

Beyond System T:

Adding other data types

There are two kinds of types

## Inductive Types

- Well-founded structure
- Elements constructed from bottom up
- Examples: Nat, Binary Trees

## CoInductive Types

- Non-well-founded structure
- Elements generated from top down

Example of CoInductive Type

Streams of bits

sequences of 0s and 1s that go on forever

## Introduction Rules

It's not enough to give just the constructors, as for inductive types:

s: BitStream

s: BitStream

0 as: BitStream

1 as: BitStream

There is no base case:

how do we start building a stream?

We need a rule to generate streams dynamically

# Introduction Rule expressing

## CoRecursion

For any type  $X$  ( $\text{Bit} = \{0, 1\}$ )

$$f: X \rightarrow \text{Bit} \quad t: X \rightarrow X$$

$$\text{corec } f \ t : X \rightarrow \text{BitStream}$$

## Elimination Rules (Observation)

$$\frac{s: \text{BitStream} \quad s: \text{BitStream}}{\text{bit } s: \text{Bit} \quad \text{next } s: \text{BitStream}}$$

## Reduction Rules:

$$\text{bit } (\text{corec } f \ t \ x) \rightsquigarrow f \ x$$

$$\text{next } (\text{corec } f \ t \ x)$$

$$\rightsquigarrow \text{corec } f \ t \ (t \ x)$$

## Examples:

$$\text{alternate} : \text{Bit} \rightarrow \text{BitStream}$$

$$\text{alternate} = \text{corec } \text{id} \ \text{flip}$$

where

id is the identity function on Bit

$$\text{flip} : \text{Bit} \rightarrow \text{Bit}$$

$$\text{flip } 0 = 1$$

$$\text{flip } 1 = 0$$

intuitively

$$\text{alternate } 0 = 0 \blacktriangleleft 1 \blacktriangleleft 0 \blacktriangleleft 1 \blacktriangleleft \dots$$

$$\text{alternate } 1 = 1 \blacktriangleleft 0 \blacktriangleleft 1 \blacktriangleleft 0 \blacktriangleleft \dots$$

Imagine corecursion as a dynamic

process:

$$f: X \rightarrow \text{Bit}$$

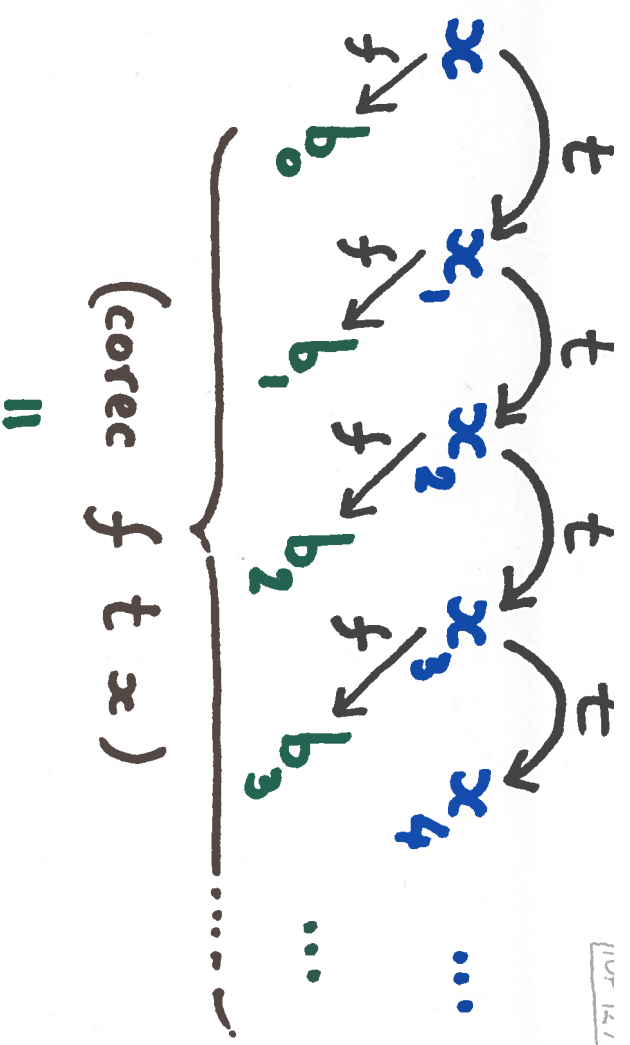
↑ a set of "process states"

function producing a bit in the current state

$$t: X \rightarrow X$$

transition function: after producing a bit, the state of the process changes from  $x$  to  $t(x)$

(corec  $f \ t \ x$ ) is the stream of elements generated in output by the process



$$b_0 \triangleright b_1 \triangleright b_2 \triangleright b_3 \triangleright \dots$$

The pair of functions  $\langle f, t \rangle$  is called a **coalgebra**

We'll see a general theory of coalgebras in a later lecture

Example:

Stream of n-repetitions of 1 separated by 0:

01<sup>0</sup>01<sup>1</sup>01<sup>2</sup>01<sup>3</sup>01<sup>4</sup>0 ...

= 0010110111011110 ...

We use as states pairs of numbers

Intuitively a state  $\langle n, m \rangle$

generate the stream

1<sup>m</sup> 0 1<sup>n</sup> 0 1<sup>n+1</sup> 0 1<sup>n+2</sup> ...

The whole stream we want

is generated by  $\langle 0, 0 \rangle$

~~X~~ = Nat x Nat

(We'll define product types next lecture)

f: X → Bit

f  $\langle n, 0 \rangle$  = 0

f  $\langle n, S_m \rangle$  = 1

t: X → X

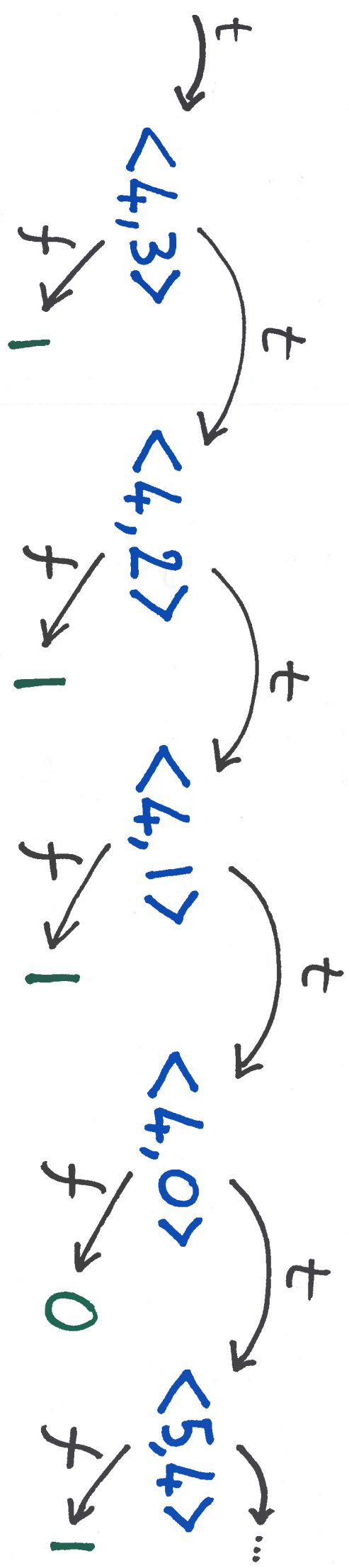
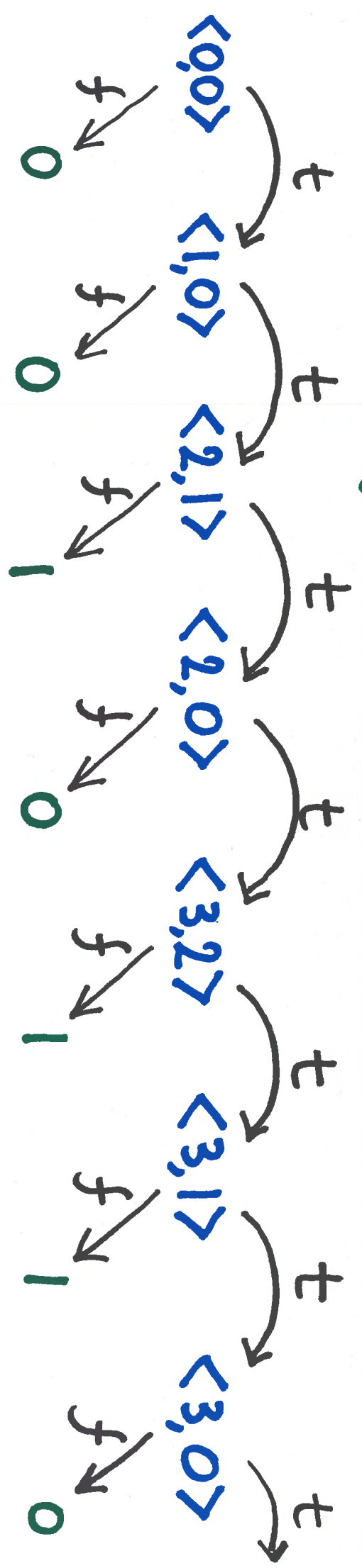
t  $\langle n, 0 \rangle$  =  $\langle n+1, n \rangle$

t  $\langle n, S_m \rangle$  =  $\langle n, m \rangle$

The stream we want is

corec f t  $\langle 0, 0 \rangle$

If we see it as a generating process:



Very often, when we want to define a stream, we need to find a general type X of states that generates all the intermediate streams.