Judgments, Inference Rules, and Inductive Definitions

Stephen A. Edwards

Columbia University

Spring 2023

There are various symbols

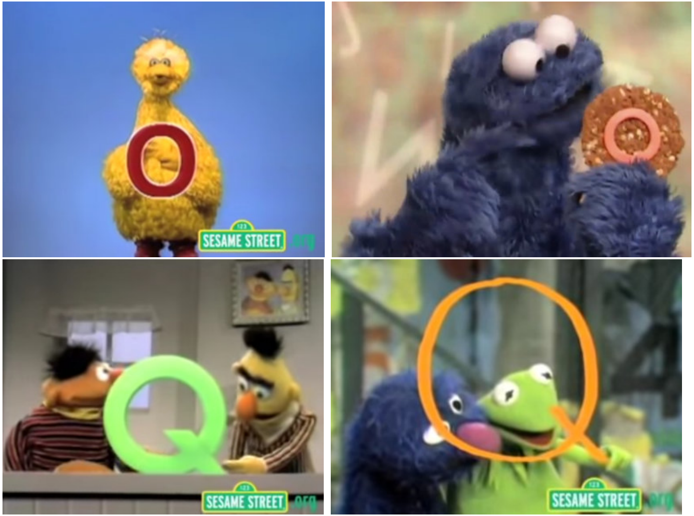
There are various symbols

Symbols may be identical, even when drawn slightly differently

There are various symbols

Symbols may be identical, even when drawn slightly differently

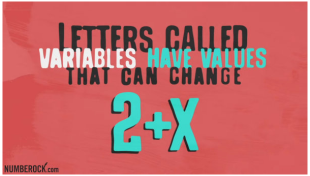
Other symbols are distinct



There are various symbols
Symbols may be identical, even when drawn slightly differently
Other symbols are distinct
Symbols arranged in a horizontal sequence are “words,” “strings,” or “expressions”

There are various symbols
Symbols may be identical, even when drawn slightly differently
Other symbols are distinct
Symbols arranged in a horizontal sequence are “words,” “strings,” or “expressions”
Symbols may represent values, operations, or relationships

There are various symbols
Symbols may be identical, even when drawn slightly differently
Other symbols are distinct
Symbols arranged in a horizontal sequence are “words,” “strings,” or “expressions”
Symbols may represent values, operations, or relationships
Some symbols are treated as variables that represent other symbols



There are various symbols

Symbols may be identical, even when drawn slightly differently

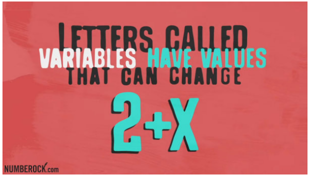
Other symbols are distinct

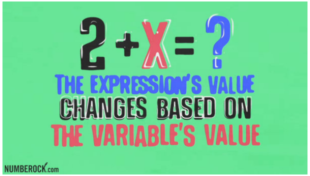
Symbols arranged in a horizontal sequence are “words,” “strings,” or “expressions”

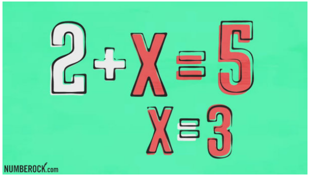
Symbols may represent values, operations, or relationships

Some symbols are treated as variables that represent other symbols

The meaning of an expression with variables depends on the variables’ values







Judgment

A *judgment* is an assertion about one or more things, typically membership in a set.

$0 \in \mathbb{N}$	0 is a member of the set of natural numbers
n nat	n is a member of the set of natural numbers
$1 + 2$ expr	$1 + 2$ is in the set of expressions
τ type	τ is in the set of types
$e : \tau$	Expression e has type τ
$\mathsf{sum}(n_1, n_2, n_3)$	Adding n_1 and n_2 gives n_3
$n_1 + n_2 = n_3$	Adding n_1 and n_2 gives n_3

Prefix; infix; and suffix syntax

Inference Rule

Premises: Judgments $\longrightarrow$
Conclusion: A Judgment $\longrightarrow$

$$\frac{\mathcal{J}_1 \quad \mathcal{J}_2 \quad \cdots \quad \mathcal{J}_k}{\mathcal{J}}\,\text{Rule-Name}$$

"If all the premises hold, the conclusion follows"

The Natural Numbers Defined Inductively by Two Inference Rules (Peano)

Axiom $\longrightarrow$ $\dfrac{}{\mathtt{0} \in \mathbb{N}}\,\text{zero}$

Judgments: $a \in \mathbb{N}$
Variables: a $\longleftarrow$ Sequences of symbols
Symbols: `0` `succ(` `)`

`0`	0
`succ(0)`	1
`succ(succ(0))`	2
`succ(succ(succ(0)))`	3
`succ(succ(succ(succ(0))))`	4

Inference Rule

Premises: Judgments $\longrightarrow$
Conclusion: A Judgment $\longrightarrow$

$$\frac{\mathcal{J}_1 \qquad \mathcal{J}_2 \qquad \cdots \qquad \mathcal{J}_k}{\mathcal{J}}\text{Rule-Name}$$

"If all the premises hold, the conclusion follows"

The Natural Numbers Defined Inductively by Two Inference Rules (Peano)

$$\frac{}{\mathtt{0} \in \mathbb{N}}\text{zero} \qquad \frac{a \in \mathbb{N}}{\mathtt{succ(}a\mathtt{)} \in \mathbb{N}}\text{successor}$$

Judgments: $a \in \mathbb{N}$
Variables: a $\longleftarrow$ Sequences of symbols
Symbols: `0` `succ(` `)`

`0`	0
`succ(0)`	1
`succ(succ(0))`	2
`succ(succ(succ(0)))`	3
`succ(succ(succ(succ(0))))`	4

Inference Rule

$$\text{Premises: Judgments} \longrightarrow \quad \frac{\mathcal{J}_1 \qquad \mathcal{J}_2 \qquad \cdots \qquad \mathcal{J}_k}{\mathcal{J}}\,\text{Rule-Name}$$
$$\text{Conclusion: A Judgment} \longrightarrow$$

"If all the premises hold, the conclusion follows"

The Natural Numbers Defined Inductively by Two Inference Rules (Peano)

$$\frac{}{\mathtt{0} \in \mathbb{N}}\,\text{zero} \qquad \frac{a \in \mathbb{N}}{\mathtt{succ}(a) \in \mathbb{N}}\,\text{successor} \longleftarrow \text{Technically a scheme}$$

Scheme: pattern with variables: replacing *a* consistently gives a rule

$$\frac{\mathtt{0} \in \mathbb{N}}{\mathtt{succ}(\mathtt{0}) \in \mathbb{N}} \qquad \frac{\mathtt{succ}(\mathtt{0}) \in \mathbb{N}}{\mathtt{succ}(\mathtt{succ}(\mathtt{0})) \in \mathbb{N}} \qquad \frac{\mathtt{true} \in \mathbb{N}}{\mathtt{succ}(\mathtt{true}) \in \mathbb{N}}$$

Which are variables? Values constrained? Variable scope: a single rule

Consistent replacement only:

$$\frac{\mathtt{foo} \in \mathbb{N}}{\mathtt{succ}(\mathtt{bar}) \in \mathbb{N}}$$

is not a rule

Inference Rule

$$\text{Premises: Judgments} \longrightarrow \quad \frac{\mathcal{J}_1 \quad \mathcal{J}_2 \quad \cdots \quad \mathcal{J}_k}{\mathcal{J}}\,\text{Rule-Name}$$
$$\text{Conclusion: A Judgment} \longrightarrow$$

"If all the premises hold, the conclusion follows"

The Natural Numbers Defined Inductively by Two Inference Rules (Peano)

$$\frac{}{\mathtt{0} \in \mathbb{N}}\,\text{zero} \qquad \frac{a \in \mathbb{N}}{\mathtt{succ}(a) \in \mathbb{N}}\,\text{successor}$$

Is succ(succ(succ(0))) a.k.a. 3 a natural number? A forward derivation

$$\frac{}{\mathtt{0} \in \mathbb{N}}\,\text{zero}$$

Inference Rule

$$\begin{array}{r} \text{Premises: Judgments} \longrightarrow \\ \text{Conclusion: A Judgment} \longrightarrow \end{array} \quad \dfrac{\mathcal{J}_1 \quad \mathcal{J}_2 \quad \cdots \quad \mathcal{J}_k}{\mathcal{J}}\,\text{Rule-Name}$$

"If all the premises hold, the conclusion follows"

The Natural Numbers Defined Inductively by Two Inference Rules (Peano)

$$\dfrac{}{\mathtt{0} \in \mathbb{N}}\,\text{zero} \qquad \dfrac{a \in \mathbb{N}}{\mathtt{succ}(a) \in \mathbb{N}}\,\text{successor}$$

Is $\mathtt{succ(succ(succ(0)))}$ a.k.a. 3 a natural number? A forward derivation

$$\dfrac{\dfrac{}{\mathtt{0} \in \mathbb{N}}\,\text{zero}}{\mathtt{succ(0)} \in \mathbb{N}}\,\text{successor} \quad \longleftarrow \text{choose } a = \mathtt{0}$$

Inference Rule

Premises: Judgments $\longrightarrow$

Conclusion: A Judgment $\longrightarrow$

$$\frac{\mathcal{J}_1 \quad \mathcal{J}_2 \quad \cdots \quad \mathcal{J}_k}{\mathcal{J}}\ \text{Rule-Name}$$

"If all the premises hold, the conclusion follows"

The Natural Numbers Defined Inductively by Two Inference Rules (Peano)

$$\frac{}{\mathtt{0} \in \mathbb{N}}\ \text{zero} \qquad \frac{a \in \mathbb{N}}{\mathtt{succ}(a) \in \mathbb{N}}\ \text{successor}$$

Is `succ(succ(succ(0)))` a.k.a. 3 a natural number? A forward derivation

$$\cfrac{\cfrac{\cfrac{}{\mathtt{0} \in \mathbb{N}}\ \text{zero}}{\mathtt{succ(0)} \in \mathbb{N}}\ \text{successor}}{\mathtt{succ(succ(0))} \in \mathbb{N}}\ \text{successor} \quad \longleftarrow \text{choose } a = \mathtt{succ(0)}$$

Inference Rule

Premises: Judgments $\longrightarrow$
Conclusion: A Judgment $\longrightarrow$

$$\frac{\mathcal{J}_1 \qquad \mathcal{J}_2 \qquad \cdots \qquad \mathcal{J}_k}{\mathcal{J}}\,\text{Rule-Name}$$

"If all the premises hold, the conclusion follows"

The Natural Numbers Defined Inductively by Two Inference Rules (Peano)

$$\frac{}{\mathtt{0} \in \mathbb{N}}\,\text{zero} \qquad\qquad \frac{a \in \mathbb{N}}{\mathtt{succ}(a) \in \mathbb{N}}\,\text{successor}$$

Is succ(succ(succ(0))) a.k.a. 3 a natural number? A forward derivation

$$\cfrac{\cfrac{\cfrac{\cfrac{}{\mathtt{0} \in \mathbb{N}}\,\text{zero}}{\mathtt{succ(0)} \in \mathbb{N}}\,\text{successor}}{\mathtt{succ(succ(0))} \in \mathbb{N}}\,\text{successor}}{\mathtt{succ(succ(succ(0)))} \in \mathbb{N}}\,\text{successor}$$

The Natural Numbers

$$\frac{}{\mathtt{0} \in \mathbb{N}}\ \text{zero} \qquad \frac{a \in \mathbb{N}}{\mathtt{succ}(a) \in \mathbb{N}}\ \text{successor}$$

```
zeroIsN      :: String -> Bool



successorIsN :: String -> Bool
```

`String` is inefficient, but let's focus on correctness first

The Natural Numbers

$$\frac{}{\mathtt{0} \in \mathbb{N}}\ \text{zero} \qquad\qquad \frac{a \in \mathbb{N}}{\mathtt{succ}(a) \in \mathbb{N}}\ \text{successor}$$

```
zeroIsN      :: String -> Bool
zeroIsN "0" = True
zeroIsN _   = False     -- Default case

successorIsN :: String -> Bool
```

The Natural Numbers

$$\frac{}{0 \in \mathbb{N}}\ \text{zero} \qquad \frac{a \in \mathbb{N}}{\mathtt{succ}(a) \in \mathbb{N}}\ \text{successor}$$

```haskell
import Data.List (stripPrefix) -- stripPrefix :: String -> String -> Maybe String
zeroIsN      :: String -> Bool
zeroIsN "0" = True
zeroIsN _   = False     -- Default case

successorIsN :: String -> Bool    -- Construct a Reverse Derivation
successorIsN s = case stripPrefix "succ(" s of  -- Of the form succ(...)?
   Just aa@(_:_) -> last aa == ')' &&           -- Prohibit the empty string
                    let a = init aa in          -- Get all but last character
                    zeroIsN a || successorIsN a -- Try both
   _             -> False
```

The Natural Numbers

$$\frac{}{0 \in \mathbb{N}}\ \text{zero} \qquad \frac{a \in \mathbb{N}}{\mathtt{succ}(a) \in \mathbb{N}}\ \text{successor}$$

```haskell
import Data.List (stripPrefix)
zeroIsN       :: String -> Bool
zeroIsN "0" = True
zeroIsN _   = False

successorIsN :: String -> Bool
successorIsN s = case match "succ(" ")" s of
  Just a -> zeroIsN a || successorIsN a
  _      -> False

match :: String -> String -> String -> Maybe String -- Helper function
match pre suff s = do a' <- stripPrefix pre s          -- Stops at Nothing
                      reverse <$> stripPrefix (reverse suff) (reverse a')
```

The Natural Numbers

$$\frac{}{\mathtt{0} \in \mathbb{N}}\ \text{zero} \qquad \frac{a \in \mathbb{N}}{\mathtt{succ}(a) \in \mathbb{N}}\ \text{successor}$$

```haskell
import Data.List (stripPrefix)

isNat :: String -> Bool                       -- Merge the two rules
isNat "0" = True                              -- zero rule
isNat s   = case match "succ(" ")" s of       -- successor rule
  Just a  -> isNat a                          -- Only one thing to check
  Nothing -> False




match :: String -> String -> String -> Maybe String
match pre suff s = do a' <- stripPrefix pre s
                      reverse <$> stripPrefix (reverse suff) (reverse a')
```

The Natural Numbers

$$\frac{}{\mathtt{0} \in \mathbb{N}}\ \text{zero} \qquad \frac{a \in \mathbb{N}}{\mathtt{succ}(a) \in \mathbb{N}}\ \text{successor}$$

```
data Nat = Zero | Succ Nat   -- Algebraic data type: either "Zero" or "Succ n"

zeroIsN      :: Nat -> Bool
zeroIsN Zero = True
zeroIsN _    = False

successorIsN          :: Nat -> Bool
successorIsN (Succ a) = zeroIsN a || successorIsN a -- Try both
successorIsN _        = False
```

The Natural Numbers

$$\frac{}{\mathtt{0} \in \mathbb{N}}\ \text{zero} \qquad \frac{a \in \mathbb{N}}{\mathtt{succ}(a) \in \mathbb{N}}\ \text{successor}$$

```haskell
data Nat = Zero | Succ Nat

isNat              :: Nat -> Bool
isNat Zero         = True          -- zero rule
isNat (Succ a) = isNat a       -- successor rule
```

`isNat` is trivial; Haskell's type system enforces it for us

Equality of Natural Numbers as an Inductive Definition

$$\frac{}{\texttt{eq(0,0)}}\ \text{equalzero} \qquad \frac{\texttt{eq(}a\texttt{,}b\texttt{)}}{\texttt{eq(succ(}a\texttt{),succ(}b\texttt{))}}\ \text{equal}$$

Judgements: $\texttt{eq(}n_1\texttt{,}n_2\texttt{)}$ "n_1 and n_2 are equal" ← a relation/a set of pairs

Variables: a b

Symbols: `0` `succ( )`

Is 3 = 3? A reverse derivation

`eq(succ(succ(succ(0))),succ(succ(succ(0))))`

Equality of Natural Numbers as an Inductive Definition

$$\frac{}{\texttt{eq(0,0)}}\ \text{equalzero} \qquad \frac{\texttt{eq(}a\texttt{,}\ b\texttt{)}}{\texttt{eq(succ(}a\texttt{),succ(}b\texttt{))}}\ \text{equal}$$

Judgements: $\texttt{eq(}n_1\texttt{,}\ n_2\texttt{)}$ "n_1 and n_2 are equal"

Variables: a b

Symbols: `0` `succ(` `)`

Is 3 = 3? A reverse derivation

$$\frac{\texttt{eq(}\quad\texttt{succ(succ(0))}\ \texttt{,}\quad\texttt{succ(succ(0))}\ \texttt{)}}{\texttt{eq(succ(succ(succ(0))),succ(succ(succ(0))))}}\ \text{equal}$$

Equality of Natural Numbers as an Inductive Definition

$$\frac{}{\texttt{eq(0,0)}}\ \text{equalzero} \qquad \frac{\texttt{eq(}a\texttt{,}b\texttt{)}}{\texttt{eq(succ(}a\texttt{),succ(}b\texttt{))}}\ \text{equal}$$

Judgements: $\texttt{eq(}n_1\texttt{,}n_2\texttt{)}$ "n_1 and n_2 are equal"

Variables: a b

Symbols: `0` `succ(  )`

Is 3 = 3? A reverse derivation

$$\cfrac{\cfrac{\texttt{eq(succ(0), succ(0))}}{\texttt{eq(succ(succ(0)), succ(succ(0)))}}\ \text{equal}}{\texttt{eq(succ(succ(succ(0))),succ(succ(succ(0))))}}\ \text{equal}$$

Equality of Natural Numbers as an Inductive Definition

$$\frac{}{\texttt{eq(0,0)}}\ \text{equalzero} \qquad \frac{\texttt{eq}(a,b)}{\texttt{eq(succ(}a\texttt{),succ(}b\texttt{))}}\ \text{equal}$$

Judgements: $\texttt{eq}(n_1, n_2)$ "n_1 and n_2 are equal"

Variables: a b

Symbols: `0` `succ( )`

Is 3 = 3? A reverse derivation

$$\cfrac{\cfrac{\cfrac{\texttt{eq(0,0)}}{\texttt{eq(succ(0),succ(0))}}\ \text{equal}}{\texttt{eq(succ(succ(0)),succ(succ(0)))}}\ \text{equal}}{\texttt{eq(succ(succ(succ(0))),succ(succ(succ(0))))}}\ \text{equal}$$

Equality of Natural Numbers as an Inductive Definition

$$\frac{}{\mathtt{eq(0,0)}}\ \text{equalzero} \qquad\qquad \frac{\mathtt{eq}(a,b)}{\mathtt{eq(succ(}a\mathtt{),succ(}b\mathtt{))}}\ \text{equal}$$

Judgements: $\mathtt{eq}(n_1, n_2)$ "n_1 and n_2 are equal"

Variables: a b

Symbols: $\mathtt{0}$ $\mathtt{succ(\ \ )}$

Is 3 = 3? A reverse derivation

$$\cfrac{\cfrac{\cfrac{\cfrac{}{\mathtt{eq(0,0)}}\ \text{equalzero}}{\mathtt{eq(succ(0),succ(0))}}\ \text{equal}}{\mathtt{eq(succ(succ(0)),succ(succ(0)))}}\ \text{equal}}{\mathtt{eq(succ(succ(succ(0))),succ(succ(succ(0))))}}\ \text{equal}$$

Equality of Natural Numbers as an Inductive Definition

$$\frac{}{\texttt{eq(0,0)}}\ \text{equalzero} \qquad \frac{\texttt{eq}(a,b)}{\texttt{eq(succ(}a\texttt{),succ(}b\texttt{))}}\ \text{equal}$$

Judgements: $\texttt{eq}(n_1, n_2)$ "n_1 and n_2 are equal"

Variables: a b

Symbols: `0` `succ( )`

Is 3 = 3? A reverse derivation

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\texttt{eq(0,0)}}\ \text{equalzero}}{\texttt{eq(succ(0),succ(0))}}\ \text{equal}}{\texttt{eq(succ(succ(0)),succ(succ(0)))}}\ \text{equal}}{\texttt{eq(succ(succ(succ(0))),succ(succ(succ(0))))}}\ \text{equal}$$

Is 1 = 2?

`eq(succ(0),succ(succ(0)))`

Equality of Natural Numbers as an Inductive Definition

$$\frac{}{\texttt{eq(0,0)}}\ \text{equalzero} \qquad \frac{\texttt{eq}(a,b)}{\texttt{eq(succ(}a\texttt{),succ(}b\texttt{))}}\ \text{equal}$$

Judgements: $\texttt{eq}(n_1, n_2)$ "n_1 and n_2 are equal"

Variables: a b

Symbols: `0` `succ( )`

Is 3 = 3? A reverse derivation

$$\cfrac{\cfrac{\cfrac{\cfrac{}{\texttt{eq(0,0)}}\ \text{equalzero}}{\texttt{eq(succ(0),succ(0))}}\ \text{equal}}{\texttt{eq(succ(succ(0)),succ(succ(0)))}}\ \text{equal}}{\texttt{eq(succ(succ(succ(0))),succ(succ(succ(0))))}}\ \text{equal}$$

Is 1 = 2?

$$\frac{\texttt{eq(0,succ(0))}}{\texttt{eq(succ(0),succ(succ(0)))}}\ \text{equal}$$

Equality of Natural Numbers as an Inductive Definition

$$\frac{}{\texttt{eq(0,0)}}\ \text{equalzero} \qquad \frac{\texttt{eq}(a,b)}{\texttt{eq(succ(}a\texttt{),succ(}b\texttt{))}}\ \text{equal}$$

Judgements: $\texttt{eq}(n_1, n_2)$ "n_1 and n_2 are equal"

Variables: a b

Symbols: `0` `succ( )`

Is 3 = 3? A reverse derivation

$$\cfrac{\cfrac{\cfrac{\cfrac{}{\texttt{eq(0,0)}}\ \text{equalzero}}{\texttt{eq(succ(0),succ(0))}}\ \text{equal}}{\texttt{eq(succ(succ(0)),succ(succ(0)))}}\ \text{equal}}{\texttt{eq(succ(succ(succ(0))),succ(succ(succ(0))))}}\ \text{equal}$$

Is 1 = 2?

$$\frac{\texttt{eq(0,succ(0))}}{\texttt{eq(succ(0),succ(succ(0)))}}\ \text{equal}$$

We are stuck: neither rule applies, so $1 \neq 2$

Equality of Natural Numbers as an Inductive Definition

$$\frac{}{\mathtt{0} = \mathtt{0}}\ \text{equalzero} \qquad \frac{a = b}{\mathtt{succ}(a) = \mathtt{succ}(b)}\ \text{equal}$$

Judgements: $n_1 = n_2$ "n_1 and n_2 are equal" ← a relation/a set of pairs

Variables: a b

Symbols: `0` `succ( )`

Is 3 = 3?

$$\cfrac{\cfrac{\cfrac{\cfrac{}{\mathtt{0} = \mathtt{0}}\ \text{equalzero}}{\mathtt{succ(0)} = \mathtt{succ(0)}}\ \text{equal}}{\mathtt{succ(succ(0))} = \mathtt{succ(succ(0))}}\ \text{equal}}{\mathtt{succ(succ(succ(0)))} = \mathtt{succ(succ(succ(0)))}}\ \text{equal}$$

Is 1 = 2?

$$\frac{\mathtt{0} = \mathtt{succ(0)}}{\mathtt{succ(0)} = \mathtt{succ(succ(0))}}\ \text{equal}$$

We are stuck: neither rule applies, so $1 \neq 2$

Equality of Natural Numbers as an Inductive Definition

$$\frac{}{\mathtt{0} = \mathtt{0}}\ \text{equalzero} \qquad \frac{a = b}{\mathtt{succ}(a) = \mathtt{succ}(b)}\ \text{equal}$$

```
data Nat = Zero | Succ Nat

natEqual :: Nat -> Nat      -> Bool
natEqual Zero     Zero      = True          -- equalzero rule
natEqual (Succ a) (Succ b) = natEqual a b  -- equal rule
natEqual _        _         = False
```

Again: single function because only one rule may ever match

Equality of Natural Numbers as an Inductive Definition

$$\frac{}{\mathtt{0} = \mathtt{0}}\ \text{equalzero} \qquad \frac{a = b}{\mathtt{succ}(a) = \mathtt{succ}(b)}\ \text{equal}$$

```haskell
data Nat = Zero | Succ Nat
  deriving Eq
```

This Haskell's default implementation of == for algebraic data types

Addition as an Inductive Definition

$$\frac{b \in \mathbb{N}}{\mathtt{sum}(\mathtt{0},\ b,\ b)}\ \text{addzero} \qquad \frac{\mathtt{sum}(a,\ b,\ c)}{\mathtt{sum}(\mathtt{succ}(a),\ b,\ \mathtt{succ}(c))}\ \text{add}$$

Judgments: $n \in \mathbb{N}$ $\quad \mathtt{sum}(n_1,\ n_2,\ n_3)$ $\longleftarrow$ a relation/a set of triples

Variables: a $\quad b$ $\quad c$

Symbols: $\mathtt{0}$ $\quad \mathtt{succ(\quad)}$

Addition as an Inductive Definition

$$\frac{b \in \mathbb{N}}{\mathtt{sum}(\mathtt{0},\, b,\, b)}\,\text{addzero} \qquad \frac{\mathtt{sum}(a,\, b,\, c)}{\mathtt{sum}(\mathtt{succ}(a),\, b,\, \mathtt{succ}(c))}\,\text{add}$$

Judgments: $n \in \mathbb{N}$ $\quad \mathtt{sum}(n_1,\, n_2,\, n_3)$

Variables: a $\quad b$ $\quad c$

Symbols: `0` `succ(` `)`

Is 2 + 1 = 3?

$\mathtt{sum}(\mathtt{succ}(\mathtt{succ}(\mathtt{0})),\, \mathtt{succ}(\mathtt{0}),\, \mathtt{succ}(\mathtt{succ}(\mathtt{succ}(\mathtt{0}))))$

Addition as an Inductive Definition

$$\frac{b \in \mathbb{N}}{\texttt{sum}(\texttt{0},\, b,\, b)}\,\text{addzero} \qquad \frac{\texttt{sum}(a,\, b,\, c)}{\texttt{sum}(\texttt{succ(}a\texttt{)},\, b,\, \texttt{succ(}c\texttt{)})}\,\text{add}$$

Judgments: $n \in \mathbb{N}$ $\quad$ $\texttt{sum}(n_1,\, n_2,\, n_3)$

Variables: a $\quad b$ $\quad c$

Symbols: `0` `succ(` `)`

Is 2 + 1 = 3?

$$\frac{\texttt{sum(}\quad \texttt{succ(0)}\;\texttt{, succ(0),}\quad \texttt{succ(succ(0))}\;\texttt{)}}{\texttt{sum(succ(succ(0)), succ(0), succ(succ(succ(0))))}}\,\text{add}$$

Addition as an Inductive Definition

$$\frac{b \in \mathbb{N}}{\mathtt{sum}(\mathtt{0},\, b,\, b)}\,\text{addzero} \qquad \frac{\mathtt{sum}(a,\, b,\, c)}{\mathtt{sum}(\mathtt{succ}(a),\, b,\, \mathtt{succ}(c))}\,\text{add}$$

Judgments: $n \in \mathbb{N}$ $\quad \mathtt{sum}(n_1,\, n_2,\, n_3)$

Variables: $a \quad b \quad c$

Symbols: `0` `succ(` `)`

Is 2 + 1 = 3?

$$\cfrac{\cfrac{\mathtt{sum}(\mathtt{0},\, \mathtt{succ(0)},\, \mathtt{succ(0)})}{\mathtt{sum}(\mathtt{succ(0)},\, \mathtt{succ(0)},\, \mathtt{succ(succ(0))})}\,\text{add}}{\mathtt{sum}(\mathtt{succ(succ(0))},\, \mathtt{succ(0)},\, \mathtt{succ(succ(succ(0)))})}\,\text{add}$$

Addition as an Inductive Definition

$$\frac{b \in \mathbb{N}}{\mathtt{sum}(\mathtt{0},\, b,\, b)}\ \text{addzero} \qquad \frac{\mathtt{sum}(a,\, b,\, c)}{\mathtt{sum}(\mathtt{succ}(a),\, b,\, \mathtt{succ}(c))}\ \text{add}$$

Judgments: $n \in \mathbb{N}$ $\quad \mathtt{sum}(n_1,\, n_2,\, n_3)$

Variables: a $\quad b$ $\quad c$

Symbols: $\mathtt{0}$ $\quad \mathtt{succ(}\quad\mathtt{)}$

Is 2 + 1 = 3?

$$\dfrac{\dfrac{\dfrac{\mathtt{succ}(\mathtt{0}) \in \mathbb{N}}{\mathtt{sum}(\mathtt{0},\, \mathtt{succ}(\mathtt{0}),\, \mathtt{succ}(\mathtt{0}))}\ \text{addzero}}{\mathtt{sum}(\mathtt{succ}(\mathtt{0}),\, \mathtt{succ}(\mathtt{0}),\, \mathtt{succ}(\mathtt{succ}(\mathtt{0})))}\ \text{add}}{\mathtt{sum}(\mathtt{succ}(\mathtt{succ}(\mathtt{0})),\, \mathtt{succ}(\mathtt{0}),\, \mathtt{succ}(\mathtt{succ}(\mathtt{succ}(\mathtt{0}))))}\ \text{add}$$

Addition as an Inductive Definition

$$\frac{b \in \mathbb{N}}{\mathtt{sum}(\mathtt{0},\, b,\, b)}\,\text{addzero} \qquad \frac{\mathtt{sum}(a,\, b,\, c)}{\mathtt{sum}(\mathtt{succ}(a),\, b,\, \mathtt{succ}(c))}\,\text{add}$$

Judgments: $n \in \mathbb{N}$ $\quad \mathtt{sum}(n_1,\, n_2,\, n_3)$

Variables: $a \quad b \quad c$

Symbols: $\mathtt{0} \quad \mathtt{succ(}\quad\mathtt{)}$

Is 2 + 1 = 3?

$$\dfrac{\dfrac{\dfrac{\dfrac{\mathtt{0} \in \mathbb{N}}{\mathtt{succ(0)} \in \mathbb{N}}\,\text{successor}}{\mathtt{sum(0,\, succ(0),\, succ(0))}}\,\text{addzero}}{\mathtt{sum(succ(0),\, succ(0),\, succ(succ(0)))}}\,\text{add}}{\mathtt{sum(succ(succ(0)),\, succ(0),\, succ(succ(succ(0))))}}\,\text{add}$$

Addition as an Inductive Definition

$$\frac{b \in \mathbb{N}}{\mathtt{sum}(\mathtt{0},\, b,\, b)}\ \text{addzero} \qquad \frac{\mathtt{sum}(a,\, b,\, c)}{\mathtt{sum}(\mathtt{succ}(a),\, b,\, \mathtt{succ}(c))}\ \text{add}$$

Judgments: $n \in \mathbb{N}$ $\quad$ $\mathtt{sum}(n_1,\, n_2,\, n_3)$

Variables: a $\quad b$ $\quad c$

Symbols: `0` $\quad$ `succ(` $\quad$ `)`

Is 2 + 1 = 3?

$$\dfrac{\dfrac{\dfrac{}{\mathtt{0} \in \mathbb{N}}\ \text{zero}}{\mathtt{succ}(\mathtt{0}) \in \mathbb{N}}\ \text{successor}}{\dfrac{\mathtt{sum}(\quad \mathtt{0} \quad, \mathtt{succ}(\mathtt{0}), \quad \mathtt{succ}(\mathtt{0}) \quad)}{\dfrac{\mathtt{sum}(\quad \mathtt{succ}(\mathtt{0}) \quad, \mathtt{succ}(\mathtt{0}), \quad \mathtt{succ}(\mathtt{succ}(\mathtt{0})) \quad)}{\mathtt{sum}(\mathtt{succ}(\mathtt{succ}(\mathtt{0})), \mathtt{succ}(\mathtt{0}), \mathtt{succ}(\mathtt{succ}(\mathtt{succ}(\mathtt{0}))))}\ \text{add}}\ \text{add}}\ \text{addzero}$$

Addition as an Inductive Definition

$$\frac{b \in \mathbb{N}}{\mathtt{0} + b = b}\,\text{addzero} \qquad \frac{a + b = c}{\mathtt{succ}(a) + b = \mathtt{succ}(c)}\,\text{add}$$

Judgments: $n \in \mathbb{N}$ $\quad n_1 + n_2 = n_3$ $\longleftarrow$ a relation/a set of triples

Variables: $a \quad b \quad c$

Symbols: $\mathtt{0} \quad \mathtt{succ(\ \ )}$

Is 2 + 1 = 3?

$$\cfrac{\cfrac{\cfrac{}{\mathtt{0} \in \mathbb{N}}\,\text{zero}}{\mathtt{succ(0)} \in \mathbb{N}}\,\text{successor}}{\cfrac{\mathtt{0} + \mathtt{succ(0)} = \mathtt{succ(0)}}{\cfrac{\mathtt{succ(0)} + \mathtt{succ(0)} = \mathtt{succ(succ(0))}}{\mathtt{succ(succ(0))} + \mathtt{succ(0)} = \mathtt{succ(succ(succ(0)))}}\,\text{add}}\,\text{add}}\,\text{addzero}$$

Addition as an Inductive Definition

$$\frac{b \in \mathbb{N}}{\mathtt{0} + b = b}\ \text{addzero} \qquad \frac{a + b = c}{\mathtt{succ}(a) + b = \mathtt{succ}(c)}\ \text{add}$$

```
data Nat = Zero | Succ Nat
  deriving Eq

sumsTo :: Nat -> Nat -> Nat     -> Bool         -- Is a + b = c?
sumsTo Zero     b b' | b == b' = True         -- addzero rule
sumsTo (Succ a) b (Succ c)     = sumsTo a b c -- add rule
sumsTo _        _ _            = False        -- E.g., (Succ a) _ Zero
```

No need to check whether $b \in \mathbb{N}$: the types enforce this

Haskell patterns can't check for equality like `sumsTo Zero b b`, so I added guard `b == 'b`

Rather awkward to ask "is this it?"

Addition as an Inductive Definition

$$\frac{b \in \mathbb{N}}{0 + b = b}\,\text{addzero} \qquad \frac{a + b = c}{\mathtt{succ}(a) + b = \mathtt{succ}(c)}\,\text{add}$$

```
data Nat = Zero | Succ Nat
  deriving (Eq, Show)

addNat :: Nat -> Nat -> Nat               -- Given a and b, what c satisfies a+b = c?
addNat Zero     b = b                     -- addzero rule
addNat (Succ a) b = Succ (addNat a b)     -- add rule
```

The dataflow makes this easy and it's obviously a total function

Addition as an Inductive Definition

$$\frac{b \in \mathbb{N}}{\mathtt{0} + b = b}\;\text{addzero} \qquad \frac{a + b = c}{\mathtt{succ}(a) + b = \mathtt{succ}(c)}\;\text{add}$$

```haskell
data Nat = Zero | Succ Nat
  deriving (Eq, Show)

subNat :: Nat -> Nat      -> Maybe Nat  -- Given c and a, what b satisfies a+b = c?
subNat c        Zero     = Just c       -- addzero rule
subNat (Succ c) (Succ a) = subNat c a   -- add rule
subNat Zero     (Succ _) = Nothing      -- failure
```

Still straightforward dataflow, but the function is no longer total

A Definition of Binary Trees

$$\frac{}{\texttt{leaf}\ \mathsf{tree}}\ \text{leaf} \qquad \frac{t_1\ \mathsf{tree} \quad t_2\ \mathsf{tree}}{\texttt{branch}(t_1\texttt{,}\,t_2\texttt{)}\ \mathsf{tree}}\ \text{branch}$$

Judgments: $t\ \mathsf{tree}$ "t is a tree"

Variables: t_1 t_2 t_1 and t_2 may be equal

Symbols: `leaf` `branch(` `,` `)`

A Definition of Binary Trees

$$\frac{}{\texttt{leaf}\ \mathsf{tree}}\ \text{leaf} \qquad \frac{t_1\ \mathsf{tree} \quad t_2\ \mathsf{tree}}{\texttt{branch(}t_1\texttt{,}t_2\texttt{)}\ \mathsf{tree}}\ \text{branch}$$

Judgments: t tree "t is a tree"

Variables: t_1 t_2 t_1 and t_2 may be equal

Symbols: `leaf` `branch( , )`

Derivations are generally tree-structured

`branch(branch(leaf,leaf),leaf)` tree

A Definition of Binary Trees

$$\frac{}{\texttt{leaf}\ \mathsf{tree}}\ \text{leaf} \qquad \frac{t_1\ \mathsf{tree} \quad t_2\ \mathsf{tree}}{\texttt{branch}(t_1\texttt{,}t_2)\ \mathsf{tree}}\ \text{branch}$$

Judgments: t tree "t is a tree"

Variables: t_1 t_2 t_1 and t_2 may be equal

Symbols: `leaf` `branch(` `,` `)`

Derivations are generally tree-structured

$$\frac{\texttt{branch(leaf,leaf)}\ \mathsf{tree} \qquad \texttt{leaf}\ \mathsf{tree}}{\texttt{branch(branch(leaf,leaf),leaf)}\ \mathsf{tree}}\ \text{branch}$$

A Definition of Binary Trees

$$\frac{}{\texttt{leaf}\ \mathsf{tree}}\ \text{leaf} \qquad\qquad \frac{t_1\ \mathsf{tree} \quad t_2\ \mathsf{tree}}{\texttt{branch(}t_1\texttt{,}t_2\texttt{)}\ \mathsf{tree}}\ \text{branch}$$

Judgments: t tree "t is a tree"

Variables: t_1 t_2 t_1 and t_2 may be equal

Symbols: `leaf` `branch(` `,` `)`

Derivations are generally tree-structured

$$\dfrac{\dfrac{\texttt{leaf}\ \mathsf{tree} \qquad \texttt{leaf}\ \mathsf{tree}}{\texttt{branch(leaf,leaf)}\ \mathsf{tree}}\ \text{branch} \qquad \dfrac{}{\texttt{leaf}\ \mathsf{tree}}\ \text{leaf}}{\texttt{branch(branch(leaf,leaf),leaf)}\ \mathsf{tree}}\ \text{branch}$$

A Definition of Binary Trees

$$\frac{}{\texttt{leaf}\ \mathsf{tree}}\ \text{leaf} \qquad\qquad \frac{t_1\ \mathsf{tree} \quad t_2\ \mathsf{tree}}{\texttt{branch(}t_1\texttt{,}t_2\texttt{)}\ \mathsf{tree}}\ \text{branch}$$

Judgments: t tree "t is a tree"

Variables: t_1 t_2 t_1 and t_2 may be equal

Symbols: `leaf` `branch(` `,` `)`

Derivations are generally tree-structured

$$\dfrac{\dfrac{\dfrac{}{\texttt{leaf}\ \mathsf{tree}}\ \text{leaf} \qquad \dfrac{}{\texttt{leaf}\ \mathsf{tree}}\ \text{leaf}}{\texttt{branch(leaf,leaf)}\ \mathsf{tree}}\ \text{branch} \qquad \dfrac{}{\texttt{leaf}\ \mathsf{tree}}\ \text{leaf}}{\texttt{branch(branch(leaf,leaf),leaf)}\ \mathsf{tree}}\ \text{branch}$$

A Definition of Binary Trees

$$\frac{}{\texttt{leaf tree}}\ \text{leaf} \qquad\qquad \frac{t_1\ \texttt{tree} \quad t_2\ \texttt{tree}}{\texttt{branch(}t_1\texttt{,}t_2\texttt{) tree}}\ \text{branch}$$

Judgments: t tree "t is a tree"

Variables: t_1 t_2 t_1 and t_2 may be equal

Symbols: `leaf` `branch( , )`

```
data Tree = Leaf | Branch Tree Tree

isTree :: Tree -> Bool
isTree Leaf          = True
isTree (Branch l r) = isTree l && isTree r   -- Must test both branches
```

Trivially true because of Haskell's types, but note two-way recursion