Components generated by linked pairs

`ClosureSpace.component`

def

Components generated by linked pairs

Statement

The component of a point in a closure space consists of all points reachable from it by a finite chain of linked pairs. It is represented as an actual subset of the original set.

Definition

Let X be a set with an extensive, monotone and idempotent closure operation $c : \mathcal{P}(X) \to \mathcal{P}(X)$ satisfying $c(\varnothing) = \varnothing$. A subset A is closed when $c(A) = A$. Define

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

Let $x \sim_c y$ mean that there is an integer $n \geq 0$ and a finite chain

$$x = x_0 \leftrightarrow_c x_1 \leftrightarrow_c \cdots \leftrightarrow_c x_n = y.$$

The case $n = 0$ is the chain with a single point, so it gives $x \sim_c x$. The **component of** x is

$$C_c(x) = \{y \in X : x \sim_c y\}.$$

The source defines $\sim_c$ as the equivalence relation generated by the linked relation. This agrees with the finite-chain description: the linked relation is symmetric, reversing a chain gives symmetry, and concatenating two chains gives transitivity. The empty chain of steps supplies reflexivity. These are exactly the operations in the cited equivalence-closure construction.

Assumptions

The only data are a closure space (X, c) and a point $x \in X$. No topology or separation condition is required. If X is empty, there is no point at which to evaluate the component function.

Conclusion

The result is the subset $C_c(x) \subseteq X$, which contains x. The separately cited equality criterion gives

$$C_c(x) = C_c(y) \quad \Longleftrightarrow \quad x \sim_c y.$$

Thus choosing another point in the same component leaves the component set unchanged. Its closure $c(C_c(x))$ is a separate object, defined separately in the source. The definition of $C_c(x)$ does not assert that this subset is closed or that it is a connected component in a topological sense.

Main citations

- The generating relation and equivalence closure: `ClosureSpace.Linked`, `ClosureSpace.componentSetoid` and `ClosureSpace.Related`, lines 30–65.
- Exact component, membership and the separate component closure: `ClosureSpace.component`, `ClosureSpace.mem_component`, `ClosureSpace.component_eq_of_related` and `ClosureSpace.componentClosure`, lines 68–84.
- Equality of component sets: `ClosureSpace.component_eq_iff_related`, lines 30–37.
- The finite equivalence-closure constructors: Mathlib `Relation.EqvGen`, lines 349–353, at the pinned mathlib commit. Its constructors insert relation steps, reflexivity, symmetry and transitivity.
- Ryotaro Tanaka, arXiv:2504.05551v1, Definition 3.20, Proposition 3.23 and Definition 3.24: the finite-chain relation; the component notation continues on page 11.

Lean source signature (exact)

The namespace is `ClosureSpace`; the surrounding carrier `α` denotes the set X.

```
def component (c : ClosureSpace α) (x : α) : Set α := {y | c.Related x y}
```

In the source	Mathematical meaning
Exact declaration `α, c : ClosureSpace α`	An arbitrary set X equipped with the extensive, monotone, idempotent closure c preserving $\varnothing$; finite-union preservation is not required.
Exact declaration `x : α`	The given starting point $x \in X$.
Exact declaration `c.Related x y`	$x \sim_c y$: the equivalence relation generated by linked pairs, or equivalently the finite-chain relation defined above, including a chain with no steps.
Exact declaration `{y | c.Related x y}`	The complete RHS is $C_c(x) = \{y \in X : x \sim_c y\}$. It ranges over all possible endpoints y while keeping x fixed.
Exact declaration `: Set α`	The returned object is this subset of X, not a quotient point or its closure $c(C_c(x))$. The latter is the separate declaration `componentClosure`.

Relation.EqvGen · fixed Mathlib definition

Fixed source, lines 346–353 · Raw UTF-8 source

```
variable (r) in
/-- `EqvGen r`: equivalence closure of `r`. -/
@[mk_iff]
inductive EqvGen : α → α → Prop
  | rel x y : r x y → EqvGen x y
  | refl x : EqvGen x x
  | symm x y : EqvGen x y → EqvGen y x
  | trans x y z : EqvGen x y → EqvGen y z → EqvGen x z
```