- **(State spaces)**
  def (S, A) is state space
  $:\leftrightarrow$ (i) S, A are countable sets;
  (ii) $A \subseteq (S \to S)$;
  (iii) $\mathrm{id} \in A$;
  (iv) forall $f, g \in A$, $g \circ f \in A$.
  -- An element $s \in S$ is called a "(valid) state", while $f \in A$ is a "(valid) action".
  -- Conditions (iii) and (iv) together say that "zero or more valid actions performed sequentially is still a valid action".
  -- For example, $S = \mathbb{N}$ and $A = \{n \mapsto n + i \mid i \in \mathbb{N}\}$ (trivially bijective to $\mathbb{N}$) form the state space of a simple counting app.
- **(Products of state spaces)**
  (S, A) is state space,
  (T, B) is state space,
  def $(S, A) \times (T, B) := (\{(s, t) \mid s \in S,\ t \in T\}, \{(s, t) \mapsto (f(s), g(t)) \mid f \in A,\ g \in B\})$.
  $\Rightarrow$ $(S, A) \times (T, B)$ is state space.
  -- Check axioms.
  -- Corollary: for any finite index set I, if (forall $i \in I$, (S_i, A_i) is state space), then so is the iterated product $\prod_{i \in I}(S_i, A_i)$.
  -- Remark: this simply means that it is possible to "decompose" a state space into a product of independent parts.
- **(Joinability)**
  (S, A) is state space,
  def (S, A) is joinable by $\wedge : S \times S \to S$
  $:\leftrightarrow$ exists partial order $\leq$ on S such that
  (i) $(S, \leq)$ is semilattice;
  (ii) forall $s \in S$ and $f \in A$, $s \leq f(s)$;
  (iii) forall $s, t \in S$, $s \wedge t$ is the join (least upper bound) of s and t.
  -- Notation: if there is no ambiguity, I will simply say "(S, A) is joinable (state space)" and use "$\wedge$" for the join operation.
  -- Remark: the join operation "$\wedge$" takes whole s as input. In practice, we seldom want to send the whole application state s over network...
- **(Products of joinable state spaces)**
  (S, A) is joinable,
  (T, B) is joinable,
  $\Rightarrow$ $(S, A) \times (T, B)$ is joinable by $(s_1, t_1), (s_2, t_2) \mapsto (s_1 \wedge s_2, t_1 \wedge t_2)$.
  -- Check axioms (using partial order $(s_1, t_1) \leq (s_2, t_2) :\leftrightarrow s_1 \leq s_2$ and $t_1 \leq t_2$).
  -- Corollary: for any finite index set I, if (forall $i \in I$, (S_i, A_i) is joinable), then so is the iterated product $\prod_{i \in I}(S_i, A_i)$.
  -- Remark: to join is to join independent parts separately.

- **(Delta- and gamma-joinability)**

  (S, A) is joinable,

  def (S, A) is delta-joinable by $\Delta : S \times A \times A \to S$

  $:\leftrightarrow$ forall $s \in S$ and $f, g \in A$, $\Delta(s, f, g) = f(s) \wedge g(s)$.

  -- "Three-way merge" using common ancestor and changes.

  -- For many data structures, such Δ can be implemented more efficiently than $\wedge$. However, this will require storing the state snapshot s in some form.

  def (S, A) is gamma-joinable by $\Gamma : S \times A \to S$

  $:\leftrightarrow$ forall $s \in S$ and $f, g \in A$, $\Gamma(f(s), g) = f(s) \wedge g(s)$.

  -- "Asymmetric merge" using "our" state and "their" changes.

  -- For some data structures, such Γ is possible to implement. If this is true, then there is no need to retain older state snapshots. A history of actions is still needed.

  -- Remark: it is easy to see that joinability implies delta- and gamma-joinability (simply implement Δ and Γ by first applying changes and then joining), so the two definitions are mathematically "meaningless". Practically, however, it is possible to have more efficient direct implementations for Δ and Γ. (In mathematics, we think "extensionally" equal functions to be "the same"; in programming, it makes sense to consider their "intensional" difference.)

- **(LWW-registers are gamma-joinable)**

  X is totally ordered set, -- The set of possible values.

  $S := (\mathbb{R} \cup \{-\infty, +\infty\}) \times X$, -- The set of timestamped values.

  $A := \{s \mapsto \max\{s, (t, x)\} \mid t \in \mathbb{R} \cup \{-\infty, +\infty\},\ x \in X\}$, -- The set of timestamped actions.

  def $\mathrm{Reg}(X)$ $:=$ (S, A).

  $\Rightarrow$ $\mathrm{Reg}(X)$ is gamma-joinable by $(s, f) \mapsto f(s)$.

  -- Easy to check (using the "natural" total order on S).

- **(LWW-graphs are gamma-joinable)**

  V, E are finite sets, -- Index sets of vertices and edges.

  X, Y are totally ordered sets, -- Sets of possible values on vertices and edges.

  def $\mathrm{Graph}(V, E, X, Y)$ $:=$ $\prod_{v \in V} \mathrm{Reg}(X \cup \{\perp\}) \times \prod_{e \in E} \mathrm{Reg}(Y \cup \{\perp\})$.

  $\Rightarrow$ $\mathrm{Graph}(V, E, X, Y)$ is gamma-joinable.

  -- By previous results (products of joinable state spaces are joinable, so are gamma-joinable; although more efficient implementation exists).

  -- Remark: a LWW-graph is just a product of many LWW-registers. A special value $\perp$ is used to indicate absence of a particular vertex or edge. In case an edge has a normal value but one of its endpoints has value $\perp$ (i.e. marked as absent), the edge is disregarded.