```
# check if a list of indices specifies a tensor contraction in ncon style
function isnconstyle(network)
    allindices = Vector{Int}()
    for ind in network
        all(i->isa(i, Integer), ind) || return false
        append!(allindices, ind)
    end
    while length(allindices) > 0
        i = pop!(allindices)
        if i > 0 # positive labels represent contractions or traces and should
            appear twice
            k = findfirst(isequal(i), allindices)
            k === nothing && return false
            l = findnext(isequal(i), allindices, k+1)
            l !== nothing && return false
            deleteat!(allindices, k)
        elseif i < 0 # negative labels represent open indices and should appear once
            findfirst(isequal(i), allindices) === nothing || return false
        else # i == 0
            return false
        end
    end
    return true
end

function ncontree(network)
    contractionindices = Vector{Vector{Int}}(undef, length(network))
    for k = 1:length(network)
        indices = network[k]
        # trace indices have already been removed, remove open indices by filtering
          on positive values
        contractionindices[k] = filter(i->i>0, indices)
    end
    partialtrees = collect(Any, 1:length(network))
    _ncontree!(partialtrees, contractionindices)
end

function _ncontree!(partialtrees, contractionindices)
    if length(partialtrees) == 1
        return partialtrees[1]
    end
    if all(isempty, contractionindices) # disconnected network
        partialtrees[end-1] = Any[partialtrees[end-1], partialtrees[end]]
        pop!(partialtrees)
        pop!(contractionindices)
    else
        let firstind = minimum(vcat(contractionindices...))
            i1 = findfirst(x->in(firstind,x), contractionindices)
            i2 = findnext(x->in(firstind,x), contractionindices, i1+1)
            @assert i1 !== nothing && i2 !== nothing
            newindices = unique2(vcat(contractionindices[i1],
contractionindices[i2]))
            newtree = Any[partialtrees[i1], partialtrees[i2]]
            partialtrees[i1] = newtree
```

```
            deleteat!(partialtrees, i2)
            contractionindices[i1] = newindices
            deleteat!(contractionindices, i2)
        end
    end
    _ncontree!(partialtrees, contractionindices)
end

function nconindexcompletion(ex)
    if isassignment(ex) || isdefinition(ex)
        lhs, rhs = getlhs(ex), getrhs(ex)
        # process left hand side
        if istensor(lhs) && istensorexpr(rhs)
            indices = getindices(rhs)

            if lhs.head == :ref && length(lhs.args) == 2 && lhs.args[2] == :(:)
                if all(isa(i, Integer) && i < 0 for i in indices)
                    lhs = Expr(:ref, lhs.args[1], sort(indices, rev=true)...)
                else
                    error("cannot automatically infer index order of left hand
side")
                end
            end
            return Expr(ex.head, lhs, rhs)
        else
            return ex
        end
    elseif ex isa Expr
        return Expr(ex.head, map(nconindexcompletion, ex.args)...)
    else
        return ex
    end
end

function resolve_traces(tensors,network)
    transformed = map(zip(tensors,network)) do (A,IA)
        IC = unique2(IA);
        if length(IC) == length(IA)
            (A,IA)
        else
            (tensortrace(copy(A),IA,IC),IC)
        end
    end

    first.(transformed), last.(transformed)
end
```