

```

cop (θ o') (φ o') = let !!!!tl, c, tr = cop θ φ in !!!!tl t'' , c , tr t''
cop (θ o') (φ os) = let !!!!tl, c, tr = cop θ φ in !!!!tl t's' , c c's , tr tsss
cop (θ os) (φ o') = let !!!!tl, c, tr = cop θ φ in !!!!tl tsss , c cs' , tr t's'
cop (θ os) (φ os) = let !!!!tl, c, tr = cop θ φ in !!!!tl tsss , c css , tr tsss
cop   oz   oz   =                               !!!! tzzz , czz , tzzz

```

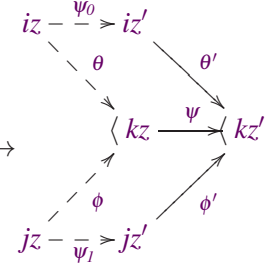
The **copU** proof goes by induction on the triangles which share ψ' and inversion of the coproduct.

A further useful property of coproduct diagrams is that we can selectively refine them by a thinning into the covered scope.

```

subCOP : (ψ : kz ⊆ kz') → Cover ov θ' φ' →
  Σ _ λ iz → Σ _ λ jz → Σ (iz ⊆ kz) λ θ → Σ (jz ⊆ kz) λ φ →
  Σ (iz ⊆ iz') λ ψ₀ → Σ (jz ⊆ jz') λ ψ₁ → Cover ov θ φ

```



The implementation is a straightforward induction on the diagram.

The payoff from coproducts is the type of *relevant pairs* — the co-de-Brujin touchstone:

<pre> record _×_R_ (S T : K̄) (ijz : Bwd K) : Set where constructor pair field outl : S ↑ ijz; outr : T ↑ ijz cover : Cover tt (thinning outl) (thinning outr) </pre>	$\begin{array}{l} \text{--}_R\text{--} : S \uparrow kz \rightarrow T \uparrow kz \rightarrow (S \times_R T) \uparrow kz \\ (s \uparrow \theta) ,_R (t \uparrow \phi) = \\ \text{let } ! \psi, \theta', \phi', -, c, - = \text{cop } \theta \phi \\ \text{in pair } (s \uparrow \theta') (t \uparrow \phi') c \uparrow \psi \end{array}$
---	---

The corresponding projections are readily definable.

<pre> outl_R : (S ×_R T) ↑ kz → S ↑ kz outl_R (pair s - - ↑ ψ) = thin ↑ ψ s </pre>	<pre> outr_R : (S ×_R T) ↑ kz → T ↑ kz outr_R (pair - t - ↑ ψ) = thin ↑ ψ t </pre>
--	--

7 Monoidal Structure of Order-Preserving Embeddings

Variable bindings extend scopes. The λ construct does just one ‘snoc’, but binding can be simultaneous, so the monoidal structure on Δ_+ induced by concatenation is what we need.

<pre> _++_ : Bwd K → Bwd K → Bwd K kz ++ [] = kz kz ++ (iz , j) = (kz ++ iz) , j </pre>	$\begin{array}{l} \text{--}++\text{--} : iz \sqsubseteq jz \rightarrow iz' \sqsubseteq jz' \rightarrow (iz ++ iz') \sqsubseteq (jz ++ jz') \\ \theta ++ \text{--} \text{oz} = \theta \\ \theta ++ \text{--} (\phi \text{os}) = (\theta ++ \text{--} \phi) \text{os} \\ \theta ++ \text{--} (\phi \text{o}') = (\theta ++ \text{--} \phi) \text{o}' \end{array}$
---	---

Concatenation further extends to **Coverings**, allowing us to build them in chunks.

```

_++_C_ : Cover ov θ φ → Cover ov θ' φ' → Cover ov (θ ++_C_ θ') (φ ++_C_ φ')
c ++_C_ (d c's) = (c ++_C_ d) c's
c ++_C_ (d cs') = (c ++_C_ d) cs'
c ++_C_ (_css {both = b} d) = _css {both = b} (c ++_C_ d)
c ++_C_ czz = c

```

One way to build such a chunk is to observe that two scopes cover their concatenation.