

Hypergraph Basics

Chelsea Edmonds and Lawrence C. Paulson

February 6, 2026

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	1
1.1 Sub hypergraphs	4
2 Hypergraph Variations	5
2.1 Non-trivial hypergraphs	6
2.2 Regular and Uniform Hypergraphs	7
2.3 Factorisations	8
2.4 Sample Graph Theory Connections	9

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 $C \implies \text{is-singleton } (f \text{ ` } C)$
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* $C \iff \text{is-singleton } (f \text{ ` } C)$
proof (*intro iffI*)
show *is-singleton* $C \implies \text{is-singleton } (f \text{ ` } C)$ **by** (*rule is-singleton-image*)
show *is-singleton* $(f \text{ ` } C) \implies \text{is-singleton } C$ **using** *assms is-singleton-image*
by (*metis bij-betw-def inv-into-image-cancel*)
qed

lemma *image-singleton*:
assumes $A \neq \{\}$
assumes $\bigwedge x. x \in A \implies f x = c$
shows $f \text{ ` } A = \{c\}$
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*) \times (*'a hyp-edge multiset*)

abbreviation *hyp-edges* :: *'a hyp-graph* \Rightarrow *'a hyp-edge multiset* **where**
hyp-edges $H \equiv \text{snd } H$

abbreviation *hyp-verts* :: *'a hyp-graph* \Rightarrow *'a set* **where**
hyp-verts $H \equiv \text{fst } H$

locale *hypersystem* = *incidence-system vertices* :: *'a set edges* :: *'a hyp-edge multiset*
for *vertices* ($\langle \mathcal{V} \rangle$) **and** *edges* ($\langle E \rangle$)

begin
Basic definitions using hypergraph language

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

definition *hdegree* :: *'a* \Rightarrow *nat* **where**
hdegree $v \equiv \text{size } \{\#e \in \# E . v \in e \#\}$

lemma *hdegree-rep-num*: *hdegree* $v = \text{point-replication-number } E v$
unfolding *hdegree-def point-replication-number-def* **by** *simp*

definition *hdegree-set* :: 'a set \Rightarrow nat **where**
hdegree-set vs \equiv size {#e \in # E. vs \subseteq 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 \Rightarrow 'a \Rightarrow bool **where**
hvert-adjacent v1 v2 \equiv \exists e . e \in # E \wedge v1 \in e \wedge v2 \in e \wedge v1 \in V \wedge v2 \in V

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

lemma *edge-adjacent-alt-def*: e1 \in # E \Longrightarrow e2 \in # E \Longrightarrow \exists x . x \in V \wedge x \in e1
 \wedge x \in e2 \Longrightarrow
hedge-adjacent e1 e2
unfolding *hedge-adjacent-def* **by** *auto*

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

definition *hmax-degree* :: nat **where**
hmax-degree \equiv Max {*hdegree* v | v . v \in V}

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

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

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

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

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

end

Sublocale for rewriting definition purposes rather than inheritance

sublocale *hypersystem* \subseteq *incidence-system* 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* \subseteq *hypersystem* V B

rewrites *hdegree* v = *point-replication-number* B v **and** *hdegree-set* vs = *points-index*
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-empty:  $bl \in \# \mathcal{B} \implies bl \neq \{\}$ 

```

```

sublocale design  $\subseteq$  inf-design
  by unfold-locales (simp add: blocks-empty)

```

```

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

```

```

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

```

```

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

```

```

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

```

```

locale fin-hypergraph = hypergraph + fin-hypersystem

```

```

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

```

```

sublocale fin-hypergraph  $\subseteq$  design  $\mathcal{V} E$ 
  using blocks-empty 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  $\mathcal{V} :: 'a \text{ set } E$ 
+
  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  $\mathcal{V} H \ EH \ \mathcal{V} E \implies \mathcal{V} = \mathcal{V} H \implies$ 
spanning-hypergraph  $\mathcal{V} H \ EH \ \mathcal{V} E$ 
  using spanning-hypergraph-def spanning-hypergraph-axioms-def by blast

```

```

context hypergraph
begin

```

```

definition is-subhypergraph :: 'a hyp-graph  $\Rightarrow$  bool where

```

is-subhypergraph $H \equiv \text{sub-hypergraph } (\text{hyp-verts } H) (\text{hyp-edges } H) \vee E$

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

unfolding *is-subhypergraph-def*

proof –

interpret *h*: *hypergraph* *hyp-verts* H *hyp-edges* H

using *assms*(\mathcal{B}) **by** *simp*

show *sub-hypergraph* (*hyp-verts* H) (*hyp-edges* H) $\vee E$

by (*unfold-locales*) (*simp-all add: assms*)

qed

definition *hypergraph-decomposition* :: 'a *hyp-graph multiset* \Rightarrow *bool* **where**

hypergraph-decomposition $S \equiv (\forall h \in\# S . \text{is-subhypergraph } h) \wedge$

partition-on-mset $E \{ \# \text{hyp-edges } h . h \in\# S \}$

definition *is-spanning-subhypergraph* :: 'a *hyp-graph* \Rightarrow *bool* **where**

is-spanning-subhypergraph $H \equiv \text{spanning-hypergraph } (\text{hyp-verts } H) (\text{hyp-edges } H)$

$\vee E$

lemma *is-spanning-subhypergraphI*: *is-subhypergraph* $H \Longrightarrow (\text{hyp-verts } H) = \mathcal{V}$

\Longrightarrow

is-spanning-subhypergraph H

unfolding *is-subhypergraph-def is-spanning-subhypergraph-def* **using** *spanning-hypergraphI*

by *blast*

lemma *spanning-subhypergraphI*: $(\text{hyp-verts } H) = \mathcal{V} \Longrightarrow (\text{hyp-edges } H) \subseteq\# E$

\Longrightarrow

hypergraph (*hyp-verts* H) (*hyp-edges* H) \Longrightarrow *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* ≥ 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  $\mathcal{V}$  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:  $\bigwedge x. x \in \mathcal{V} \implies \text{hdegree } x = d$ 

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

locale kuniform-hypergraph = hypergraph +
  fixes k :: nat
  assumes uniform:  $\bigwedge e. e \in \# E \implies \text{card } e = k$ 

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

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

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* $\mathcal{V} E k$
by *unfold-locales*

sublocale *fin-kuniform-regular-hypgraph-nt* \subseteq *fin-dregular-hypergraph* $\mathcal{V} E k$
by *unfold-locales*

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

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

sublocale *t-design* \subseteq *block-balanced-design*
by *unfold-locales*

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

sublocale *fin-kuniform-regular-hypgraph-nt* \subseteq *fin-kuniform-hypergraph-nt* $\mathcal{V} E k$
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* \subseteq *block-design* $\mathcal{V} E k$
rewrites *point-replication-number* $E v = \text{hdegree } v$ **and** *points-index* $E vs = \text{hdegree-set } vs$
using *uniform* **by** (*unfold-locales*)
(*simp-all add: point-replication-number-def hdegree-def hdegree-set-def points-index-def E-empty*)

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

context *hypergraph*
begin

definition *is-d-factor* :: 'a hyp-graph \Rightarrow bool **where**
is-d-factor $H \equiv (\exists d. d\text{-factor } (hyp\text{-verts } H) (hyp\text{-edges } H) \mathcal{V} E d)$

definition *d-factorisation* :: 'a hyp-graph multiset \Rightarrow bool **where**
d-factorisation $S \equiv hypergraph\text{-decomposition } S \wedge (\forall h \in \# S. is\text{-}d\text{-factor } h)$
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  $\bigwedge b. b \in \# \text{ mset-set } E \implies b \subseteq V$  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.