VectorSpace

Holden Lee*

March 17, 2025

Abstract

I present a formalisation of basic linear algebra based completely on locales, building off HOL-Algebra. It includes the following:

1. basic definitions: linear combinations, span, linear independence
2. linear transformations
3. interpretation of function spaces as vector spaces
4. direct sum of vector spaces, sum of subspaces
5. the replacement theorem
6. existence of bases in finite-dimensional vector spaces, definition of dimension
7. rank-nullity theorem.

Note that some concepts are actually defined and proved for modules as they also apply there.

In the process, I also prove some basic facts about rings, modules, and fields, as well as finite sums in monoids/modules.

Note that infinite-dimensional vector spaces are supported, but dimension is only supported for finite-dimensional vector spaces.

The proofs are standard; the proofs of the replacement theorem and rank-nullity theorem roughly follow the presentation in [**?**]. The rank-nullity theorem generalises the existing development in [**?**] (originally using type classes, now using a mix of type classes and locales).

Contents

*This work was funded by the Post-Masters Consultancy and the Computer Laboratory at the University of Cambridge.

1 Basic facts about rings and modules

theory *RingModuleFacts*
imports *Main*
HOL−Algebra.Module
HOL−Algebra.Coset

begin

1.1 Basic facts

In a field, every nonzero element has an inverse.

lemma (**in** *field*) *inverse-exists* [*simp*, *intro*]:
assumes *h1*: $a \in carrier\ R$ **and** *h2*: $a \neq \mathbf{0}_R$
shows $inv_R\ a \in carrier\ R$
⟨*proof*⟩

Multiplication by 0 in R gives 0. (Note that this fact encompasses smult-l-null as this is for module while that is for algebra, so smult-l-null is redundant.)

lemma (**in** *module*) *lmult-0* [*simp*]:
assumes *1*: $m \in carrier\ M$
shows $\mathbf{0}_R \odot_M m = \mathbf{0}_M$
⟨*proof*⟩

Multiplication by 0 in M gives 0.

lemma (**in** *module*) *rmult-0* [*simp*]:
assumes *0*: $r \in carrier\ R$
shows $r \odot_M \mathbf{0}_M = \mathbf{0}_M$
⟨*proof*⟩

Multiplication by -1 is the same as negation. May be useful as a simp rule.

lemma (**in** *module*) *smult-minus-1*:
  fixes v
  assumes $0{:}v \in carrier\ M$
  shows $(\ominus_R \mathbf{1}_R) \odot_M v = (\ominus_M\ v)$

⟨*proof*⟩

The version with equality reversed.

lemmas (**in** *module*) *smult-minus-1-back* = *smult-minus-1*[*THEN sym*]

-1 is not 0

lemma (**in** *field*) *neg-1-not-0* [*simp*]: $\ominus_R \mathbf{1}_R \neq \mathbf{0}_R$
⟨*proof*⟩

Note smult-assoc1 is the wrong way around for simplification. This is the reverse of smult-assoc1.

lemma (**in** *module*) *smult-assoc-simp*:
$[|\ a \in carrier\ R;\ b \in carrier\ R;\ x \in carrier\ M\ |] ==>$
  $a \odot_M (b \odot_M x) = (a \otimes b) \odot_M x$
⟨*proof*⟩

lemmas (**in** *abelian-group*) *show-r-zero*= *add.l-cancel-one*
lemmas (**in** *abelian-group*) *show-l-zero*= *add.r-cancel-one*

A nontrivial ring has $0 \neq 1$.

lemma (**in** *ring*) *nontrivial-ring* [*simp*]:
  assumes $carrier\ R \neq \{\mathbf{0}_R\}$
  shows $\mathbf{0}_R \neq \mathbf{1}_R$
⟨*proof*⟩

Use as simp rule. To show $a - b = 0$, it suffices to show $a = b$.

lemma (**in** *abelian-group*) *minus-other-side* [*simp*]:
  $[\![a \in carrier\ G;\ b \in carrier\ G]\!] \Longrightarrow (a \ominus_G b = \mathbf{0}_G) = (a = b)$
  ⟨*proof*⟩

1.2 Units group

Define the units group $R^\times$ and show it is actually a group.

definition *units-group*::('a,'b) *ring-scheme* ⇒ 'a *monoid*
  where $units\text{-}group\ R = (\!|carrier = Units\ R,\ mult = (\lambda x\ y.\ x \otimes_R y),\ one = \mathbf{1}_R|\!)$

The units form a group.

lemma (**in** *ring*) *units-form-group*: *group* (*units-group* R)
⟨*proof*⟩

The units of a *cring* form a commutative group.

lemma (**in** *cring*) *units-form-cgroup*: *comm-group* (*units-group* R)
⟨*proof*⟩

end

2 Basic lemmas about functions

theory *FunctionLemmas*

imports *Main*
HOL−Library.FuncSet
begin

These are used in simplification. Note that the difference from Pi-mem is that the statement about the function comes first, so Isabelle can more easily figure out what S is.

lemma *PiE-mem2*: $f \in S \rightarrow_E T \Longrightarrow x \in S \Longrightarrow f\, x \in T$
⟨*proof*⟩
lemma *Pi-mem2*: $f \in S \rightarrow T \Longrightarrow x \in S \Longrightarrow f\, x \in T$
⟨*proof*⟩

end

3 Sums in monoids

theory *MonoidSums*

imports *Main*
HOL−Algebra.Module
RingModuleFacts
FunctionLemmas
begin

We build on the finite product simplifications in FiniteProduct.thy and the analogous ones for finite sums (see "lemmas" in Ring.thy).

Use as an intro rule

lemma (**in** *comm-monoid*) *factors-equal*:
$[\![a=b;\ c=d]\!] \Longrightarrow a \otimes_G c = b \otimes_G d$
⟨*proof*⟩

lemma (**in** *comm-monoid*) *extend-prod*:
fixes $a\ A\ S$
assumes *fin*: *finite* S **and** *subset*: $A \subseteq S$ **and** a: $a \in A \rightarrow$ *carrier* G
shows $(\bigotimes_G\ x \in S.\ (\mathit{if}\ x \in A\ \mathit{then}\ a\ x\ \mathit{else}\ \mathbf{1}_G)) = (\bigotimes_G\ x \in A.\ a\ x)$
(**is** $(\bigotimes_G\ x \in S.\ ?b\ x) = (\bigotimes_G\ x \in A.\ a\ x)$)
⟨*proof*⟩

Scalar multiplication distributes over scalar multiplication (on left).

lemma (**in** *module*) *finsum-smult*:
[| $c \in$ *carrier* R; $g \in A \rightarrow$ *carrier* M |] ==>
$(c \odot_M$ *finsum* $M\ g\ A) =$ *finsum* M (%$x.\ c \odot_M g\ x)\ A$
⟨*proof*⟩

Scalar multiplication distributes over scalar multiplication (on right).

lemma (**in** *module*) *finsum-smult-r*:
[| $v \in$ *carrier* M; $f \in A \rightarrow$ *carrier* R |] ==>
(*finsum* $R\ f\ A \odot_M v) =$ *finsum* M (%$x.\ f\ x \odot_M v)\ A$
⟨*proof*⟩

A sequence of lemmas that shows that the product does not depend on the ambient group. Note I had to dig back into the definitions of foldSet to show this.

lemma *foldSet-not-depend*:
fixes $A\ E$
assumes *h1*: $D \subseteq E$
shows *foldSetD* $D\ f\ e \subseteq$*foldSetD* $E\ f\ e$
⟨*proof*⟩

lemma *foldD-not-depend*:
fixes $D\ E\ B\ f\ e\ A$
assumes *h1*: *LCD* $B\ D\ f$ **and** *h2*: *LCD* $B\ E\ f$ **and** *h3*: $D \subseteq E$ **and** *h4*: $e \in D$ **and** *h5*: $A \subseteq B$ **and** *h6*: *finite* B
shows *foldD* $D\ f\ e\ A =$ *foldD* $E\ f\ e\ A$
⟨*proof*⟩

lemma (**in** *comm-monoid*) *finprod-all1*[*simp*]:
assumes *all1*: $\bigwedge a.\ a \in A \Longrightarrow f\ a = \mathbf{1}_G$
shows $(\bigotimes_G\ a \in A.\ f\ a) = \mathbf{1}_G$

⟨*proof*⟩

context *abelian-monoid*
begin
lemmas *summands-equal* = *add.factors-equal*

lemmas *extend-sum* = *add.extend-prod*
lemmas *finsum-all0* = *add.finprod-all1*
end

end

4 Linear Combinations

theory *LinearCombinations*
imports *Main*
  HOL−Algebra.Module
  HOL−Algebra.Coset
  RingModuleFacts
  MonoidSums
  FunctionLemmas
begin

4.1 Lemmas for simplification

The following are helpful in certain simplifications (esp. congruence rules). Warning: arbitrary use leads to looping.

lemma (**in** *ring*) *coeff-in-ring*:
  $[\![a{\in}A{\rightarrow}carrier\ R;\ x{\in}A]\!] \Longrightarrow a\ x \in carrier\ R$
⟨*proof*⟩

lemma (**in** *ring*) *coeff-in-ring2*:
  $[\![\ x{\in}A;a{\in}A{\rightarrow}carrier\ R]\!] \Longrightarrow a\ x \in carrier\ R$
⟨*proof*⟩

lemma *ring-subset-carrier*:
  $[\![x \in A;\ A{\subseteq}carrier\ R]\!] \Longrightarrow x \in carrier\ R$
⟨*proof*⟩

lemma *disj-if*:
  $[\![A{\cap}B{=}\{\};\ x{\in}\ B]\!] \Longrightarrow (\textit{if}\ x{\in}A\ \textit{then}\ f\ x\ \textit{else}\ g\ x) = g\ x$
⟨*proof*⟩

lemmas (**in** *module*) *sum-simp* = *ring-subset-carrier*

4.2 Linear combinations

A linear combination is $\sum_{v\in A} a_v v$. $(a_v)_{v\in S}$ is a function $A \to K$, where $A \subseteq K$.

definition (**in** *module*) *lincomb*::$['c \Rightarrow 'a,\ 'c\ set]{\Rightarrow}\ 'c$
where *lincomb a A* = $(\bigoplus_M\ v{\in}A.\ (a\ v \odot_M v))$

lemma (**in** *module*) *summands-valid*:
  fixes *A a*
  assumes *h2*: $A \subseteq$ *carrier M* **and** *h3*: $a \in (A \to$ *carrier R*)
  shows $\forall\ v \in A.\ (((a\ v) \odot_M v) \in$ *carrier M*)
⟨*proof*⟩

lemma (**in** *module*) *lincomb-closed* [*simp*, *intro*]:
  fixes *S a*
  assumes *h2*: $S \subseteq$ *carrier M* **and** *h3*: $a \in (S \to$ *carrier R*)
  shows *lincomb a S* $\in$ *carrier M*
⟨*proof*⟩

lemma (**in** *comm-monoid*) *finprod-cong2*:
  [| $A = B$;
    !!i. $i \in B$ ==> $f\ i = g\ i$; $f \in B \to$ *carrier G*|] ==>
finprod G f A = *finprod G g B*
⟨*proof*⟩

lemmas (**in** *abelian-monoid*) *finsum-cong2* = *add.finprod-cong2*

lemma (**in** *module*) *lincomb-cong*:
  assumes *h2*: $A=B$ **and** *h3*: $A \subseteq$ *carrier M*
    and *h4*: $\bigwedge v.\ v \in A \Longrightarrow a\ v = b\ v$ **and** *h5*: $b \in B \to$ *carrier R*
  shows *lincomb a A* = *lincomb b B*
⟨*proof*⟩

lemma (**in** *module*) *lincomb-union*:
  fixes *a A B*
  assumes *h1*: *finite* $(A \cup B)$ **and** *h3*: $A \cup B \subseteq$ *carrier M*
    and *h4*: $A \cap B = \{\}$ **and** *h5*: $a \in (A \cup B \to$ *carrier R*)
  shows *lincomb a* $(A \cup B)$ = *lincomb a A* $\oplus_M$ *lincomb a B*
⟨*proof*⟩

This is useful as a simp rule sometimes, for combining linear combinations.

lemma (**in** *module*) *lincomb-union2*:
  fixes *a b A B*
  assumes *h1*: *finite* $(A \cup B)$ **and** *h3*: $A \cup B \subseteq$ *carrier M*
    and *h4*: $A \cap B = \{\}$ **and** *h5*: $a \in A \to$ *carrier R* **and** *h6*: $b \in B \to$ *carrier R*
  shows *lincomb a A* $\oplus_M$ *lincomb b B* = *lincomb* ($\lambda v.$ *if* $(v \in A)$ *then a v else b v*) $(A \cup B)$
    (**is** *lincomb a A* $\oplus_M$ *lincomb b B* = *lincomb ?c* $(A \cup B)$)
⟨*proof*⟩

lemma (**in** *module*) *lincomb-del2*:
  fixes *S a v*
  assumes *h1*: *finite S* **and** *h2*: $S \subseteq$ *carrier M* **and** *h3*: $a \in (S \to$ *carrier R*) **and** *h4*:$v \in S$

shows *lincomb a S* = ((*a v*) $\odot_M$ *v*) $\oplus_M$ *lincomb a* (*S*−{*v*})
⟨*proof*⟩

lemma (**in** *module*) *lincomb-insert*:
fixes *S a v*
assumes *h1*: *finite S* **and** *h2*: *S*⊆ *carrier M* **and** *h3*: *a*∈(*S*∪{*v*}→*carrier R*) **and** *h4*:*v*∉*S* **and**
h5:*v*∈ *carrier M*
shows *lincomb a* (*S*∪{*v*}) = ((*a v*) $\odot_M$ *v*) $\oplus_M$ *lincomb a S*
⟨*proof*⟩

lemma (**in** *module*) *lincomb-elim-if* [*simp*]:
fixes *b c S*
assumes *h1*: *S* ⊆ *carrier M* **and** *h2*: ⋀*v*. *v*∈*S*⟹ ¬*P v* **and** *h3*: *c*∈*S*→*carrier R*
shows *lincomb* (λ*w*. *if P w then b w else c w*) *S* = *lincomb c S*
⟨*proof*⟩

lemma (**in** *module*) *lincomb-smult*:
fixes *A c*
assumes *h2*: *A*⊆*carrier M* **and** *h3*: *a*∈*A*→*carrier R* **and** *h4*: *c*∈*carrier R*
shows *lincomb* (λ*w*. *c*$\otimes_R$ *a w*) *A* = *c*$\odot_M$ (*lincomb a A*)
⟨*proof*⟩

4.3 Linear dependence and independence.

A set S in a module/vectorspace is linearly dependent if there is a finite set $A \subseteq S$ and coefficients $(a_v)_{v\in A}$ such that $sum_{v\in A} a_v v = 0$ and for some v, $a_v \neq 0$.

definition (**in** *module*) *lin-dep* **where**
lin-dep S = (∃ *A a v*. (*finite A* ∧ *A*⊆*S* ∧ (*a*∈ (*A*→*carrier R*)) ∧ (*lincomb a A* = $\mathbf{0}_M$) ∧ (*v*∈*A*) ∧ (*a v*≠ $\mathbf{0}_R$)))

abbreviation (**in** *module*) *lin-indpt*::′*c set* ⇒ *bool*
where *lin-indpt S* ≡ ¬*lin-dep S*

In the finite case, we can take $A = S$. This may be more convenient (e.g., when adding two linear combinations.

lemma (**in** *module*) *finite-lin-dep*:
fixes *S*
assumes *finS*:*finite S* **and** *ld*: *lin-dep S* **and** *inC*: *S*⊆*carrier M*
shows ∃ *a v*. (*a*∈ (*S*→*carrier R*)) ∧ (*lincomb a S* = $\mathbf{0}_M$) ∧ (*v*∈*S*) ∧ (*a v*≠ $\mathbf{0}_R$)
⟨*proof*⟩

Criteria of linear dependency in a easy format to apply: apply (rule lin-dep-crit)

lemma (**in** *module*) *lin-dep-crit*:
fixes *A S a v*
assumes *fin*: *finite A* **and** *subset*: *A*⊆*S* **and** *h1*: (*a*∈ (*A*→*carrier R*)) **and** *h2*: *v*∈ *A*
and *h3*:*a v*≠ $\mathbf{0}_R$ **and** *h4*: (*lincomb a A* = $\mathbf{0}_M$)
shows *lin-dep S*
⟨*proof*⟩

If $\sum_{v\in A} a_v v = 0$ implies $a_v = 0$ for all $v \in S$, then A is linearly independent.

lemma (**in** *module*) *finite-lin-indpt2*:
fixes *A*
assumes *A-fin*: *finite A* **and** *AinC*: *A*⊆*carrier M* **and**
lc0: ⋀*a*. *a*∈ (*A*→*carrier R*) ⟹ (*lincomb a A* = $\mathbf{0}_M$) ⟹ (∀ *v*∈*A*. *a v*= $\mathbf{0}_R$)
shows *lin-indpt A*
⟨*proof*⟩

Any set containing 0 is linearly dependent.

lemma (**in** *module*) *zero-lin-dep*:
assumes *0*: $\mathbf{0}_M \in S$ **and** *nonzero*: *carrier R* ≠ {$\mathbf{0}_R$}
shows *lin-dep S*
⟨*proof*⟩

lemma (**in** *module*) *zero-nin-lin-indpt*:
assumes *h2*: *S*⊆*carrier M* **and** *li*: ¬(*lin-dep S*) **and** *nonzero*: *carrier R* ≠ {$\mathbf{0}_R$}
shows $\mathbf{0}_M \notin S$
⟨*proof*⟩

The *span* of S is the set of linear combinations with $A \subseteq S$.

definition (**in** *module*) *span*::′*c set* ⇒′*c set*
where *span S* = {*lincomb a A* | *a A*. *finite A* ∧ *A*⊆*S* ∧ *a*∈ (*A*→*carrier R*)}

The *span* interpreted as a module or vectorspace.

abbreviation (**in** *module*) *span-vs*::′*c set* ⇒ (′*a*,′*c*,′*d*) *module-scheme*
where *span-vs S* ≡ *M* ⦇*carrier* := *span S*⦈

In the finite case, we can take $A = S$ without loss of generality.

lemma (**in** *module*) *finite-span*:
assumes *fin*: *finite S* **and** *inC*: *S*⊆*carrier M*
shows *span S* = {*lincomb a S* | *a*. *a*∈ (*S*→*carrier R*)}
⟨*proof*⟩

If $v \in$ span S, then we can find a linear combination. This is in an easy to apply format (e.g. obtain a A where...)

lemma (**in** *module*) *in-span*:
fixes $S\ v$
assumes *h2*: $S \subseteq carrier\ V$ **and** *h3*: $v \in span\ S$
shows $\exists\, a\ A.\ (A \subseteq S \wedge (a \in A \rightarrow carrier\ R) \wedge (lincomb\ a\ A = v))$
⟨*proof*⟩

In the finite case, we can take $A = S$.

lemma (**in** *module*) *finite-in-span*:
fixes $S\ v$
assumes *fin*: *finite* S **and** *h2*: $S \subseteq carrier\ M$ **and** *h3*: $v \in span\ S$
shows $\exists\, a.\ (a \in S \rightarrow carrier\ R) \wedge (lincomb\ a\ S = v)$
⟨*proof*⟩

If a subset is linearly independent, then any linear combination that is 0 must have a nonzero coefficient outside that set.

lemma (**in** *module*) *lincomb-must-include*:
fixes $A\ S\ T\ b\ v$
assumes *inC*: $T \subseteq carrier\ M$ **and** *li*: *lin-indpt* S **and** *Ssub*: $S \subseteq T$ **and** *Ssub*: $A \subseteq T$
and *fin*: *finite* A
and *b*: $b \in A \rightarrow carrier\ R$ **and** *lc*: $lincomb\ b\ A = \mathbf{0}_M$ **and** *v-in*: $v \in A$
and *nz-coeff*: $b\ v \neq \mathbf{0}_R$
shows $\exists\, w \in A - S.\ b\ w \neq \mathbf{0}_R$
⟨*proof*⟩

A generating set is a set such that the span of S is all of M.

abbreviation (**in** *module*) *gen-set*::$'c\ set \Rightarrow bool$
where *gen-set* $S \equiv (span\ S = carrier\ M)$

4.4 Submodules

lemma *module-criteria*:
fixes R **and** M
assumes *cring*: *cring* R
and *zero*: $\mathbf{0}_M \in carrier\ M$
and *add*: $\forall\, v\ w.\ v \in carrier\ M \wedge w \in carrier\ M \longrightarrow v \oplus_M w \in carrier\ M$
and *neg*: $\forall\, v \in carrier\ M.\ (\exists\, neg\text{-}v \in carrier\ M.\ v \oplus_M neg\text{-}v = \mathbf{0}_M)$
and *smult*: $\forall\, c\ v.\ c \in carrier\ R \wedge v \in carrier\ M \longrightarrow c \odot_M v \in carrier\ M$
and *comm*: $\forall\, v\ w.\ v \in carrier\ M \wedge w \in carrier\ M \longrightarrow v \oplus_M w = w \oplus_M v$
and *assoc*: $\forall\, v\ w\ x.\ v \in carrier\ M \wedge w \in carrier\ M \wedge x \in carrier\ M \longrightarrow (v \oplus_M w) \oplus_M x = v \oplus_M (w \oplus_M x)$
and *add-id*: $\forall\, v \in carrier\ M.\ (v \oplus_M \mathbf{0}_M = v)$
and *compat*: $\forall\, a\ b\ v.\ a \in carrier\ R \wedge b \in carrier\ R \wedge v \in carrier\ M \longrightarrow (a \otimes_R b) \odot_M v = a \odot_M (b \odot_M v)$

and *smult-id*: $\forall$ *v*$\in$*carrier M*. ($\mathbf{1}_R \odot_M v = v$)
and *dist-f*: $\forall$ *a b v*. *a*$\in$ *carrier R* $\wedge$ *b*$\in$ *carrier R* $\wedge$ *v*$\in$*carrier M*$\longrightarrow$ $(a \oplus_R b) \odot_M v = (a \odot_M v) \oplus_M (b \odot_M v)$
and *dist-add*: $\forall$ *a v w*. *a*$\in$ *carrier R* $\wedge$ *v*$\in$*carrier M* $\wedge$ *w*$\in$*carrier M*$\longrightarrow$ $a \odot_M (v \oplus_M w) = (a \odot_M v) \oplus_M (a \odot_M w)$
shows *module R M*
⟨*proof*⟩

A submodule is $N \subseteq M$ that is closed under addition and scalar multiplication, and contains 0 (so is not empty).

locale *submodule* =
fixes *R* **and** *N* **and** *M* (**structure**)
assumes *module*: *module R M*
and *subset*: *N* $\subseteq$ *carrier M*
and *m-closed* [*intro*, *simp*]: ⟦*v* $\in$ *N*; *w* $\in$ *N*⟧ $\Longrightarrow$ *v* $\oplus$ *w* $\in$ *N*
and *zero-closed* [*simp*]: $\mathbf{0}$ $\in$ *N*
and *smult-closed* [*intro*, *simp*]: ⟦*c* $\in$ *carrier R*; *v* $\in$ *N*⟧ $\Longrightarrow$ *c*$\odot$*v* $\in$ *N*

abbreviation (**in** *module*) *md*::'*c set* $\Rightarrow$ ('*a*, '*c*, '*d*) *module-scheme*
where *md N* $\equiv$ *M*⦇*carrier* :=*N*⦈

lemma (**in** *module*) *carrier-vs-is-self* [*simp*]:
carrier (*md N*) = *N*
⟨*proof*⟩

lemma (**in** *module*) *submodule-is-module*:
fixes *N*::'*c set*
assumes *0*: *submodule R N M*
shows *module R* (*md N*)
⟨*proof*⟩

$N_1 + N_2 = \{x + y | x \in N_1, y \in N_2\}$

definition (**in** *module*) *submodule-sum*:: ['*c set*, '*c set*] $\Rightarrow$ '*c set*
where *submodule-sum N1 N2* = (λ (*x*,*y*). *x* $\oplus_M$ *y*) '{(*x*,*y*). *x*$\in$ *N1* $\wedge$ *y*$\in$ *N2*}

A module homomorphism $M \to N$ preserves addition and scalar multiplication.

definition *module-hom*:: [('*a*, '*c0*) *ring-scheme*,
('*a*,'*b1*,'*c1*) *module-scheme*, ('*a*,'*b2*,'*c2*) *module-scheme*] $\Rightarrow$('*b1*$\Rightarrow$'*b2*) *set*
where *module-hom R M N* = {*f*.
((*f*$\in$ *carrier M* $\to$ *carrier N*)
$\wedge$ ($\forall$ *m1 m2*. *m1*$\in$*carrier M*$\wedge$ *m2*$\in$*carrier M* $\longrightarrow$ *f* (*m1* $\oplus_M$ *m2*) = (*f m1*) $\oplus_N$ (*f m2*))
$\wedge$ ($\forall$ *r m*. *r*$\in$*carrier R*$\wedge$ *m*$\in$*carrier M* $\longrightarrow$*f* (*r* $\odot_M$ *m*) = *r* $\odot_N$ (*f m*)))}

lemma *module-hom-closed*: $f \in$ *module-hom R M N* $\Longrightarrow f \in$ *carrier M* $\to$ *carrier N*
⟨*proof*⟩

lemma *module-hom-add*: $[\![f \in$ *module-hom R M N*; $m1 \in$ *carrier M*; $m2 \in$ *carrier M* $]\!] \Longrightarrow f\ (m1 \oplus_M m2) = (f\ m1) \oplus_N (f\ m2)$
⟨*proof*⟩

lemma *module-hom-smult*: $[\![f \in$ *module-hom R M N*; $r \in$ *carrier R*; $m \in$ *carrier M* $]\!] \Longrightarrow f\ (r \odot_M m) = r \odot_N (f\ m)$
⟨*proof*⟩

locale *mod-hom* =
  M?: *module R M* + *N?*: *module R N*
    for *R* **and** *M* **and** *N* +
  fixes *f*
  assumes *f-hom*: $f \in$ *module-hom R M N*
  notes *f-add* [*simp*] = *module-hom-add* [*OF f-hom*]
    and *f-smult* [*simp*] = *module-hom-smult* [*OF f-hom*]

Some basic simplification rules for module homomorphisms.

context *mod-hom*
begin

lemma *f-im* [*simp*, *intro*]:
assumes $v \in$ *carrier M*
shows $f\ v \in$ *carrier N*
⟨*proof*⟩

definition *im*:: *'e set*
  where *im* = *f*‘(*carrier M*)

definition *ker*:: *'c set*
  where *ker* = $\{v.\ v \in$ *carrier M* & $f\ v = \mathbf{0}_N\}$

lemma *f0-is-0*[*simp*]: $f\ \mathbf{0}_M = \mathbf{0}_N$
⟨*proof*⟩

lemma *f-neg* [*simp*]: $v \in$ *carrier M* $\Longrightarrow f\ (\ominus_M v) = \ominus_N f\ v$
  ⟨*proof*⟩

lemma *f-minus* [*simp*]: $[\![v \in$ *carrier M*; $w \in$ *carrier M*$]\!] \Longrightarrow f\ (v \ominus_M w) = f\ v \ominus_N f\ w$
  ⟨*proof*⟩

lemma *ker-is-submodule*: *submodule R ker M*
⟨*proof*⟩

lemma *im-is-submodule*: *submodule R im N*

$\langle proof \rangle$

lemma (**in** *mod-hom*) *f-ker*:
 $v \in ker \Longrightarrow f\ v = \mathbf{0}_N$
$\langle proof \rangle$
end

We will show that for any set S, the space of functions $S \to K$ forms a vector space.

definition (**in** *ring*) *func-space*:: $'z\ set \Rightarrow ('a,('z \Rightarrow 'a))\ module$
 where $func\text{-}space\ S = (|carrier = S \rightarrow_E carrier\ R,$
 $mult = (\lambda\ f\ g.\ restrict\ (\lambda v.\ \mathbf{0}_R)\ S),$
 $one = \ restrict\ (\lambda v.\ \mathbf{0}_R)\ S,$
 $zero = restrict\ (\lambda v.\ \mathbf{0}_R)\ S,$
 $add = (\lambda\ f\ g.\ restrict\ (\lambda v.\ f\ v \oplus_R g\ v)\ S),$
 $smult = (\lambda\ c\ f.\ restrict\ (\lambda v.\ c \otimes_R f\ v)\ S)|)$

lemma (**in** *cring*) *func-space-is-module*:
 fixes S
 shows *module R* (*func-space S*)
$\langle proof \rangle$

Note: one can define M^n from this.

A linear combination is a module homomorphism from the space of coefficients to the module, $(a_v) \mapsto \sum_{v \in S} a_v v$.

lemma (**in** *module*) *lincomb-is-mod-hom*:
 fixes S
 assumes *h*: *finite S* **and** *h2*: $S \subseteq carrier\ M$
 shows *mod-hom R* (*func-space S*) M ($\lambda a.\ lincomb\ a\ S$)
$\langle proof \rangle$

lemma (**in** *module*) *lincomb-sum*:
 assumes *A-fin*: *finite A* **and** *AinC*: $A \subseteq carrier\ M$ **and** *a-fun*: $a \in A \rightarrow carrier$ R **and**
 b-fun: $b \in A \rightarrow carrier\ R$
 shows $lincomb\ (\lambda v.\ a\ v \oplus_R b\ v)\ A = lincomb\ a\ A \oplus_M lincomb\ b\ A$
$\langle proof \rangle$

The negative of a function is just pointwise negation.

lemma (**in** *cring*) *func-space-neg*:
 fixes f
 assumes $f \in\ carrier\ (func\text{-}space\ S)$
 shows $\ominus_{func\text{-}space\ S} f = (\lambda\ v.\ if\ (v \in S)\ then\ \ominus_R f\ v\ else\ undefined)$
$\langle proof \rangle$

Ditto for subtraction. Note the above is really a special case, when a is the 0 function.

lemma (**in** *module*) *lincomb-diff*:
assumes *A-fin*: *finite A* **and** *AinC*: $A \subseteq$ *carrier M* **and** *a-fun*: $a \in A \rightarrow$ *carrier R* **and**
b-fun: $b \in A \rightarrow$ *carrier R*
shows *lincomb* $(\lambda v.\ a\ v \ominus_R b\ v)\ A =$ *lincomb a A* $\ominus_M$ *lincomb b A*
⟨*proof*⟩

The union of nested submodules is a submodule. We will use this to show that span of any set is a submodule.

lemma (**in** *module*) *nested-union-vs*:
fixes $I\ N\ N'$
assumes *subm*: $\bigwedge i.\ i \in I \Longrightarrow$ *submodule R* $(N\ i)\ M$
and *max-exists*: $\bigwedge i\ j.\ i \in I \Longrightarrow j \in I \Longrightarrow (\exists\, k.\ k \in I \wedge N\ i \subseteq N\ k \wedge N\ j \subseteq N\ k)$
and *uni*: $N' = (\bigcup\ i \in I.\ N\ i)$
and *ne*: $I \neq \{\}$
shows *submodule R* N' *M*
⟨*proof*⟩

lemma (**in** *module*) *span-is-monotone*:
fixes *S T*
assumes *subs*: $S \subseteq T$
shows *span S* $\subseteq$ *span T*
⟨*proof*⟩

lemma (**in** *module*) *span-is-submodule*:
fixes *S*
assumes *h2*: $S \subseteq$ *carrier M*
shows *submodule R* (*span S*) *M*
⟨*proof*⟩

A finite sum does not depend on the ambient module. This can be done for monoid, but "submonoid" isn't currently defined. (It can be copied, however, for groups. . .) This lemma requires a somewhat annoying lemma foldD-not-depend. Then we show that linear combinations, linear independence, span do not depend on the ambient module.

lemma (**in** *module*) *finsum-not-depend*:
fixes *a A N*
assumes *h1*: *finite A* **and** *h2*: $A \subseteq N$ **and** *h3*: *submodule R N M* **and** *h4*: $f{:}A \rightarrow N$
shows $(\bigoplus_{(md\ N)} v \in A.\ f\ v) = (\bigoplus_M v \in A.\ f\ v)$
⟨*proof*⟩

lemma (**in** *module*) *lincomb-not-depend*:
fixes *a A N*
assumes *h1*: *finite A* **and** *h2*: $A \subseteq N$ **and** *h3*: *submodule R N M*

and *h4*: $a{:}A{\rightarrow}carrier\ R$
  shows *lincomb a A = module.lincomb (md N) a A*
⟨*proof*⟩

lemma (**in** *module*) *span-li-not-depend*:
  fixes *S N*
  assumes *h2*: $S{\subseteq}N$ **and** *h3*: *submodule R N M*
  shows *module.span R (md N) S = module.span R M S*
    and *module.lin-dep R (md N) S = module.lin-dep R M S*
⟨*proof*⟩

lemma (**in** *module*) *span-is-subset*:
  fixes *S N*
  assumes *h2*: $S{\subseteq}N$ **and** *h3*: *submodule R N M*
  shows *span* $S \subseteq N$
⟨*proof*⟩

lemma (**in** *module*) *span-is-subset2*:
  fixes *S*
  assumes *h2*: $S{\subseteq}carrier\ M$
  shows *span* $S \subseteq$ *carrier M*
⟨*proof*⟩

lemma (**in** *module*) *in-own-span*:
  fixes *S*
  assumes *inC*: $S{\subseteq}carrier\ M$
  shows $S \subseteq$ *span S*
⟨*proof*⟩

lemma (**in** *module*) *supset-ld-is-ld*:
  fixes *A B*
  assumes *ld*: *lin-dep A* **and** *sub*: $A \subseteq B$
  shows *lin-dep B*
⟨*proof*⟩

lemma (**in** *module*) *subset-li-is-li*:
  fixes *A B*
  assumes *li*: *lin-indpt A* **and** *sub*: $B \subseteq A$
  shows *lin-indpt B*
⟨*proof*⟩

lemma (**in** *mod-hom*) *hom-sum*:
  fixes *A B g*
  assumes *h2*: $A{\subseteq}carrier\ M$ **and** *h3*: $g{:}A{\rightarrow}carrier\ M$
  shows $f\ (\bigoplus_M a{\in}A.\ g\ a) = (\bigoplus_N a{\in}A.\ f\ (g\ a))$
⟨*proof*⟩

end

5 The direct sum of modules.

theory *SumSpaces*
imports *Main*
HOL−Algebra.Module
HOL−Algebra.Coset
RingModuleFacts
MonoidSums
FunctionLemmas
LinearCombinations
begin

We define the direct sum $M_1 \oplus M_2$ of 2 vector spaces as the set $M_1 \times M_2$ under componentwise addition and scalar multiplication.

definition *direct-sum*:: $('a,'b,\ 'd)$ *module-scheme* $\Rightarrow$ $('a,\ 'c,\ 'e)$ *module-scheme* $\Rightarrow$$('a,\ ('b\times'c))$ *module*
where *direct-sum M1 M2* = (|*carrier* = *carrier M1* × *carrier M2*,
mult = $(\lambda\ v\ w.\ (\mathbf{0}_{M1},\ \mathbf{0}_{M2}))$,
one = $(\mathbf{0}_{M1},\ \mathbf{0}_{M2})$,
zero = $(\mathbf{0}_{M1},\ \mathbf{0}_{M2})$,
add = $(\lambda\ v\ w.\ (fst\ v \oplus_{M1} fst\ w,\ snd\ v \oplus_{M2} snd\ w))$,
smult = $(\lambda\ c\ v.\ (c \odot_{M1} fst\ v,\ c \odot_{M2} snd\ v))$|)

lemma *direct-sum-is-module*:
fixes *R M1 M2*
assumes *h1*: *module R M1* **and** *h2*: *module R M2*
shows *module R* (*direct-sum M1 M2*)
⟨*proof*⟩

definition *inj1*:: $('a,'b)$ *module* $\Rightarrow$ $('a,\ 'c)$ *module* $\Rightarrow$$('b\Rightarrow('b\times'c))$
where *inj1 M1 M2* = $(\lambda v.\ (v,\ \mathbf{0}_{M2}))$

definition *inj2*:: $('a,'b)$ *module* $\Rightarrow$ $('a,\ 'c)$ *module* $\Rightarrow$$('c\Rightarrow('b\times'c))$
where *inj2 M1 M2* = $(\lambda v.\ (\mathbf{0}_{M1},\ v))$

lemma *inj1-hom*:
fixes *R M1 M2*
assumes *h1*: *module R M1* **and** *h2*: *module R M2*
shows *mod-hom R M1* (*direct-sum M1 M2*) (*inj1 M1 M2*)
⟨*proof*⟩

lemma *inj2-hom*:
fixes *R M1 M2*
assumes *h1*: *module R M1* **and** *h2*: *module R M2*
shows *mod-hom R M2* (*direct-sum M1 M2*) (*inj2 M1 M2*)

⟨*proof*⟩

For submodules $M_1, M_2 \subseteq M$, the map $M_1 \oplus M_2 \to M$ given by $(m_1, m_2) \mapsto m_1 + m_2$ is linear.

lemma (**in** *module*) *sum-map-hom*:
 fixes *M1 M2*
 assumes *h1*: *submodule R M1 M* **and** *h2*: *submodule R M2 M*
 shows *mod-hom R* (*direct-sum* (*md M1*) (*md M2*)) *M* (λ *v*. (*fst v*) $\oplus_M$ (*snd v*))
⟨*proof*⟩

lemma (**in** *module*) *sum-is-submodule*:
 fixes *N1 N2*
 assumes *h1*: *submodule R N1 M* **and** *h2*: *submodule R N2 M*
 shows *submodule R* (*submodule-sum N1 N2*) *M*
⟨*proof*⟩

lemma (**in** *module*) *in-sum*:
 fixes *N1 N2*
 assumes *h1*: *submodule R N1 M* **and** *h2*: *submodule R N2 M*
 shows *N1* ⊆ *submodule-sum N1 N2*
⟨*proof*⟩

lemma (**in** *module*) *msum-comm*:
 fixes *N1 N2*
 assumes *h1*: *submodule R N1 M* **and** *h2*: *submodule R N2 M*
 shows (*submodule-sum N1 N2*) = (*submodule-sum N2 N1*)
⟨*proof*⟩

If $M_1, M_2 \subseteq M$ are submodules, then $M_1 + M_2$ is the minimal subspace such that both $M_1 \subseteq M$ and $M_2 \subseteq M$.

lemma (**in** *module*) *sum-is-minimal*:
 fixes *N N1 N2*
 assumes *h1*: *submodule R N1 M* **and** *h2*: *submodule R N2 M* **and** *h3*: *submodule R N M*
 shows (*submodule-sum N1 N2*) ⊆ *N* ⟷ *N1* ⊆ *N* ∧ *N2* ⊆ *N*
⟨*proof*⟩

$\text{span} A \cup B = \text{span} A + \text{span} B$

lemma (**in** *module*) *span-union-is-sum*:
 fixes *A B*
 assumes *h2*: *A*⊆*carrier M* **and** *h3*: *B*⊆*carrier M*
 shows *span* (*A*∪ *B*) = *submodule-sum* (*span A*) (*span B*)
⟨*proof*⟩

end

6 Basic theory of vector spaces, using locales

theory *VectorSpace*
imports *Main*
  HOL−Algebra.Module
  HOL−Algebra.Coset
  RingModuleFacts
  MonoidSums
  LinearCombinations
  SumSpaces
begin

6.1 Basic definitions and facts carried over from modules

A *vectorspace* is a module where the ring is a field. Note that we switch notation from (R, M) to (K, V).

locale *vectorspace* =
  module?: *module K V* + *field?*: *field K*
  for *K* **and** *V*

A *subspace* of a vectorspace is a nonempty subset that is closed under addition and scalar multiplication. These properties have already been defined in submodule. Caution: W is a set, while V is a module record. To get W as a vectorspace, write vs W.

locale *subspace* =
  fixes *K* **and** *W* **and** *V* (**structure**)
  assumes *vs*: *vectorspace K V*
    and *submod*: *submodule K W V*

lemma (**in** *vectorspace*) *is-module*[*simp*]:
  subspace K W V$\Longrightarrow$*submodule K W V*
⟨*proof*⟩

We introduce some basic facts and definitions copied from module. We introduce some abbreviations, to match convention.

abbreviation (**in** *vectorspace*) *vs*::′*c set* ⇒ (′*a*, ′*c*, ′*d*) *module-scheme*
  where *vs W* ≡ *V*⦇*carrier* :=*W*⦈

lemma (**in** *vectorspace*) *carrier-vs-is-self* [*simp*]:
  carrier (*vs W*) = *W*
  ⟨*proof*⟩

lemma (**in** *vectorspace*) *subspace-is-vs*:
  fixes *W*::′*c set*

assumes *0*: *subspace K W V*
shows *vectorspace K* (*vs W*)
⟨*proof*⟩

abbreviation (**in** *module*) *subspace-sum*:: [*'c set*, *'c set*] ⇒ *'c set*
where *subspace-sum W1 W2* ≡*submodule-sum W1 W2*

lemma (**in** *vectorspace*) *vs-zero-lin-dep*:
assumes *h2*: *S*⊆*carrier V* **and** *h3*: *lin-indpt S*
shows $\mathbf{0}_V \notin S$
⟨*proof*⟩

A *linear-map* is a module homomorphism between 2 vectorspaces over the same field.

locale *linear-map* =
V?: *vectorspace K V* + *W?*: *vectorspace K W*
+ *mod-hom?*: *mod-hom K V W T*
for *K* **and** *V* **and** *W* **and** *T*

context *linear-map*
begin
lemmas *T-hom* = *f-hom*
lemmas *T-add* = *f-add*
lemmas *T-smult* = *f-smult*
lemmas *T-im* = *f-im*
lemmas *T-neg* = *f-neg*
lemmas *T-minus* = *f-minus*
lemmas *T-ker* = *f-ker*

abbreviation *imT*:: *'e set*
where *imT* ≡ *mod-hom.im*

abbreviation *kerT*:: *'c set*
where *kerT* ≡ *mod-hom.ker*

lemmas *T0-is-0*[*simp*] = *f0-is-0*

lemma *kerT-is-subspace*: *subspace K ker V*
⟨*proof*⟩

lemma *imT-is-subspace*: *subspace K imT W*
⟨*proof*⟩
end

lemma *vs-criteria*:
fixes *K* **and** *V*
assumes *field*: *field K*
and *zero*: $\mathbf{0}_V$∈ *carrier V*
and *add*: ∀ *v w*. *v*∈*carrier V* ∧ *w*∈*carrier V* ⟶ $v \oplus_V w$∈ *carrier*

V
and *neg*: $\forall$ *v*$\in$*carrier V*. ($\exists$ *neg-v*$\in$*carrier V*. $v\oplus_V$*neg-v*=$\mathbf{0}_V$)
and *smult*: $\forall$ *c v*. *c*$\in$ *carrier K* $\wedge$ *v*$\in$*carrier V* $\longrightarrow$ $c\odot_V v \in$ *carrier V*
and *comm*: $\forall$ *v w*. *v*$\in$*carrier V* $\wedge$ *w*$\in$*carrier V* $\longrightarrow$ $v\oplus_V w$=$w\oplus_V v$
and *assoc*: $\forall$ *v w x*. *v*$\in$*carrier V* $\wedge$ *w*$\in$*carrier V* $\wedge$ *x*$\in$*carrier V* $\longrightarrow$ $(v\oplus_V w)\oplus_V x$= $v\oplus_V (w\oplus_V x)$
and *add-id*: $\forall$ *v*$\in$*carrier V*. $(v\oplus_V \mathbf{0}_V = v)$
and *compat*: $\forall$ *a b v*. *a*$\in$ *carrier K* $\wedge$ *b*$\in$ *carrier K* $\wedge$ *v*$\in$*carrier V* $\longrightarrow$ $(a\otimes_K b)\odot_V v = a\odot_V (b\odot_V v)$
and *smult-id*: $\forall$ *v*$\in$*carrier V*. $(\mathbf{1}_K \odot_V v = v)$
and *dist-f*: $\forall$ *a b v*. *a*$\in$ *carrier K* $\wedge$ *b*$\in$ *carrier K* $\wedge$ *v*$\in$*carrier V* $\longrightarrow$ $(a\oplus_K b)\odot_V v = (a\odot_V v) \oplus_V (b\odot_V v)$
and *dist-add*: $\forall$ *a v w*. *a*$\in$ *carrier K* $\wedge$ *v*$\in$*carrier V* $\wedge$ *w*$\in$*carrier V* $\longrightarrow$ $a\odot_V (v\oplus_V w) = (a\odot_V v) \oplus_V (a\odot_V w)$
shows *vectorspace K V*
⟨*proof*⟩

For any set S, the space of functions $S \to K$ forms a vector space.

lemma (**in** *vectorspace*) *func-space-is-vs*:
fixes *S*
shows *vectorspace K* (*func-space S*)
⟨*proof*⟩

lemma *direct-sum-is-vs*:
fixes *K V1 V2*
assumes *h1*: *vectorspace K V1* **and** *h2*: *vectorspace K V2*
shows *vectorspace K* (*direct-sum V1 V2*)
⟨*proof*⟩

lemma *inj1-linear*:
fixes *K V1 V2*
assumes *h1*: *vectorspace K V1* **and** *h2*: *vectorspace K V2*
shows *linear-map K V1* (*direct-sum V1 V2*) (*inj1 V1 V2*)
⟨*proof*⟩

lemma *inj2-linear*:
fixes *K V1 V2*
assumes *h1*: *vectorspace K V1* **and** *h2*: *vectorspace K V2*
shows *linear-map K V2* (*direct-sum V1 V2*) (*inj2 V1 V2*)
⟨*proof*⟩

For subspaces $V_1, V_2 \subseteq V$, the map $V_1 \oplus V_2 \to V$ given by $(v_1, v_2) \mapsto v_1 + v_2$ is linear.

lemma (**in** *vectorspace*) *sum-map-linear*:
fixes *V1 V2*

assumes *h1*: *subspace K V1 V* **and** *h2*: *subspace K V2 V*
shows *linear-map K* (*direct-sum* (*vs V1*) (*vs V2*)) *V* (λ *v*. (*fst v*) $\oplus_V$ (*snd v*))
⟨*proof*⟩

lemma (**in** *vectorspace*) *sum-is-subspace*:
fixes *W1 W2*
assumes *h1*: *subspace K W1 V* **and** *h2*: *subspace K W2 V*
shows *subspace K* (*subspace-sum W1 W2*) *V*
⟨*proof*⟩

If $W_1, W_2 \subseteq V$ are subspaces, $W_1 \subseteq W_1 + W_2$

lemma (**in** *vectorspace*) *in-sum-vs*:
fixes *W1 W2*
assumes *h1*: *subspace K W1 V* **and** *h2*: *subspace K W2 V*
shows *W1* $\subseteq$ *subspace-sum W1 W2*
⟨*proof*⟩

lemma (**in** *vectorspace*) *vsum-comm*:
fixes *W1 W2*
assumes *h1*: *subspace K W1 V* **and** *h2*: *subspace K W2 V*
shows (*subspace-sum W1 W2*) = (*subspace-sum W2 W1*)
⟨*proof*⟩

If $W_1, W_2 \subseteq V$ are subspaces, then $W_1 + W_2$ is the minimal subspace such that both $W_1 \subseteq W$ and $W_2 \subseteq W$.

lemma (**in** *vectorspace*) *vsum-is-minimal*:
fixes *W W1 W2*
assumes *h1*: *subspace K W1 V* **and** *h2*: *subspace K W2 V* **and** *h3*: *subspace K W V*
shows (*subspace-sum W1 W2*) $\subseteq$ *W* $\longleftrightarrow$ *W1* $\subseteq$ *W* $\wedge$ *W2* $\subseteq$ *W*
⟨*proof*⟩

lemma (**in** *vectorspace*) *span-is-subspace*:
fixes *S*
assumes *h2*: *S*$\subseteq$*carrier V*
shows *subspace K* (*span S*) *V*
⟨*proof*⟩

6.1.1 Facts specific to vector spaces

If $av = w$ and $a \neq 0$, $v = a^{-1}w$.

lemma (**in** *vectorspace*) *mult-inverse*:
assumes *h1*: *a*$\in$*carrier K* **and** *h2*: *v*$\in$*carrier V* **and** *h3*: *a* $\odot_V$ *v* = *w* **and** *h4*: *a*$\neq\mathbf{0}_K$
shows *v* = (inv_K *a*)$\odot_V$ *w*
⟨*proof*⟩

If $w \in S$ and $\sum_{w \in S} a_w w = 0$, we have $v = \sum_{w \notin S} a_v^{-1} a_w w$

lemma (**in** *vectorspace*) *lincomb-isolate*:
fixes *A v*
assumes *h1*: *finite A* **and** *h2*: *A⊆carrier V* **and** *h3*: *a∈A→carrier K* **and** *h4*: *v∈A*
and *h5*: $a\ v \neq \mathbf{0}_K$ **and** *h6*: *lincomb a A*=$\mathbf{0}_V$
shows *v=lincomb* $(\lambda w.\ \ominus_K(inv_K\ (a\ v)) \otimes_K a\ w)$ $(A-\{v\})$ **and** *v∈ span* $(A-\{v\})$
⟨*proof*⟩

The map $(S \to K) \mapsto V$ given by $(a_v)_{v \in S} \mapsto \sum_{v \in S} a_v v$ is linear.

lemma (**in** *vectorspace*) *lincomb-is-linear*:
fixes *S*
assumes *h*: *finite S* **and** *h2*: *S⊆carrier V*
shows *linear-map K* (*func-space S*) *V* (λ*a. lincomb a S*)
⟨*proof*⟩

6.2 Basic facts about span and linear independence

If S is linearly independent, then $v \in \text{span} S$ iff $S \cup \{v\}$ is linearly dependent.

theorem (**in** *vectorspace*) *lin-dep-iff-in-span*:
fixes *A v S*
assumes *h1*: *S ⊆ carrier V* **and** *h2*: *lin-indpt S* **and** *h3*: *v∈ carrier V* **and** *h4*: *v∉S*
shows *v∈ span S* ⟷ *lin-dep* $(S \cup \{v\})$
⟨*proof*⟩

If $v \in \text{span} A$ then $\text{span} A = \text{span}(A \cup \{v\})$

lemma (**in** *vectorspace*) *already-in-span*:
fixes *v A*
assumes *inC*: *A⊆carrier V* **and** *inspan*: *v∈span A*
shows *span A= span* $(A \cup \{v\})$
⟨*proof*⟩

6.3 The Replacement Theorem

If $A, B \subseteq V$ are finite, A is linearly independent, B generates W, and $A \subseteq W$, then there exists $C \subseteq V$ disjoint from A such that $\text{span}(A \cup C) = W$ and $|C| \leq |B| - |A|$. In other words, we can complete any linearly independent set to a generating set of W by adding at most $|B| - |A|$ more elements.

theorem (**in** *vectorspace*) *replacement*:
fixes *A B*
assumes *h1*: *finite A*
and *h2*: *finite B*

and *h3*: $B \subseteq$ *carrier V*
and *h4*: *lin-indpt A*
and *h5*: $A \subseteq$ *span B*
shows $\exists C$. *finite C* $\wedge$ $C \subseteq$ *carrier V* $\wedge$ $C \subseteq$ *span B* $\wedge$ $C \cap A = \{\}$ $\wedge$ *int* (*card C*) $\leq$ (*int* (*card B*)) $-$ (*int* (*card A*)) $\wedge$ (*span* ($A \cup C$) $=$ *span B*)
(**is** $\exists C$. *?P A B C*)

⟨*proof*⟩

6.4 Defining dimension and bases.

Finite dimensional is defined as having a finite generating set.

definition (**in** *vectorspace*) *fin-dim*:: *bool*
where *fin-dim* = ($\exists$ *A*. ((*finite A*) $\wedge$ ($A \subseteq$ *carrier V*) $\wedge$ (*gen-set A*)))

The dimension is the size of the smallest generating set. For equivalent characterizations see below.

definition (**in** *vectorspace*) *dim*:: *nat*
where *dim* = (*LEAST n*. ($\exists$ *A*. ((*finite A*) $\wedge$ (*card A* = *n*) $\wedge$ ($A \subseteq$ *carrier V*) $\wedge$ (*gen-set A*))))

A *basis* is a linearly independent generating set.

definition (**in** *vectorspace*) *basis*:: *'c set* $\Rightarrow$ *bool*
where *basis A* = ((*lin-indpt A*) $\wedge$ (*gen-set A*)$\wedge$ ($A \subseteq$ *carrier V*))

From the replacement theorem, any linearly independent set is smaller than any generating set.

lemma (**in** *vectorspace*) *li-smaller-than-gen*:
fixes *A B*
assumes *h1*: *finite A* **and** *h2*: *finite B* **and** *h3*: $A \subseteq$ *carrier V* **and** *h4*: $B \subseteq$ *carrier V*
and *h5*: *lin-indpt A* **and** *h6*: *gen-set B*
shows *card A* $\leq$ *card B*
⟨*proof*⟩

The dimension is the cardinality of any basis. (In particular, all bases are the same size.)

lemma (**in** *vectorspace*) *dim-basis*:
fixes *A*
assumes *fin*: *finite A* **and** *h2*: *basis A*
shows *dim* = *card A*
⟨*proof*⟩

A *maximal* set with respect to P is such that if $B \supseteq A$ and P is also satisfied for B, then $B = A$.

definition *maximal*::$'a$ *set* $\Rightarrow$ ($'a$ *set* $\Rightarrow$ *bool*) $\Rightarrow$ *bool*
  where *maximal* A P = $((P\ A) \wedge (\forall B.\ B \supseteq A \wedge P\ B \longrightarrow B = A))$

A *minimal* set with respect to P is such that if $B \subseteq A$ and P is also satisfied for B, then $B = A$.

definition *minimal*::$'a$ *set* $\Rightarrow$ ($'a$ *set* $\Rightarrow$ *bool*) $\Rightarrow$ *bool*
  where *minimal* A P = $((P\ A) \wedge (\forall B.\ B \subseteq A \wedge P\ B \longrightarrow B = A))$

A maximal linearly independent set is a generating set.

lemma (**in** *vectorspace*) *max-li-is-gen*:
  fixes A
  assumes *h1*: *maximal* A ($\lambda S.\ S \subseteq$ *carrier* V $\wedge$ *lin-indpt* S)
  shows *gen-set* A
⟨*proof*⟩

A minimal generating set is linearly independent.

lemma (**in** *vectorspace*) *min-gen-is-li*:
  fixes A
  assumes *h1*: *minimal* A ($\lambda S.\ S \subseteq$ *carrier* V $\wedge$ *gen-set* S)
  shows *lin-indpt* A
⟨*proof*⟩

Given that some finite set satisfies P, there is a minimal set that satisfies P.

lemma *minimal-exists*:
  fixes A P
  assumes *h1*: *finite* A **and** *h2*: P A
  shows $\exists B.\ B \subseteq A \wedge$ *minimal* B P
⟨*proof*⟩

If V is finite-dimensional, then any linearly independent set is finite.

lemma (**in** *vectorspace*) *fin-dim-li-fin*:
  assumes *fd*: *fin-dim* **and** *li*: *lin-indpt* A **and** *inC*: $A \subseteq$ *carrier* V
  shows *fin*: *finite* A
⟨*proof*⟩

If V is finite-dimensional (has a finite generating set), then a finite basis exists.

lemma (**in** *vectorspace*) *finite-basis-exists*:
  assumes *h1*: *fin-dim*
  shows $\exists \beta.$ *finite* β $\wedge$ *basis* β
⟨*proof*⟩

The proof is as follows.

1. Because V is finite-dimensional, there is a finite generating set (we took this as our definition of finite-dimensional).

2. Hence, there is a minimal $\beta \subseteq A$ such that β generates V.
3. β is linearly independent because a minimal generating set is linearly independent.

Finally, β is a basis because it is both generating and linearly independent.

Any linearly independent set has cardinality at most equal to the dimension.

lemma (**in** *vectorspace*) *li-le-dim*:
  fixes A
  assumes *fd*: *fin-dim* **and** *c*: $A \subseteq$ *carrier* V **and** *l*: *lin-indpt* A
  shows *finite* A *card* $A \leq$ *dim*
$\langle proof \rangle$

Any generating set has cardinality at least equal to the dimension.

lemma (**in** *vectorspace*) *gen-ge-dim*:
  fixes A
  assumes *fa*: *finite* A **and** *c*: $A \subseteq$ *carrier* V **and** *l*: *gen-set* A
  shows *card* $A \geq$ *dim*
$\langle proof \rangle$

If there is an upper bound on the cardinality of sets satisfying P, then there is a maximal set satisfying P.

lemma *maximal-exists*:
  fixes P B N
  assumes *maxc*: $\bigwedge A.\ P\ A \Longrightarrow$ *finite* $A \wedge$ (*card* $A \leq N$) **and** *b*: $P\ B$
  shows $\exists A.$ *finite* $A \wedge$ *maximal* A P
$\langle proof \rangle$

Any maximal linearly independent set is a basis.

lemma (**in** *vectorspace*) *max-li-is-basis*:
  fixes A
  assumes *h1*: *maximal* A ($\lambda S.\ S \subseteq$ *carrier* $V \wedge$ *lin-indpt* S)
  shows *basis* A
$\langle proof \rangle$

Any minimal linearly independent set is a generating set.

lemma (**in** *vectorspace*) *min-gen-is-basis*:
  fixes A
  assumes *h1*: *minimal* A ($\lambda S.\ S \subseteq$ *carrier* $V \wedge$ *gen-set* S)
  shows *basis* A
$\langle proof \rangle$

Any linearly independent set with cardinality at least the dimension is a basis.

lemma (**in** *vectorspace*) *dim-li-is-basis*:
fixes *A*
assumes *fd*: *fin-dim* **and** *fa*: *finite A* **and** *ca*: *A⊆carrier V* **and** *li*: *lin-indpt A*
and *d*: *card A* ≥ *dim*
shows *basis A*
⟨*proof*⟩

Any generating set with cardinality at most the dimension is a basis.

lemma (**in** *vectorspace*) *dim-gen-is-basis*:
fixes *A*
assumes *fa*: *finite A* **and** *ca*: *A⊆carrier V* **and** *li*: *gen-set A*
and *d*: *card A* ≤ *dim*
shows *basis A*
⟨*proof*⟩

β is a basis iff for all $v \in V$, there exists a unique $(a_v)_{v\in S}$ such that $\sum_{v\in S} a_v v = v$.

lemma (**in** *vectorspace*) *basis-criterion*:
assumes *A-fin*: *finite A* **and** *AinC*: *A⊆carrier V*
shows *basis A* ⟷ (∀ *v*. *v*∈ *carrier V* ⟶(∃! *a*. *a*∈*A* $\rightarrow_E$ *carrier K* ∧ *lincomb a A* = *v*))
⟨*proof*⟩

lemma (**in** *linear-map*) *surj-imp-imT-carrier*:
assumes *surj*: *T*‘ (*carrier V*) = *carrier W*
shows (*imT*) = *carrier W*
⟨*proof*⟩

6.5 The rank-nullity (dimension) theorem

If V is finite-dimensional and $T : V \to W$ is a linear map, then $\dim(\mathrm{im}(T)) + \dim(\ker(T)) = \dim V$. Moreover, we prove that if T is surjective linear-map between V and W, where V is finite-dimensional, then also W is finite-dimensional.

theorem (**in** *linear-map*) *rank-nullity-main*:
assumes *fd*: *V.fin-dim*
shows (*vectorspace.dim K* (*W.vs imT*)) + (*vectorspace.dim K* (*V.vs kerT*)) = *V.dim*
T ‘ (*carrier V*) = *carrier W* ⟹ *W.fin-dim*
⟨*proof*⟩

theorem (**in** *linear-map*) *rank-nullity*:
assumes *fd*: *V.fin-dim*
shows (*vectorspace.dim K* (*W.vs imT*)) + (*vectorspace.dim K* (*V.vs kerT*)) = *V.dim*
⟨*proof*⟩

end