum-verts j,enum-verts i*))/*of-nat d*)

lemma *card-n*: *CARD*('n) = *n*
 \langle *proof* \rangle

lemma *symmetric-A*: *transpose A* = *A*
 \langle *proof* \rangle

lemma *g-step-conv*:
 $(\chi \ i.$ *g-step f* (*enum-verts i*)) = *A *v* ($\chi \ i.$ *f* (*enum-verts i*))
 \langle *proof* \rangle

lemma *g-inner-conv*:
 $g\text{-inner } f \ g = (\chi \ i.$ *f* (*enum-verts i*)) \cdot ($\chi \ i.$ *g* (*enum-verts i*))
 \langle *proof* \rangle

lemma *g-norm-conv*:
 $g\text{-norm } f = \text{norm } (\chi \ i.$ *f* (*enum-verts i*))
 \langle *proof* \rangle

end

lemma *eg-tts-1*:
assumes *regular-graph G*
assumes $\exists (f :: ('n :: finite) \Rightarrow 'a)$ *g*. *type-definition* *f g* (*verts G*)
shows *regular-graph-tts* *TYPE*('n) *G*
 \langle *proof* \rangle

context *regular-graph*
begin

lemma *remove-finite-premise-aww*:
assumes $\exists (Rep :: 'n \Rightarrow 'a)$ *Abs*. *type-definition* *Rep Abs* (*verts G*)
shows *class.finite* *TYPE*('n)
 \langle *proof* \rangle

```

lemma remove-finite-premise:
  (class.finite TYPE('n)  $\implies \exists (Rep :: 'n \Rightarrow 'a)$  Abs. type-definition Rep Abs (verts G)  $\implies PROP$ 
  Q)
   $\equiv (\exists (Rep :: 'n \Rightarrow 'a)$  Abs. type-definition Rep Abs (verts G)  $\implies PROP$  Q)
  (is ?L  $\equiv$  ?R)
  <proof>

end

end

```

5 Algebra-only Theorems

This section verifies the linear algebraic counter-parts of the graph-theoretic theorems about Random walks. The graph-theoretic results are then derived in Section 9.

```

theory Expander-Graphs-Algebra

```

```

imports

```

```

  HOL-Library.Monad-Syntax
  Expander-Graphs-TTS

```

```

begin

```

```

lemma pythagoras:

```

```

  fixes v w :: 'a::real-inner

```

```

  assumes v  $\cdot$  w = 0

```

```

  shows norm (v+w)2 = norm v2 + norm w2

```

```

  <proof>

```

```

definition diag :: ('a :: zero)n  $\Rightarrow$  'an

```

```

  where diag v = ( $\chi$  i j. if i = j then (v $ i) else 0)

```

```

definition ind-vec :: 'n set  $\Rightarrow$  realn

```

```

  where ind-vec S = ( $\chi$  i. of_bool( i  $\in$  S))

```

```

lemma diag-mult-eq: diag x ** diag y = diag (x * y)

```

```

  <proof>

```

```

lemma diag-vec-mult-eq: diag x *v y = x * y

```

```

  <proof>

```

```

definition matrix-norm-bound :: realnm  $\Rightarrow$  real  $\Rightarrow$  bool

```

```

  where matrix-norm-bound A l = ( $\forall x$ . norm (A *v x)  $\leq$  l * norm x)

```

```

lemma matrix-norm-boundI:

```

```

  assumes  $\bigwedge x$ . norm (A *v x)  $\leq$  l * norm x

```

```

  shows matrix-norm-bound A l

```

```

  <proof>

```

```

lemma matrix-norm-boundD:

```

```

  assumes matrix-norm-bound A l

```

```

  shows norm (A *v x)  $\leq$  l * norm x

```

```

  <proof>

```

```

lemma matrix-norm-bound-nonneg:

```

```

  fixes A :: realnm

```

```

  assumes matrix-norm-bound A l

```

```

  shows l  $\geq$  0

```

<proof>

lemma *matrix-norm-bound-0*:
 assumes *matrix-norm-bound* A 0
 shows $A = (0::\text{real}^n \times \text{real}^m)$

<proof>

lemma *matrix-norm-bound-diag*:
 fixes $x :: \text{real}^n$
 assumes $\bigwedge i. |x \ \$ \ i| \leq l$
 shows *matrix-norm-bound* (*diag* x) l

<proof>

lemma *vector-scaleR-matrix-ac-2*: $b \ *_R \ (A::\text{real}^n \times \text{real}^m) \ *v \ x = b \ *_R \ (A \ *v \ x)$

<proof>

lemma *matrix-norm-bound-scale*:
 assumes *matrix-norm-bound* A l
 shows *matrix-norm-bound* ($b \ *_R \ A$) ($|b| \ * \ l$)

<proof>

definition *nonneg-mat* $:: \text{real}^n \times \text{real}^m \Rightarrow \text{bool}$
 where *nonneg-mat* $A = (\forall i \ j. A \ \$ \ i \ \$ \ j \geq 0)$

lemma *nonneg-mat-1*:
 shows *nonneg-mat* (*mat* 1)

<proof>

lemma *nonneg-mat-prod*:
 assumes *nonneg-mat* A *nonneg-mat* B
 shows *nonneg-mat* ($A \ ** \ B$)

<proof>

lemma *nonneg-mat-transpose*:
 nonneg-mat (*transpose* A) = *nonneg-mat* A

<proof>

definition *spec-bound* $:: \text{real}^n \times \text{real}^n \Rightarrow \text{real} \Rightarrow \text{bool}$
 where *spec-bound* $M \ l = (l \geq 0 \wedge (\forall v. v \cdot 1 = 0 \longrightarrow \text{norm} (M \ *v \ v) \leq l \ * \ \text{norm} \ v))$

lemma *spec-boundD1*:
 assumes *spec-bound* $M \ l$
 shows $0 \leq l$

<proof>

lemma *spec-boundD2*:
 assumes *spec-bound* $M \ l$
 assumes $v \cdot 1 = 0$
 shows $\text{norm} (M \ *v \ v) \leq l \ * \ \text{norm} \ v$

<proof>

lemma *spec-bound-mono*:
 assumes *spec-bound* $M \ \alpha \ \alpha \leq \beta$
 shows *spec-bound* $M \ \beta$

<proof>

definition *markov* $:: \text{real}^n \times \text{real}^n \Rightarrow \text{bool}$
 where *markov* $M = (\text{nonneg-mat} \ M \wedge M \ *v \ 1 = 1 \wedge 1 \ *v \ M = 1)$

lemma *markov-symI*:
assumes *nonneg-mat A transpose A = A A *v 1 = 1*
shows *markov A*
 ⟨*proof*⟩

lemma *markov-apply*:
assumes *markov M*
shows *M *v 1 = 1 1 v* M = 1*
 ⟨*proof*⟩

lemma *markov-transpose*:
markov A = markov (transpose A)
 ⟨*proof*⟩

fun *matrix-pow* **where**
matrix-pow M 0 = mat 1 |
*matrix-pow M (Suc n) = M ** (matrix-pow M n)*

lemma *markov-orth-inv*:
assumes *markov A*
shows *inner (A *v x) 1 = inner x 1*
 ⟨*proof*⟩

lemma *markov-id*:
markov (mat 1)
 ⟨*proof*⟩

lemma *markov-mult*:
assumes *markov A markov B*
shows *markov (A ** B)*
 ⟨*proof*⟩

lemma *markov-matrix-pow*:
assumes *markov A*
shows *markov (matrix-pow A k)*
 ⟨*proof*⟩

lemma *spec-bound-prod*:
assumes *markov A markov B*
assumes *spec-bound A la spec-bound B lb*
shows *spec-bound (A ** B) (la*lb)*
 ⟨*proof*⟩

lemma *spec-bound-pow*:
assumes *markov A*
assumes *spec-bound A l*
shows *spec-bound (matrix-pow A k) (l^k)*
 ⟨*proof*⟩

fun *intersperse* :: *'a ⇒ 'a list ⇒ 'a list*
where
intersperse x [] = [] |
intersperse x (y#[]) = y#[] |
intersperse x (y#z#zs) = y#x#intersperse x (z#zs)

lemma *intersperse-snoc*:
assumes *xs ≠ []*
shows *intersperse z (xs@[y]) = intersperse z xs@[z,y]*

<proof>

lemma *foldl-intersperse*:

assumes $xs \neq []$

shows $foldl\ f\ a\ ((intersperse\ x\ xs)@[x]) = foldl\ (\lambda y\ z.\ f\ (f\ y\ z)\ x)\ a\ xs$

<proof>

lemma *foldl-intersperse-2*:

shows $foldl\ f\ a\ (intersperse\ y\ (x\#\#xs)) = foldl\ (\lambda x\ z.\ f\ (f\ x\ y)\ z)\ (f\ a\ x)\ xs$

<proof>

context *regular-graph-tts*

begin

definition *stat* :: $real^{n^2}$

where $stat = (1 / real\ CARD(n)) *_{\mathbb{R}} 1$

definition *J* :: $(c :: field)^{n^2}$

where $J = (\chi\ i\ j.\ of\text{-}nat\ 1 / of\text{-}nat\ CARD(n))$

lemma *inner-1-1*: $1 \cdot (1::real^{n^2}) = CARD(n)$

<proof>

definition *proj-unit* :: $real^{n^2} \Rightarrow real^{n^2}$

where $proj\text{-}unit\ v = (1 \cdot v) *_{\mathbb{R}} stat$

definition *proj-rem* :: $real^{n^2} \Rightarrow real^{n^2}$

where $proj\text{-}rem\ v = v - proj\text{-}unit\ v$

lemma *proj-rem-orth*: $1 \cdot (proj\text{-}rem\ v) = 0$

<proof>

lemma *split-vec*: $v = proj\text{-}unit\ v + proj\text{-}rem\ v$

<proof>

lemma *apply-J*: $J * v\ x = proj\text{-}unit\ x$

<proof>

lemma *spec-bound-J*: $spec\text{-}bound\ (J :: real^{n^2}) = 0$

<proof>

lemma *matrix-decomposition-lemma-aux*:

fixes $A :: real^{n^2}$

assumes *markov* A

shows $spec\text{-}bound\ A\ l \iff matrix\text{-}norm\text{-}bound\ (A - (1-l) *_{\mathbb{R}} J)\ l\ (is\ ?L \iff ?R)$

<proof>

lemma *matrix-decomposition-lemma*:

fixes $A :: real^{n^2}$

assumes *markov* A

shows $spec\text{-}bound\ A\ l \iff (\exists E.\ A = (1-l) *_{\mathbb{R}} J + l *_{\mathbb{R}} E \wedge matrix\text{-}norm\text{-}bound\ E\ 1 \wedge l \geq 0)$

(*is* $?L \iff ?R$)

<proof>

lemma *hitting-property-alg*:

fixes $S :: (n :: finite)\ set$

assumes *l-range*: $l \in \{0..1\}$

defines $P \equiv \text{diag } (\text{ind-vec } S)$
defines $\mu \equiv \text{card } S / \text{CARD}('n)$
assumes $\bigwedge M. M \in \text{set } Ms \implies \text{spec-bound } M \ l \ \wedge \ \text{markov } M$
shows $\text{foldl } (\lambda x M. P *v (M *v x)) (P *v \text{stat}) Ms \cdot 1 \leq (\mu + l * (1 - \mu))^{\wedge} (\text{length } Ms + 1)$
 <proof>

lemma *upto-append*:
assumes $i \leq j \ j \leq k$
shows $[i..<j]@ [j..<k] = [i..<k]$
 <proof>

definition *bool-list-split* :: *bool list* \Rightarrow (*nat list* \times *nat*)
where *bool-list-split* $xs = \text{foldl } (\lambda (ys,z) x. (\text{if } x \text{ then } (ys@[z],0) \text{ else } (ys,z+1))) ([],0) xs$

lemma *bool-list-split*:
assumes *bool-list-split* $xs = (ys,z)$
shows $xs = \text{concat } (\text{map } (\lambda k. \text{replicate } k \ \text{False}@[\text{True}]) \ ys) @ \text{replicate } z \ \text{False}$
 <proof>

lemma *bool-list-split-count*:
assumes *bool-list-split* $xs = (ys,z)$
shows $\text{length } (\text{filter } \text{id } xs) = \text{length } ys$
 <proof>

lemma *foldl-concat*:
 $\text{foldl } f \ a \ (\text{concat } xss) = \text{foldl } (\lambda y \ xs. \text{foldl } f \ y \ xs) \ a \ xss$
 <proof>

lemma *hitting-property-alg-2*:
fixes $S :: ('n :: \text{finite}) \ \text{set}$ **and** $l :: \text{nat}$
fixes $M :: \text{real}^{\wedge n} \ \text{real}^{\wedge n}$
assumes $\alpha\text{-range}: \alpha \in \{0..1\}$
assumes $I \subseteq \{..<l\}$
defines $P \ i \equiv (\text{if } i \in I \text{ then } \text{diag } (\text{ind-vec } S) \text{ else } \text{mat } 1)$
defines $\mu \equiv \text{real } (\text{card } S) / \text{real } (\text{CARD}('n))$
assumes *spec-bound* $M \ \alpha$ *markov* M
shows
 $\text{foldl } (\lambda x \ M. M *v x) \ \text{stat } (\text{intersperse } M \ (\text{map } P \ [0..<l])) \cdot 1 \leq (\mu + \alpha * (1 - \mu))^{\wedge} \text{card } I$
 (**is** ?L \leq ?R)
 <proof>

lemma *uniform-property-alg*:
fixes $x :: ('n :: \text{finite})$ **and** $l :: \text{nat}$
assumes $i < l$
defines $P \ j \equiv (\text{if } j = i \text{ then } \text{diag } (\text{ind-vec } \{x\}) \text{ else } \text{mat } 1)$
assumes *markov* M
shows $\text{foldl } (\lambda x \ M. M *v x) \ \text{stat } (\text{intersperse } M \ (\text{map } P \ [0..<l])) \cdot 1 = 1 / \text{CARD}('n)$
 (**is** ?L = ?R)
 <proof>

end

lemma *foldl-matrix-mult-expand*:
fixes $Ms :: (('r :: \{\text{semiring-1}, \text{comm-monoid-mult}\})^{\wedge} a^{\wedge} a) \ \text{list}$
shows $(\text{foldl } (\lambda x \ M. M *v x) \ a \ Ms) \ \$ \ k = (\sum x \mid \text{length } x = \text{length } Ms + 1 \ \wedge \ x! \ \text{length } Ms = k. (\prod i < \text{length } Ms. (Ms ! i) \ $(x ! (i+1)) \ $(x ! i)) * a \ $(x ! 0))$
 <proof>

```

lemma foldl-matrix-mult-expand-2:
  fixes  $M_s :: (\text{real}^a)^a \text{ list}$ 
  shows  $(\text{foldl } (\lambda x M. M * v x) a M_s) \cdot 1 = (\sum x \mid \text{length } x = \text{length } M_s + 1. \prod_{i < \text{length } M_s. (M_s ! i) \$ (x ! (i+1)) \$ (x ! i)} * a \$ (x ! 0))$ 
  (is  $?L = ?R$ )
  <proof>

end

```

6 Spectral Theory

This section establishes the correspondence of the variationally defined expansion parameters with the definitions using the spectrum of the stochastic matrix. Additionally stronger results for the expansion parameters are derived.

```

theory Expander-Graphs-Eigenvalues
imports
  Expander-Graphs-Algebra
  Expander-Graphs-TTS
  Perron-Frobenius.HMA-Connect
  Commuting-Hermitian.Commuting-Hermitian
begin

unbundle intro-cong-syntax

hide-const Matrix-Legacy.transpose
hide-const Matrix-Legacy.row
hide-const Matrix-Legacy.mat
hide-const Matrix.mat
hide-const Matrix.row
hide-fact Matrix-Legacy.row-def
hide-fact Matrix-Legacy.mat-def
hide-fact Matrix.vec-eq-iff
hide-fact Matrix.mat-def
hide-fact Matrix.row-def
no-notation Matrix.scalar-prod (infix  $\langle \cdot \rangle$  70)
no-notation Ordered-Semiring.max ( $\langle \text{Max} \rangle$ )

```

```

lemma mult-right-mono':  $y \geq (0 :: \text{real}) \implies x \leq z \vee y = 0 \implies x * y \leq z * y$ 
  <proof>

```

```

lemma poly-prod-zero:
  fixes  $x :: 'a :: \text{idom}$ 
  assumes poly  $(\prod_{a \in \# xs. [- a, 1:]}) x = 0$ 
  shows  $x \in \# xs$ 
  <proof>

```

```

lemma poly-prod-inj-aux-1:
  fixes  $xs \ ys :: ('a :: \text{idom}) \text{ multiset}$ 
  assumes  $x \in \# xs$ 
  assumes  $(\prod_{a \in \# xs. [- a, 1:]}) = (\prod_{a \in \# ys. [- a, 1:]})$ 
  shows  $x \in \# ys$ 
  <proof>

```

```

lemma poly-prod-inj-aux-2:
  fixes  $xs \ ys :: ('a :: \text{idom}) \text{ multiset}$ 
  assumes  $x \in \# xs \cup \# ys$ 
  assumes  $(\prod_{a \in \# xs. [- a, 1:]}) = (\prod_{a \in \# ys. [- a, 1:]})$ 

```

shows $x \in\# xs \cap\# ys$
 ⟨proof⟩

lemma *poly-prod-inj*:
fixes $xs\ ys :: ('a :: idom)\ multiset$
assumes $(\prod a \in\#xs. [-\ a,\ 1:]) = (\prod a \in\#ys. [-\ a,\ 1:])$
shows $xs = ys$
 ⟨proof⟩

definition *eigenvalues* $:: ('a :: comm-ring-1)\ ^n\ ^n \Rightarrow 'a\ multiset$
where
eigenvalues $A = (SOME\ as.\ charpoly\ A = (\prod a \in\#as. [-\ a,\ 1:]) \wedge\ size\ as = CARD\ ('n))$

lemma *char-poly-factorized-hma*:
fixes $A :: complex\ ^n\ ^n$
shows $\exists\ as.\ charpoly\ A = (\prod a \leftarrow as. [-\ a,\ 1:]) \wedge\ length\ as = CARD\ ('n)$
 ⟨proof⟩

lemma *eigvals-poly-length*:
fixes $A :: complex\ ^n\ ^n$
shows
 $charpoly\ A = (\prod a \in\#eigenvalues\ A. [-\ a,\ 1:])$ (**is** ?A)
 $size\ (eigenvalues\ A) = CARD\ ('n)$ (**is** ?B)
 ⟨proof⟩

lemma *similar-matrix-eigvals*:
fixes $A\ B :: complex\ ^n\ ^n$
assumes *similar-matrix* $A\ B$
shows $eigenvalues\ A = eigenvalues\ B$
 ⟨proof⟩

definition *upper-triangular-hma* $:: 'a :: zero\ ^n\ ^n \Rightarrow bool$
where *upper-triangular-hma* $A \equiv$
 $\forall\ i.\ \forall\ j.\ (to-nat\ j < Bij-Nat.to-nat\ i \longrightarrow A\ \$h\ i\ \$h\ j = 0)$

lemma *for-all-reindex2*:
assumes $range\ f = A$
shows $(\forall\ x \in A.\ \forall\ y \in A.\ P\ x\ y) \longleftrightarrow (\forall\ x\ y.\ P\ (f\ x)\ (f\ y))$
 ⟨proof⟩

lemma *upper-triangular-hma*:
fixes $A :: ('a :: zero)\ ^n\ ^n$
shows $upper-triangular\ (from-hma_m\ A) = upper-triangular-hma\ A$ (**is** ?L = ?R)
 ⟨proof⟩

lemma *from-hma-carrier*:
fixes $A :: 'a\ ^n\ ^n\ ^m\ ^m$
shows $from-hma_m\ A \in carrier-mat\ (CARD\ ('m))\ (CARD\ ('n))$
 ⟨proof⟩

definition *diag-mat-hma* $:: 'a\ ^n\ ^n \Rightarrow 'a\ multiset$
where *diag-mat-hma* $A = image-mset\ (\lambda i.\ A\ \$h\ i\ \$h\ i)$ (*mset-set UNIV*)

lemma *diag-mat-hma*:
fixes $A :: 'a\ ^n\ ^n$
shows $mset\ (diag-mat\ (from-hma_m\ A)) = diag-mat-hma\ A$ (**is** ?L = ?R)
 ⟨proof⟩

definition *adjoint-hma* :: $\text{complex}^{\wedge m} \wedge n \Rightarrow \text{complex}^{\wedge n} \wedge m$ **where**
adjoint-hma $A = \text{map-matrix } \text{cnj } (\text{transpose } A)$

lemma *adjoint-hma-eq*: $\text{adjoint-hma } A \ \$h\ i\ \$h\ j = \text{cnj } (A \ \$h\ j\ \$h\ i)$
 $\langle \text{proof} \rangle$

lemma *adjoint-hma*:
fixes $A :: \text{complex}^{\wedge (n::\text{finite})} \wedge (m::\text{finite})$
shows $\text{mat-adjoint } (\text{from-hma}_m A) = \text{from-hma}_m (\text{adjoint-hma } A)$
 $\langle \text{proof} \rangle$

definition *cinner* **where** $\text{cinner } v\ w = \text{scalar-product } v (\text{map-vector } \text{cnj } w)$

context

includes *lifting-syntax*

begin

lemma *cinner-hma*:
fixes $x\ y :: \text{complex}^{\wedge n}$
shows $\text{cinner } x\ y = (\text{from-hma}_v x) \cdot c (\text{from-hma}_v y)$ (**is** $?L = ?R$)
 $\langle \text{proof} \rangle$

lemma *cinner-hma-transfer[transfer-rule]*:
 $(\text{HMA-V} \implies \text{HMA-V} \implies (=)) (\cdot c) \text{cinner}$
 $\langle \text{proof} \rangle$

lemma *adjoint-hma-transfer[transfer-rule]*:
 $(\text{HMA-M} \implies \text{HMA-M}) (\text{mat-adjoint}) \text{adjoint-hma}$
 $\langle \text{proof} \rangle$

end

lemma *adjoint-adjoint-id[simp]*: $\text{adjoint-hma } (\text{adjoint-hma } A) = A$
 $\langle \text{proof} \rangle$

lemma *adjoint-def-alter-hma*:
 $\text{cinner } (A * v\ v) w = \text{cinner } v (\text{adjoint-hma } A * v\ w)$
 $\langle \text{proof} \rangle$

lemma *cinner-0*: $\text{cinner } 0\ 0 = 0$
 $\langle \text{proof} \rangle$

lemma *cinner-scale-left*: $\text{cinner } (a * s\ v) w = a * \text{cinner } v\ w$
 $\langle \text{proof} \rangle$

lemma *cinner-scale-right*: $\text{cinner } v (a * s\ w) = \text{cnj } a * \text{cinner } v\ w$
 $\langle \text{proof} \rangle$

lemma *norm-of-real*:
shows $\text{norm } (\text{map-vector } \text{complex-of-real } v) = \text{norm } v$
 $\langle \text{proof} \rangle$

definition *unitary-hma* :: $\text{complex}^{\wedge n} \wedge n \Rightarrow \text{bool}$
where $\text{unitary-hma } A \iff A ** \text{adjoint-hma } A = \text{Finite-Cartesian-Product.mat } 1$

definition *unitarily-equiv-hma* **where**
 $\text{unitarily-equiv-hma } A\ B\ U \equiv (\text{unitary-hma } U \wedge \text{similar-matrix-wit } A\ B\ U (\text{adjoint-hma } U))$

definition *diagonal-mat* :: ('a::zero)^(n::finite)^n ⇒ bool **where**
diagonal-mat A ≡ (∀ i. ∀ j. i ≠ j → A \$ i \$ j = 0)

lemma *diagonal-mat-ex*:
assumes *diagonal-mat* A
shows A = diag (χ i. A \$ i \$ i)
⟨proof⟩

lemma *diag-diagonal-mat[simp]*: *diagonal-mat* (diag x)
⟨proof⟩

lemma *diag-imp-upper-tri*: *diagonal-mat* A ⇒ *upper-triangular-hma* A
⟨proof⟩

definition *unitary-diag* **where**
unitary-diag A b U ≡ *unitarily-equiv-hma* A (diag b) U

definition *real-diag-decomp-hma* **where**
real-diag-decomp-hma A d U ≡ *unitary-diag* A d U ∧
(∀ i. d \$ i ∈ Reals)

definition *hermitian-hma* :: complex^n^n ⇒ bool **where**
hermitian-hma A = (*adjoint-hma* A = A)

lemma *from-hma-one*:
*from-hma*_m (mat 1 :: (('a::{one,zero})^n^n)) = 1_m CARD('n)
⟨proof⟩

lemma *from-hma-mult*:
fixes A :: ('a :: semiring-1)^m^n
fixes B :: 'a^k^m::finite
shows *from-hma*_m A * *from-hma*_m B = *from-hma*_m (A ** B)
⟨proof⟩

lemma *hermitian-hma*:
hermitian-hma A = *hermitian* (*from-hma*_m A)
⟨proof⟩

lemma *unitary-hma*:
fixes A :: complex^n^n
shows *unitary-hma* A = *unitary* (*from-hma*_m A) (is ?L = ?R)
⟨proof⟩

lemma *unitary-hmaD*:
fixes A :: complex^n^n
assumes *unitary-hma* A
shows *adjoint-hma* A ** A = mat 1 (is ?A) A ** *adjoint-hma* A = mat 1 (is ?B)
⟨proof⟩

lemma *unitary-hma-adjoint*:
assumes *unitary-hma* A
shows *unitary-hma* (*adjoint-hma* A)
⟨proof⟩

lemma *unitarily-equiv-hma*:
fixes A :: complex^n^n
shows *unitarily-equiv-hma* A B U =
unitarily-equiv (*from-hma*_m A) (*from-hma*_m B) (*from-hma*_m U)

(is ?L = ?R)
 <proof>

lemma *Matrix-diagonal-matD*:
 assumes *Matrix.diagonal-mat* A
 assumes $i < \text{dim-row } A$ $j < \text{dim-col } A$
 assumes $i \neq j$
 shows $A \text{ \$(\$ (i,j) = 0}$
 <proof>

lemma *diagonal-mat-hma*:
 fixes $A :: ('a :: \text{zero})^{\wedge}('n :: \text{finite})^{\wedge}n$
 shows $\text{diagonal-mat } A = \text{Matrix.diagonal-mat } (\text{from-hma}_m A)$ (is ?L = ?R)
 <proof>

lemma *unitary-diag-hma*:
 fixes $A :: \text{complex}^{\wedge}n^{\wedge}n$
 shows $\text{unitary-diag } A \text{ d } U =$
 $\text{Spectral-Theory-Complements.unitary-diag } (\text{from-hma}_m A) (\text{from-hma}_m (\text{diag } d)) (\text{from-hma}_m$
 $U)$
 <proof>

lemma *real-diag-decomp-hma*:
 fixes $A :: \text{complex}^{\wedge}n^{\wedge}n$
 shows $\text{real-diag-decomp-hma } A \text{ d } U =$
 $\text{real-diag-decomp } (\text{from-hma}_m A) (\text{from-hma}_m (\text{diag } d)) (\text{from-hma}_m U)$
 <proof>

lemma *diagonal-mat-diag-ex-hma*:
 assumes *Matrix.diagonal-mat* A $A \in \text{carrier-mat } \text{CARD}('n) \text{ CARD} ('n :: \text{finite})$
 shows $\text{from-hma}_m (\text{diag } (\chi (i::'n). A \text{ \$(\$ (to-nat } i, \text{to-nat } i)))) = A$
 <proof>

theorem *commuting-hermitian-family-diag-hma*:
 fixes $Af :: (\text{complex}^{\wedge}n^{\wedge}n) \text{ set}$
 assumes *finite* Af
 and $Af \neq \{\}$
 and $\bigwedge A. A \in Af \implies \text{hermitian-hma } A$
 and $\bigwedge A B. A \in Af \implies B \in Af \implies A ** B = B ** A$
 shows $\exists U. \forall A \in Af. \exists B. \text{real-diag-decomp-hma } A B U$
 <proof>

lemma *char-poly-upper-triangular*:
 fixes $A :: \text{complex}^{\wedge}n^{\wedge}n$
 assumes *upper-triangular-hma* A
 shows $\text{charpoly } A = (\prod a \in \# \text{diag-mat-hma } A. [:- a, 1:])$
 <proof>

lemma *upper-tri-eigvals*:
 fixes $A :: \text{complex}^{\wedge}n^{\wedge}n$
 assumes *upper-triangular-hma* A
 shows $\text{eigenvalues } A = \text{diag-mat-hma } A$
 <proof>

lemma *cinner-self*:
 fixes $v :: \text{complex}^{\wedge}n$
 shows $\text{cinner } v v = \text{norm } v^2$
 <proof>

lemma *unitary-iso*:

assumes *unitary-hma* U

shows $\text{norm } (U *v v) = \text{norm } v$

$\langle \text{proof} \rangle$

lemma (**in** *semiring-hom*) *mult-mat-vec-hma*:

$\text{map-vector hom } (A *v v) = \text{map-matrix hom } A *v \text{ map-vector hom } v$

$\langle \text{proof} \rangle$

lemma (**in** *semiring-hom*) *mat-hom-mult-hma*:

$\text{map-matrix hom } (A ** B) = \text{map-matrix hom } A ** \text{map-matrix hom } B$

$\langle \text{proof} \rangle$

context *regular-graph-tts*

begin

lemma *to-nat-less-n*: $\text{to-nat } (x::'n) < n$

$\langle \text{proof} \rangle$

lemma *to-nat-from-nat*: $x < n \implies \text{to-nat } (\text{from-nat } x :: 'n) = x$

$\langle \text{proof} \rangle$

lemma *hermitian-A*: *hermitian-hma* A

$\langle \text{proof} \rangle$

lemma *nonneg-A*: *nonneg-mat* A

$\langle \text{proof} \rangle$

lemma *g-step-1*:

assumes $v \in \text{verts } G$

shows $g\text{-step } (\lambda \cdot 1) v = 1$ (**is** $?L = ?R$)

$\langle \text{proof} \rangle$

lemma *markov*: *markov* $(A :: \text{real}^n{}^n)$

$\langle \text{proof} \rangle$

lemma *nonneg-J*: *nonneg-mat* J

$\langle \text{proof} \rangle$

lemma *J-eigvals*: *eigenvalues* $J = \{\#1::\text{complex}\# \} + \text{replicate-mset } (n - 1) 0$

$\langle \text{proof} \rangle$

lemma *J-markov*: *markov* J

$\langle \text{proof} \rangle$

lemma *markov-complex-apply*:

assumes *markov* M

shows $(\text{map-matrix complex-of-real } M) *v (1 :: \text{complex}^n) = 1$ (**is** $?L = ?R$)

$\langle \text{proof} \rangle$

lemma *J-A-comm-real*: $J ** A = A ** (J :: \text{real}^n{}^n)$

$\langle \text{proof} \rangle$

lemma *J-A-comm*: $J ** A = A ** (J :: \text{complex}^n{}^n)$ (**is** $?L = ?R$)

$\langle \text{proof} \rangle$

definition $\gamma_a :: 'n \text{ itself} \implies \text{real}$ **where**

$\gamma_a = (\text{if } n > 1 \text{ then Max-mset (image-mset cmod (eigenvalues } A - \{\#1\#\}) \text{) else } 0)$

definition $\gamma_2 :: 'n \text{ itself} \Rightarrow \text{real where}$

$\gamma_2 = (\text{if } n > 1 \text{ then Max-mset } \{\# \text{ Re } x. x \in \# (\text{eigenvalues } A - \{\#1\#\})\# \text{ else } 0)$

lemma *J-sym: hermitian-hma J*

<proof>

lemma

shows *evs-real: set-mset (eigenvalues A::complex multiset) $\subseteq \mathbb{R}$ (is ?R1)*

and *ev-1: (1::complex) $\in \#$ eigenvalues A*

and *γ_a -ge-0: $\gamma_a \text{ TYPE } ('n) \geq 0$*

and *find-any-ev:*

$\forall \alpha \in \# \text{ eigenvalues } A - \{\#1\#\}. \exists v. \text{cinner } v \ 1 = 0 \wedge v \neq 0 \wedge A * v \ v = \alpha * s \ v$

and *γ_a -bound: $\forall v. \text{cinner } v \ 1 = 0 \longrightarrow \text{norm } (A * v \ v) \leq \gamma_a \text{ TYPE } ('n) * \text{norm } v$*

and *γ_2 -bound: $\forall (v::\text{real}^n). v \cdot 1 = 0 \longrightarrow v \cdot (A * v \ v) \leq \gamma_2 \text{ TYPE } ('n) * \text{norm } v^2$*

<proof>

lemma *find-any-real-ev:*

assumes *complex-of-real $\alpha \in \#$ eigenvalues A - $\{\#1\#\}$*

shows $\exists v. v \cdot 1 = 0 \wedge v \neq 0 \wedge A * v \ v = \alpha * s \ v$

<proof>

lemma *size-evs:*

size (eigenvalues A - $\{\#1::\text{complex}\#\}) = n - 1$

<proof>

lemma *find- γ_2 :*

assumes $n > 1$

shows $\gamma_a \text{ TYPE } ('n) \in \# \text{ image-mset cmod (eigenvalues } A - \{\#1::\text{complex}\#\})$

<proof>

lemma *γ_2 -real-ev:*

assumes $n > 1$

shows $\exists v. (\exists \alpha. \text{abs } \alpha = \gamma_a \text{ TYPE } ('n) \wedge v \cdot 1 = 0 \wedge v \neq 0 \wedge A * v \ v = \alpha * s \ v)$

<proof>

lemma *γ_a -real-bound:*

fixes $v :: \text{real}^n$

assumes $v \cdot 1 = 0$

shows $\text{norm } (A * v \ v) \leq \gamma_a \text{ TYPE } ('n) * \text{norm } v$

<proof>

lemma *Λ_e -eq- Λ : $\Lambda_a = \gamma_a \text{ TYPE } ('n)$*

<proof>

lemma *γ_2 -ev:*

assumes $n > 1$

shows $\exists v. v \cdot 1 = 0 \wedge v \neq 0 \wedge A * v \ v = \gamma_2 \text{ TYPE } ('n) * s \ v$

<proof>

lemma *Λ_2 -eq- γ_2 : $\Lambda_2 = \gamma_2 \text{ TYPE } ('n)$*

<proof>

lemma *expansionD2:*

assumes $g\text{-inner } f \ (\lambda-. 1) = 0$

shows $g\text{-norm } (g\text{-step } f) \leq \Lambda_a * g\text{-norm } f \text{ (is ?L } \leq \text{ ?R)}$

<proof>

lemma *rayleigh-bound*:
fixes $v :: \text{real}^n$
shows $|v \cdot (A * v)| \leq \text{norm } v^2$
 $\langle \text{proof} \rangle$

The following implies that two-sided expanders are also one-sided expanders.

lemma $\Lambda_2\text{-range}$: $|\Lambda_2| \leq \Lambda_a$
 $\langle \text{proof} \rangle$

end

lemmas (**in** *regular-graph*) *expansionD2* =
regular-graph-tts.expansionD2[*OF eg-tts-1*,
internalize-sort 'n :: finite, OF - regular-graph-axioms,
unfolded remove-finite-premise, cancel-type-definition, OF verts-non-empty]

lemmas (**in** *regular-graph*) $\Lambda_2\text{-range}$ =
regular-graph-tts. $\Lambda_2\text{-range}$ [*OF eg-tts-1*,
internalize-sort 'n :: finite, OF - regular-graph-axioms,
unfolded remove-finite-premise, cancel-type-definition, OF verts-non-empty]

unbundle *no intro-cong-syntax*

end

7 Cheeger Inequality

The Cheeger inequality relates edge expansion (a combinatorial property) with the second largest eigenvalue.

theory *Expander-Graphs-Cheeger-Inequality*
imports *Expander-Graphs-Eigenvalues*
begin

unbundle *intro-cong-syntax*
hide-const *Quantum.T*

context *regular-graph*
begin

lemma *edge-expansionD2*:
assumes $m = \text{card } (S \cap \text{verts } G) \ 2 * m \leq n$
shows $\Lambda_e * m \leq \text{real } (\text{card } (\text{edges-betw } S \ (-S)))$
 $\langle \text{proof} \rangle$

lemma *edges-betw-sym*:
 $\text{card } (\text{edges-betw } S \ T) = \text{card } (\text{edges-betw } T \ S)$ (**is** $?L = ?R$)
 $\langle \text{proof} \rangle$

lemma *edges-betw-reg*:
assumes $S \subseteq \text{verts } G$
shows $\text{card } (\text{edges-betw } S \ UNIV) = \text{card } S * d$ (**is** $?L = ?R$)
 $\langle \text{proof} \rangle$

The following proof follows Hoory et al. [4, §4.5.1].

lemma *cheeger-aux-2*:

assumes $n > 1$
shows $\Lambda_e \geq d*(1-\Lambda_2)/2$
 $\langle proof \rangle$

end

lemma *surj-onI*:
assumes $\bigwedge x. x \in B \implies g x \in A \wedge f (g x) = x$
shows $B \subseteq f ' A$
 $\langle proof \rangle$

lemma *find-sorted-bij-1*:
fixes $g :: 'a \Rightarrow ('b :: linorder)$
assumes *finite* S
shows $\exists f. \text{bij-betw } f \{..< \text{card } S\} S \wedge \text{mono-on } \{..< \text{card } S\} (g \circ f)$
 $\langle proof \rangle$

lemma *find-sorted-bij-2*:
fixes $g :: 'a \Rightarrow ('b :: linorder)$
assumes *finite* S
shows $\exists f. \text{bij-betw } f S \{..< \text{card } S\} \wedge (\forall x y. x \in S \wedge y \in S \wedge f x < f y \longrightarrow g x \leq g y)$
 $\langle proof \rangle$

context *regular-graph-tts*
begin

Normalized Laplacian of the graph

definition L **where** $L = \text{mat } 1 - A$

lemma *L-pos-semidefinite*:
fixes $v :: \text{real } ^n$
shows $v \cdot (L * v) \geq 0$
 $\langle proof \rangle$

The following proof follows Hoory et al. [4, §4.5.2].

lemma *cheeger-aux-1*:
assumes $n > 1$
shows $\Lambda_e \leq d * \text{sqrt } (2 * (1-\Lambda_2))$
 $\langle proof \rangle$

end

context *regular-graph*
begin

lemmas *cheeger-aux-1* =
 $\text{regular-graph-tts.cheeger-aux-1} [OF \text{ eg-tts-1,}$
 $\text{internalize-sort } 'n :: \text{finite, } OF - \text{regular-graph-axioms,}$
 $\text{unfolded remove-finite-premise, cancel-type-definition, } OF \text{ verts-non-empty}]$

theorem *cheeger-inequality*:
assumes $n > 1$
shows $\Lambda_e \in \{d * (1 - \Lambda_2) / 2.. d * \text{sqrt } (2 * (1 - \Lambda_2))\}$
 $\langle proof \rangle$

unbundle *no intro-cong-syntax*

end

end

8 Margulis Gabber Galil Construction

This section formalizes the Margulis-Gabber-Galil expander graph, which is defined on the product space $\mathbb{Z}_n \times \mathbb{Z}_n$. The construction is an adaptation of graph introduced by Margulis [8], for which he gave a non-constructive proof of its spectral gap. Later Gabber and Galil [3] adapted the graph and derived an explicit spectral gap, i.e., that the second largest eigenvalue is bounded by $\frac{5}{8}\sqrt{2}$. The proof was later improved by Jimbo and Marouka [6] using Fourier Analysis. Hoory et al. [4, §8] present a slight simplification of that proof (due to Boppala) which this formalization is based on.

theory *Expander-Graphs-MGG*

imports

HOL-Analysis.Complex-Transcendental

HOL-Decision-Proc.Approximation

Expander-Graphs-Definition

begin

datatype ('a, 'b) *arc* = *Arc* (*arc-tail*: 'a) (*arc-head*: 'a) (*arc-label*: 'b)

fun *mgg-graph-step* :: *nat* \Rightarrow (*int* \times *int*) \Rightarrow (*nat* \times *int*) \Rightarrow (*int* \times *int*)

where *mgg-graph-step* *n* (*i,j*) (*l*, σ) =

[((*i*+ σ *(2**j*+0)) mod *int n*, *j*), (*i*, (*j*+ σ *(2**i*+0)) mod *int n*)
, ((*i*+ σ *(2**j*+1)) mod *int n*, *j*), (*i*, (*j*+ σ *(2**i*+1)) mod *int n*)] ! *l*

definition *mgg-graph* :: *nat* \Rightarrow (*int* \times *int*, (*int* \times *int*, *nat* \times *int*) *arc*) *pre-digraph* **where**

mgg-graph *n* =

(| *verts* = {0..*n*} \times {0..*n*},

arcs = (λ (*t,l*). (*Arc* *t* (*mgg-graph-step* *n* *t* *l*) *l*))'({0..*int n*} \times {0..*int n*}) \times ({..*4*} \times {-1,1})),

tail = *arc-tail*,

head = *arc-head* |)

locale *margulis-gaber-galil* =

fixes *m* :: *nat*

assumes *m-gt-0*: *m* > 0

begin

abbreviation *G* **where** *G* \equiv *mgg-graph* *m*

lemma *wf-digraph*: *wf-digraph* (*mgg-graph* *m*)

<proof>

lemma *mgg-finite*: *fin-digraph* (*mgg-graph* *m*)

<proof>

interpretation *fin-digraph* *mgg-graph* *m*

<proof>

definition *arcs-pos* :: (*int* \times *int*, *nat* \times *int*) *arc* *set*

where *arcs-pos* = (λ (*t,l*). (*Arc* *t* (*mgg-graph-step* *m* *t* (*l*,1)) (*l*, 1)))'(*verts* *G* \times {..*4*})

definition *arcs-neg* :: (*int* \times *int*, *nat* \times *int*) *arc* *set*

where *arcs-neg* = (λ (*h,l*). (*Arc* (*mgg-graph-step* *m* *h* (*l*,1)) *h* (*l*,-1)))'(*verts* *G* \times {..*4*})

lemma *arcs-sym*:

arcs *G* = *arcs-pos* \cup *arcs-neg*

<proof>

lemma *sym: symmetric-multi-graph (mgg-graph m)*

<proof>

lemma *out-deg:*

assumes $v \in \text{verts } G$

shows $\text{out-degree } G \ v = 8$

<proof>

lemma *verts-ne:*

$\text{verts } G \neq \{\}$

<proof>

sublocale *regular-graph mgg-graph m*

<proof>

lemma *d-eq-8: d = 8*

<proof>

We start by introducing Fourier Analysis on the torus $\mathbb{Z}_n \times \mathbb{Z}_n$. The following is too specialized for a general AFP entry.

lemma *g-inner-sum-left:*

assumes *finite I*

shows $g\text{-inner } (\lambda x. (\sum i \in I. f \ i \ x)) \ g = (\sum i \in I. g\text{-inner } (f \ i) \ g)$

<proof>

lemma *g-inner-sum-right:*

assumes *finite I*

shows $g\text{-inner } f \ (\lambda x. (\sum i \in I. g \ i \ x)) = (\sum i \in I. g\text{-inner } f \ (g \ i))$

<proof>

lemma *g-inner-reindex:*

assumes *bij-betw h (verts G) (verts G)*

shows $g\text{-inner } f \ g = g\text{-inner } (\lambda x. (f \ (h \ x))) \ (\lambda x. (g \ (h \ x)))$

<proof>

definition $\omega_F :: \text{real} \Rightarrow \text{complex}$ **where** $\omega_F \ x = \text{cis } (2 * \pi * x / m)$

lemma *ω_F -simps:*

$\omega_F \ (x + y) = \omega_F \ x * \omega_F \ y$

$\omega_F \ (x - y) = \omega_F \ x * \omega_F \ (-y)$

$\text{conj } (\omega_F \ x) = \omega_F \ (-x)$

<proof>

lemma *ω_F -cong:*

fixes $x \ y :: \text{int}$

assumes $x \bmod m = y \bmod m$

shows $\omega_F \ (\text{of-int } x) = \omega_F \ (\text{of-int } y)$

<proof>

lemma *cis-eq-1-imp:*

assumes $\text{cis } (2 * \pi * x) = 1$

shows $x \in \mathbb{Z}$

<proof>

lemma *ω_F -eq-1-iff:*

fixes $x :: \text{int}$

shows $\omega_F x = 1 \longleftrightarrow x \bmod m = 0$
<proof>

definition *FT* :: $(int \times int \Rightarrow complex) \Rightarrow (int \times int \Rightarrow complex)$
where $FT f v = g\text{-inner } f (\lambda x. \omega_F (fst x * fst v + snd x * snd v))$

lemma *FT-altdef*: $FT f (u,v) = g\text{-inner } f (\lambda x. \omega_F (fst x * u + snd x * v))$
<proof>

lemma *FT-add*: $FT (\lambda x. f x + g x) v = FT f v + FT g v$
<proof>

lemma *FT-zero*: $FT (\lambda x. 0) v = 0$
<proof>

lemma *FT-sum*:
assumes *finite I*
shows $FT (\lambda x. (\sum i \in I. f i x)) v = (\sum i \in I. FT (f i) v)$
<proof>

lemma *FT-scale*: $FT (\lambda x. c * f x) v = c * FT f v$
<proof>

lemma *FT-cong*:
assumes $\bigwedge x. x \in \text{verts } G \implies f x = g x$
shows $FT f = FT g$
<proof>

lemma *parseval*:
 $g\text{-inner } f g = g\text{-inner } (FT f) (FT g) / m^{\wedge 2}$ (**is** ?L = ?R)
<proof>

lemma *plancharel*:
 $(\sum v \in \text{verts } G. \text{norm } (f v)^{\wedge 2}) = (\sum v \in \text{verts } G. \text{norm } (FT f v)^{\wedge 2}) / m^{\wedge 2}$ (**is** ?L = ?R)
<proof>

lemma *FT-swap*:
 $FT (\lambda x. f (snd x, fst x)) (u,v) = FT f (v,u)$
<proof>

lemma *mod-add-mult-eq*:
fixes $a x y :: int$
shows $(a + x * (y \bmod m)) \bmod m = (a+x*y) \bmod m$
<proof>

definition *periodic* **where** $periodic f = (\forall x y. f (x,y) = f (x \bmod int m, y \bmod int m))$

lemma *periodicD*:
assumes *periodic f*
shows $f (x,y) = f (x \bmod m, y \bmod m)$
<proof>

lemma *periodic-comp*:
assumes *periodic f*
shows *periodic* $(\lambda x. g (f x))$
<proof>

lemma *periodic-cong*:

fixes $x y u v :: \text{int}$
assumes $\text{periodic } f$
assumes $x \bmod m = u \bmod m \ y \bmod m = v \bmod m$
shows $f(x, y) = f(u, v)$
 $\langle \text{proof} \rangle$

lemma $\text{periodic-FT: periodic } (FT f)$
 $\langle \text{proof} \rangle$

lemma $FT\text{-sheer-aux:}$
fixes $u v c d :: \text{int}$
assumes $\text{periodic } f$
shows $FT(\lambda x. f(\text{fst } x, \text{snd } x + c * \text{fst } x + d))(u, v) = \omega_F(d * v) * FT f(u - c * v, v)$
(is ?L = ?R)
 $\langle \text{proof} \rangle$

lemma $FT\text{-sheer:}$
fixes $u v c d :: \text{int}$
assumes $\text{periodic } f$
shows
 $FT(\lambda x. f(\text{fst } x, \text{snd } x + c * \text{fst } x + d))(u, v) = \omega_F(d * v) * FT f(u - c * v, v)$ **(is ?A)**
 $FT(\lambda x. f(\text{fst } x, \text{snd } x + c * \text{fst } x))(u, v) = FT f(u - c * v, v)$ **(is ?B)**
 $FT(\lambda x. f(\text{fst } x + c * \text{snd } x + d, \text{snd } x))(u, v) = \omega_F(d * u) * FT f(u, v - c * u)$ **(is ?C)**
 $FT(\lambda x. f(\text{fst } x + c * \text{snd } x, \text{snd } x))(u, v) = FT f(u, v - c * u)$ **(is ?D)**
 $\langle \text{proof} \rangle$

definition $T_1 :: \text{int} \times \text{int} \Rightarrow \text{int} \times \text{int}$ **where** $T_1 x = ((\text{fst } x + 2 * \text{snd } x) \bmod m, \text{snd } x)$
definition $S_1 :: \text{int} \times \text{int} \Rightarrow \text{int} \times \text{int}$ **where** $S_1 x = ((\text{fst } x - 2 * \text{snd } x) \bmod m, \text{snd } x)$
definition $T_2 :: \text{int} \times \text{int} \Rightarrow \text{int} \times \text{int}$ **where** $T_2 x = (\text{fst } x, (\text{snd } x + 2 * \text{fst } x) \bmod m)$
definition $S_2 :: \text{int} \times \text{int} \Rightarrow \text{int} \times \text{int}$ **where** $S_2 x = (\text{fst } x, (\text{snd } x - 2 * \text{fst } x) \bmod m)$

definition $\gamma\text{-aux} :: \text{int} \times \text{int} \Rightarrow \text{real} \times \text{real}$
where $\gamma\text{-aux } x = (|\text{fst } x / m - 1/2|, |\text{snd } x / m - 1/2|)$

definition $\text{compare} :: \text{real} \times \text{real} \Rightarrow \text{real} \times \text{real} \Rightarrow \text{bool}$
where $\text{compare } x y = (\text{fst } x \leq \text{fst } y \wedge \text{snd } x \leq \text{snd } y \wedge x \neq y)$

The value here is different from the value in the source material. This is because the proof in Hoory [4, §8] only establishes the bound $\frac{73}{80}$ while this formalization establishes the improved bound of $\frac{5}{8}\sqrt{2}$.

definition $\alpha :: \text{real}$ **where** $\alpha = \text{sqrt } 2$

lemma $\alpha\text{-inv: } 1/\alpha = \alpha/2$
 $\langle \text{proof} \rangle$

definition $\gamma :: \text{int} \times \text{int} \Rightarrow \text{int} \times \text{int} \Rightarrow \text{real}$
where $\gamma x y = (\text{if compare } (\gamma\text{-aux } x) (\gamma\text{-aux } y) \text{ then } \alpha \text{ else } (\text{if compare } (\gamma\text{-aux } y) (\gamma\text{-aux } x) \text{ then } (1 / \alpha) \text{ else } 1))$

lemma $\gamma\text{-sym: } \gamma x y * \gamma y x = 1$
 $\langle \text{proof} \rangle$

lemma $\gamma\text{-nonneg: } \gamma x y \geq 0$
 $\langle \text{proof} \rangle$

definition $\tau :: \text{int} \Rightarrow \text{real}$ **where** $\tau x = |\cos(\text{pi} * x / m)|$

definition $\gamma' :: \text{real} \Rightarrow \text{real} \Rightarrow \text{real}$

where $\gamma' x y = (\text{if } \text{abs } (x - 1/2) < \text{abs } (y - 1/2) \text{ then } \alpha \text{ else } (\text{if } \text{abs } (x-1/2) > \text{abs } (y-1/2) \text{ then } (1 / \alpha) \text{ else } 1))$

definition $\varphi :: \text{real} \Rightarrow \text{real} \Rightarrow \text{real}$

where $\varphi x y = \gamma' y (\text{frac}(y-2*x)) + \gamma' y (\text{frac } (y+2*x))$

lemma γ' -cases:

$\text{abs } (x-1/2) = \text{abs } (y-1/2) \implies \gamma' x y = 1$
 $\text{abs } (x-1/2) > \text{abs } (y-1/2) \implies \gamma' x y = 1/\alpha$
 $\text{abs } (x-1/2) < \text{abs } (y-1/2) \implies \gamma' x y = \alpha$
 $\langle \text{proof} \rangle$

lemma *if-cong-direct*:

assumes $a = b$
assumes $c = d'$
assumes $e = f$
shows $(\text{if } a \text{ then } c \text{ else } e) = (\text{if } b \text{ then } d' \text{ else } f)$
 $\langle \text{proof} \rangle$

lemma γ' -cong:

assumes $\text{abs } (x-1/2) = \text{abs } (u-1/2)$
assumes $\text{abs } (y-1/2) = \text{abs } (v-1/2)$
shows $\gamma' x y = \gamma' u v$
 $\langle \text{proof} \rangle$

lemma *add-swap-cong*:

fixes $x y u v :: 'a :: \text{ab-semigroup-add}$
assumes $x = y \ u = v$
shows $x + u = v + y$
 $\langle \text{proof} \rangle$

lemma *frac-cong*:

fixes $x y :: \text{real}$
assumes $x - y \in \mathbb{Z}$
shows $\text{frac } x = \text{frac } y$
 $\langle \text{proof} \rangle$

lemma *frac-expand*:

fixes $x :: \text{real}$
shows $\text{frac } x = (\text{if } x < (-1) \text{ then } (x - \lfloor x \rfloor) \text{ else } (\text{if } x < 0 \text{ then } (x+1) \text{ else } (\text{if } x < 1 \text{ then } x \text{ else } (\text{if } x < 2 \text{ then } (x-1) \text{ else } (x - \lfloor x \rfloor))))))$
 $\langle \text{proof} \rangle$

lemma *one-minus-frac*:

fixes $x :: \text{real}$
shows $1 - \text{frac } x = (\text{if } x \in \mathbb{Z} \text{ then } 1 \text{ else } \text{frac } (-x))$
 $\langle \text{proof} \rangle$

lemma *abs-rev-cong*:

fixes $x y :: \text{real}$
assumes $x = -y$
shows $\text{abs } x = \text{abs } y$
 $\langle \text{proof} \rangle$

lemma *cos-pi-ge-0*:

assumes $x \in \{-1/2.. 1/2\}$
shows $\cos (\pi * x) \geq 0$
 $\langle \text{proof} \rangle$

The following is the first step in establishing Eq. 15 in Hoory et al. [4, §8]. Afterwards using various symmetries (diagonal, x-axis, y-axis) the result will follow for the entire square $[0, 1] \times [0, 1]$.

lemma *fun-bound-real-3*:

assumes $0 \leq x \leq y \leq 1/2$ $(x,y) \neq (0,0)$

shows $|\cos(\pi x)| * \varphi x y + |\cos(\pi y)| * \varphi y x \leq 2.5 * \text{sqrt } 2$ (**is** $?L \leq ?R$)

<proof>

Extend to square $[0, \frac{1}{2}] \times [0, \frac{1}{2}]$ using symmetry around $x=y$ axis.

lemma *fun-bound-real-2*:

assumes $x \in \{0..1/2\}$ $y \in \{0..1/2\}$ $(x,y) \neq (0,0)$

shows $|\cos(\pi x)| * \varphi x y + |\cos(\pi y)| * \varphi y x \leq 2.5 * \text{sqrt } 2$ (**is** $?L \leq ?R$)

<proof>

Extend to $x > \frac{1}{2}$ using symmetry around $x = \frac{1}{2}$ axis.

lemma *fun-bound-real-1*:

assumes $x \in \{0..<1\}$ $y \in \{0..1/2\}$ $(x,y) \neq (0,0)$

shows $|\cos(\pi x)| * \varphi x y + |\cos(\pi y)| * \varphi y x \leq 2.5 * \text{sqrt } 2$ (**is** $?L \leq ?R$)

<proof>

Extend to $y > \frac{1}{2}$ using symmetry around $y = \frac{1}{2}$ axis.

lemma *fun-bound-real*:

assumes $x \in \{0..<1\}$ $y \in \{0..<1\}$ $(x,y) \neq (0,0)$

shows $|\cos(\pi x)| * \varphi x y + |\cos(\pi y)| * \varphi y x \leq 2.5 * \text{sqrt } 2$ (**is** $?L \leq ?R$)

<proof>

lemma *mod-to-frac*:

fixes $x :: \text{int}$

shows $\text{real-of-int } (x \text{ mod } m) = m * \text{frac } (x/m)$ (**is** $?L = ?R$)

<proof>

lemma *fun-bound*:

assumes $v \in \text{verts } G$ $v \neq (0,0)$

shows $\tau(\text{fst } v) * (\gamma v (S_2 v) + \gamma v (T_2 v)) + \tau(\text{snd } v) * (\gamma v (S_1 v) + \gamma v (T_1 v)) \leq 2.5 * \text{sqrt } 2$
(**is** $?L \leq ?R$)

<proof>

Equation 15 in Proof of Theorem 8.8

lemma *hoory-8-8*:

fixes $f :: \text{int} \times \text{int} \Rightarrow \text{real}$

assumes $\bigwedge x. f x \geq 0$

assumes $f (0,0) = 0$

assumes *periodic* f

shows $g\text{-inner } f (\lambda x. f(S_2 x) * \tau(\text{fst } x) + f(S_1 x) * \tau(\text{snd } x)) \leq 1.25 * \text{sqrt } 2 * g\text{-norm } f^2$
(**is** $?L \leq ?R$)

<proof>

lemma *hoory-8-7*:

fixes $f :: \text{int} \times \text{int} \Rightarrow \text{complex}$

assumes $f (0,0) = 0$

assumes *periodic* f

shows $\text{norm}(g\text{-inner } f (\lambda x. f(S_2 x) * (1 + \omega_F(\text{fst } x)) + f(S_1 x) * (1 + \omega_F(\text{snd } x))))$
 $\leq (2.5 * \text{sqrt } 2) * (\sum v \in \text{verts } G. \text{norm } (f v)^2)$ (**is** $?L \leq ?R$)

<proof>

lemma *hoory-8-3*:

assumes $g\text{-inner } f (\lambda\cdot. 1) = 0$
assumes *periodic* f
shows $|\sum_{(x,y)\in\text{verts } G}. f(x,y)*(f(x+2*y,y)+f(x+2*y+1,y)+f(x,y+2*x)+f(x,y+2*x+1))|$
 $\leq (2.5 * \text{sqrt } 2) * g\text{-norm } f^2$ (**is** $?L \leq ?R$)
 <proof>

Inequality stated before Theorem 8.3 in Hoory.

lemma *mgg-numerical-radius-aux*:
assumes $g\text{-inner } f (\lambda\cdot. 1) = 0$
shows $|\sum a \in \text{arcs } G. f(\text{head } G a) * f(\text{tail } G a)| \leq (5 * \text{sqrt } 2) * g\text{-norm } f^2$ (**is** $?L \leq ?R$)
 <proof>

definition *MGG-bound* :: *real*
where $MGG\text{-bound} = 5 * \text{sqrt } 2 / 8$

Main result: Theorem 8.2 in Hoory.

lemma *mgg-numerical-radius*: $\Lambda_a \leq MGG\text{-bound}$
 <proof>

end

end

9 Random Walks

theory *Expander-Graphs-Walks*

imports

Expander-Graphs-Algebra
Expander-Graphs-Eigenvalues
Expander-Graphs-TTS
Constructive-Chernoff-Bound

begin

unbundle *intro-cong-syntax*

no-notation *Matrix.vec-index* (**infixl** $\langle \$ \rangle$ 100)

hide-const *Matrix.vec-index*

hide-const *Matrix.vec*

no-notation *Matrix.scalar-prod* (**infix** $\langle \cdot \rangle$ 70)

fun *walks'* :: $('a, 'b) \text{ pre-digraph} \Rightarrow \text{nat} \Rightarrow ('a \text{ list}) \text{ multiset}$

where

$walks' G 0 = \text{image-mset } (\lambda x. [x]) (\text{mset-set } (\text{verts } G))$ |

$walks' G (\text{Suc } n) =$

$\text{concat-mset } \{ \# \{ \# w @ [z]. z \in \# \text{ vertices-from } G (\text{last } w) \# \}. w \in \# \text{ walks' } G n \# \}$

definition $walks G l = (\text{case } l \text{ of } 0 \Rightarrow \{ \# [] \# \} \mid \text{Suc } pl \Rightarrow \text{walks' } G pl)$

lemma *Union-image-mono*: $(\bigwedge x. x \in A \Rightarrow f x \subseteq g x) \Rightarrow \bigcup (f ' A) \subseteq \bigcup (g ' A)$

<proof>

context *fin-digraph*

begin

lemma *count-walks'*:

assumes $\text{set } xs \subseteq \text{verts } G$

assumes $\text{length } xs = l+1$

shows $\text{count } (\text{walks}' G l) xs = (\prod i \in \{..<l\}. \text{count } (\text{edges } G) (xs ! i, xs ! (i+1)))$
 <proof>

lemma *count-walks*:

assumes $\text{set } xs \subseteq \text{verts } G$

assumes $\text{length } xs = l \ l > 0$

shows $\text{count } (\text{walks } G l) xs = (\prod i \in \{..<l-1\}. \text{count } (\text{edges } G) (xs ! i, xs ! (i+1)))$

<proof>

lemma *set-walks'*:

$\text{set-mset } (\text{walks}' G l) \subseteq \{xs. \text{set } xs \subseteq \text{verts } G \wedge \text{length } xs = (l+1)\}$

<proof>

lemma *set-walks*:

$\text{set-mset } (\text{walks } G l) \subseteq \{xs. \text{set } xs \subseteq \text{verts } G \wedge \text{length } xs = l\}$

<proof>

lemma *set-walks-2*:

assumes $xs \in \# \text{walks}' G l$

shows $\text{set } xs \subseteq \text{verts } G \ \text{xs} \neq []$

<proof>

lemma *set-walks-3*:

assumes $xs \in \# \text{walks } G l$

shows $\text{set } xs \subseteq \text{verts } G \ \text{length } xs = l$

<proof>

end

lemma *measure-pmf-of-multiset*:

assumes $A \neq \{\#\}$

shows $\text{measure } (\text{pmf-of-multiset } A) S = \text{real } (\text{size } (\text{filter-mset } (\lambda x. x \in S) A)) / \text{size } A$
 (**is** ?L = ?R)

<proof>

lemma *pmf-of-multiset-image-mset*:

assumes $A \neq \{\#\}$

shows $\text{pmf-of-multiset } (\text{image-mset } f A) = \text{map-pmf } f (\text{pmf-of-multiset } A)$

<proof>

context *regular-graph*

begin

lemma *size-walks'*:

$\text{size } (\text{walks}' G l) = \text{card } (\text{verts } G) * d^l$

<proof>

lemma *size-walks*:

$\text{size } (\text{walks } G l) = (\text{if } l > 0 \text{ then } n * d^{l-1} \text{ else } 1)$

<proof>

lemma *walks-nonempty*:

$\text{walks } G l \neq \{\#\}$

<proof>

end

context *regular-graph-tts*

begin

lemma *g-step-remains-orth*:

assumes *g-inner* $f (\lambda\cdot. 1) = 0$

shows *g-inner* (*g-step* f) ($\lambda\cdot. 1$) = 0 (**is** ?L = ?R)

<proof>

lemma *spec-bound*:

spec-bound $A \Lambda_a$

<proof>

A spectral expansion rule that does not require orthogonality of the vector for the stationary distribution:

lemma *expansionD3*:

$|g\text{-inner } f (g\text{-step } f)| \leq \Lambda_a * g\text{-norm } f^{\wedge 2} + (1 - \Lambda_a) * g\text{-inner } f (\lambda\cdot. 1)^{\wedge 2} / n$ (**is** ?L ≤ ?R)

<proof>

definition *ind-mat where* *ind-mat* $S = \text{diag } (\text{ind-vec } (\text{enum-verts } - ' S))$

lemma *walk-distr*:

measure (*pmf-of-multiset* (*walks* $G l$)) $\{\omega. (\forall i < l. \omega ! i \in S i)\} =$

foldl ($\lambda x M. M * v x$) *stat* (*intersperse* $A (\text{map } (\lambda i. \text{ind-mat } (S i)) [0..<l])) \cdot 1$

(**is** ?L = ?R)

<proof>

lemma *hitting-property*:

assumes $S \subseteq \text{verts } G$

assumes $I \subseteq \{..<l\}$

defines $\mu \equiv \text{real } (\text{card } S) / \text{card } (\text{verts } G)$

shows *measure* (*pmf-of-multiset* (*walks* $G l$)) $\{w. \text{set } (\text{nths } w I) \subseteq S\} \leq (\mu + \Lambda_a * (1 - \mu))^{\wedge \text{card } I}$

(**is** ?L ≤ ?R)

<proof>

lemma *uniform-property*:

assumes $i < l \ x \in \text{verts } G$

shows *measure* (*pmf-of-multiset* (*walks* $G l$)) $\{w. w ! i = x\} = 1 / \text{real } (\text{card } (\text{verts } G))$

(**is** ?L = ?R)

<proof>

end

context *regular-graph*

begin

lemmas *expansionD3* =

regular-graph-tts.expansionD3[*OF* *eg-tts-1*,

internalize-sort 'n :: finite, OF - regular-graph-axioms,

unfolded remove-finite-premise, cancel-type-definition, OF verts-non-empty]

lemmas *g-step-remains-orth* =

regular-graph-tts.g-step-remains-orth[*OF* *eg-tts-1*,

internalize-sort 'n :: finite, OF - regular-graph-axioms,

unfolded remove-finite-premise, cancel-type-definition, OF verts-non-empty]

lemmas *hitting-property* =

regular-graph-tts.hitting-property[*OF* *eg-tts-1*,

internalize-sort 'n :: finite, OF - regular-graph-axioms,

unfolded remove-finite-premise, cancel-type-definition, OF verts-non-empty]

```

lemmas uniform-property-2 =
  regular-graph-tts.uniform-property[OF eg-tts-1,
    internalize-sort 'n :: finite, OF - regular-graph-axioms,
    unfolded remove-finite-premise, cancel-type-definition, OF verts-non-empty]

theorem uniform-property:
  assumes  $i < l$ 
  shows  $\text{map-pmf } (\lambda w. w ! i) (\text{pmf-of-multiset } (\text{walks } G l)) = \text{pmf-of-set } (\text{verts } G)$  (is ?L = ?R)
  <proof>

lemma uniform-property-gen:
  fixes  $S :: 'a \text{ set}$ 
  assumes  $S \subseteq \text{verts } G$   $i < l$ 
  defines  $\mu \equiv \text{real } (\text{card } S) / \text{card } (\text{verts } G)$ 
  shows  $\text{measure } (\text{pmf-of-multiset } (\text{walks } G l)) \{w. w ! i \in S\} = \mu$  (is ?L = ?R)
  <proof>

theorem kl-chernoff-property:
  assumes  $l > 0$ 
  assumes  $S \subseteq \text{verts } G$ 
  defines  $\mu \equiv \text{real } (\text{card } S) / \text{card } (\text{verts } G)$ 
  assumes  $\gamma \leq 1$   $\mu + \Lambda_a * (1 - \mu) \in \{0 < .. \gamma\}$ 
  shows  $\text{measure } (\text{pmf-of-multiset } (\text{walks } G l)) \{w. \text{real } (\text{card } \{i \in \{..<l\}. w ! i \in S\}) \geq \gamma * l\}$ 
     $\leq \text{exp } (- \text{real } l * \text{KL-div } \gamma (\mu + \Lambda_a * (1 - \mu)))$  (is ?L  $\leq$  ?R)
  <proof>

end

unbundle no intro-cong-syntax

end

```

10 Graph Powers

```

theory Expander-Graphs-Power-Construction
imports
  Expander-Graphs-Walks
  Graph-Theory.Arc-Walk
begin

unbundle intro-cong-syntax

fun is-arc-walk :: ('a, 'b) pre-digraph  $\Rightarrow$  'a  $\Rightarrow$  'b list  $\Rightarrow$  bool
  where
    is-arc-walk G - [] = True |
    is-arc-walk G y (x#xs) = (is-arc-walk G (head G x) xs  $\wedge$  tail G x = y  $\wedge$  x  $\in$  arcs G)

definition arc-walk-head :: ('a, 'b) pre-digraph  $\Rightarrow$  ('a  $\times$  'b list)  $\Rightarrow$  'a
  where
    arc-walk-head G x = (if snd x = [] then fst x else head G (last (snd x)))

lemma is-arc-walk-snoc:
  is-arc-walk G y (xs@[x])  $\longleftrightarrow$  is-arc-walk G y xs  $\wedge$  x  $\in$  out-arcs G (arc-walk-head G (y,xs))
  <proof>

lemma is-arc-walk-set:

```

assumes *is-arc-walk* $G u w$
shows $set w \subseteq arcs G$
 $\langle proof \rangle$

lemma (**in** *wf-digraph*) *awalk-is-arc-walk*:
assumes $u \in verts G$
shows $is-arc-walk G u w \longleftrightarrow awalk u w (awlast u w)$
 $\langle proof \rangle$

definition *arc-walks* :: (*'a, 'b*) *pre-digraph* $\Rightarrow nat \Rightarrow ('a \times 'b list) set$
where
 $arc-walks G l = \{(u,w). u \in verts G \wedge is-arc-walk G u w \wedge length w = l\}$

lemma *arc-walks-len*:
assumes $x \in arc-walks G l$
shows $length (snd x) = l$
 $\langle proof \rangle$

lemma (**in** *wf-digraph*) *awhd-of-arc-walk*:
assumes $w \in arc-walks G l$
shows $awhd (fst w) (snd w) = fst w$
 $\langle proof \rangle$

lemma (**in** *wf-digraph*) *awlast-of-arc-walk*:
assumes $w \in arc-walks G l$
shows $awlast (fst w) (snd w) = arc-walk-head G w$
 $\langle proof \rangle$

lemma (**in** *wf-digraph*) *arc-walk-head-wellformed*:
assumes $w \in arc-walks G l$
shows $arc-walk-head G w \in verts G$
 $\langle proof \rangle$

lemma (**in** *wf-digraph*) *arc-walk-tail-wellformed*:
assumes $w \in arc-walks G l$
shows $fst w \in verts G$
 $\langle proof \rangle$

lemma (**in** *fin-digraph*) *arc-walks-fin*:
 $finite (arc-walks G l)$
 $\langle proof \rangle$

lemma (**in** *wf-digraph*) *awalk-verts-unfold*:
assumes $w \in arc-walks G l$
shows $awalk-verts (fst w) (snd w) = fst w \# map (head G) (snd w) (is ?L = ?R)$
 $\langle proof \rangle$

lemma (**in** *fin-digraph*) *arc-walks-map-walks'*:
 $walks' G l = image-mset (case-prod awalk-verts) (mset-set (arc-walks G l))$
 $\langle proof \rangle$

lemma (**in** *fin-digraph*) *arc-walks-map-walks*:
 $walks G (l+1) = image-mset (case-prod awalk-verts) (mset-set (arc-walks G l))$
 $\langle proof \rangle$

lemma (**in** *wf-digraph*)
assumes $awalk u a v \ length a = l l > 0$
shows $awalk-ends: tail G (hd a) = u \ head G (last a) = v$

<proof>

definition *graph-power* :: ('a, 'b) pre-digraph \Rightarrow nat \Rightarrow ('a, ('a \times 'b list)) pre-digraph
where *graph-power* G l =
(| *verts* = *verts* G, *arcs* = *arc-walks* G l, *tail* = *fst*, *head* = *arc-walk-head* G |)

lemma (in *wf-digraph*) *graph-power-wf*:
wf-digraph (*graph-power* G l)
<proof>

lemma (in *fin-digraph*) *graph-power-fin*:
fin-digraph (*graph-power* G l)
<proof>

lemma (in *fin-digraph*) *graph-power-count-edges*:
fixes l v w
defines S \equiv {x. length x=l+1 \wedge set x \subseteq verts G \wedge hd x=v \wedge last x=w}
shows count (*edges* (*graph-power* G l)) (v,w) = ($\sum x \in S. (\prod i < l. \text{count}(\text{edges } G)(x!i, x!(i+1)))$)
(is ?L = ?R)
<proof>

lemma (in *fin-digraph*) *graph-power-sym-aux*:
assumes *symmetric-multi-graph* G
assumes v \in *verts* (*graph-power* G l) w \in *verts* (*graph-power* G l)
shows card (*arcs-betw* (*graph-power* G l) v w) = card (*arcs-betw* (*graph-power* G l) w v)
(is ?L = ?R)
<proof>

lemma (in *fin-digraph*) *graph-power-sym*:
assumes *symmetric-multi-graph* G
shows *symmetric-multi-graph* (*graph-power* G l)
<proof>

lemma (in *fin-digraph*) *graph-power-out-degree'*:
assumes reg: $\bigwedge v. v \in \text{verts } G \implies \text{out-degree } G v = d$
assumes v \in *verts* (*graph-power* G l)
shows *out-degree* (*graph-power* G l) v = $d \wedge l$ (is ?L = ?R)
<proof>

lemma (in *regular-graph*) *graph-power-out-degree*:
assumes v \in *verts* (*graph-power* G l)
shows *out-degree* (*graph-power* G l) v = $d \wedge l$ (is ?L = ?R)
<proof>

lemma (in *regular-graph*) *graph-power-regular*:
regular-graph (*graph-power* G l)
<proof>

lemma (in *regular-graph*) *graph-power-degree*:
regular-graph.d (*graph-power* G l) = $d \wedge l$ (is ?L = ?R)
<proof>

lemma (in *regular-graph*) *graph-power-step*:
assumes x \in *verts* G
shows *regular-graph.g-step* (*graph-power* G l) f x = (*g-step* $\wedge l$) f x
<proof>

lemma (in *regular-graph*) *graph-power-expansion*:

regular-graph. Λ_a (graph-power G l) $\leq \Lambda_a \hat{\sim} l$
 ⟨proof⟩

unbundle *no intro-cong-syntax*

end

11 Strongly Explicit Expander Graphs

In some applications, representing an expander graph using a data structure (for example as an adjacency lists) would be prohibitive. For such cases strongly explicit expander graphs (SEE) are relevant. These are expander graphs, which can be represented implicitly using a function that computes for each vertex its neighbors in space and time logarithmic w.r.t. to the size of the graph. An application can for example sample a random walk, from a SEE using such a function efficiently. An example of such a graph is the Margulis construction from Section 8. This section presents the latter as a SEE but also shows that two graph operations that preserve the SEE property, in particular the graph power construction from Section 10 and a compression scheme introduced by Murtagh et al. [9, Theorem 20]. Combining all of the above it is possible to construct strongly explicit expander graphs of *every size* and spectral gap.

theory *Expander-Graphs-Strongly-Explicit*

imports

Expander-Graphs-Power-Construction

Expander-Graphs-MGG

Finite-Fields.Finite-Fields-Nth-Root-Code

begin

unbundle *intro-cong-syntax*

no-notation *Digraph.dominates* ($\leftarrow \rightarrow_1 \rightarrow$ [100,100] 40)

record *strongly-explicit-expander* =

see-size :: nat

see-degree :: nat

see-step :: nat \Rightarrow nat \Rightarrow nat

definition *graph-of* :: *strongly-explicit-expander* \Rightarrow (nat, (nat,nat) arc) *pre-digraph*

where *graph-of* e =

\langle *verts* = $\{..<see-size\ e\}$,

arcs = $(\lambda(v, i). \text{Arc } v \text{ (see-step } e \ i \ v) \ i) \text{ ' } (\{..<see-size\ e\} \times \{..<see-degree\ e\})$,

tail = *arc-tail*,

head = *arc-head* \rangle

definition *is-expander* $e \ \Lambda_a \longleftrightarrow$

regular-graph (*graph-of* e) \wedge *regular-graph*. Λ_a (*graph-of* e) $\leq \Lambda_a$

lemma *is-expander-mono*:

assumes *is-expander* $e \ a \ a \leq b$

shows *is-expander* $e \ b$

⟨proof⟩

lemma *graph-of-finI*:

assumes *see-step* $e \in (\{..<see-degree\ e\} \rightarrow (\{..<see-size\ e\} \rightarrow \{..<see-size\ e\}))$

shows *fin-digraph* (*graph-of* e)

⟨proof⟩

lemma *edges-graph-of*:

$edges(graph-of\ e) = \{\#(v, see-step\ e\ i\ v). (v, i) \in \#mset-set\ (\{..<see-size\ e\} \times \{..<see-degree\ e\})\#\}$
 $\langle proof \rangle$

lemma *out-degree-see*:

assumes $v \in verts\ (graph-of\ e)$
shows $out-degree\ (graph-of\ e)\ v = see-degree\ e\ (\mathbf{is}\ ?L = ?R)$
 $\langle proof \rangle$

lemma *card-arc-walks-see*:

assumes $fin-digraph\ (graph-of\ e)$
shows $card\ (arc-walks\ (graph-of\ e)\ n) = see-degree\ e^{\wedge}n * see-size\ e\ (\mathbf{is}\ ?L = ?R)$
 $\langle proof \rangle$

lemma *regular-graph-degree-eq-see-degree*:

assumes $regular-graph\ (graph-of\ e)$
shows $regular-graph.d\ (graph-of\ e) = see-degree\ e\ (\mathbf{is}\ ?L = ?R)$
 $\langle proof \rangle$

The following introduces the compression scheme, described in [9, Theorem 20].

fun *see-compress* :: $nat \Rightarrow strongly-explicit-expander \Rightarrow strongly-explicit-expander$

where $see-compress\ m\ e =$
 $\lfloor see-size = m, see-degree = see-degree\ e * 2$
 $, see-step = (\lambda k\ v.$
 $\quad if\ k < see-degree\ e$
 $\quad\quad then\ (see-step\ e\ k\ v)\ mod\ m$
 $\quad\quad else\ (if\ v+m < see-size\ e\ then\ (see-step\ e\ (k-see-degree\ e)\ (v+m))\ mod\ m\ else\ v) \rfloor$

lemma *edges-of-compress*:

fixes $e\ m$
assumes $2*m \geq see-size\ e\ m \leq see-size\ e$
defines $A \equiv \{\#(x\ mod\ m, y\ mod\ m). (x, y) \in \#edges\ (graph-of\ e)\#\}$
defines $B \equiv repeat-mset\ (see-degree\ e)\ \{\#(x, x). x \in \#(mset-set\ \{see-size\ e - m..<m\})\#\}$
shows $edges\ (graph-of\ (see-compress\ m\ e)) = A + B\ (\mathbf{is}\ ?L = ?R)$
 $\langle proof \rangle$

lemma *see-compress-sym*:

assumes $2*m \geq see-size\ e\ m \leq see-size\ e$
assumes $symmetric-multi-graph\ (graph-of\ e)$
shows $symmetric-multi-graph\ (graph-of\ (see-compress\ m\ e))$
 $\langle proof \rangle$

lemma *see-compress*:

assumes $is-expander\ e\ \Lambda_a$
assumes $2*m \geq see-size\ e\ m \leq see-size\ e$
shows $is-expander\ (see-compress\ m\ e)\ (\Lambda_a/2 + 1/2)$
 $\langle proof \rangle$

The graph power of a strongly explicit expander graph is itself a strongly explicit expander graph.

fun *to-digits* :: $nat \Rightarrow nat \Rightarrow nat \Rightarrow nat\ list$

where
 $to-digits\ -\ 0\ - = []\ |$
 $to-digits\ b\ (Suc\ l)\ k = (k\ mod\ b)\#\ to-digits\ b\ l\ (k\ div\ b)$

fun *from-digits* :: $nat \Rightarrow nat\ list \Rightarrow nat$

where
 $from-digits\ b\ [] = 0\ |$

$from-digits\ b\ (x\#\!xs) = x + b * from-digits\ b\ xs$

lemma *to-from-digits*:

assumes $length\ xs = n$ set $xs \subseteq \{..<b\}$

shows $to-digits\ b\ n\ (from-digits\ b\ xs) = xs$

<proof>

lemma *from-digits-range*:

assumes $length\ xs = n$ set $xs \subseteq \{..<b\}$

shows $from-digits\ b\ xs < b^n$

<proof>

lemma *from-digits-inj*:

inj-on $(from-digits\ b)\ \{xs.\ set\ xs \subseteq \{..<b\} \wedge length\ xs = n\}$

<proof>

fun *see-power* :: $nat \Rightarrow strongly-explicit-expander \Rightarrow strongly-explicit-expander$

where *see-power* $l\ e =$

$(\ | see-size = see-size\ e, see-degree = see-degree\ e^l$

$, see-step = (\lambda k\ v.\ foldl\ (\lambda y\ x.\ see-step\ e\ x\ y)\ v\ (to-digits\ (see-degree\ e)\ l\ k))\)$

lemma *graph-power-iso-see-power*:

assumes *fin-digraph* $(graph-of\ e)$

shows *digraph-iso* $(graph-power\ (graph-of\ e)\ n)\ (graph-of\ (see-power\ n\ e))$

<proof>

lemma *see-power*:

assumes *is-expander* $e\ \Lambda_a$

shows *is-expander* $(see-power\ n\ e)\ (\Lambda_a\ \hat{\ }n)$

<proof>

The Margulis Construction from Section 8 is a strongly explicit expander graph.

definition *mgg-vert* :: $nat \Rightarrow nat \Rightarrow (int \times int)$

where $mgg-vert\ n\ x = (x\ mod\ n, x\ div\ n)$

definition *mgg-vert-inv* :: $nat \Rightarrow (int \times int) \Rightarrow nat$

where $mgg-vert-inv\ n\ x = nat\ (fst\ x) + nat\ (snd\ x) * n$

lemma *mgg-vert-inv*:

assumes $n > 0$ $x \in \{0..<int\ n\} \times \{0..<int\ n\}$

shows $mgg-vert\ n\ (mgg-vert-inv\ n\ x) = x$

<proof>

definition *mgg-arc* :: $nat \Rightarrow (nat \times int)$

where $mgg-arc\ k = (k\ mod\ 4, if\ k \geq 4\ then\ (-1)\ else\ 1)$

definition *mgg-arc-inv* :: $(nat \times int) \Rightarrow nat$

where $mgg-arc-inv\ x = (nat\ (fst\ x) + 4 * of-bool\ (snd\ x < 0))$

lemma *mgg-arc-inv*:

assumes $x \in \{..<4\} \times \{-1, 1\}$

shows $mgg-arc\ (mgg-arc-inv\ x) = x$

<proof>

definition *see-mgg* :: $nat \Rightarrow strongly-explicit-expander$ **where**

$see-mgg\ n = (\ | see-size = n^2, see-degree = 8,$

$see-step = (\lambda i\ v.\ mgg-vert-inv\ n\ (mgg-graph-step\ n\ (mgg-vert\ n\ v)\ (mgg-arc\ i)))\)$

lemma *mgg-graph-iso*:
assumes $n > 0$
shows *digraph-iso* (*mgg-graph* n) (*graph-of* (*see-mgg* n))
 \langle *proof* \rangle

lemma *see-mgg*:
assumes $n > 0$
shows *is-expander* (*see-mgg* n) ($5 * \text{sqrt } 2 / 8$)
 \langle *proof* \rangle

Using all of the above it is possible to construct strongly explicit expanders of every size and spectral gap with asymptotically optimal degree.

definition *see-standard-aux*
where *see-standard-aux* $n = \text{see-compress } n (\text{see-mgg } (\text{nat } \lceil \text{sqrt } n \rceil))$

lemma *see-standard-aux*:
assumes $n > 0$
shows
is-expander (*see-standard-aux* n) ($(8+5 * \text{sqrt } 2) / 16$) (**is** ?A)
see-degree (*see-standard-aux* n) = 16 (**is** ?B)
see-size (*see-standard-aux* n) = n (**is** ?C)
 \langle *proof* \rangle

lemma *nth-root-nat-eq*:
assumes $k > 0$
shows *nth-root-nat* $k n = \text{nat } \lfloor \text{root } k n \rfloor$
 \langle *proof* \rangle

lemma *nat-sqrt-code*: *nth-root-nat* 2 $n + \text{of-bool } (\neg(\text{is-nth-power-nat } 2 n)) = \text{nat } \lceil \text{sqrt } n \rceil$
(**is** ?L = ?R)
 \langle *proof* \rangle

lemma *see-standard-aux-code* [code]:
see-standard-aux $n = \text{see-compress } n (\text{see-mgg } (\text{nth-root-nat } 2 n + \text{of-bool } (\neg \text{is-nth-power-nat } 2 n)))$
 \langle *proof* \rangle

definition *see-standard-power*
where *see-standard-power* $x = (\text{if } x \leq (0::\text{real}) \text{ then } 0 \text{ else } \text{nat } \lceil \ln x / \ln 0.95 \rceil)$

lemma *see-standard-power*:
assumes $\Lambda_a > 0$
shows $0.95^{\text{see-standard-power } \Lambda_a} \leq \Lambda_a$ (**is** ?L \leq ?R)
 \langle *proof* \rangle

lemma *see-standard-power-eval*[code]:
see-standard-power $x = (\text{if } x \leq 0 \vee x \geq 1 \text{ then } 0 \text{ else } (1 + \text{see-standard-power } (x/0.95)))$
 \langle *proof* \rangle

definition *see-standard* :: $\text{nat} \Rightarrow \text{real} \Rightarrow \text{strongly-explicit-expander}$
where *see-standard* $n \Lambda_a = \text{see-power } (\text{see-standard-power } \Lambda_a) (\text{see-standard-aux } n)$

theorem *see-standard*:
assumes $n > 0 \ \Lambda_a > 0$
shows *is-expander* (*see-standard* $n \Lambda_a$) Λ_a
and *see-size* (*see-standard* $n \Lambda_a$) = n
and *see-degree* (*see-standard* $n \Lambda_a$) = $16^{\wedge} (\text{nat } \lceil \ln \Lambda_a / \ln 0.95 \rceil)$ (**is** ?C)
 \langle *proof* \rangle

```

fun see-sample-walk :: strongly-explicit-expander  $\Rightarrow$  nat  $\Rightarrow$  nat  $\Rightarrow$  nat list
where
  see-sample-walk e 0 x = [x] |
  see-sample-walk e (Suc l) x = (let w = see-sample-walk e l (x div (see-degree e)) in
    w@[see-step e (x mod (see-degree e)) (last w)])

```

```

theorem see-sample-walk:
fixes e l
assumes fin-digraph (graph-of e)
defines r  $\equiv$  see-size e * see-degree e  $\wedge$  l
shows {# see-sample-walk e l k. k  $\in$  # mset-set {.. $r$ } #} = walks' (graph-of e) l
<proof>

```

```

unbundle no intro-cong-syntax

```

```

end

```

12 Expander Walks as Pseudorandom Objects

```

theory Pseudorandom-Objects-Expander-Walks
imports
  Universal-Hash-Families.Pseudorandom-Objects
  Expander-Graphs.Expander-Graphs-Strongly-Explicit
begin

```

```

unbundle intro-cong-syntax
hide-const (open) Quantum.T
hide-fact (open) SN-Orders.of-nat-mono

```

```

definition expander-pro ::
  nat  $\Rightarrow$  real  $\Rightarrow$  ('a,'b) pseudorandom-object-scheme  $\Rightarrow$  (nat  $\Rightarrow$  'a) pseudorandom-object
where expander-pro l  $\wedge$  S = (
  let e = see-standard (pro-size S)  $\wedge$  in
    ( pro-last = see-size e * see-degree e  $\wedge$  (l-1) - 1,
      pro-select = ( $\lambda$  i j. pro-select S (see-sample-walk e (l-1) i ! j mod pro-size S)) )
  )

```

```

open-bundle expander-pseudorandom-object-syntax
begin
notation expander-pro (<mathcal{E}>)
end

```

```

unbundle expander-pseudorandom-object-syntax

```

```

context
fixes l :: nat
fixes  $\wedge$  :: real
fixes P :: 'a pseudorandom-object
assumes l-gt-0: l > 0
assumes  $\wedge$ -gt-0:  $\wedge$  > 0
begin

```

```

private definition e where e = see-standard (pro-size P)  $\wedge$ 

```

```

private lemma expander-pro-alt:
  expander-pro l  $\wedge$  P =

```

$\langle \text{proof} \rangle$ $\text{pro-last} = \text{see-size } e * \text{see-degree } e^{(l-1)} - 1,$
 $\text{pro-select} = (\lambda i j. \text{pro-select } P (\text{see-sample-walk } e (l-1) i ! j \text{ mod } \text{pro-size } P)) \rangle$
 $\langle \text{proof} \rangle$ **lemmas** $\text{see-standard} = \text{see-standard} [\text{OF } \text{pro-size-gt-0}[\text{where } S=P] \ \Lambda\text{-gt-0}]$

interpretation E : *regular-graph graph-of e*

$\langle \text{proof} \rangle$ **lemma** $e\text{-deg-gt-0}$: *see-degree e > 0*

$\langle \text{proof} \rangle$ **lemma** $e\text{-size-gt-0}$: *see-size e > 0*

$\langle \text{proof} \rangle$ **lemma** $\text{expander-sample-size}$: $\text{pro-size } (\mathcal{E} \ l \ \Lambda \ P) = \text{see-size } e * \text{see-degree } e^{(l-1)}$

$\langle \text{proof} \rangle$ **lemma** $\text{sample-pro-expander-walks}$:

defines $R \equiv \text{map-pmf } (\lambda xs \ i. \text{pro-select } P (xs ! i \text{ mod } \text{pro-size } P))$

$(\text{pmf-of-multiset } (\text{walks } (\text{graph-of } e) \ l))$

shows $\text{sample-pro } (\text{expander-pro } l \ \Lambda \ P) = R$

$\langle \text{proof} \rangle$

lemma $\text{expander-pro-range}$: $\text{pro-select } (\text{expander-pro } l \ \Lambda \ P) \ i \ j \in \text{pro-set } P$

$\langle \text{proof} \rangle$

lemma $\text{expander-uniform-property}$:

assumes $i < l$

shows $\text{map-pmf } (\lambda w. \ w \ i) (\text{sample-pro } (\text{expander-pro } l \ \Lambda \ P)) = \text{sample-pro } P \ (\text{is } ?L = ?R)$

$\langle \text{proof} \rangle$

lemma $\text{expander-kl-chernoff-bound}$:

assumes $\text{measure } (\text{sample-pro } P) \ \{w. \ T \ w\} \leq \mu$

assumes $c \leq 1 \ \mu + \Lambda * (1-\mu) \leq c \ \mu \leq 1$

shows $\text{measure } (\text{sample-pro } (\text{expander-pro } l \ \Lambda \ P)) \ \{w. \ \text{real } (\text{card } \{i \in \{..<l\}. \ T \ (w \ i)\}) \geq c * l\}$
 $\leq \text{exp } (- \text{real } l * \text{KL-div } c \ (\mu + \Lambda * (1-\mu))) \ (\text{is } ?L \leq ?R)$

$\langle \text{proof} \rangle$

lemma expander-pro-size :

$\text{pro-size } (\text{expander-pro } l \ \Lambda \ P) = \text{pro-size } P * (16 \wedge ((l-1) * \text{nat } \lceil \ln \ \Lambda / \ln \ (19 / 20) \rceil))$

$(\text{is } ?L = ?R)$

$\langle \text{proof} \rangle$

context

fixes $\gamma :: \text{real}$

assumes $\gamma\text{-ge-0}$: $\gamma \geq 0$

begin

lemma $\text{expander-chernoff-bound-one-sided}$:

assumes $AE \ x \ \text{in } \text{sample-pro } P. \ f \ x \in \{0,1::\text{real}\}$

assumes $(\int x. \ f \ x \ \partial \text{sample-pro } P) \leq \mu$

shows $\text{measure } (\text{expander-pro } l \ \Lambda \ P) \ \{w. \ (\sum i < l. \ f \ (w \ i)) / l - \mu \geq \gamma + \Lambda\} \leq \text{exp } (- \ 2 * \text{real } l * \gamma^2)$

$(\text{is } ?L \leq ?R)$

$\langle \text{proof} \rangle$

lemma $\text{expander-chernoff-bound-1}$:

assumes $AE \ x \ \text{in } \text{sample-pro } P. \ f \ x \in \{0,1::\text{real}\}$

defines $\mu \equiv (\int x. \ f \ x \ \partial \text{sample-pro } P)$

shows $\text{measure } (\mathcal{E} \ l \ \Lambda \ P) \ \{w. \ |(\sum i < l. \ f \ (w \ i)) / l - \mu| \geq \gamma + \Lambda\} \leq 2 * \text{exp } (- \ 2 * \text{real } l * \gamma^2)$

$(\text{is } ?L \leq ?R)$

$\langle \text{proof} \rangle$

lemma $\text{expander-chernoff-bound}$:

assumes $AE \ x \ \text{in } \text{sample-pro } P. \ f \ x \in \{0,1::\text{real}\}$

defines $\mu \equiv \text{measure-pmf.expectation } P \ f$

shows $\mathcal{P}(\omega \ \text{in } \mathcal{E} \ l \ \Lambda \ P. \ |(\sum i < l. \ f \ (\omega \ i)) / l - \mu| \geq \gamma + \Lambda) \leq 2 * \text{exp } (- \ 2 * \text{real } l * \gamma^2)$

(is ?L ≤ ?R)
<proof>

lemma *expander-chernoff-bound-2*:

assumes *AE x in sample-pro P. f x ∈ {0,1::real}*

defines $\mu \equiv \text{measure-pmf.expectation } P f$

shows $\mathcal{P}(\omega \text{ in sample-pro } (\mathcal{E} \ l \ \Lambda \ P). |(\sum_{i < l} f(\omega \ i)) - \text{real } l * \mu| \geq \text{real } l * (\gamma + \Lambda)) \leq 2 * \exp(-2 * \text{real } l * \gamma^2)$

(is ?L ≤ ?R)

<proof>

end

end

unbundle *no intro-cong-syntax*

end

References

- [1] J. Divasón, O. Kunar, R. Thiemann, and A. Yamada. Perron-frobenius theorem for spectral radius analysis. *Archive of Formal Proofs*, May 2016. https://isa-afp.org/entries/Perron_Frobenius.html, Formal proof development.
- [2] M. Echenim. Simultaneous diagonalization of pairwise commuting hermitian matrices. *Archive of Formal Proofs*, July 2022. https://isa-afp.org/entries/Commuting_Hermitian.html, Formal proof development.
- [3] O. Gabber and Z. Galil. Explicit constructions of linear-sized superconcentrators. *Journal of Computer and System Sciences*, 22(3):407–420, 1981.
- [4] S. Hoory and N. Linial. Expander graphs and their applications. *Bulletin of the American Mathematical Society*, 43:439–561, 2006.
- [5] R. Impagliazzo and V. Kabanets. Constructive proofs of concentration bounds. In M. Serna, R. Shaltiel, K. Jansen, and J. Rolim, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, pages 617–631, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.
- [6] S. Jimbo and A. Maruoka. Expanders obtained from affine transformations. *Combinatorica*, 7(4), 1987.
- [7] O. Kuncar and A. Popescu. From types to sets by local type definition in higher-order logic. *Journal of Automated Reasoning*, 62:237 – 260, 2016.
- [8] G. A. Margulis. Explicit construction of a concentrator. *Probl. Peredachi Inf.*, 9(4):71–80, 1973.
- [9] J. Murtagh, O. Reingold, A. Sidford, and S. Vadhan. Deterministic Approximation of Random Walks in Small Space. In D. Achlioptas and L. A. Végh, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2019)*, volume 145 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 42:1–42:22, Dagstuhl, Germany, 2019. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.
- [10] L. Noschinski. Graph theory. *Archive of Formal Proofs*, April 2013. https://isa-afp.org/entries/Graph_Theory.html, Formal proof development.
- [11] S. P. Vadhan. Pseudorandomness. *Foundations and Trends(R) in Theoretical Computer Science*, 7(13):1–336, 2012.