Hypergraph Basics

Chelsea Edmonds and Lawrence C. Paulson

June 20, 2024

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* $(f \text{ ` } C)$
  ⟨*proof*⟩

lemma *bij-betw-singleton-image*:
  assumes *bij-betw* $f\ A\ B$
  assumes $C \subseteq A$
  shows *is-singleton* $C \longleftrightarrow$ *is-singleton* $(f \text{ ` } C)$
⟨*proof*⟩

lemma *image-singleton*:
  assumes $A \neq \{\}$
  assumes $\bigwedge x.\ x \in A \Longrightarrow f\ x = c$
  shows $f \text{ ` } A = \{c\}$
  ⟨*proof*⟩

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*
  ⟨*proof*⟩

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*

$\langle proof \rangle$

definition *hvert-adjacent* :: $'a \Rightarrow 'a \Rightarrow bool$ **where**
$hvert\text{-}adjacent\ v1\ v2 \equiv \exists\ e\ .\ e \in\#\ E \land v1 \in e \land v2 \in e \land v1 \in \mathcal{V} \land v2 \in \mathcal{V}$

definition *hedge-adjacent* :: $'a\ hyp\text{-}edge \Rightarrow 'a\ hyp\text{-}edge \Rightarrow bool$ **where**
$hedge\text{-}adjacent\ e1\ e2 \equiv e1 \cap e2 \neq \{\} \land e1 \in\#\ E \land e2 \in\#\ E$

lemma *edge-adjacent-alt-def*: $e1 \in\#\ E \Longrightarrow e2 \in\#\ E \Longrightarrow \exists\ x\ .\ x \in \mathcal{V} \land x \in e1 \land x \in e2 \Longrightarrow$
$hedge\text{-}adjacent\ e1\ e2$
$\langle proof \rangle$

definition *hneighborhood* :: $'a \Rightarrow 'a\ set$ **where**
$hneighborhood\ x \equiv \{v \in \mathcal{V}\ .\ hvert\text{-}adjacent\ x\ v\}$

definition *hmax-degree* :: *nat* **where**
$hmax\text{-}degree \equiv Max\ \{hdegree\ v \mid v.\ v \in \mathcal{V}\}$

definition *hrank* :: *nat* **where**
$hrank \equiv Max\ \{card\ e \mid e\ .\ e \in\#\ E\}$

definition *hcorank* :: *nat* **where**
$hcorank = Min\ \{card\ e \mid e\ .\ e \in\#\ E\}$

definition *hedge-neighbourhood* :: $'a \Rightarrow 'a\ hyp\text{-}edge\ multiset$ **where**
$hedge\text{-}neighbourhood\ x \equiv \{\#\ e \in\#\ E\ .\ x \in e\ \#\}$

lemma *degree-alt-neigbourhood*: $hdegree\ x = size\ (hedge\text{-}neighbourhood\ x)$
$\langle proof \rangle$

definition *hinduced-edges*:: $'a\ set \Rightarrow 'a\ hyp\text{-}edge\ multiset$ **where**
$hinduced\text{-}edges\ V' = \{\#e \in\#\ E\ .\ e \subseteq V'\#\}$

end

Sublocale for rewriting definition purposes rather than inheritance

sublocale *hypersystem* $\subseteq$ *incidence-system* $\mathcal{V}\ E$
rewrites $point\text{-}replication\text{-}number\ E\ v = hdegree\ v$ **and** $points\text{-}index\ E\ vs = hdegree\text{-}set\ vs$
$\langle proof \rangle$

Reverse sublocale to establish equality

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

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* $\subseteq$ *inf-design*
$\langle proof \rangle$

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

sublocale *finite-incidence-system* $\subseteq$ *fin-hypersystem* $\mathcal{V}$ $\mathcal{B}$
$\langle proof \rangle$

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

sublocale *inf-design* $\subseteq$ *hypergraph* $\mathcal{V}$ $\mathcal{B}$
$\langle proof \rangle$

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

sublocale *design* $\subseteq$ *fin-hypergraph* $\mathcal{V}$ $\mathcal{B}$
$\langle proof \rangle$

sublocale *fin-hypergraph* $\subseteq$ *design* $\mathcal{V}$ E
$\langle proof \rangle$

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 $E \Longrightarrow V = VH \Longrightarrow$ *spanning-hypergraph* VH EH V E
$\langle proof \rangle$

context *hypergraph*
begin

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

lemma *is-subhypergraphI*:
assumes (*hyp-verts* $H \subseteq \mathcal{V}$)
assumes (*hyp-edges* $H \subseteq\# E$)
assumes *hypergraph* (*hyp-verts* H) (*hyp-edges* H)
shows *is-subhypergraph* H
$\langle proof \rangle$

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
⟨*proof*⟩

lemma *spanning-subhypergraphI*: (*hyp-verts H*) = $\mathcal{V}$ ⟹ (*hyp-edges H*) ⊆# *E* ⟹
hypergraph (*hyp-verts H*) (*hyp-edges H*) ⟹ *is-spanning-subhypergraph H*
⟨*proof*⟩

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*: $\mathcal{V}$ ≠ {}

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

locale *hypergraph-ne* = *hypergraph* +
assumes *E-nempty*: *E* ≠ {#}
begin

lemma *V-nempty*: $\mathcal{V} \neq \{\}$
⟨*proof*⟩

lemma *sizeE-not-zero*: *size E* $\neq$ *0*
⟨*proof*⟩

end

sublocale *hypergraph-ne* $\subseteq$ *hyper-system-ne*
⟨*proof*⟩

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*
⟨*proof*⟩

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

lemma *order-gt-zero*: *horder* > *0*
⟨*proof*⟩

lemma *order-ge-one*: *horder* $\geq$ *1*
⟨*proof*⟩

end

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

lemma *order-gt-one*: *horder* > *1*
⟨*proof*⟩

lemma *order-ge-two*: *horder* $\geq$ *2*
⟨*proof*⟩

end

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

sublocale *fin-hypergraph-ne* $\subseteq$ *fin-hypersystem-vne*
⟨*proof*⟩

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

sublocale *fin-hypergraph-nt* $\subseteq$ *fin-hypersystem-nt*
⟨*proof*⟩

sublocale *fin-hypergraph-ne* $\subseteq$ *proper-design* $\mathcal{V}$ E
⟨*proof*⟩

sublocale *proper-design* $\subseteq$ *fin-hypergraph-ne* $\mathcal{V}$ $\mathcal{B}$
⟨*proof*⟩

2.2 Regular and Uniform Hypergraphs

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

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

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

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

locale *almost-regular-hypergraph* = *hypergraph* +
assumes $\bigwedge x\ y\ .\ x \in \mathcal{V} \Longrightarrow y \in \mathcal{V} \Longrightarrow |\ hdegree\ x - hdegree\ y\ | \leq 1$

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* $\mathcal{V}$ E
for $\mathcal{V}$ E k

sublocale *fin-kuniform-regular-hypgraph-nt* $\subseteq$ *fin-kuniform-hypergraph* $\mathcal{V}$ E k
⟨*proof*⟩

sublocale *fin-kuniform-regular-hypgraph-nt* $\subseteq$ *fin-dregular-hypergraph* $\mathcal{V}$ E k
⟨*proof*⟩

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

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

sublocale *t-design* $\subseteq$ *block-balanced-design*
⟨*proof*⟩

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

sublocale *fin-kuniform-regular-hypgraph-nt* $\subseteq$ *fin-kuniform-hypergraph-nt* $\mathcal{V}$ *E k*
⟨*proof*⟩

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* $\subseteq$ *block-design* $\mathcal{V}$ *E k*
rewrites *point-replication-number E v* = *hdegree v* **and** *points-index E vs* = *hdegree-set vs*
⟨*proof*⟩

sublocale *fin-kuniform-regular-hypgraph-nt* $\subseteq$ *regular-block-design* $\mathcal{V}$ *E k k*
rewrites *point-replication-number E v* =*hdegree v* **and** *points-index E vs* = *hdegree-set vs*
⟨*proof*⟩

2.3 Factorisations

locale *d-factor* = *spanning-hypergraph* + *dregular-hypergraph* $\mathcal{V}$*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*) $\mathcal{V}$ *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*⟩

sublocale *fin-bipartite-graph* $\subseteq$ *fin-hypersystem-vne V mset-set E*
⟨*proof*⟩

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.