Vanilla Haskell: Values, types and kinds are strictly separated.

Values
- `5 :: Int`
- `True :: Bool`
- `λx → not x :: Bool → Bool`
- `Just True :: Maybe Bool`

Types have kinds
- `*` (kinds)
- `* → *` (functions)
- `(a → *), *` (booleans)
- `(a → a) → *` (functions)
- `(a → a) → *` (functions)
- `Either (a → a) (b → b) → *` (unions)
- `Bool :: *`
- `[a] :: *` (lists)
- `Maybe (a → a) :: *` (optionals)
- `Int :: *`

With data kinds, the separation is lifted:

Values have types
- `VNil :: Vector n`
- `VCons True VNil :: Vector (S n) Bool`
- `S :: Int`
- `True :: Bool`

Values can appear at type position
- `Vec s (VNil (S n)) Bool`
- `Vec s (VCons True VNil) Bool`
- `Vec s (VCons True VNil) Bool`

Types at kind positions
- `*` (kinds)
- `* → *` (functions)
- `(a → *)` (booleans)
- `(a → a) → *` (functions)
- `(a → a) → *` (functions)
- `Either (a → a) (b → b) → *` (unions)
- `Rel (S n) Bool`
- `Vec s (VNil (S n)) Bool`
- `Vec s (VCons True VNil) Bool`