```
"""
    struct HomSpace{S<:ElementarySpace,P1<:CompositeSpace{S},P2<:CompositeSpace{S}}
        codomain::P1
        domain::P2
    end

Represents the linear space of morphisms with codomain of type `P1` and domain of
type `P2`.
"""
struct HomSpace{S<:ElementarySpace, P1<:CompositeSpace{S}, P2<:CompositeSpace{S}}
    codomain::P1
    domain::P2
end
codomain(W::HomSpace) = W.codomain
domain(W::HomSpace) = W.domain

dual(W::HomSpace) = HomSpace(dual(W.domain), dual(W.codomain))
Base.adjoint(W::HomSpace{<:EuclideanSpace}) = HomSpace(W.domain, W.codomain)

Base.hash(W::HomSpace, h::UInt) = hash(domain(W), hash(codomain(W), h))
Base.:(==)(W1::HomSpace, W2::HomSpace) =
    (W1.codomain == W2.codomain) && (W1.domain == W2.domain)

spacetype(W::HomSpace) = spacetype(typeof(W))
sectortype(W::HomSpace) = sectortype(typeof(W))
field(W::HomSpace) = field(typeof(W))

spacetype(::Type{<:HomSpace{S}}) where S = S
field(L::Type{<:HomSpace}) = field(spacetype(L))
sectortype(L::Type{<:HomSpace}) = sectortype(spacetype(L))

const TensorSpace{S} = Union{S, ProductSpace{S}}
const TensorMapSpace{S, N₁, N₂} = HomSpace{S, ProductSpace{S,N₁},
ProductSpace{S,N₂}}

Base.getindex(W::TensorMapSpace{<:IndexSpace,N₁,N₂}, i) where {N₁,N₂} =
    i <= N₁ ? codomain(W)[i] : dual(domain(W)[i-N₁])

→(dom::TensorSpace{S}, codom::TensorSpace{S}) where {S<:ElementarySpace} =
    HomSpace(ProductSpace(codom), ProductSpace(dom))

←(codom::TensorSpace{S}, dom::TensorSpace{S}) where {S<:ElementarySpace} =
    HomSpace(ProductSpace(codom), ProductSpace(dom))

function Base.show(io::IO, W::HomSpace)
    if length(W.codomain) == 1
        print(io, W.codomain[1])
    else
        print(io, W.codomain)
    end
    print(io, " ← ")
    if length(W.domain) == 1
        print(io, W.domain[1])
    else
```

```
        print(io, W.domain)
    end
end

"""
    blocksectors(W::HomSpace)

Return an iterator over the different unique coupled sector labels, i.e. the intersection
of the different fusion outputs that can be obtained by fusing the sectors present in the
domain, as well as from the codomain.
"""
function blocksectors(W::HomSpace)
    sectortype(W) === Trivial &&
        return TrivialOrEmptyIterator(dim(domain(W)) == 0 || dim(codomain(W)) == 0)
    return intersect(blocksectors(codomain(W)), blocksectors(domain(W)))
end

"""
    dim(W::HomSpace)

Return the total dimension of a `HomSpace`, i.e. the number of linearly independent
morphisms that can be constructed within this space.
"""
function dim(W::HomSpace)
    d = 0
    for c in blocksectors(W)
        d += blockdim(codomain(W), c) * blockdim(domain(W), c)
    end
    return d
end
```