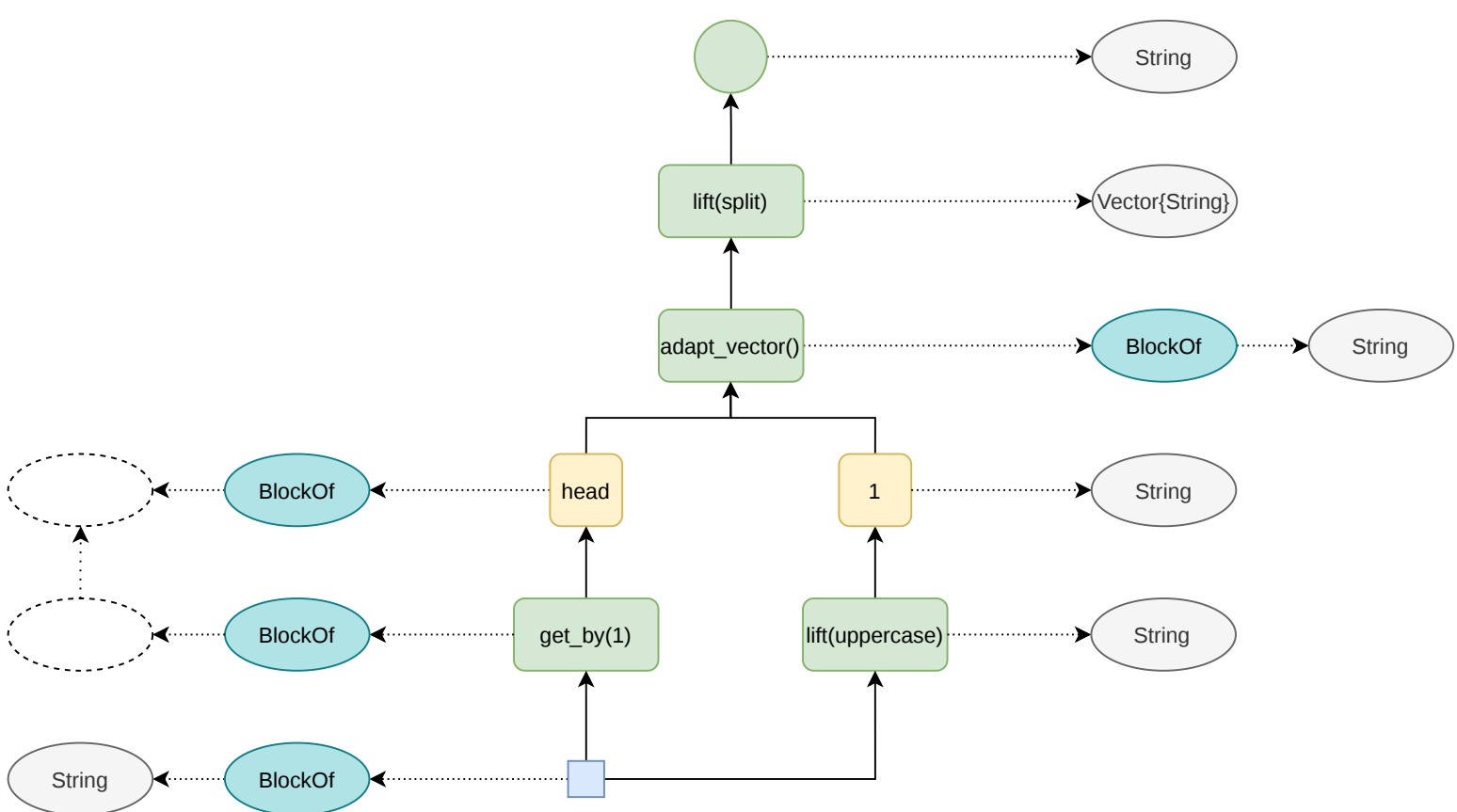


The diagram illustrates the transformation of a flat table into a hierarchical tree structure through a series of steps:

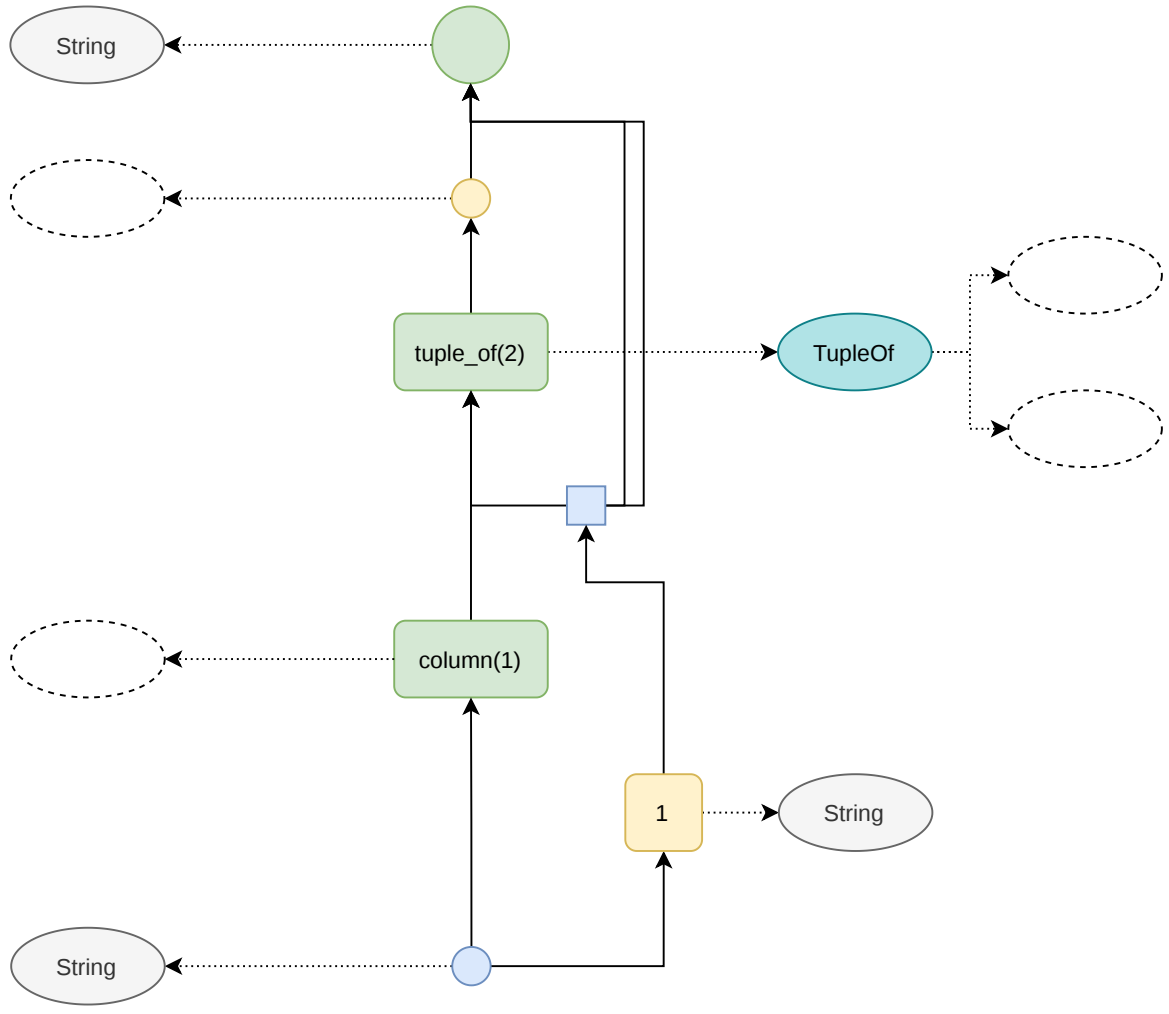
- Initial State:** A flat table with two columns: an index (1) and a value ("Hello World").
- Transformation:** An arrow points to a table where the value is a string array: `String["Hello", "World"]`.
- Indexing:** An arrow points to a table with two rows. The first row has index 1 and value 1. The second row has index 2 and value 3. This represents the first row of the original table being indexed.
- Splitting:** An arrow points to a tree structure. The root node has two children. The left child is a table with two rows (index 1, value 1; index 2, value 3). The right child is a table with two rows (index 1, value "Hello"; index 2, value "World").
- Normalization:** An arrow points to a tree structure. The root node has two children. The left child is a table with two rows (index 1, value 1; index 2, value 2). The right child is a table with one row (index 1, value "HELLO").
- Final State:** An arrow points to a tree structure. The root node has two children. The left child is a table with two rows (index 1, value 1; index 2, value 2). The right child is a table with one row (index 1, value 1).



wrap()



chain\_of(tuple\_of(2), column(1))



sieve\_by0

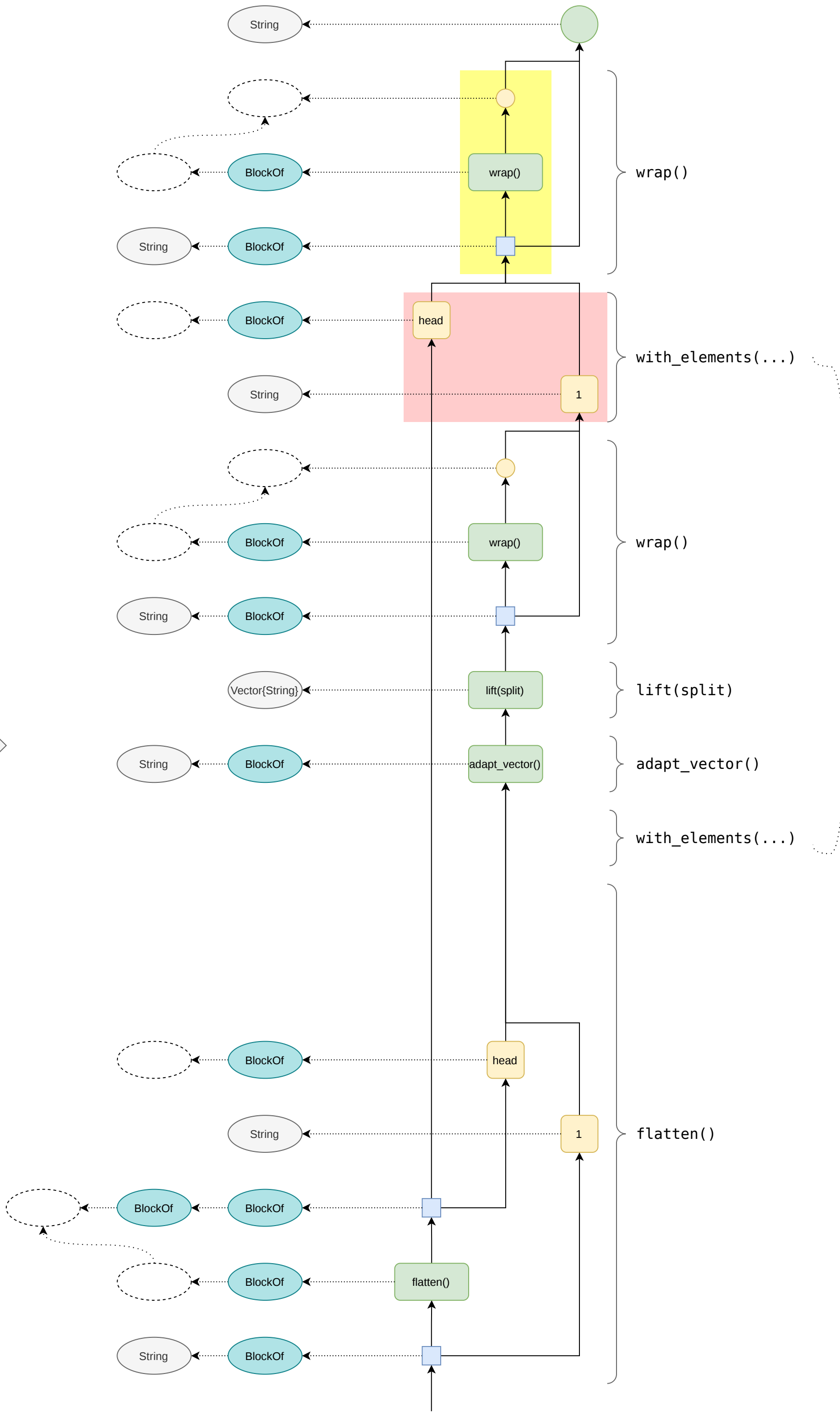


chain\_of(wrap(), block\_length())



@query "Hello World" split(it)

```
chain_of(
  wrap(),
  with_elements(
    chain_of(
      wrap(),
      lift(split),
      adapt_vector()),
    flatten())
```



untrace(n::NodeRef, guard::NodeRef)::Tuple{Pipeline,Vector{NodeRef}}

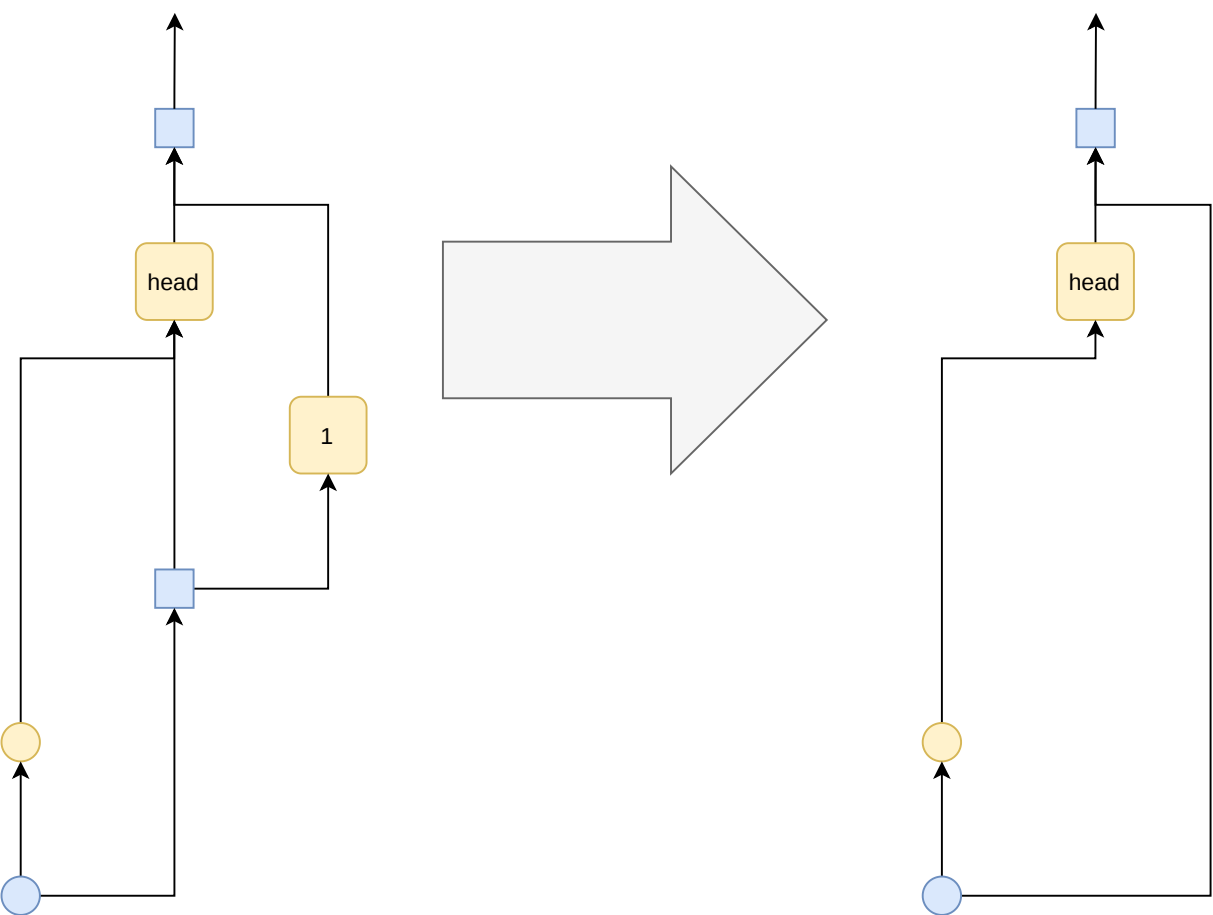






@query "Hello World" split(it)

```
chain_of(
  wrap(),
  with_elements(
    chain_of(
      wrap(),
      lift(split),
      adapt_vector()),
    flatten())
```







untrace(n::NodeRef, guard::NodeRef)::Tuple{Pipeline,Vector{NodeRef}}

`{it ^ 2, (it ^ 2) ^ 3}` `chain_of(f, tuple_of(g, h)) => tuple_of(chain_of(f, g), chain_of(f, h))`



`untrace(n::NodeRef, guard::NodeRef)::Tuple{Pipeline,Vector{NodeRef}}`

@query "Hello World" split(it)

```
chain_of(
  wrap(),
  with_elements(wrap()),
  with_elements(lift(split)),
  with_elements(adapt_vector()),
  flatten())
```

