

The diagram illustrates the transformation of a flat table into a hierarchical tree structure through four stages, connected by large gray arrows:

- Stage 1:** A flat table with two columns: an index column (1) and a value column ("Hello World").
- Stage 2:** A flat table with two columns: an index column (1) and a value column (String["Hello", "World"]).
- Stage 3:** A hierarchical tree structure. The root node (empty box) has two children:
 - Left child: A table with two rows. Row 1 has index 1 and value 1. Row 2 has index 2 and value 3.
 - Right child: A table with two rows. Row 1 has index 1 and value "Hello". Row 2 has index 2 and value "World".
- Stage 4:** A hierarchical tree structure. The root node (empty box) has two children:
 - Left child: A table with two rows. Row 1 has index 1 and value 1. Row 2 has index 2 and value 2.
 - Right child: A table with one row. Row 1 has index 1 and value "HELLO".

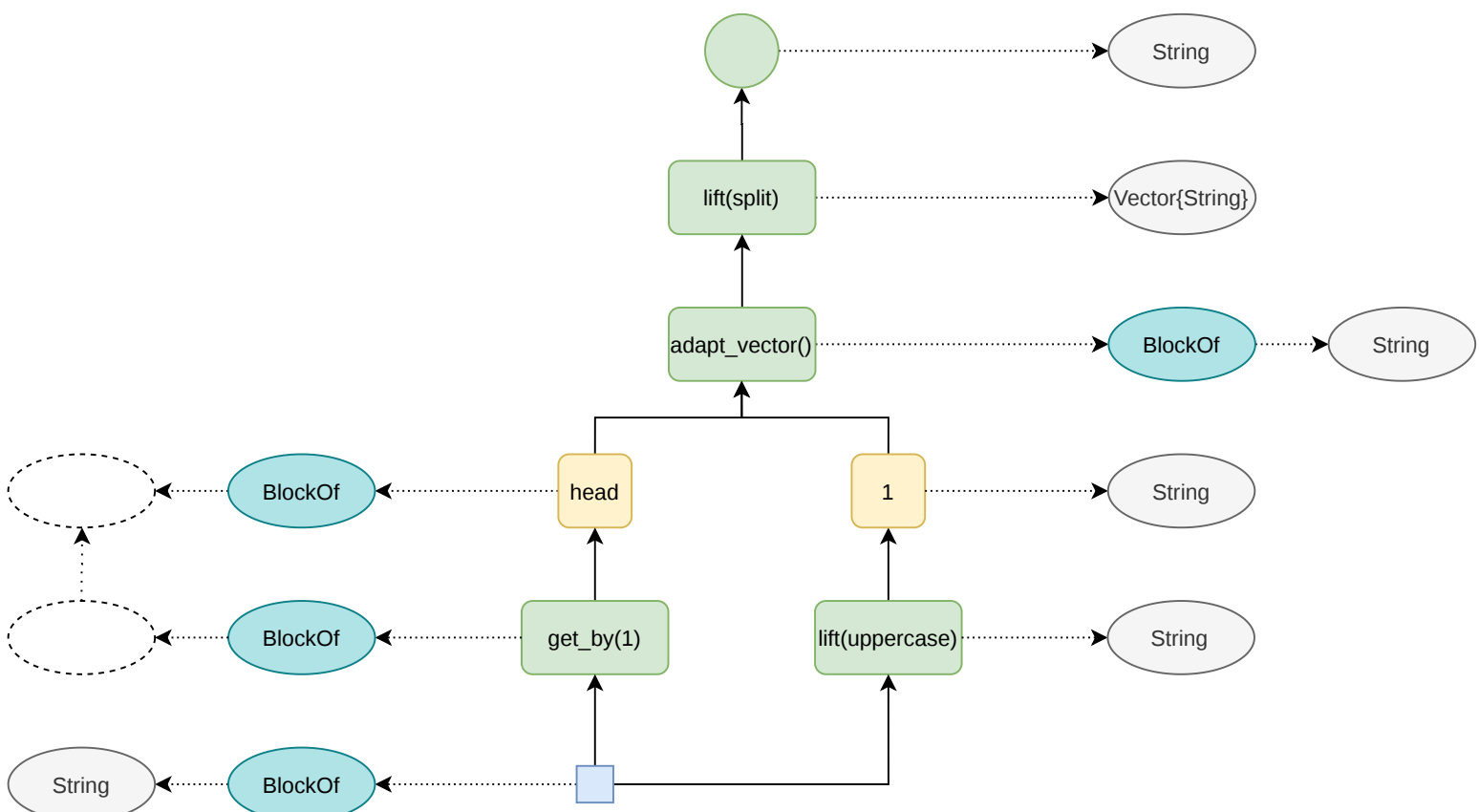
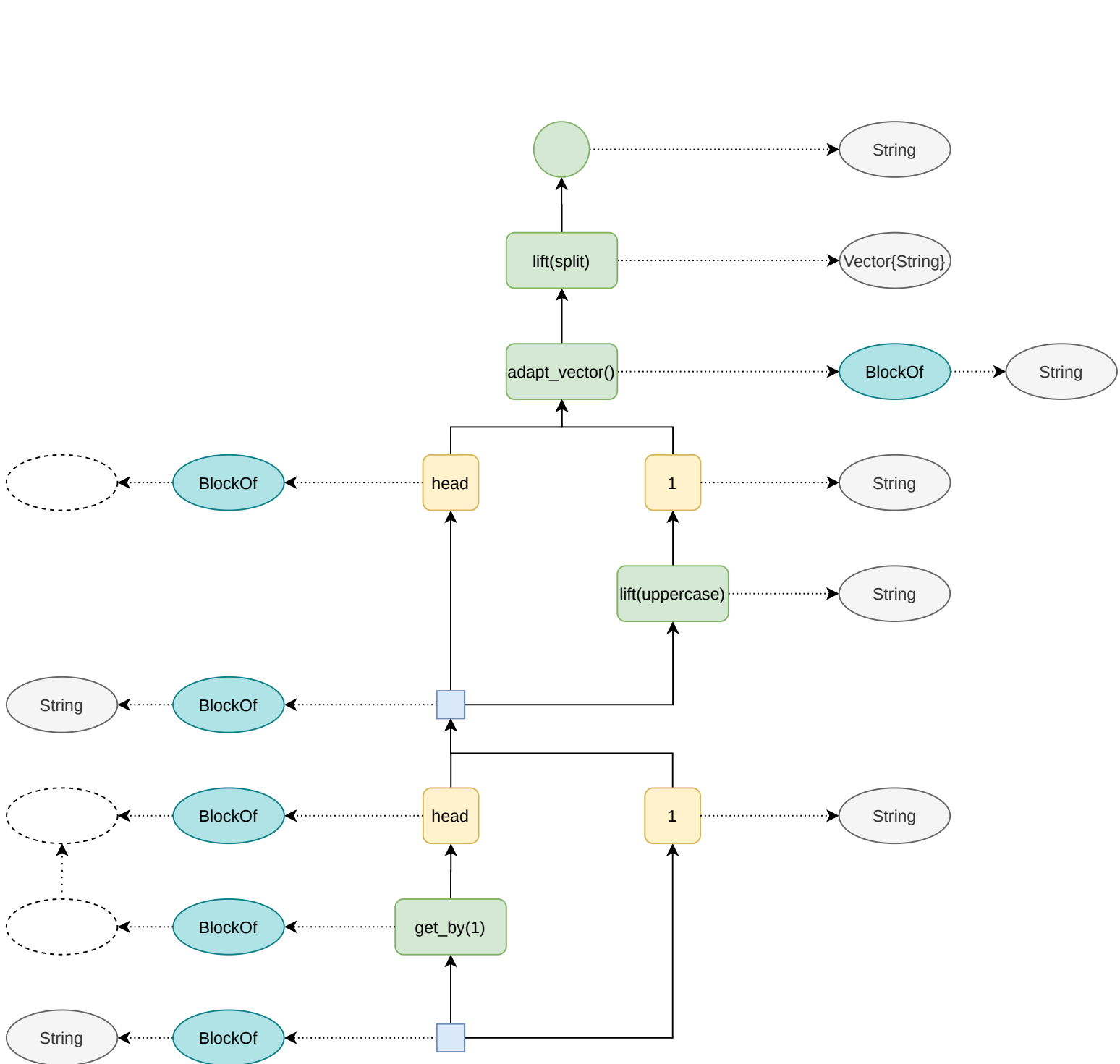
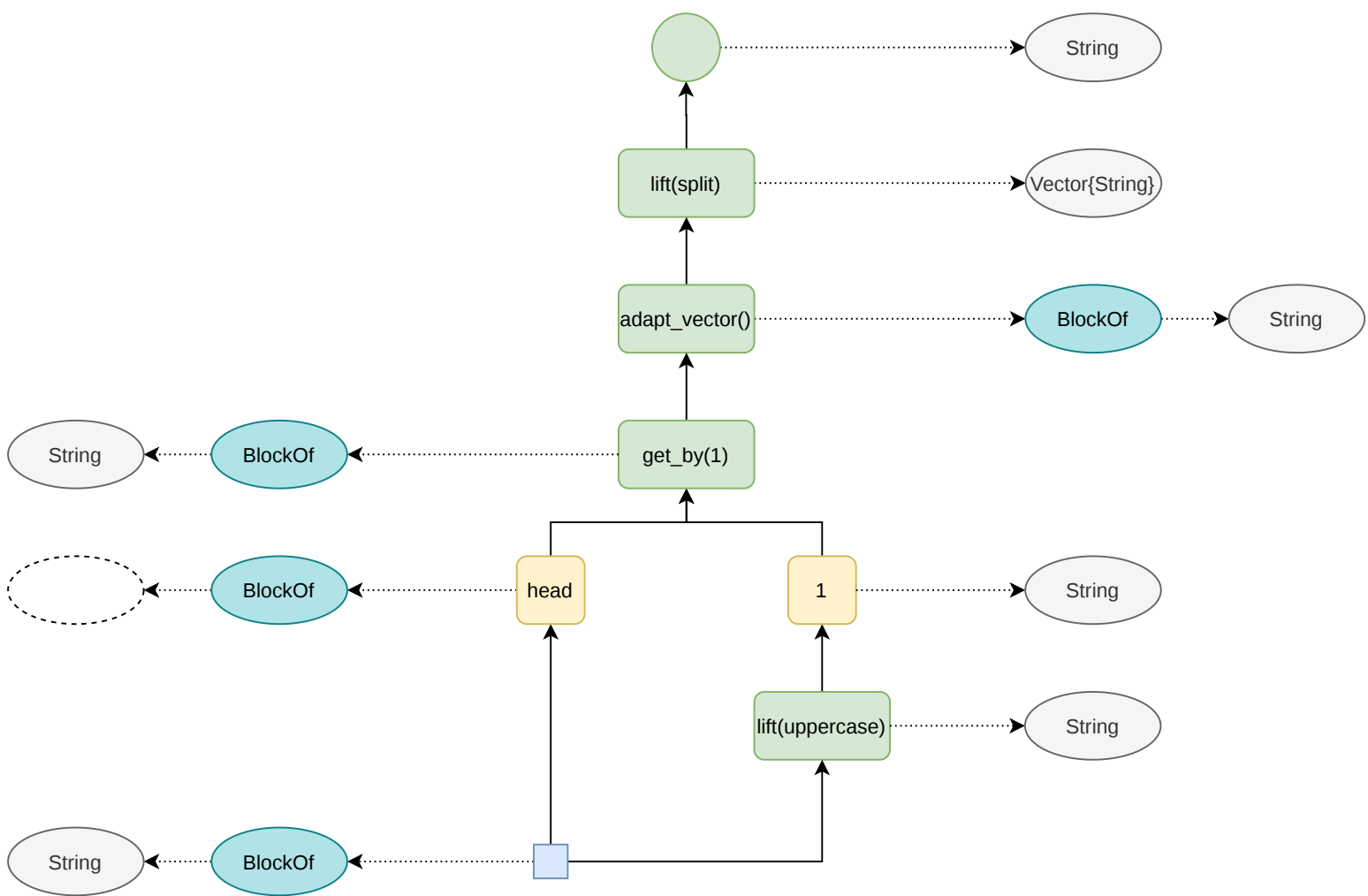
Below Stage 3, a large gray arrow points down to a table with two rows. Row 1 has index 1 and value "HELLO". Row 2 has index 2 and value "WORLD".

Below Stage 4, a hierarchical tree structure is shown. The root node (empty box) has two children:

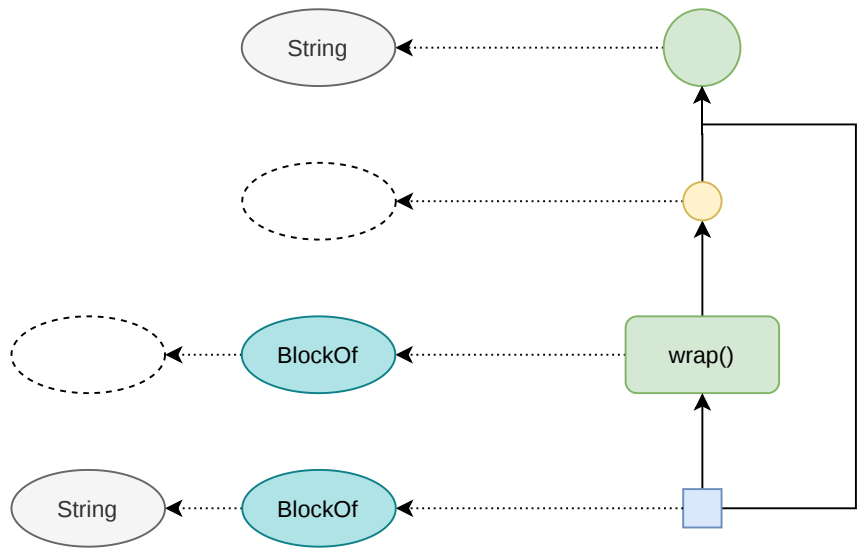
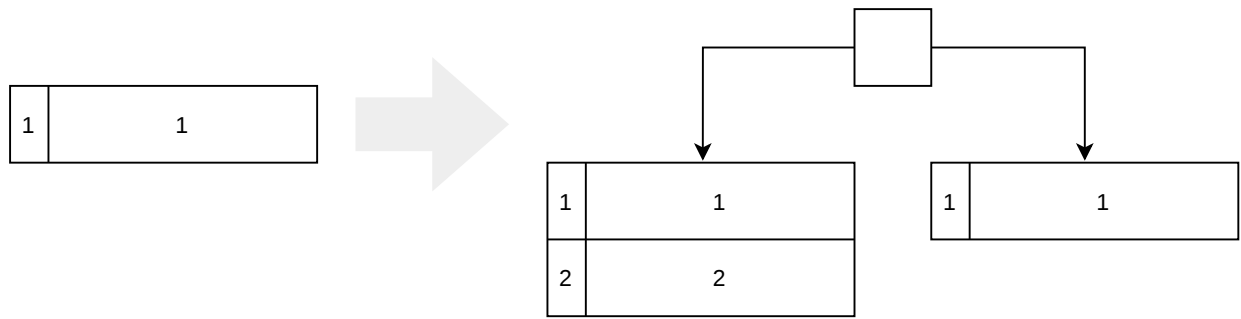
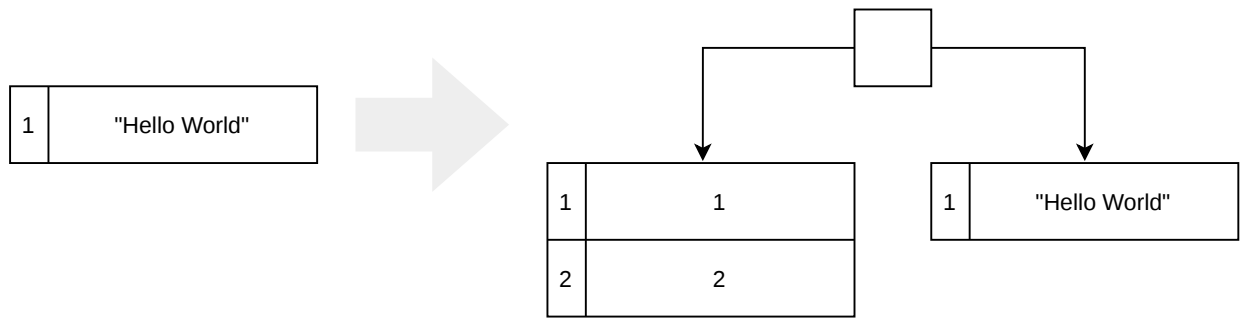
- Left child: A table with two rows. Row 1 has index 1 and value 1. Row 2 has index 2 and value 3.
- Right child: A table with two rows. Row 1 has index 1 and value 1. Row 2 has index 2 and value 2.

Below Stage 5, a hierarchical tree structure is shown. The root node (empty box) has two children:

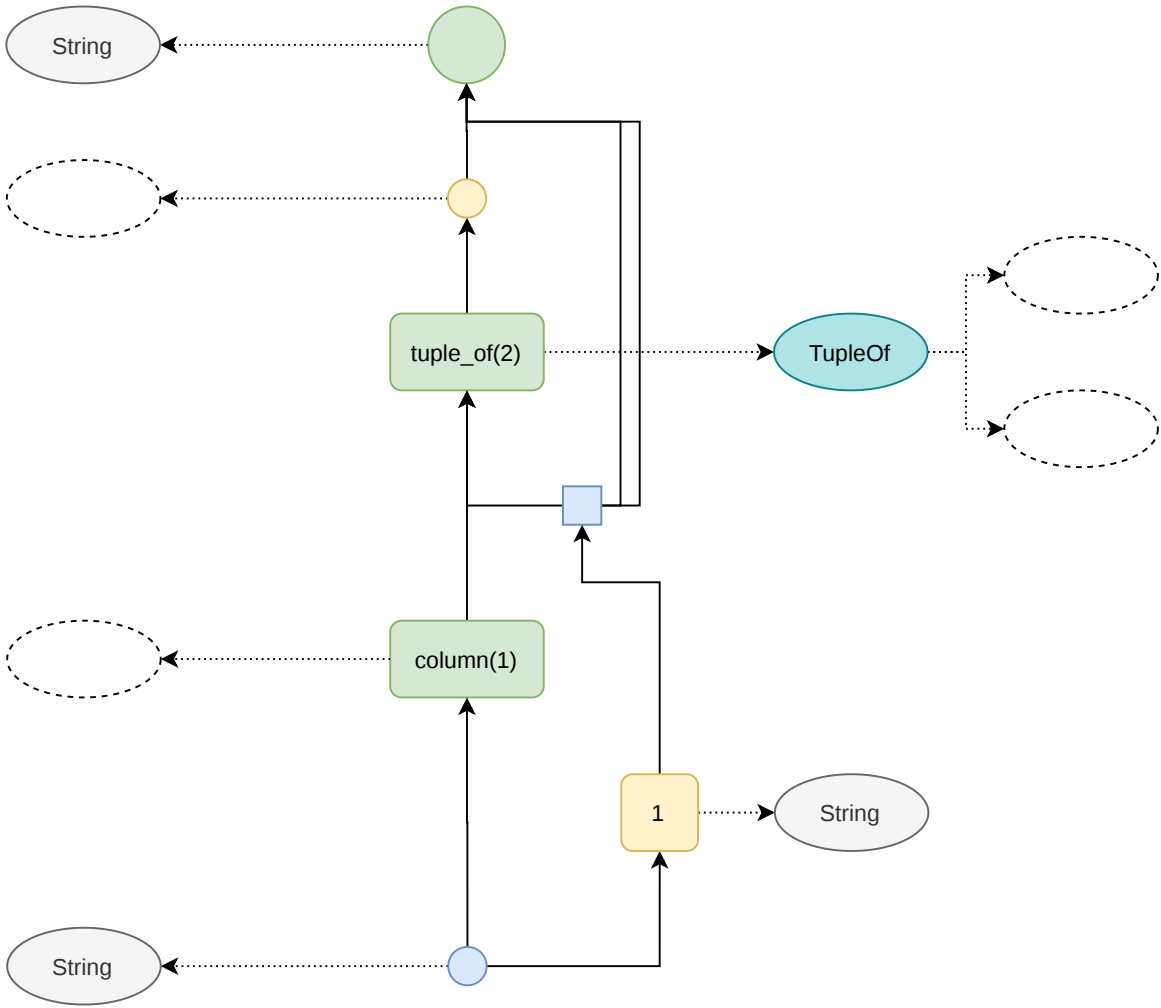
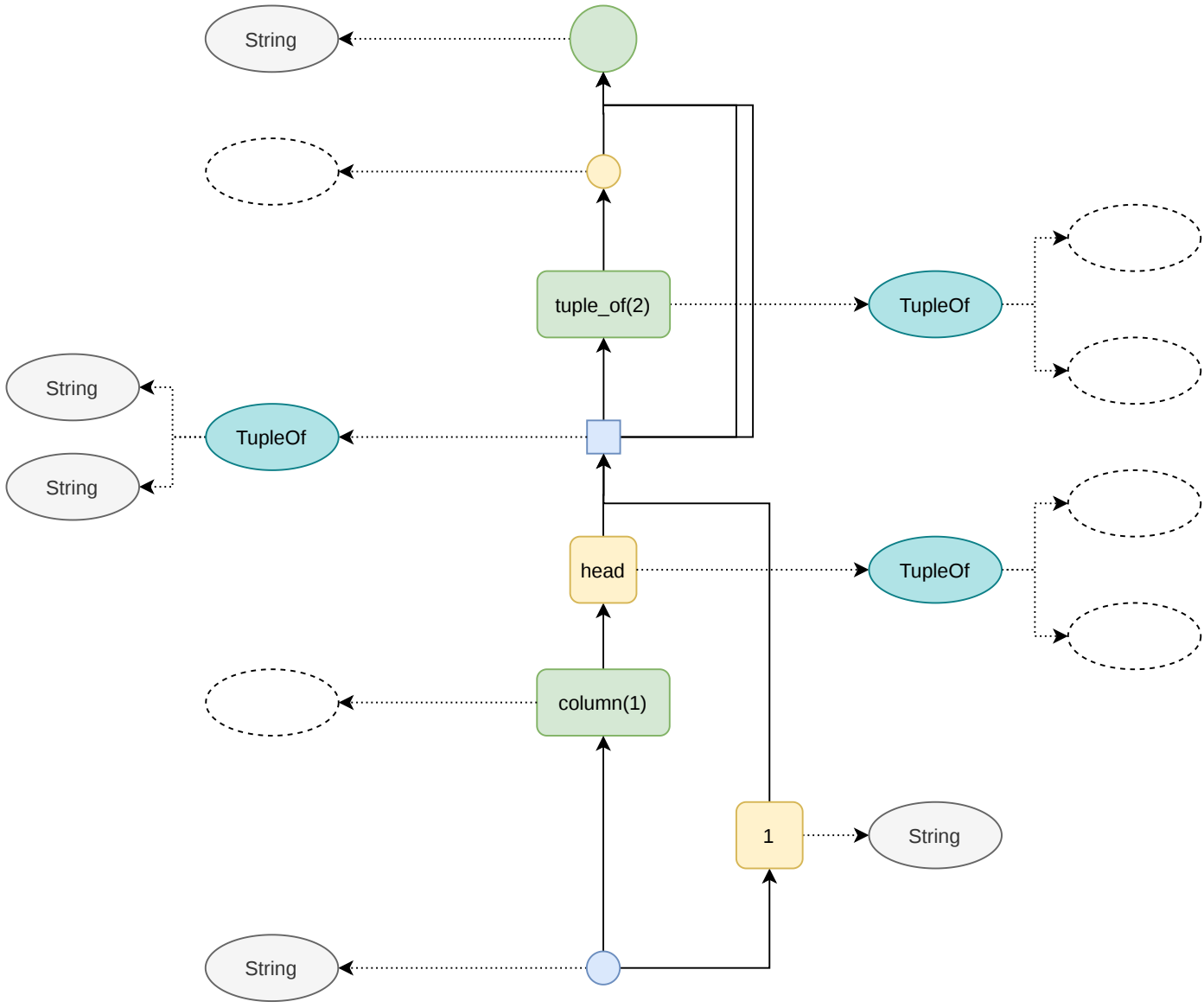
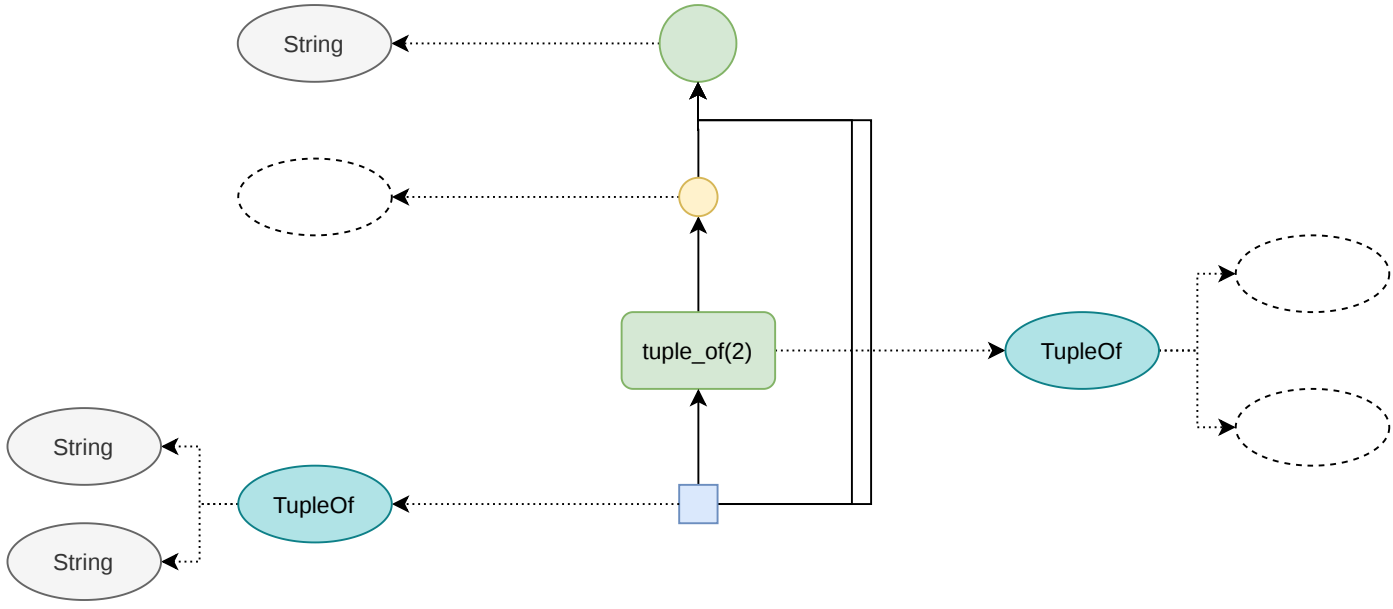
- Left child: A table with two rows. Row 1 has index 1 and value 1. Row 2 has index 2 and value 2.
- Right child: A table with one row. Row 1 has index 1 and value 1.



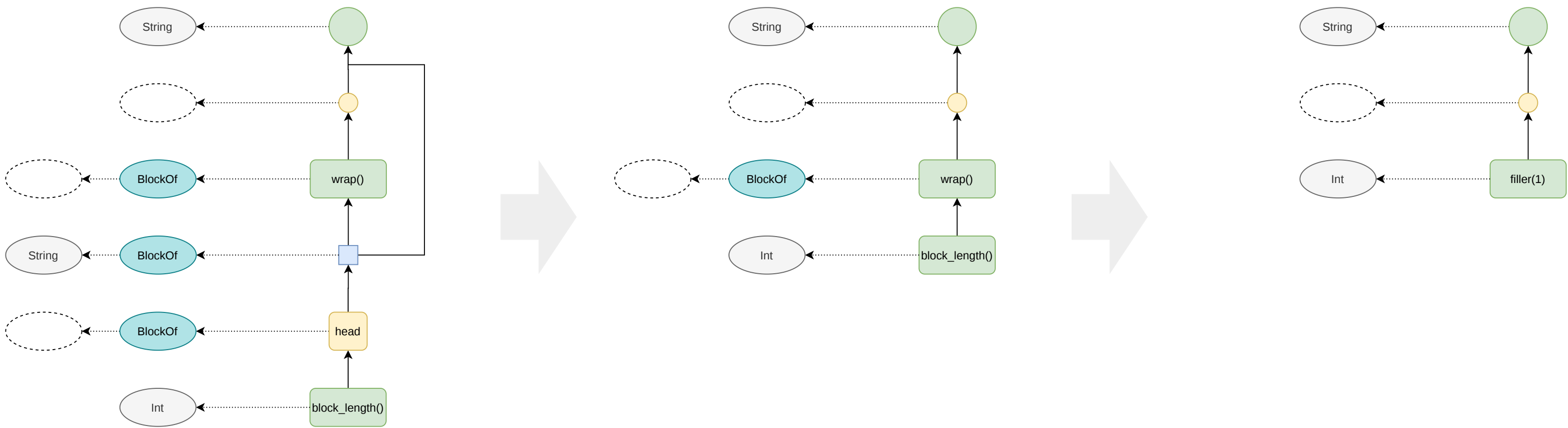
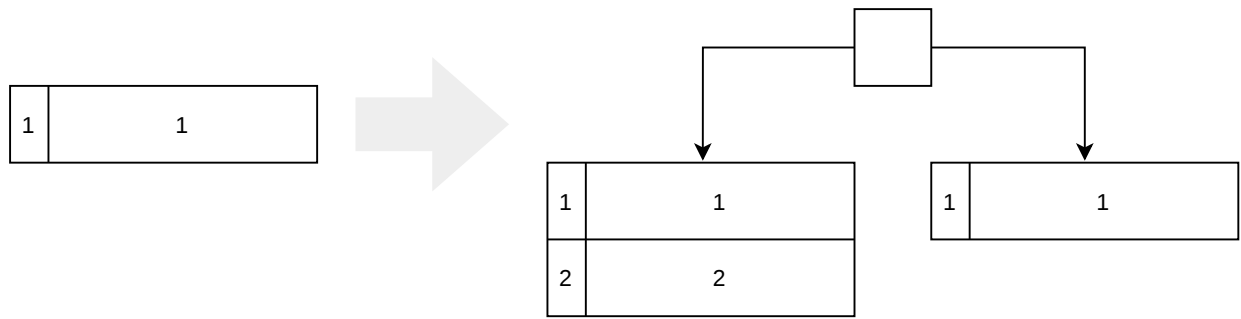
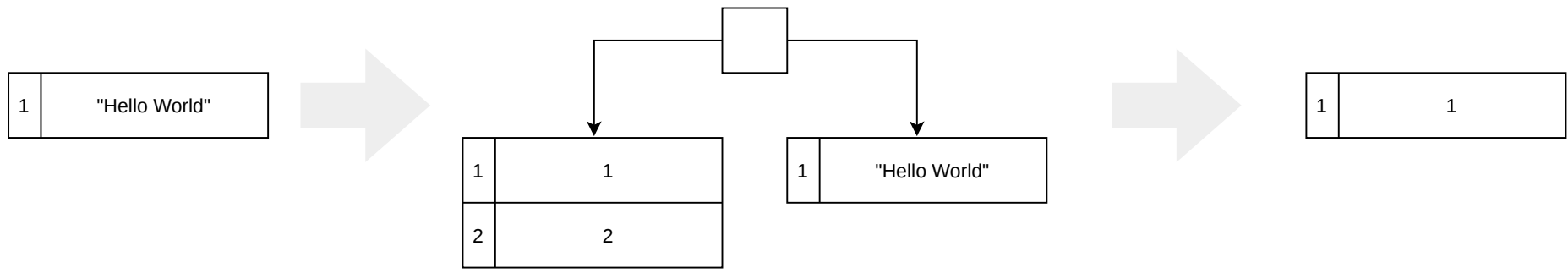
wrap0



chain_of(tuple_of(2), column(1))

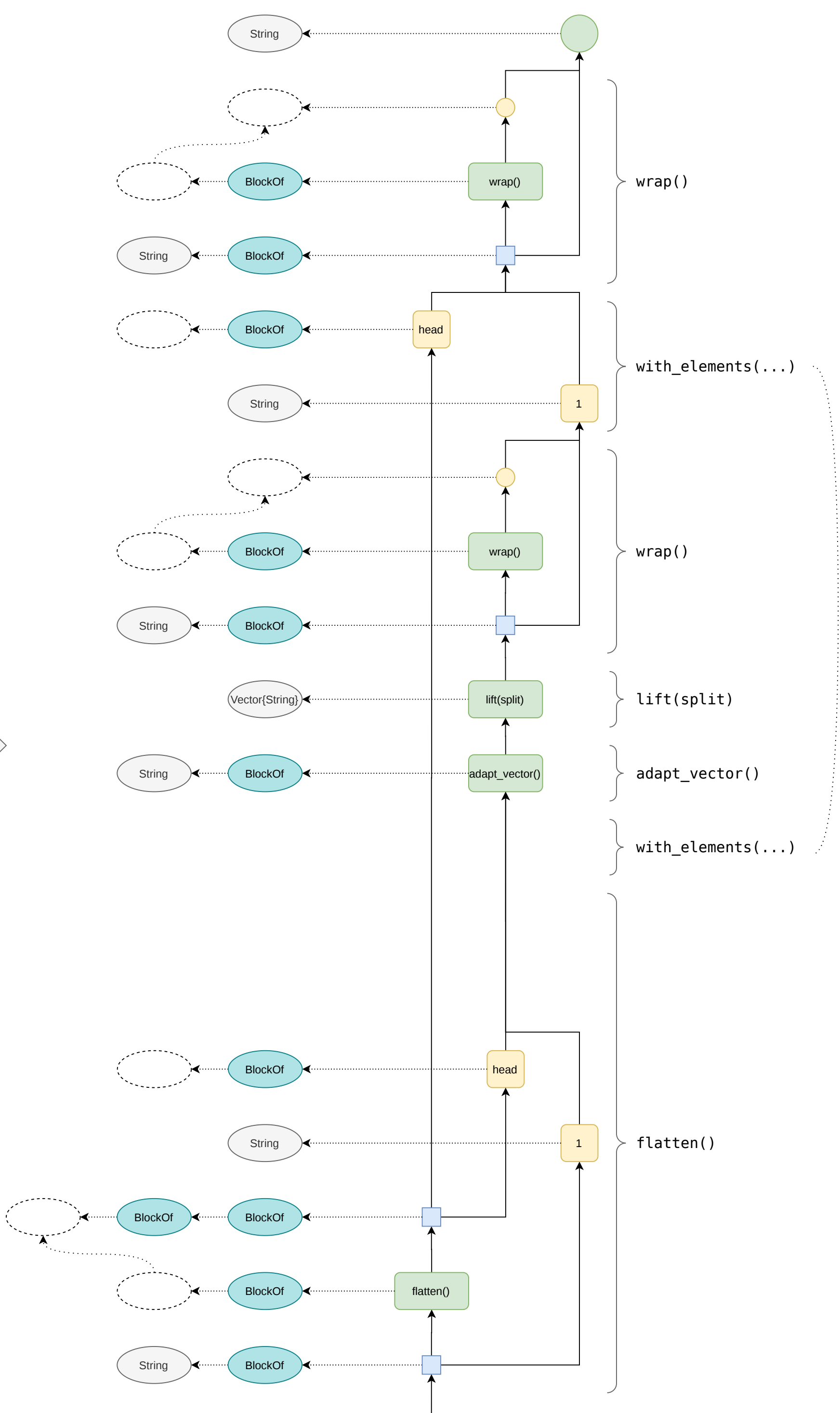
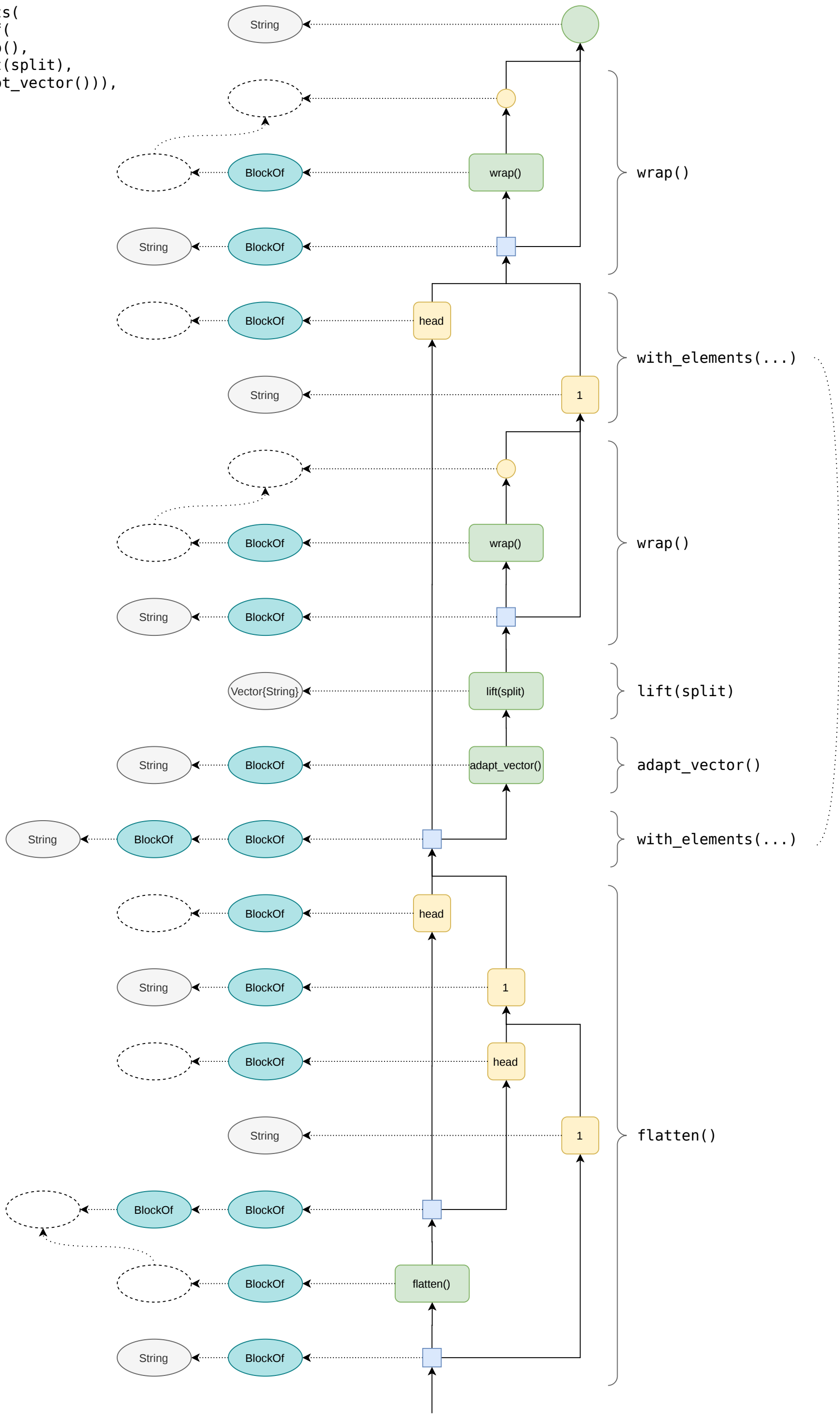


chain_of(wrap(), block_length())



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

