Functions—Take 2

SetA → SetB

Functions—Take 2

(a : SetA) → SetB

Functions—Take 2

(a: SetA) → SetB

Dependent Functions

(a : SetA) → SetB a

Dependent Functions

$(a : \mathsf{SetA}) \to \mathsf{SetB}\ a$

Dependent Functions

(a : SetA) → SetB a

Dependent Functions

(a : SetA) → SetB a

SetB : ? → ?

Dependent Functions

(a : SetA) → SetB a

SetB : ?

Dependent Functions

(a : SetA) → SetB a

SetB : ? → ?

Dependent Functions

(a : SetA) → SetB a

SetB : ? → ?

Dependent Functions

$(a : \mathrm{SetA}) \to \mathrm{SetB}\ a$

$\mathrm{SetB} : \mathrm{SetA} \to ?$

Dependent Functions

(a : SetA) → SetB a

SetB : SetA → ?

Dependent Functions

(a : SetA) → SetB a

SetB : SetA → ?

```
data SetName : Set where
```

Dependent Functions

(a : SetA) → SetB a

SetB : SetA → ?

data SetName : Set where

Dependent Functions

(a : SetA) → SetB a

SetB : SetA → Set

data SetName : Set where

Dependent Functions

(a : SetA) → SetB a

SetB : SetA → Set

```
data SetName : Set where
```

Dependent Types

(a : SetA) → SetB a

SetB : SetA → Set

data SetName : Set where

Dependent Types

(a : SetA) → SetB a

SetB : SetA → Set

data SetB : Set where

Dependent Types

(a : SetA) → SetB a

SetB : SetA → Set

```
data SetB : Set where
```

Dependent Types

(a : SetA) → SetB a

SetB : SetA → Set

data SetB : Set where

Dependent Types

(a : SetA) → SetB a

SetB : SetA → Set

data SetB : SetA → Set where

Dependent Types

(a : SetA) → SetB a

SetB : SetA → Set

data SetB : SetA → Set where

Dependent Types

(a : SetA) → SetB a

data SetB : SetA → Set where

Dependent Types

```
(a : SetA) → SetB a
```

```
data SetB : SetA → Set where
  constructor1…
```

Dependent Types

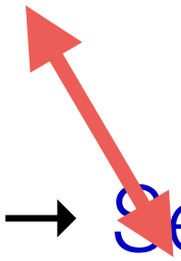
```
(a : SetA) → SetB a
```

```
data SetB : SetA → Set where
  constructor1 : SetB
```

Dependent Types

```
(a : SetA) → SetB a

data SetB : SetA → Set where
  constructor1 : SetB
```



Dependent Types

```
(a : SetA) → SetB a

    data SetA : Set where cstrA : SetA

    data SetB : SetA → Set where

      constructor1 : SetB
```

Dependent Types

```
(a : SetA) → SetB a

    data SetA : Set where cstrA : SetA

    data SetB : SetA → Set where

      constructor1 : SetB cstrA
```

Dependent Types

```
(a : SetA) → SetB a
```

```
data SetB : SetA → Set where
  constructor1 : (a : SetA) → SetB a
```

Dependent Types

(a : SetA) → SetB a

```
data SetB : SetA → Set where
  constructor1 : (a : SetA) → SetB a
```

Dependent Types

```
data SetB : SetA → Set where
  constructor1 : (a : SetA) → SetB a
```

Dependent Types

```
data SetA : Set where cstrA : SetA
data SetB : SetA → Set where
  constructor1 : SetB cstrA
```

Dependent Types

```
data SetA : Set where cstrA : SetA

data SetB : SetA → Set where
  constructor1 : SetB cstrA
```

```
value : SetB cstrA
```

Dependent Types

```
data SetA : Set where cstrA : SetA
data SetB : SetA → Set where
  constructor1 : SetB cstrA
```

```
value : SetB cstrA
value = constructor1
```

Dependent Types

```
data SetA : Set where cstrA : SetA

data SetB : SetA → Set where

  constructor1 : (a : SetA) → SetB a
```

Dependent Types

```
data SetA : Set where cstrA : SetA

data SetB : SetA → Set where

  constructor1 : (a : SetA) → SetB a
```

```
value : SetB cstrA

value = constructor1 cstrA
```

Dependent Types

```
data SetA : Set where cstrA : SetA
data SetB : SetA → Set where
  constructor1 : (a : SetA) → SetB a

value : SetB cstrA
value = constructor1 cstrA
```

Dependent Types

```
data SetA : Set where cstrA : SetA

data SetB : SetA → Set where
  constructor1 : (a : SetA) → SetB a
```

```
value : SetB cstrA
value = constructor1 cstrA
```

Dependent Functions

(a : SetA) → SetB a → SetC

Dependent Functions

(a : SetA) → (b : SetB a) → SetC

Dependent Functions

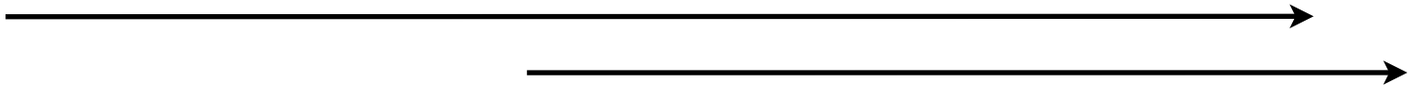
(a : SetA) → (b : SetB a) → SetC b

Dependent Functions

(a : SetA) → (b : SetB a) → SetC a

Dependent Functions

(a : SetA) → (b : SetB a) → SetC a b



Dependent Functions

(a : SetA) → (b : SetB a) → SetC a b

SetC : ?

Dependent Functions

(a : SetA) → (b : SetB a) → SetC a b

SetC : ? → ? → ?

Dependent Functions

(a : SetA) → (b : SetB a) → SetC a b

SetC : SetA → ? → ?

Dependent Functions

(a : SetA) → (b : SetB a) → SetC a b

SetC : SetA → SetB → ?

Dependent Functions

(a : SetA) → (b : SetB a) → SetC a b

SetC : SetA → SetB → Set

Dependent Functions

(a : SetA) → (b : SetB a) → SetC a b

SetC : SetA → SetB → Set

Dependent Types

```
data Name : (a : A) → (b : B a) → Set where
  constructor1 : (a : A) → (b : B a) → Name a b
  constructor2 : (a : A) → (b : B a) → Name a b
```

Dependent Types

```
data Name : (a : A) → (b : B a) → Set where
  constructor1 : (a : A) → (b : B a) → Name a b
  constructor2 : (a : A) → (b : B a) → Name a b
```

Dependent Types

```
data Name : (a : A) → (b : B a) → Set where
  constructor1 : (a : A) → (b : B a) → Name a b
  constructor2 : (a : A) → (b : B a) → Name a b
```

Dependent Types

```
data Name : (a : A) → (b : B a) → Set where
  constructor1 : (a : A) → (b : B a) → Name a b
  constructor2 : (a : A) → (b : B a) → Name a b
```

Dependent Types

```
data Name (a : A) : (b : B a) → Set where
  constructor1 : (b : B a) → Name a b
  constructor2 : (b : B a) → Name a b
```

Dependent Types

```
data Name (a : A) : (b : B a) → Set where
  constructor1 : (b : B a) → Name a b
  constructor2 : (b : B a) → Name a b
```

Constructors Limit Indexes

```
data Name : (a : A) → (b : B a) → Set where
  constructor1 : (a : A) → (b : B a) → Name a b
  constructor2 : (a : A) → (b : B a) → Name a b
```