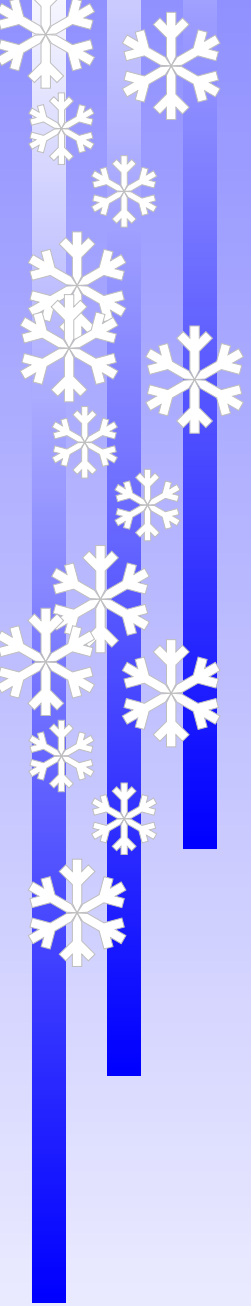


Subsets in formalised linear algebra

Jasper Stein

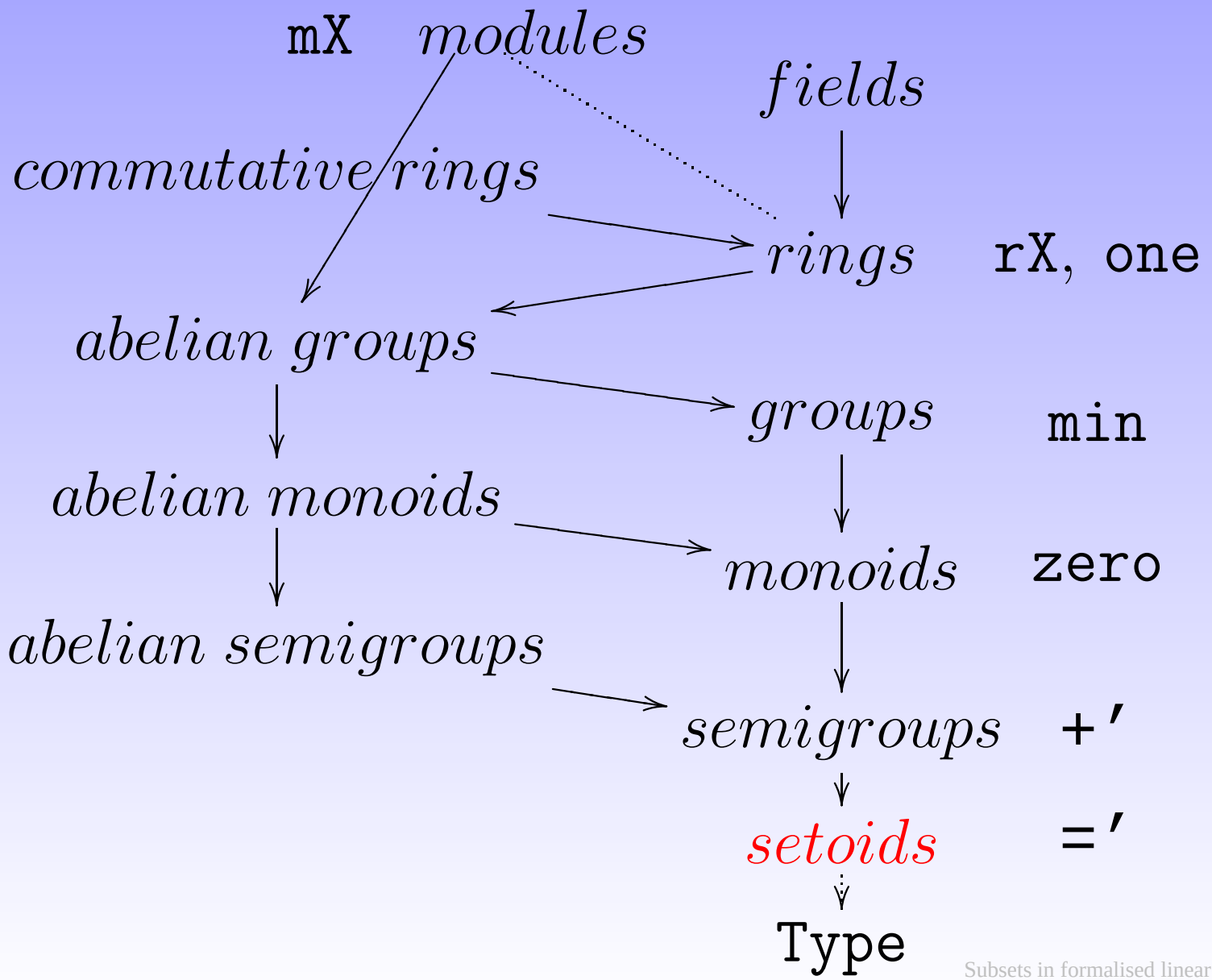
Contents

- ✓ Introduction
- ✓ The setoid formalism
- ✓ Up and down the subset ladder
- ✓ Across the subset ladder
- ✓ Conclusions

Introduction

- ✓ LinAlg: linear algebra in Coq
- ✓ vectorspaces, subspaces, linear combinations, linear (in)dependence, bases and dimension
- ✓ Book: *Linear Algebra*, 2nd ed. (chapter 1) by Friedberg, Insel and Spence
- ✓ Extends Loïc Pottier's Algebra

The setoid formalism from Algebra



The setoid formalism (contd.)

- ✓ A *setoid* is a triple $\mathtt{A} := \langle \mathtt{X}, =', \mathtt{p_A} \rangle$ where $\mathtt{p_A}$ proves that $='$ is an equivalence relation on X.
- ✓ *(setoid-)compatibility*:

  $\mathtt{a} =' \mathtt{a'}$ `in` $\mathtt{A} \rightarrow$

  - ⋆ $(\mathtt{f\ a}) =' (\mathtt{f\ a'})$ `in B`
  - ⋆ $(\mathtt{P\ a}) \iff (\mathtt{P\ a'})$
- ✓ Use this type for functions:

```
Record Map (A B:Setoid) :Type :=
{ Ap :> A -> B ;
  prf :  a =' a' -> Ap a =' Ap a'}
```

- ✓ Similarly `Predicate A`

Subsets(1)

The setoid `part_set A` has:

- ✓ carrier `Predicate A`
- ✓ equality $\mathtt{P} =' \mathtt{Q} \overset{\text{def}}{\iff} [\forall \mathtt{a} : \mathtt{A}.(\mathtt{P\ a}) \iff (\mathtt{Q\ a})]$

Every such Predicate `P : (part_set A)` induces:

```
Record subtype (P : part_set A) :=
  { subtype_elt : A
  ; subtype_prf : P subtype_elt }
```

so essentially $\mathtt{P} \rightsquigarrow \Sigma\, \mathtt{a} : \mathtt{A}.\ \mathtt{P(a)}$

and `subtype_elt` $\cong \pi_1$

Subsets (2)

Any predicate `B : part_set A` is coerced to a (sub)setoid `B : Setoid`, with:

✓ carrier `subtype B` $(\cong \Sigma a : A.\ (B\ a))$

✓ equality `x =' x' in B` $\overset{\text{def}}{\iff}$

$(\texttt{subtype\_elt x}) =' (\texttt{subtype\_elt x'})$ `in A`

(i.e. $\langle a, H_a \rangle =' \langle a', H_{a'} \rangle \overset{\text{def}}{\iff} a =' a'$)

Coercions make the following well-typed:

```
A : Setoid ;        a : A

B : part_set A ;    x : B
```

whence $x \equiv \langle a, H_a \rangle$ with $a : A;\ H_a : P\ a$

Dealing with subsets

✓ type theory:

$$\texttt{B} : (\texttt{part\_set A}) \iff \texttt{B} : \texttt{Setoid} \cong \langle \Sigma\ \texttt{a} : \texttt{A}.\ \texttt{P(a)},\ \ldots,\ \ldots \rangle$$

$$\texttt{x} : \texttt{B} \iff \texttt{x} \cong \langle \texttt{a}, \texttt{H}_\texttt{a} \rangle$$

$$\texttt{x} : \texttt{B} \quad \texttt{f} : \texttt{A} \to \texttt{Z} \rightsquigarrow (\texttt{f x}) \quad \text{↯↯↯}$$

✓ Compare with set theory:

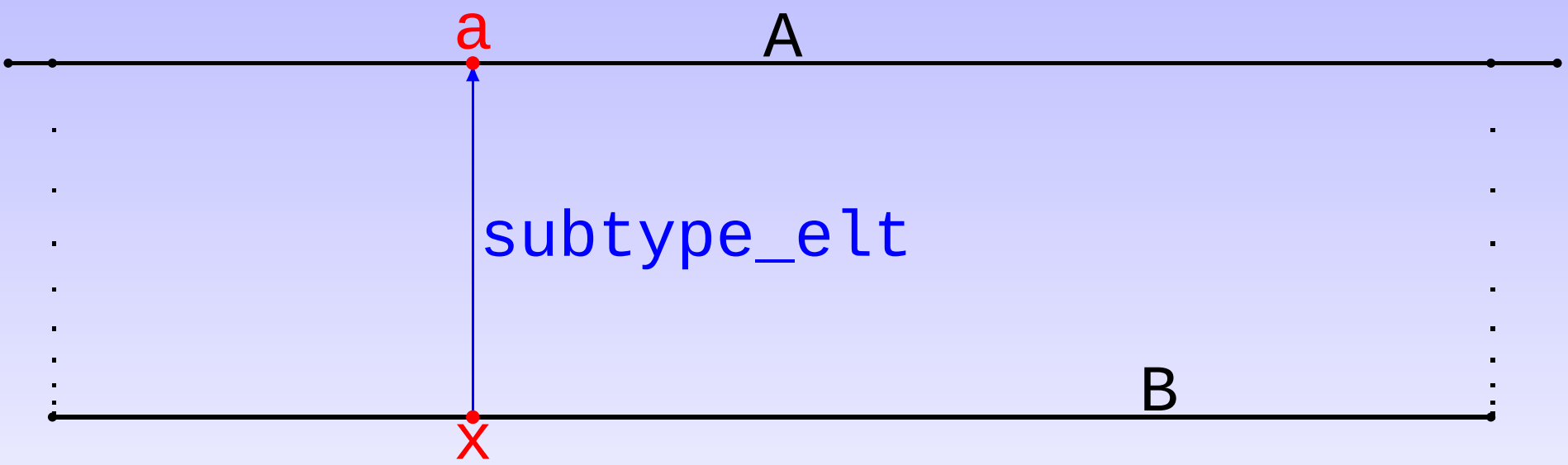
$$B \subset A \iff \forall x \in B . x \in A$$

$$a \in B \quad f : A \to Z \rightsquigarrow f(a) \in Z \quad OK$$

We need `subtype_elt` $\cong \pi_1$ as a coercion

Dealing with subsets

- ✓ $\mathtt{x} \equiv \langle \mathtt{a}, \mathtt{H_a} \rangle$
- ✓ `x` and `suptype_elt x` both represent a
- ✓ `subtype_elt x` $\equiv$ `a` lives "one level higher":



Map_embed

To say "X *is linearly dependent*" we need:

$X \subset V$	`X : part_set V`
$\langle a_0, \ldots, a_{n-1} \rangle \in \mathbb{F}^n$	`a : (seq n F)`
$\langle v_0, \ldots, v_{n-1} \rangle \in X^n$	`v : (seq n X)`

for which

$$\sum_{i<n} a_i \cdot v_i = 0_V \qquad OK$$

however:

```
(sum (mult_by_scalars a v))
```

Map_embed

`X : (part_set V)` is an *ordinary setoid*:

- ✓ `a : (seq n F); v : (seq n X)` is OK, but
- ✓ no multiplication (a_i `mX` v_i is undefined)
- ✓ no addition (even "just" `sum v` is undefined)
- ✓ we must apply `subtype_elt` to each v_i

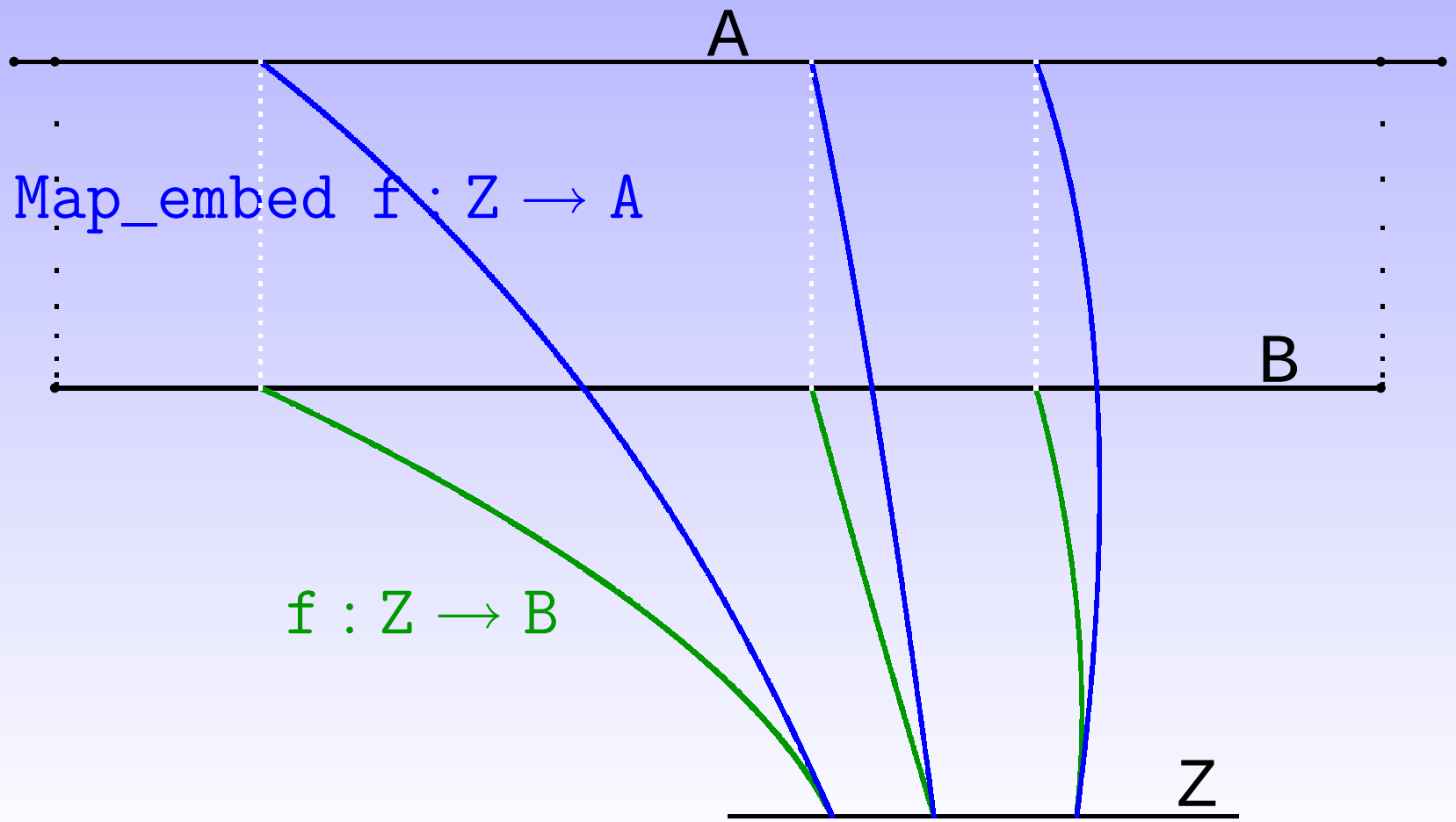
More generally, whenever `B:(part_set A)`,

`Map_embed` : `(Map Z B)` $\longrightarrow$ `(Map Z A)`

`(sum (mult_by_scalars a (Map_embed v)))` OK

Map_embed

- ✓ Both `f` and `Map_embed f` represent f
- ✓ `Map_embed f` lives "one level higher"



inject_subsets

Theorem 1.9: let $W \subset V$*. If* W *generates* V*, then* $\exists W_0 \subset W$ *that is a basis for* V

```
Definition is_basis (X : part_set V) :=
 generates X (full V) ∧ lin_indep X
```

```
W : (part_set V)
W₀ : (part_set W)
(is_basis W₀)
```

`W₀` lives 2 levels below `V`, not 1.

inject_subsets

$$\texttt{w} : \texttt{W}_0 \rightsquigarrow (\texttt{subtype\_elt w}) : \texttt{W}$$
$$\rightsquigarrow (\texttt{subtype\_elt (subtype\_elt w))} : \texttt{V}$$

These form a subset of V corresponding to $\texttt{W}_0$.

Generally, for $\texttt{B} : (\texttt{part\_set A})$,

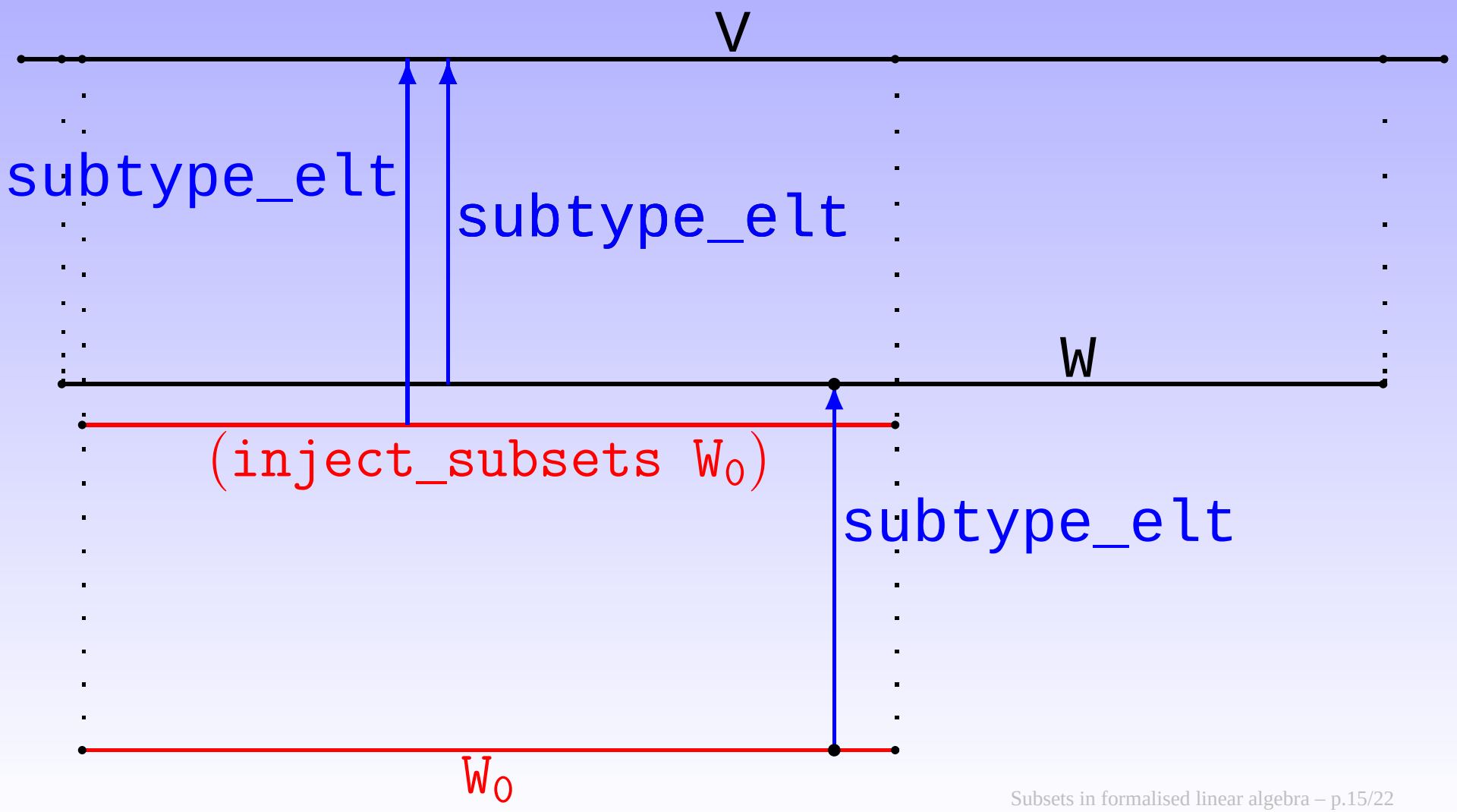
$$\texttt{inject\_subsets} :$$
$$(\texttt{part\_set B}) \longrightarrow (\texttt{part\_set A})$$

(inject_subsets C) is the Predicate based on

$$\lambda \texttt{a} : \texttt{A}.\, [\exists \texttt{c} : \texttt{C}.\, \texttt{a} =' (\texttt{subtype\_elt (subtype\_elt c))}]$$

inject_subsets

`W0` and `inject_subsets W0` both represent W_0
`inject_subsets W0` lives "one level higher":



(un) inject_subsetsify

Let W be a subspace of V. Suppose $X \subset W$.
Then X is linearly dependent in W iff it is in V.

```
Lemma inject_subsets_lin_dep :
    forall (W : subspace V) (X : part_set W),
lin_dep X  <->  lin_dep (inject_subsets X).
```

```
lin_dep X ~>   v : (seq n          X          )
   we need :   v : (seq n (inject_subsets X))
```

(inject_subsets X) represents the same X as X, one level higher.

(un) inject_subsetsify

- ✓ we have `v` : (`seq n X`)

(*un*)**inject_subsetsify**

- ✓ we have `v` : (`seq n X`)
- ✓ `v` : (`seq n` (`inject_subsets X`)) needed.

(*un*)**inject_subsetsify**

- ✓ we have `v` : (`seq n X`)
- ✓ `v` : (`seq n` (`inject_subsets X`)) needed.
- ✓ (`inject_subsets X`) is one level above `X`.

(un) inject_subsetsify

- ✓ we have `v : (seq n X)`
- ✓ `v : (seq n (inject_subsets X))` needed.
- ✓ `(inject_subsets X)` is one level above `X`.
- ✓ `Map_embed` lifts sequences one level.

(un) inject_subsetsify

- ✓ we have `v : (seq n X)`
- ✓ `v : (seq n (inject_subsets X))` needed.
- ✓ `(inject_subsets X)` is one level above `X`.
- ✓ `Map_embed` lifts sequences one level.
- ✓ `Map_embed v : (seq n W)` *wrong!*

(un) inject_subsetsify

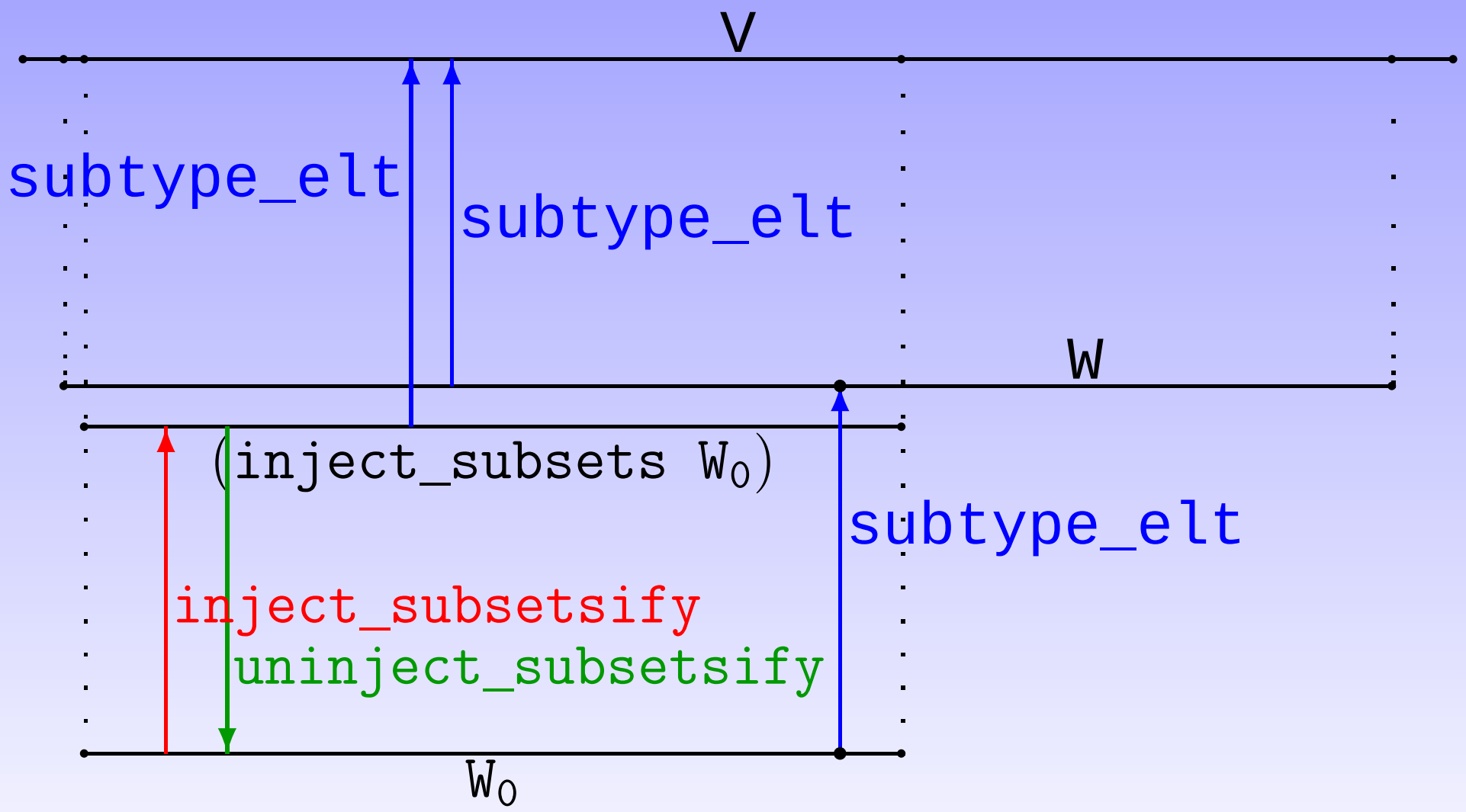
- ✓ we have `v` : (`seq n X`)
- ✓ `v` : (`seq n (inject_subsets X)`) needed.
- ✓ (`inject_subsets X`) is one level above `X`.
- ✓ `Map_embed` lifts sequences one level.
- ✓ `Map_embed v` : (`seq n W`) *wrong!*

new function:

`inject_subsetsify` : $W_0 \longrightarrow$ (`inject_subsets` W_0)

plus its inverse `uninject_subsetsify`

*(un)*inject_subsetsify



map_{between|to}_equal_subsets

Subsets may be given in different guises:

- ✓ $\mathtt{X}_1 := (\mathtt{span}\ \mathtt{W})$ v. $\mathtt{X}_2 := (\mathtt{span}\ (\mathtt{span}\ \mathtt{W}))$
- ✓ $\mathtt{X}_1 := \{\mathtt{a}, \mathtt{b}, \mathtt{c}\}$ v. $\mathtt{X}_2 :=$ the set of $\langle \mathtt{a}, \mathtt{c}, \mathtt{c}, \mathtt{b}, \mathtt{a}, \mathtt{b} \rangle$

Corresponding `Predicates` are different too!

$$\mathtt{x} : \mathtt{X}_1 \equiv \langle \mathtt{a}, \mathtt{H}_\mathtt{a} \rangle \qquad \mathtt{x} : \mathtt{X}_2 \equiv \langle \mathtt{a}, \mathtt{H}'_\mathtt{a} \rangle$$

where $\mathtt{H}_\mathtt{a}$ and $\mathtt{H}'_\mathtt{a}$ have different types.

$$\mathtt{map\_between\_equal\_subsets} : \mathtt{X}_1 =' \mathtt{X}_2 \to \mathtt{X}_1 \to \mathtt{X}_2$$

$$\mathtt{map\_to\_equal\_subsets} : \mathtt{X}_1 =' \mathtt{X}_2 \to (\mathtt{Map\ Z\ X}_1) \to (\mathtt{Map\ Z\ X}_2)$$

Map_include

Lemma: suppose $X_1 \subset X_2$ *both are subsets of* V*. Then* $span(X_1) \subset span(X_2)$

From `w : (span (inject_subsets X1))` we get

`a : (seq n F); v : (seq n (inject_subsets X1))`

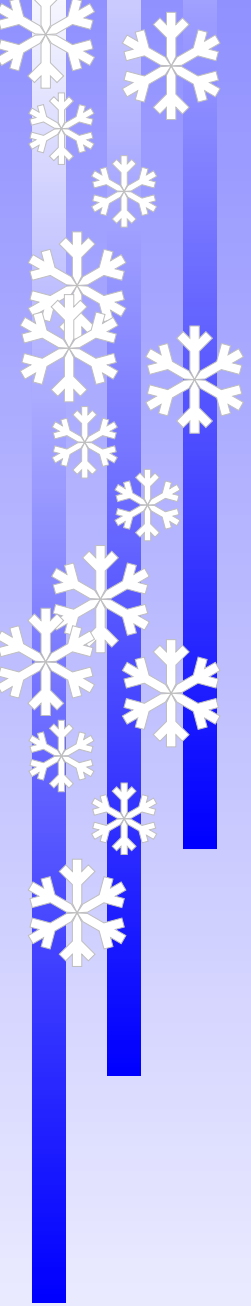
that we must turn into

`a : (seq n F); v : (seq n X2)`

```
Map_include : (included X1 X2) →
                    (Map Z X1) → (Map Z X2)
```

Conclusions

- ✓ In math, vectors can be members of several subsets besides the space itself
- ✓ Due to Coq's unique typing, the setoid formalism can accomodate subsets only so well
- ✓ Mathematical objects (vectors, sequences, ...) need several representations
- ✓ In LinAlg I used 8 different "pure type-cast" operations
- ✓ Even the subset relation $\subset$ is formalised in 2 different ways



The End

Thank you!

Questions?