Pairwise linked subsets of a closure space

`ClosureSpace.IsLinkedSet`

def

Defines subsets whose every pair of points is linked in the ambient closure space.

Statement

Let (X, c) be a closure space and let $S \subseteq X$. The linked-set predicate requires every two points of S to be linked with respect to the ambient closure c. It includes the empty set and singletons.

Definition

A set $A \subseteq X$ is closed when $c(A) = A$. The ambient linked relation is

$$x \sim_c y \quad \Longleftrightarrow \quad x = y \text{ or } c(\{x, y\}) \neq \{x, y\}.$$

Thus the linked-set condition is

$$\forall x \in S,\ \forall y \in S, \qquad x \sim_c y.$$

Both points are chosen from the same set S, while the closure of their pair is taken in the given ambient space X. Distinct points must form a nonclosed two-point set. When the two points coincide, the equality alternative already makes them linked.

This is a direct pairwise condition. The separate relation of belonging to the same component is generated by finite chains of linked pairs. A chain may pass through intermediate points; it does not replace the pairwise test in this definition.

Assumptions

The inputs are the closure space (X, c) and an arbitrary subset S. Nonemptiness, two or more elements, closedness and maximality are not required. No finite-character or finite-union property is imposed on the closure.

Conclusion

The definition returns a proposition about all pairs in S, with their membership conditions explicitly quantified. It neither chooses a distinguished point nor asserts that S is a whole component or a maximal linked subset.

The cited empty-set and singleton lemmas are separate consequences, not extra clauses of the definition. For the empty set there is no $x \in S$ to test. For $S = \{a\}$, any two members both equal a, so the equality alternative of the linked relation applies. These two arguments explain why the

definition admits these cases without a nonemptiness convention.

Main citations

- Linked-set predicate: the complete definition, lines 30–31.
- The ambient linked relation and its reflexivity, lines 30–32; see also the existing Linked points Card.
- Empty and singleton suppliers, lines 34–41, for the two separate consequences explained above.
- Closedness as a closure equality, lines 60–61.
- The generated relation and its components, lines 48–52 and 67–68; the existing Closure components Card treats this different notion.

Lean source signature (exact)

Exact declaration. The complete defining right-hand side is shown below.

```
def IsLinkedSet (c : ClosureSpace α) (S : Set α) : Prop :=
  ∀ x ∈ S, ∀ y ∈ S, c.Linked x y
```

In the source	Mathematical meaning
Exact declaration `{α : Type u}` in the surrounding scope	The arbitrary ambient carrier X. The nearby carrier `β` and universe `v` are unused by this declaration.
Exact declaration `(c : ClosureSpace α)`	The given ambient closure; the pair is not tested in a newly chosen topology or subspace closure.
Exact declaration `(S : Set α)`	The one subset S whose pairs are tested.
Exact declaration `∀ x ∈ S, ∀ y ∈ S`	For every x with $x \in S$, and every y with $y \in S$; equal points are included. No nonemptiness or distinctness condition is inserted.
Exact declaration `c.Linked x y`	The equality-or-nonclosed-pair condition in the separate complete `Linked` definition below.
Exact declaration `: Prop :=` and the whole right-hand side	A predicate asserting this condition for all pairs, not a data structure choosing points, a maximal set or a component.

Exact surrounding scope and supporting definitions

The following source excerpts are separate from the declaration above. Their line ranges are not concatenated into a synthetic signature.

Source LinkedSet.lean, lines 23–27:

```
universe u v

namespace ClosureSpace

variable {α : Type u} {β : Type v}
```

Source Components.lean, lines 23–32:

```
universe u v

namespace ClosureSpace

variable {α : Type u} {β : Type v}

/-- Equality or a nonclosed two-point set. -/
def Linked (c : ClosureSpace α) (x y : α) : Prop := x = y ∨ ¬ c.IsClosed {x, y}

theorem linked_refl (c : ClosureSpace α) (x : α) : c.Linked x x := Or.inl rfl
```

Source LinkedSet.lean, lines 34–41:

```
theorem isLinkedSet_empty (c : ClosureSpace α) : c.IsLinkedSet ∅ := by
  intro x hx
  exact hx.elim

/-- Reflexivity of `Linked` makes every singleton an R-set. -/
theorem isLinkedSet_singleton (c : ClosureSpace α) (x : α) : c.IsLinkedSet {x} := by
  rintro y rfl z rfl
  exact c.linked_refl _
```

Source Basic.lean, lines 60–61:

```
theorem isClosed_iff (c : ClosureSpace α) (s : Set α) :
    c.IsClosed s ↔ c.closure s = s := c.toClosureOperator.isClosed_iff
```

Source Components.lean, lines 48–52:

```
/-- The equivalence relation generated by linked pairs. -/
def componentSetoid (c : ClosureSpace α) : Setoid α := Relation.EqvGen.setoid c.Linked

/-- Belonging to the same finite-chain component. -/
abbrev Related (c : ClosureSpace α) (x y : α) : Prop := (c.componentSetoid).r x y
```

Source Components.lean, lines 67–68:

```
/-- The equivalence class as an ambient subset. -/
def component (c : ClosureSpace α) (x : α) : Set α := {y | c.Related x y}
```

Exact declaration for the source guide

```
def IsLinkedSet (c : ClosureSpace α) (S : Set α) : Prop :=
  ∀ x ∈ S, ∀ y ∈ S, c.Linked x y

/-- The empty set is an R-set, with no nonemptiness convention added. -/
```