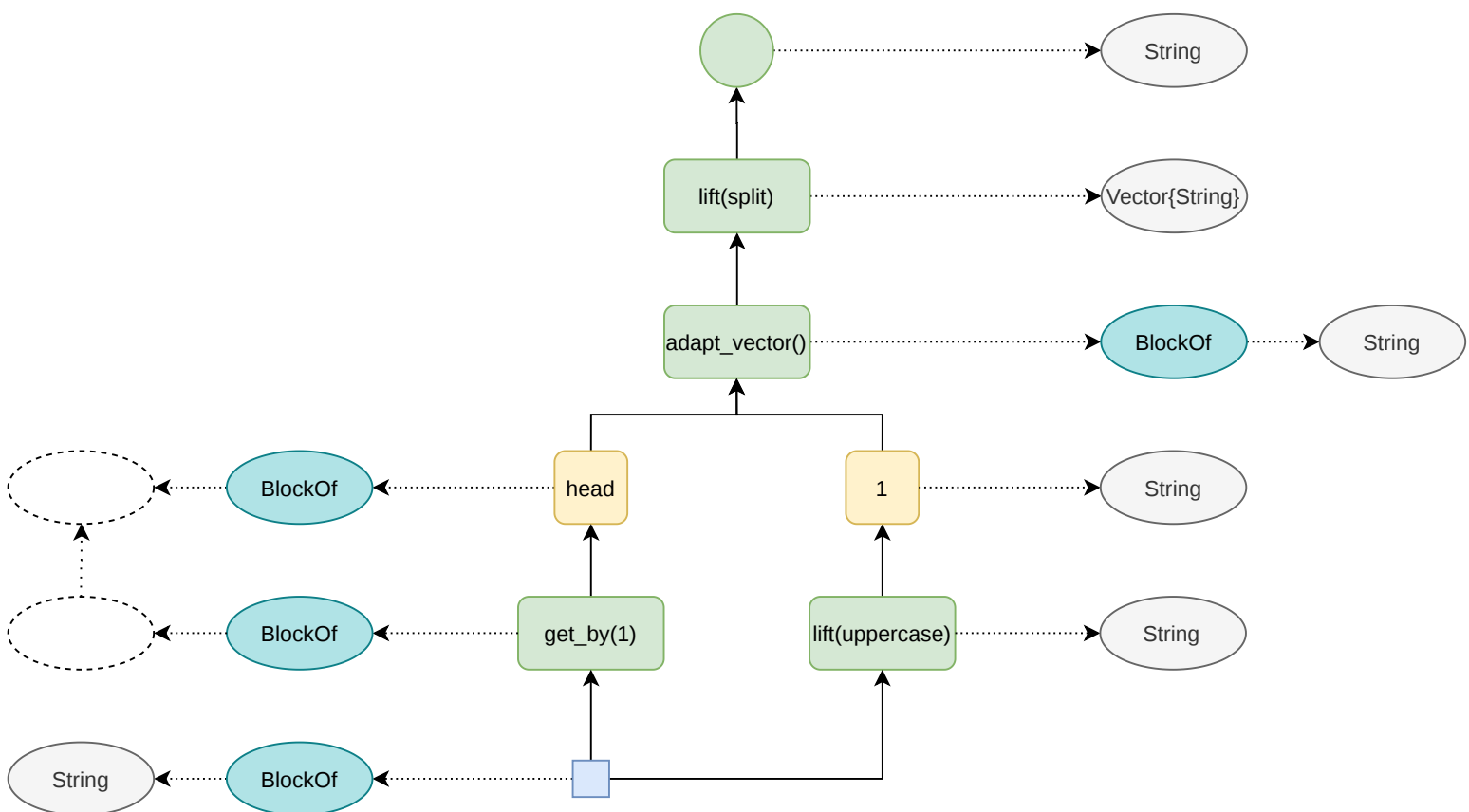
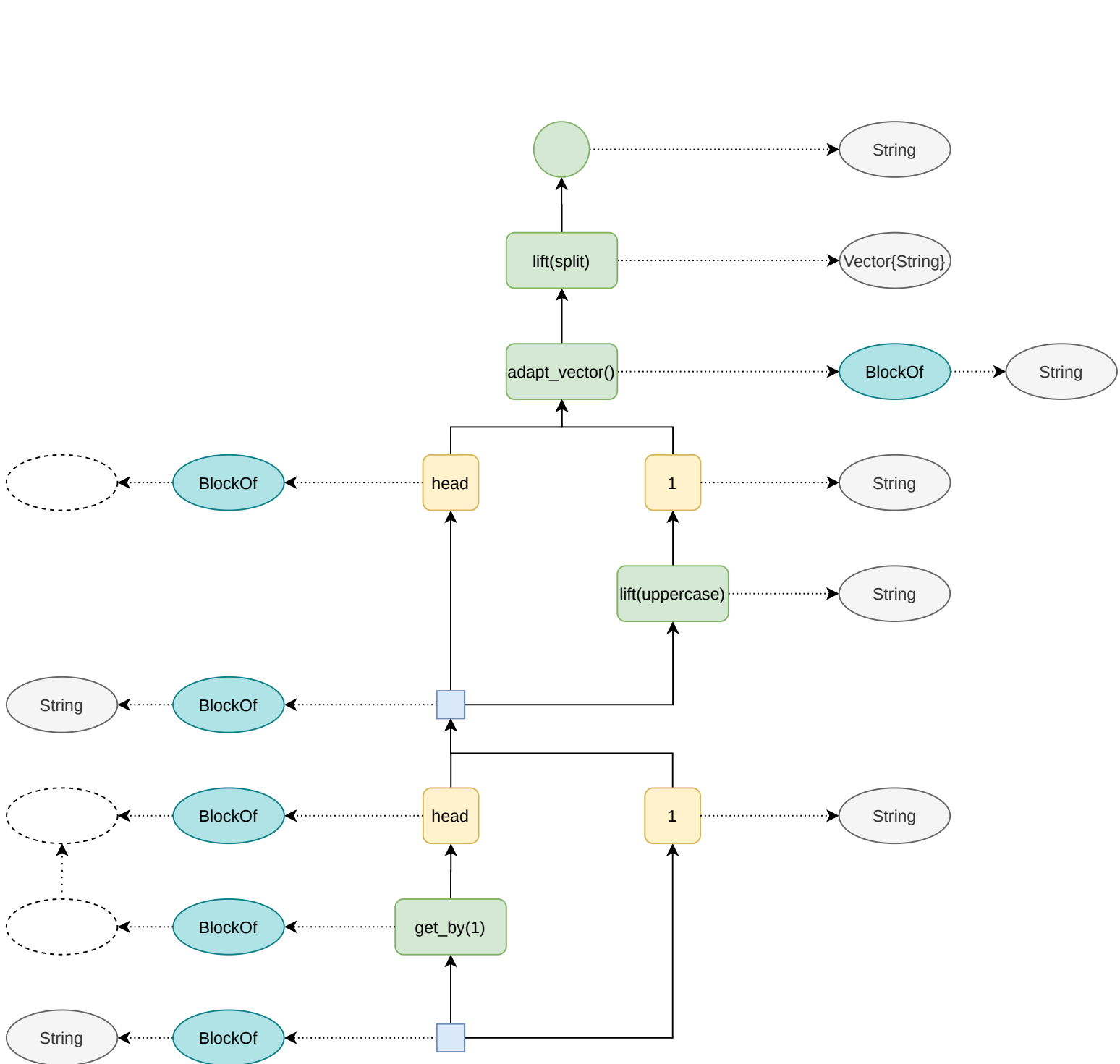
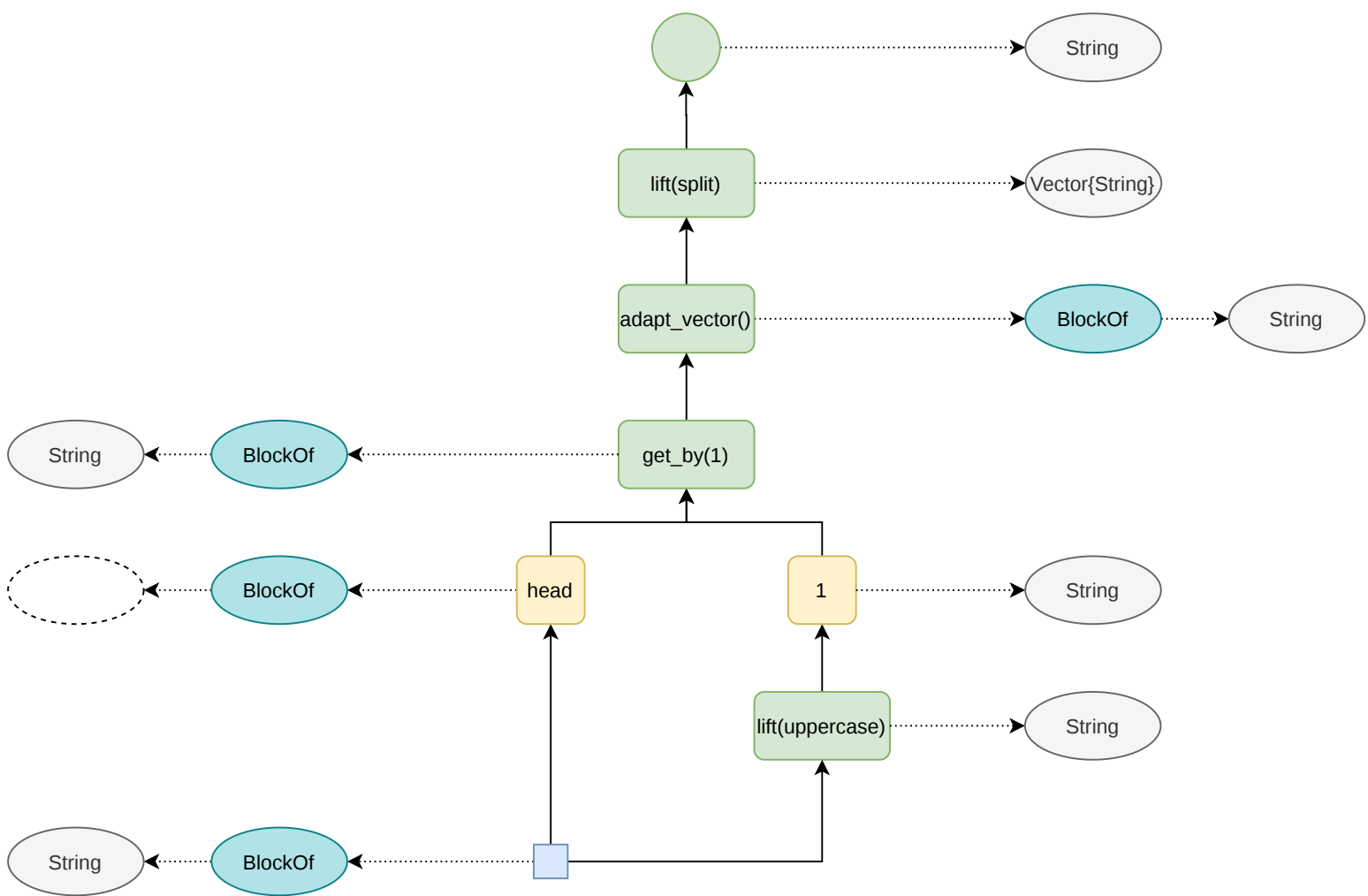
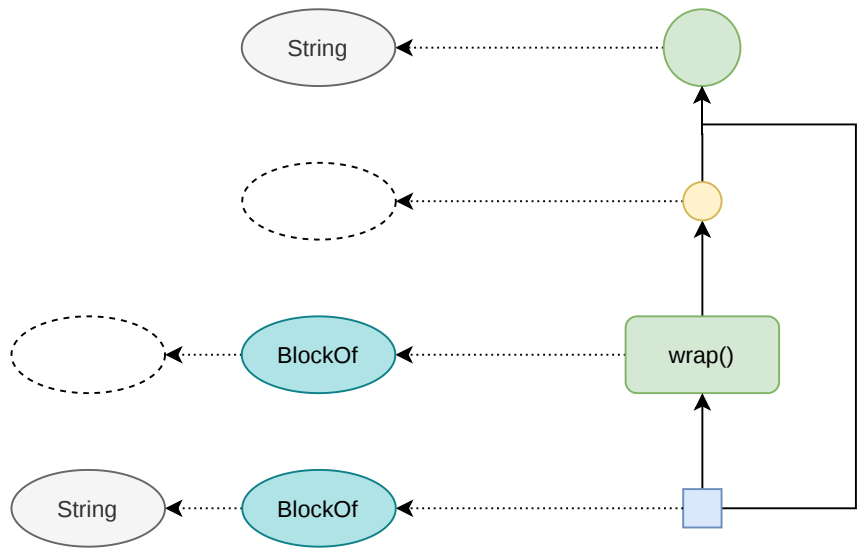
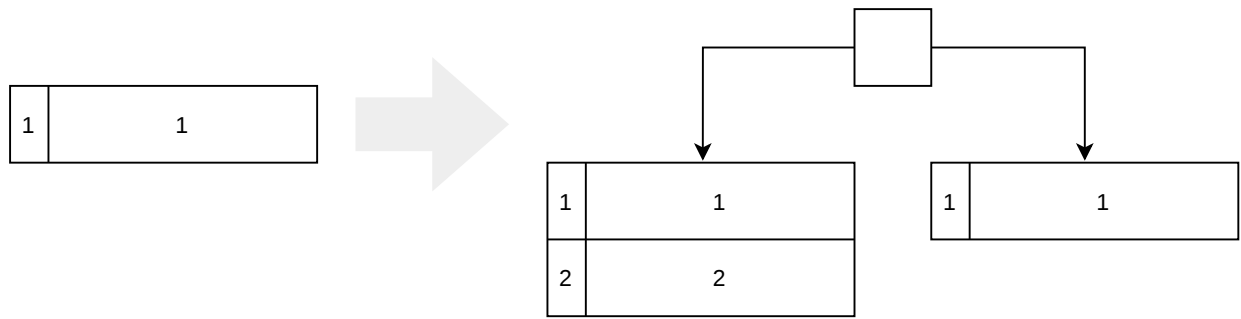
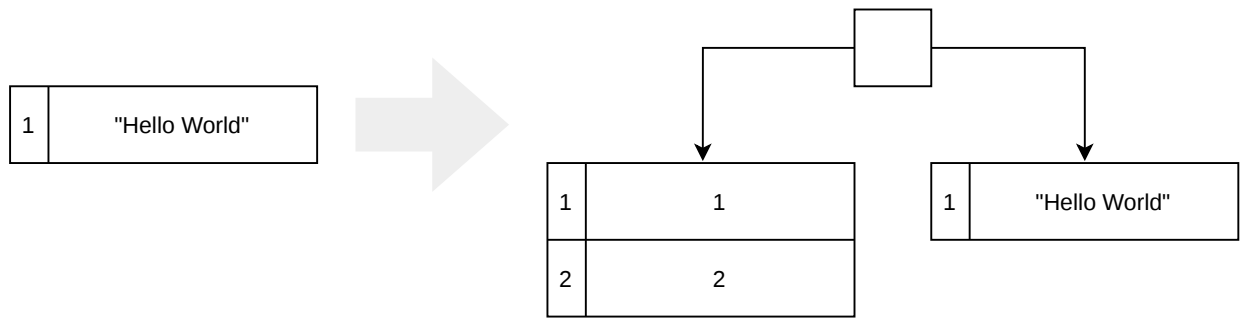


The diagram illustrates the transformation of a flat table into a hierarchical tree structure through a series of steps:

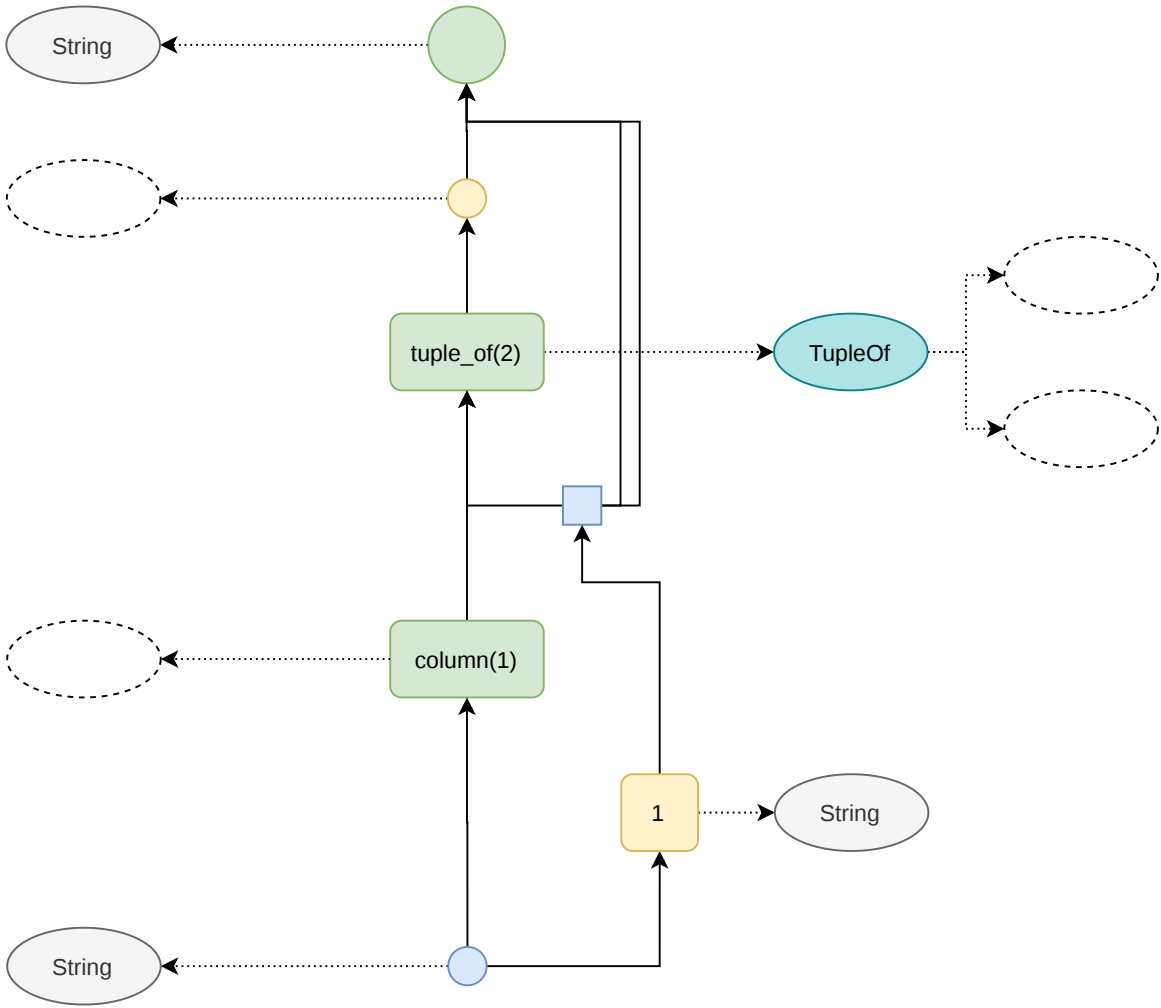
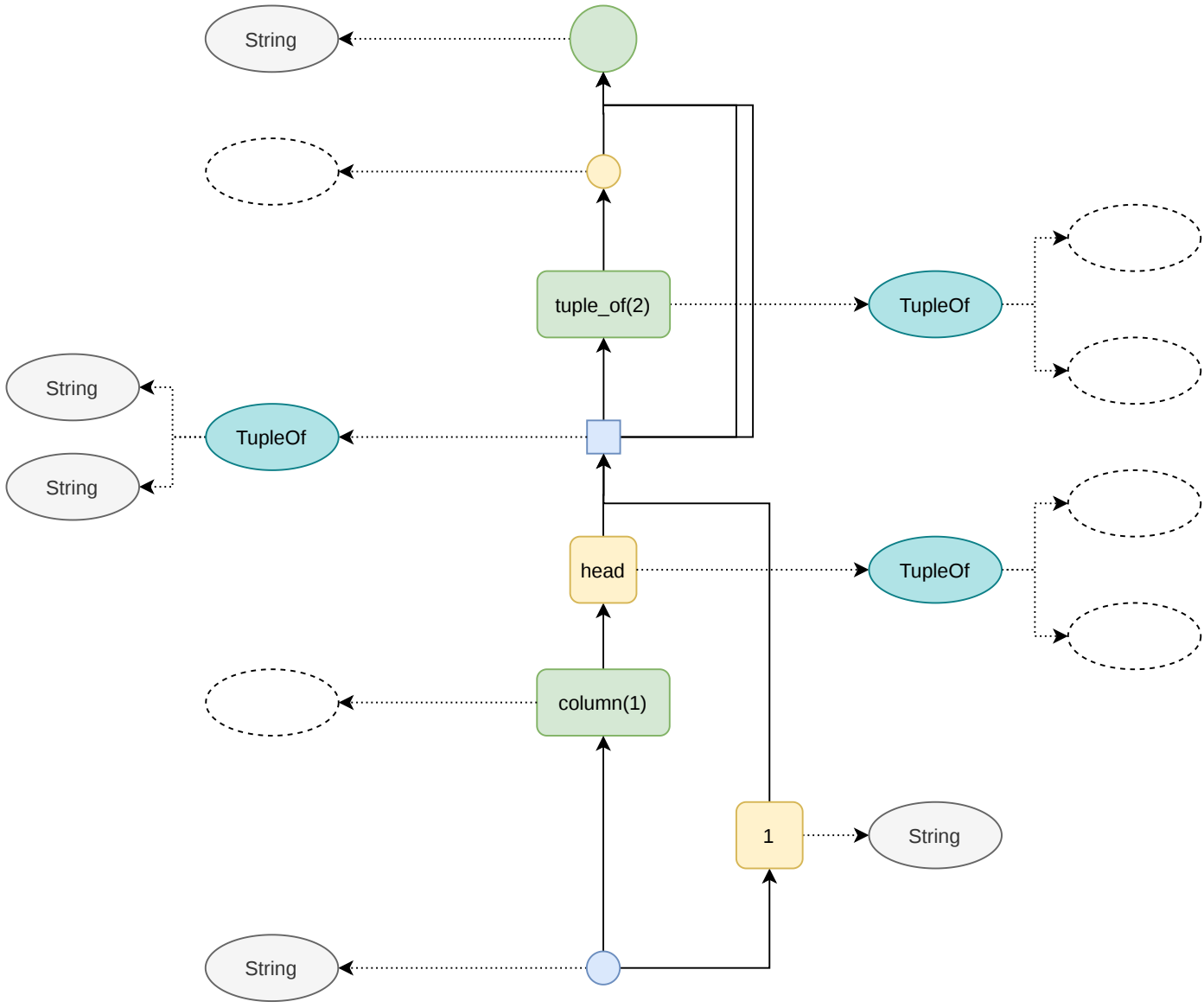
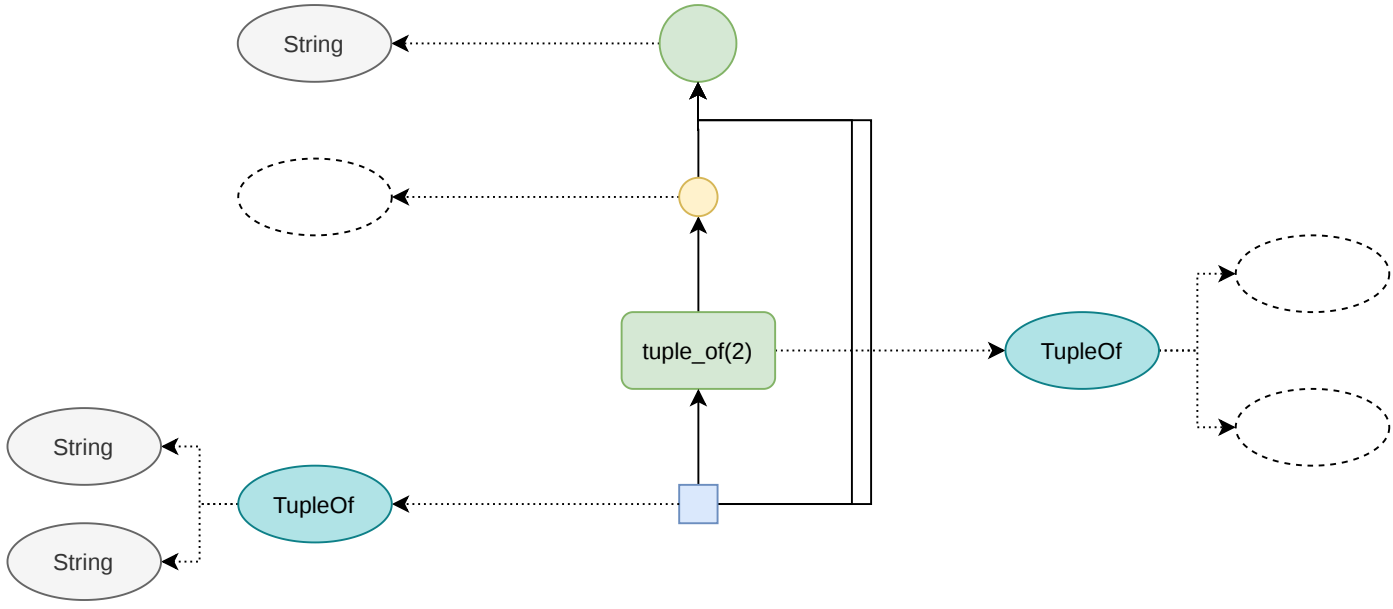
- Initial State:** A flat table with two columns: an index (1) and a value ("Hello World").
- Step 1:** The value is converted to a JSON string: `String["Hello", "World"]`.
- Step 2:** The string is parsed into a hierarchical structure. The root node branches into two children:
 - Child 1: A table with index 1 and value 1.
 - Child 2: A table with index 1 and value "Hello", and index 2 and value "World".
- Step 3:** The values are converted to uppercase:
 - Child 1: A table with index 1 and value 1.
 - Child 2: A table with index 1 and value "HELLO", and index 2 and value "WORLD".
- Step 4:** The structure is further transformed. The root node branches into two children:
 - Child 1: A table with index 1 and value 1, and index 2 and value 2.
 - Child 2: A table with index 1 and value "HELLO".
- Step 5:** The values are converted to lowercase:
 - Child 1: A table with index 1 and value 1, and index 2 and value 2.
 - Child 2: A table with index 1 and value "hello".
- Step 6:** The structure is further transformed. The root node branches into two children:
 - Child 1: A table with index 1 and value 1, and index 2 and value 3.
 - Child 2: A table with index 1 and value 1, and index 2 and value 2.
- Step 7:** The final structure is a tree where the root node branches into two children:
 - Child 1: A table with index 1 and value 1, and index 2 and value 2.
 - Child 2: A table with index 1 and value 1.



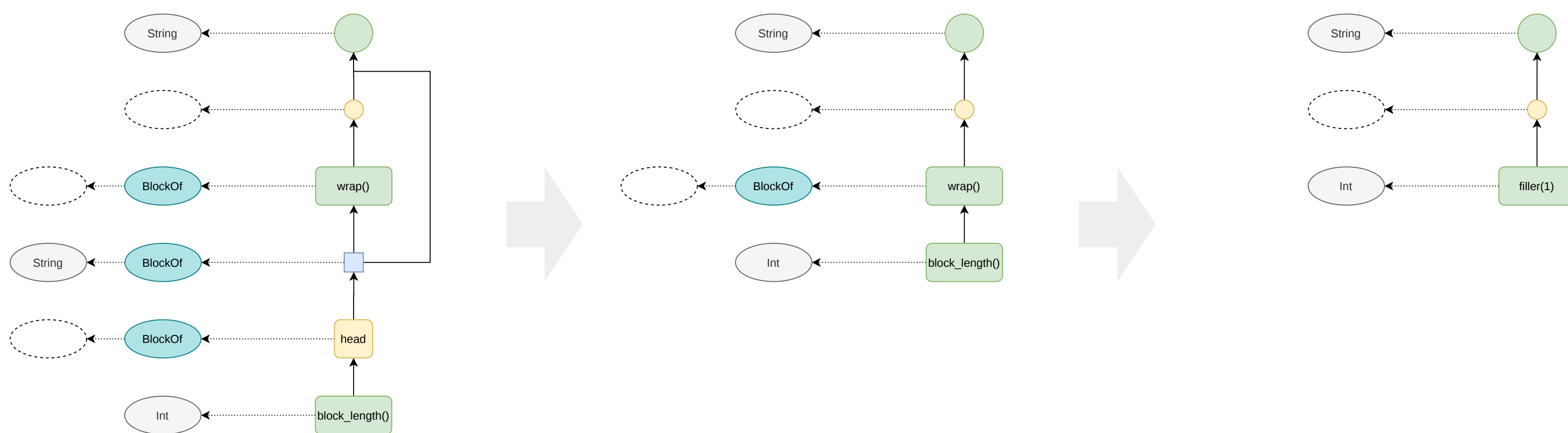
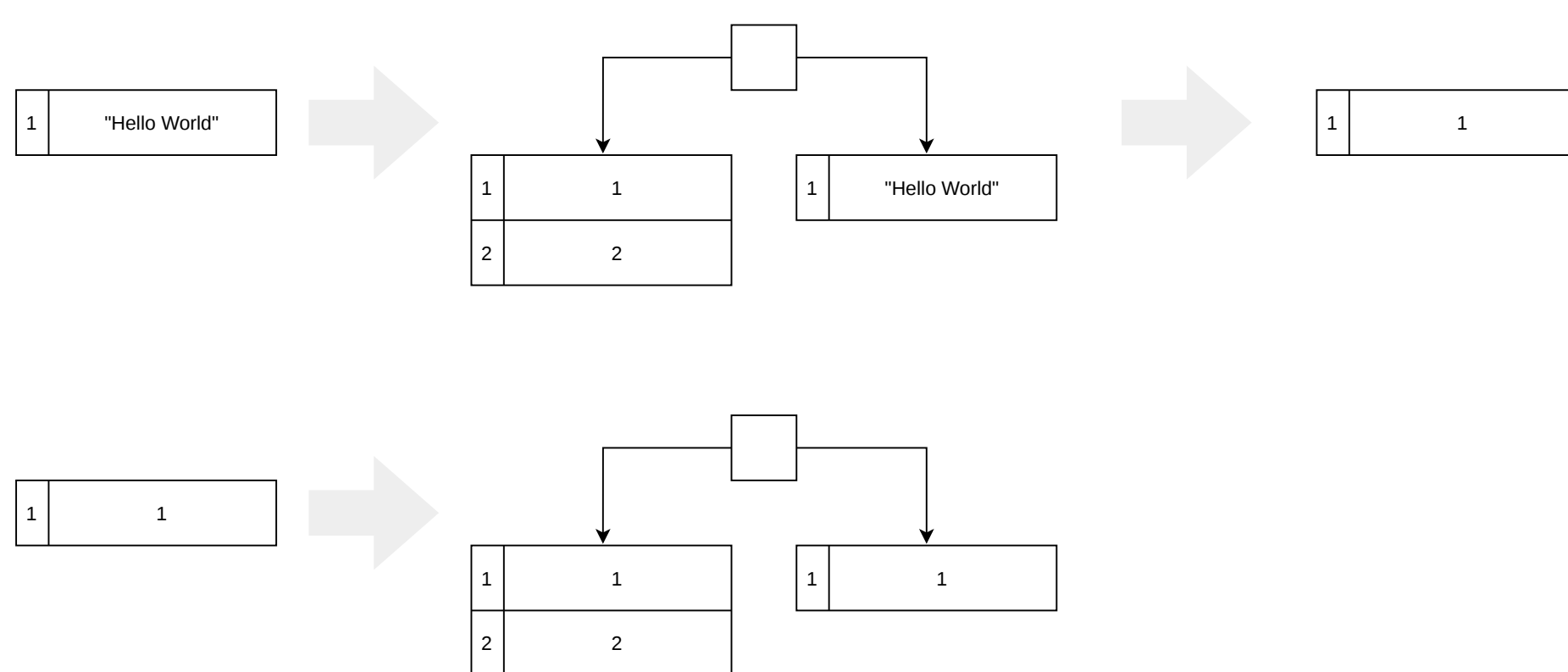
wrap()



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

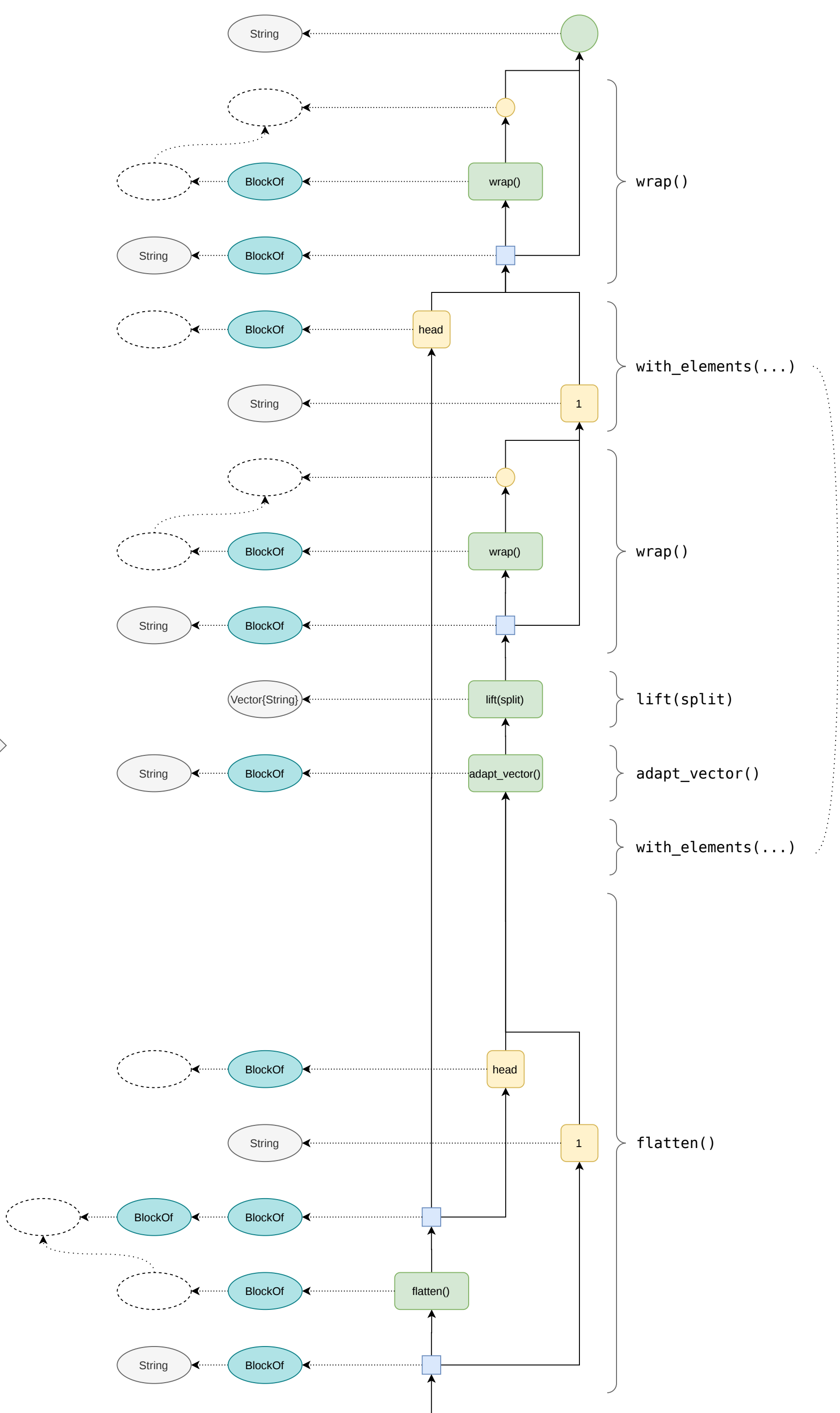
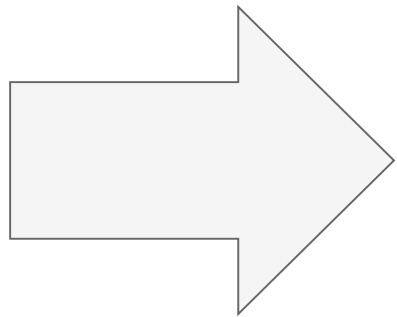
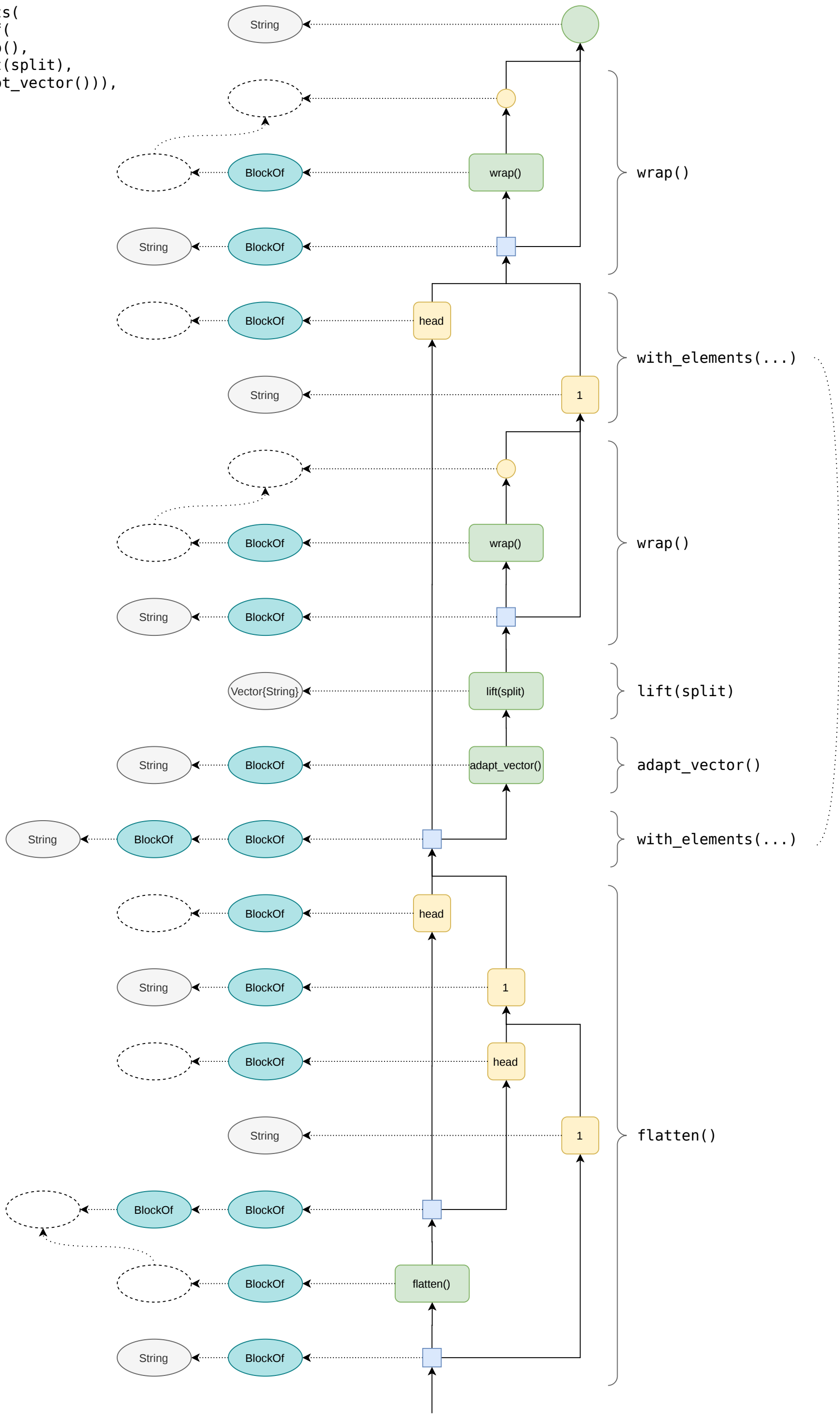


```
chain_of(wrap(), block_length())
```



@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())
```

