Hypergraph Basics

Chelsea Edmonds and Lawrence C. Paulson

March 17, 2025

Abstract

This entry is a simple extension of our previous entry for Combinatorial design theory [1], which presents new and existing concepts using hypergraph language. Both designs and hypergraphs are types of incident set systems, hence have the same underlying foundation. However, they are often used in different contexts, and some definitions are as such unique. This library uses locales to rewrite equivalent definitions and build a basic hypergraph hierarchy with direct links to equivalent design theory concepts to avoid repetition, further demonstrating the power of the "locale-centric" approach. The library includes all standard definitions (order, degree etc.), as well as some extensions on hypergraph decompositions and spanning subhypergraphs.

Contents

1 Basic Hypergraphs

Converting Design theory to hypergraph notation. Hypergraphs have technically already been formalised

theory *Hypergraph*
  imports
    Design-Theory.Block-Designs
    Design-Theory.Sub-Designs
    Fishers-Inequality.Design-Extras
begin

lemma *is-singleton-image*:
is-singleton $C \Longrightarrow$ *is-singleton* $(f$ ‘ $C)$
by (*metis image-empty image-insert is-singletonE is-singletonI*)

lemma *bij-betw-singleton-image*:
assumes *bij-betw f A B*
assumes $C \subseteq A$
shows *is-singleton* $C \longleftrightarrow$ *is-singleton* $(f$ ‘ $C)$
proof (*intro iffI*)
show *is-singleton* $C \Longrightarrow$ *is-singleton* $(f$ ‘ $C)$ **by** (*rule is-singleton-image*)
show *is-singleton* $(f$ ‘ $C) \Longrightarrow$ *is-singleton* C **using** *assms is-singleton-image*
by (*metis bij-betw-def inv-into-image-cancel*)
qed

lemma *image-singleton*:
assumes $A \neq \{\}$
assumes $\bigwedge x.\ x \in A \Longrightarrow f\ x = c$
shows f ‘ $A = \{c\}$
using *assms*(*1*) *assms*(*2*) **by** *blast*

type-synonym *colour* = *nat*

type-synonym *′a hyp-edge* = *′a set*

type-synonym *′a hyp-graph* = (*′a set*) × (*′a hyp-edge multiset*)

abbreviation *hyp-edges* :: *′a hyp-graph* ⇒ *′a hyp-edge multiset* **where**
hyp-edges H ≡ *snd H*

abbreviation *hyp-verts* :: *′a hyp-graph* ⇒ *′a set* **where**
hyp-verts H ≡ *fst H*

locale *hypersystem* = *incidence-system vertices* :: *′a set edges* :: *′a hyp-edge multiset*

for *vertices* (‹$\mathcal{V}$›) **and** *edges* (‹E›)

begin

Basic definitions using hypergraph language

abbreviation *horder* :: *nat* **where**
horder ≡ *card* ($\mathcal{V}$)

definition *hdegree* :: *′a* ⇒ *nat* **where**
hdegree v ≡ *size* {#*e* ∈# *E* . *v* ∈ *e* #}

lemma *hdegree-rep-num*: *hdegree v* = *point-replication-number E v*
unfolding *hdegree-def point-replication-number-def* **by** *simp*

definition *hdegree-set* :: $'a$ *set* ⇒ *nat* **where**
hdegree-set vs ≡ *size* {#*e* ∈# *E*. *vs* ⊆ *e*#}

lemma *hdegree-set-points-index*: *hdegree-set vs* = *points-index E vs*
unfolding *hdegree-set-def points-index-def* **by** *simp*

definition *hvert-adjacent* :: $'a$ ⇒ $'a$ ⇒ *bool* **where**
hvert-adjacent v1 v2 ≡ ∃ *e* . *e* ∈# *E* ∧ *v1* ∈ *e* ∧ *v2* ∈ *e* ∧ *v1* ∈ $\mathcal{V}$ ∧ *v2* ∈ $\mathcal{V}$

definition *hedge-adjacent* :: $'a$ *hyp-edge* ⇒ $'a$ *hyp-edge* ⇒ *bool* **where**
hedge-adjacent e1 e2 ≡ *e1* ∩ *e2* ≠ {} ∧ *e1* ∈# *E* ∧ *e2* ∈# *E*

lemma *edge-adjacent-alt-def*: *e1* ∈# *E* ⟹ *e2* ∈# *E* ⟹ ∃ *x* . *x* ∈ $\mathcal{V}$ ∧ *x* ∈ *e1* ∧ *x* ∈ *e2* ⟹
hedge-adjacent e1 e2
unfolding *hedge-adjacent-def* **by** *auto*

definition *hneighborhood* :: $'a$ ⇒ $'a$ *set* **where**
hneighborhood x ≡ {*v* ∈ $\mathcal{V}$. *hvert-adjacent x v*}

definition *hmax-degree* :: *nat* **where**
hmax-degree ≡ *Max* {*hdegree v* | *v*. *v* ∈ $\mathcal{V}$}

definition *hrank* :: *nat* **where**
hrank ≡ *Max* {*card e* | *e* . *e* ∈# *E*}

definition *hcorank* :: *nat* **where**
hcorank = *Min* {*card e* | *e* . *e* ∈# *E*}

definition *hedge-neighbourhood* :: $'a$ ⇒ $'a$ *hyp-edge multiset* **where**
hedge-neighbourhood x ≡ {# *e* ∈# *E* . *x* ∈ *e* #}

lemma *degree-alt-neigbourhood*: *hdegree x* = *size* (*hedge-neighbourhood x*)
using *hedge-neighbourhood-def* **by** (*simp add*: *hdegree-def*)

definition *hinduced-edges*:: $'a$ *set* ⇒ $'a$ *hyp-edge multiset* **where**
hinduced-edges V' = {#*e* ∈# *E* . *e* ⊆ V'#}

end

Sublocale for rewriting definition purposes rather than inheritance

sublocale *hypersystem* ⊆ *incidence-system* $\mathcal{V}$ *E*
rewrites *point-replication-number E v* = *hdegree v* **and** *points-index E vs* = *hdegree-set vs*
by (*unfold-locales*) (*simp-all add*: *hdegree-rep-num hdegree-set-points-index*)

Reverse sublocale to establish equality

sublocale *incidence-system* ⊆ *hypersystem* $\mathcal{V}$ $\mathcal{B}$
rewrites *hdegree v* = *point-replication-number* $\mathcal{B}$ *v* **and** *hdegree-set vs* = *points-index* $\mathcal{B}$ *vs*

proof (*unfold-locales*)
interpret *hs*: *hypersystem* $\mathcal{V}$ $\mathcal{B}$ **by** (*unfold-locales*)
show *hs.hdegree v* = $\mathcal{B}$ *rep v* **using** *hs.hdegree-rep-num* **by** *simp*
show *hs.hdegree-set vs* = $\mathcal{B}$ *index vs* **using** *hs.hdegree-set-points-index* **by** *simp*
qed

Missing design identified in the design theory hierarchy

locale *inf-design* = *incidence-system* +
assumes *blocks-nempty*: $bl \in\# \mathcal{B} \Longrightarrow bl \neq \{\}$

sublocale *design* ⊆ *inf-design*
by *unfold-locales* (*simp add*: *blocks-nempty*)

locale *fin-hypersystem* = *hypersystem* + *finite-incidence-system* $\mathcal{V}$ *E*

sublocale *finite-incidence-system* ⊆ *fin-hypersystem* $\mathcal{V}$ $\mathcal{B}$
by *unfold-locales*

locale *hypergraph* = *hypersystem* + *inf-design* $\mathcal{V}$ *E*

sublocale *inf-design* ⊆ *hypergraph* $\mathcal{V}$ $\mathcal{B}$
by *unfold-locales* (*simp add*: *wellformed*)

locale *fin-hypergraph* = *hypergraph* + *fin-hypersystem*

sublocale *design* ⊆ *fin-hypergraph* $\mathcal{V}$ $\mathcal{B}$
by *unfold-locales*

sublocale *fin-hypergraph* ⊆ *design* $\mathcal{V}$ *E*
using *blocks-nempty* **by** (*unfold-locales*) *simp*

1.1 Sub hypergraphs

Sub hypergraphs and related concepts (spanning hypergraphs etc)

locale *sub-hypergraph* = *sub*: *hypergraph* $\mathcal{V}H$ *EH* + *orig*: *hypergraph* $\mathcal{V}$ :: *'a set E*
+
sub-set-system $\mathcal{V}H$ *EH* $\mathcal{V}$ *E* **for** $\mathcal{V}H$ *EH* $\mathcal{V}$ *E*

locale *spanning-hypergraph* = *sub-hypergraph* +
assumes $\mathcal{V} = \mathcal{V}H$

lemma *spanning-hypergraphI*: *sub-hypergraph VH EH V E* ⟹ *V* = *VH* ⟹ *spanning-hypergraph VH EH V E*
using *spanning-hypergraph-def spanning-hypergraph-axioms-def* **by** *blast*

context *hypergraph*
begin

definition *is-subhypergraph* :: *'a hyp-graph* ⇒ *bool* **where**

is-subhypergraph H ≡ *sub-hypergraph* (*hyp-verts H*) (*hyp-edges H*) $\mathcal{V}$ *E*

lemma *is-subhypergraphI*:
  assumes (*hyp-verts H* ⊆ $\mathcal{V}$)
  assumes (*hyp-edges H* ⊆# *E*)
  assumes *hypergraph* (*hyp-verts H*) (*hyp-edges H*)
  shows *is-subhypergraph H*
  unfolding *is-subhypergraph-def*
proof −
  interpret *h*: *hypergraph hyp-verts H hyp-edges H*
    using *assms*(*3*) **by** *simp*
  show *sub-hypergraph* (*hyp-verts H*) (*hyp-edges H*) $\mathcal{V}$ *E*
    by (*unfold-locales*) (*simp-all add*: *assms*)
qed

definition *hypergraph-decomposition* :: *'a hyp-graph multiset* ⇒ *bool* **where**
hypergraph-decomposition S ≡ (∀ *h* ∈# *S* . *is-subhypergraph h*) ∧
  partition-on-mset E {#*hyp-edges h* . *h* ∈# *S*#}

definition *is-spanning-subhypergraph* :: *'a hyp-graph* ⇒ *bool* **where**
is-spanning-subhypergraph H ≡ *spanning-hypergraph* (*hyp-verts H*) (*hyp-edges H*)
$\mathcal{V}$ *E*

lemma *is-spanning-subhypergraphI*: *is-subhypergraph H* ⟹ (*hyp-verts H*) = $\mathcal{V}$
⟹
    is-spanning-subhypergraph H
 unfolding *is-subhypergraph-def is-spanning-subhypergraph-def* **using** *spanning-hypergraphI*
by *blast*

lemma *spanning-subhypergraphI*: (*hyp-verts H*) = $\mathcal{V}$ ⟹ (*hyp-edges H*) ⊆# *E*
⟹
  hypergraph (*hyp-verts H*) (*hyp-edges H*) ⟹ *is-spanning-subhypergraph H*
  using *is-spanning-subhypergraphI* **by** (*simp add*: *is-subhypergraphI*)

end
end

2 Hypergraph Variations

This section presents many different types of hypergraphs, introducing conditions such as non-triviality, regularity, and uniform. Additionally, it briefly formalises decompositions

theory *Hypergraph-Variations*
  imports
    Hypergraph
    Undirected-Graph-Theory.Bipartite-Graphs
begin

2.1 Non-trivial hypergraphs

Non empty (ne) implies that the vertex (and edge) set is not empty. Non trivial typically requires at least two edges

locale *hyper-system-vne* = *hypersystem* +
 assumes *V-nempty*: $\mathcal{V} \neq \{\}$

locale *hyper-system-ne* = *hyper-system-vne* +
 assumes *E-nempty*: $E \neq \{\#\}$

locale *hypergraph-ne* = *hypergraph* +
 assumes *E-nempty*: $E \neq \{\#\}$
begin

lemma *V-nempty*: $\mathcal{V} \neq \{\}$
 using *wellformed E-nempty blocks-nempty* **by** *fastforce*

lemma *sizeE-not-zero*: *size E* $\neq$ *0*
 using *E-nempty* **by** *auto*

end

sublocale *hypergraph-ne* $\subseteq$ *hyper-system-ne*
 by (*unfold-locales*) (*simp-all add*: *V-nempty E-nempty*)

locale *hyper-system-ns* = *hypersystem* +
 assumes *V-not-single*: $\neg$ *is-singleton* $\mathcal{V}$

locale *hypersystem-nt* = *hyper-system-ne* + *hyper-system-ns*

locale *hypergraph-nt* = *hypergraph-ne* + *hyper-system-ns*

sublocale *hypergraph-nt* $\subseteq$ *hypersystem-nt*
 by (*unfold-locales*)

locale *fin-hypersystem-vne* = *fin-hypersystem* + *hyper-system-vne*
begin

lemma *order-gt-zero*: *horder* > *0*
 using *V-nempty finite-sets* **by** *auto*

lemma *order-ge-one*: *horder* $\geq$ *1*
 using *order-gt-zero* **by** *auto*

end

locale *fin-hypersystem-nt* = *fin-hypersystem-vne* + *hypersystem-nt*
begin

lemma *order-gt-one*: *horder* > *1*
  using *V-nempty V-not-single*
  by (*simp add*: *finite-sets is-singleton-altdef nat-neq-iff*)

lemma *order-ge-two*: *horder* ≥ *2*
  using *order-gt-one* **by** *auto*

end

locale *fin-hypergraph-ne* = *fin-hypergraph* + *hypergraph-ne*

sublocale *fin-hypergraph-ne* ⊆ *fin-hypersystem-vne*
  by *unfold-locales*

locale *fin-hypergraph-nt* = *fin-hypergraph* + *hypergraph-nt*

sublocale *fin-hypergraph-nt* ⊆ *fin-hypersystem-nt*
  by (*unfold-locales*)

sublocale *fin-hypergraph-ne* ⊆ *proper-design* $\mathcal{V}$ *E*
  using *blocks-nempty sizeE-not-zero* **by** *unfold-locales simp*

sublocale *proper-design* ⊆ *fin-hypergraph-ne* $\mathcal{V}$ $\mathcal{B}$
  using *blocks-nempty design-blocks-nempty* **by** *unfold-locales simp*

2.2 Regular and Uniform Hypergraphs

locale *dregular-hypergraph* = *hypergraph* +
  fixes *d*
  assumes *const-degree*: $\bigwedge$ *x*. *x* ∈ $\mathcal{V}$ ⟹ *hdegree x* = *d*

locale *fin-dregular-hypergraph* = *dregular-hypergraph* + *fin-hypergraph*

locale *kuniform-hypergraph* = *hypergraph* +
  fixes *k* :: *nat*
  assumes *uniform*: $\bigwedge$ *e* . *e* ∈# *E* ⟹ *card e* = *k*

locale *fin-kuniform-hypergraph* = *kuniform-hypergraph* + *fin-hypergraph*

locale *almost-regular-hypergraph* = *hypergraph* +
  assumes $\bigwedge$ *x y* . *x* ∈ $\mathcal{V}$ ⟹ *y* ∈ $\mathcal{V}$ ⟹ | *hdegree x* − *hdegree y* | ≤ *1*

locale *kuniform-regular-hypgraph* = *kuniform-hypergraph* $\mathcal{V}$ *E k* + *dregular-hypergraph* $\mathcal{V}$ *E k*
  for $\mathcal{V}$ *E k*

locale *fin-kuniform-regular-hypgraph-nt* = *kuniform-regular-hypgraph* $\mathcal{V}$ *E k* + *fin-hypergraph-nt*

𝒱 *E*
for 𝒱 *E k*

sublocale *fin-kuniform-regular-hypgraph-nt* ⊆ *fin-kuniform-hypergraph* 𝒱 *E k*
by *unfold-locales*

sublocale *fin-kuniform-regular-hypgraph-nt* ⊆ *fin-dregular-hypergraph* 𝒱 *E k*
by *unfold-locales*

locale *block-balanced-design* = *block-design* + *t-wise-balance*

locale *regular-block-design* = *block-design* + *constant-rep-design*

sublocale *t-design* ⊆ *block-balanced-design*
by *unfold-locales*

locale *fin-kuniform-hypergraph-nt* = *fin-kuniform-hypergraph* + *fin-hypergraph-nt*

sublocale *fin-kuniform-regular-hypgraph-nt* ⊆ *fin-kuniform-hypergraph-nt* 𝒱 *E k*
by *unfold-locales*

Note that block designs are defined as non-trivial and finite as they automatically build on the proper design locale

sublocale *fin-kuniform-hypergraph-nt* ⊆ *block-design* 𝒱 *E k*
rewrites *point-replication-number E v* = *hdegree v* **and** *points-index E vs* = *hdegree-set vs*
using *uniform* **by** (*unfold-locales*)
(*simp-all add*: *point-replication-number-def hdegree-def hdegree-set-def points-index-def E-nempty*)

sublocale *fin-kuniform-regular-hypgraph-nt* ⊆ *regular-block-design* 𝒱 *E k k*
rewrites *point-replication-number E v* =*hdegree v* **and** *points-index E vs* = *hdegree-set vs*
using *const-degree* **by** (*unfold-locales*)
(*simp-all add*: *point-replication-number-def hdegree-def hdegree-set-def points-index-def*)

2.3 Factorisations

locale *d-factor* = *spanning-hypergraph* + *dregular-hypergraph* 𝒱*H EH d* **for** *d*

context *hypergraph*
begin

definition *is-d-factor* :: *'a hyp-graph* ⇒ *bool* **where**
is-d-factor H ≡ (∃ *d*. *d-factor* (*hyp-verts H*) (*hyp-edges H*) 𝒱 *E d*)

definition *d-factorisation* :: *'a hyp-graph multiset* ⇒ *bool* **where**
d-factorisation S ≡ *hypergraph-decomposition S* ∧ (∀ *h* ∈# *S*. *is-d-factor h*)
end

2.4 Sample Graph Theory Connections

sublocale *fin-graph-system* $\subseteq$ *fin-hypersystem V mset-set E*
  rewrites *hedge-adjacent* = *edge-adj*
proof (*unfold-locales*)
  show $\bigwedge b.\ b \in\# \ mset\text{-}set\ E \Longrightarrow b \subseteq V$ **using** *wellformed fin-edges* **by** *simp*
  then interpret *hs*: *hypersystem V mset-set E*
    by *unfold-locales* (*simp add*: *fin-edges*)
  show *hs.hedge-adjacent* = *edge-adj*
    unfolding *hs.hedge-adjacent-def edge-adj-def*
    by (*simp add*: *fin-edges*)
qed(*simp add*: *finV*)

sublocale *fin-bipartite-graph* $\subseteq$ *fin-hypersystem-vne V mset-set E*
  using *X-not-empty Y-not-empty partitions-ss*(*2*) **by** *unfold-locales* (*auto*)

end
theory *Hypergraph-Basics-Root*
  imports
    Hypergraph
    Hypergraph-Variations
begin
end

References

[1] C. Edmonds and L. C. Paulson. Combinatorial design theory. *Archive of Formal Proofs*, August 2021. https://isa-afp.org/entries/Design_Theory.html, Formal proof development.