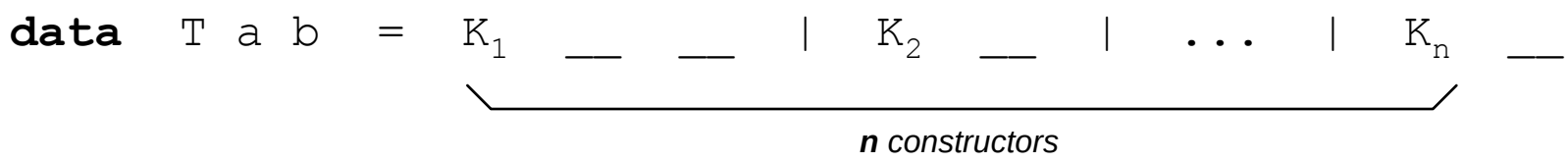
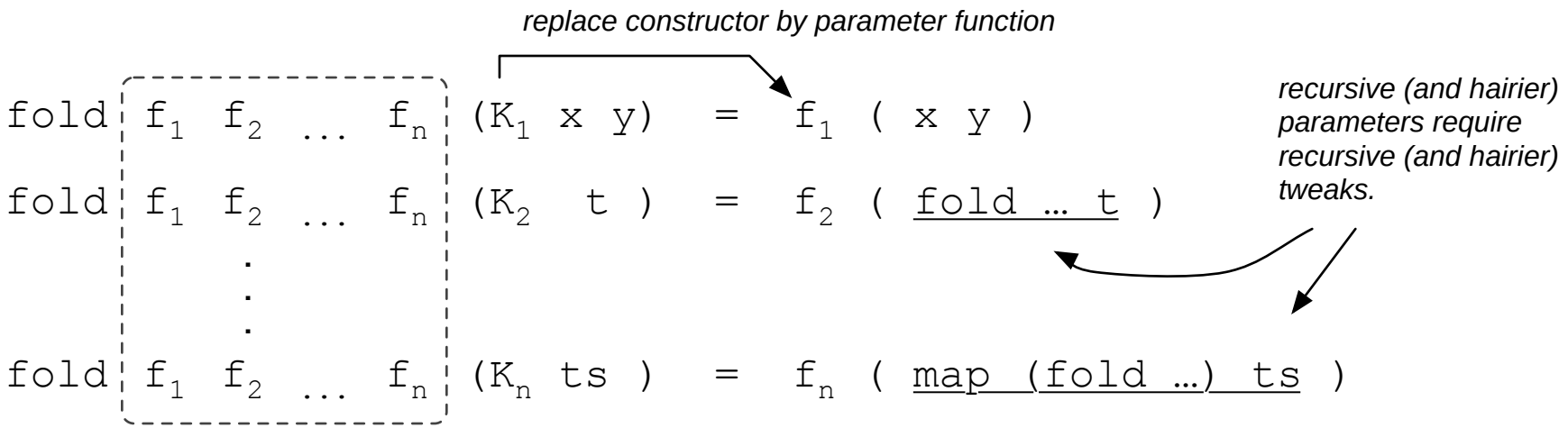
A data type definition defines a type (T), possibly with some parameters ('component' or element types; here a and b).

Values of the new type are being built from one of several constructors (the K's), each of which takes zero or more arguments.



There are several types of argument types which a constructor may take:
- **none** at all (a 'constant' constructor)
- a **fixed type** (such as Int)
- a **parameter** (such as a or b)
- a **recursive use** of T a b
- something **hairier** (e.g., a list of Ts, or even a T of non-a/b's, or both ...)

The fold function for a data type T takes n parameter functions (where n is the number of constructors T has) and a **single parameter** (typically a pattern) of type T. It also has n clauses, where in each clause the parameter of type T has a different constructor.

The *result* in each clause looks the same as the pattern for the T parameter, but the constructor **has been replaced** by a call to the corresponding function, on **possibly tweaked** parameters ("none", "fixed" and "parameter" arguments stay the same; "recursive" arguments are placed by recursive fold calls (with all function parameters); hairier ones are harder).