```ts
const len = (s: string): number => s.length
const isZero = (n: number): boolean => n === 0

function pipe<A, B, C>(
  f1: (x: A) => B,
  f2: (x: B) => C
) {
  return (arg: A) => f2(f1(arg))
}

const isEmpty = pipe(len, isZero)
const res = isEmpty('hello')
```

```ts
function fuji<TS extends RuleType, A, B = A>(r1: Rule<TS, A, B>):
Fuji<TS, B>

function fuji<TS extends RuleType, A, B = A, C = B>(
  r1: Rule<TS, A, B>,
  r2: Rule<TS, B, C>
): Fuji<TS, C>

function fuji<TS extends RuleType, A, B = A, C = B, D = C>(
  r1: Rule<TS, A, B>,
  r2: Rule<TS, B, C>,
  r3: Rule<TS, C, D>
): Fuji<TS, D>

function fuji<TS extends RuleType, A, B = A, C = B, D = C, E = D>(
  r1: Rule<TS, A, B>,
  r2: Rule<TS, B, C>,
  r3: Rule<TS, C, D>,
  r4: Rule<TS, D, E>
): Fuji<TS, E>

function fuji<TS extends RuleType, A, B = A, C = B, D = C, E = D, F =
E>(
  r1: Rule<TS, A, B>,
  r2: Rule<TS, B, C>,
  r3: Rule<TS, C, D>,
  r4: Rule<TS, D, E>,
  r5: Rule<TS, E, F>
): Fuji<TS, F>
```