Recursive Data Definition

example: *nat*

Recursive Data Definition

example: *nat*

can be constructed by starting with 0 and repeatedly adding 1

Recursive Data Definition

example: *nat*

can be constructed by starting with 0 and repeatedly adding 1

construction axiom 0: *nat*

Recursive Data Definition

example: *nat*

can be constructed by starting with 0 and repeatedly adding 1

construction axiom 0: *nat*

construction axiom *nat*+1: *nat*

Recursive Data Definition

example: *nat*

can be constructed by starting with 0 and repeatedly adding 1

construction axiom 0: *nat*

construction axiom *nat*$+1$: *nat*

⊤

Recursive Data Definition

example: *nat*

can be constructed by starting with 0 and repeatedly adding 1

construction axiom $\quad 0: nat$

construction axiom $\quad nat+1: nat$

$\top$ $\quad$ by the axiom, $0: nat$

$\Rightarrow \quad 0: nat$

Recursive Data Definition

example: *nat*

can be constructed by starting with 0 and repeatedly adding 1

construction axiom	$0: nat$
construction axiom	$nat+1: nat$

	$\top$	by the axiom, $0: nat$
$\Rightarrow$	$0: nat$	add 1 to each side
$\Rightarrow$	$0+1: nat+1$	

Recursive Data Definition

example: *nat*

can be constructed by starting with 0 and repeatedly adding 1

construction axiom $0: nat$

construction axiom $nat+1: nat$

	$\top$	by the axiom, $0: nat$
$\Rightarrow$	$0: nat$	add 1 to each side
$\Rightarrow$	$0+1: nat+1$	by arithmetic, $0+1 = 1$; by the axiom, $nat+1: nat$
$\Rightarrow$	$1: nat$	

Recursive Data Definition

example: *nat*

can be constructed by starting with 0 and repeatedly adding 1

construction axiom $0: nat$

construction axiom $nat+1: nat$

	$\top$	by the axiom, $0: nat$
$\Rightarrow$	$0: nat$	add 1 to each side
$\Rightarrow$	$0+1: nat+1$	by arithmetic, $0+1 = 1$; by the axiom, $nat+1: nat$
$\Rightarrow$	$1: nat$	add 1 to each side
$\Rightarrow$	$1+1: nat+1$	

Recursive Data Definition

example: *nat*

can be constructed by starting with 0 and repeatedly adding 1

construction axiom $0: nat$

construction axiom $nat+1: nat$

	$\top$	by the axiom, $0: nat$
$\Rightarrow$	$0: nat$	add 1 to each side
$\Rightarrow$	$0+1: nat+1$	by arithmetic, $0+1 = 1$; by the axiom, $nat+1: nat$
$\Rightarrow$	$1: nat$	add 1 to each side
$\Rightarrow$	$1+1: nat+1$	by arithmetic, $1+1 = 2$; by the axiom, $nat+1: nat$
$\Rightarrow$	$2: nat$	

Recursive Data Definition

example: *nat*

can be constructed by starting with 0 and repeatedly adding 1

construction axiom	$0: nat$
construction axiom	$nat+1: nat$

	$\top$	by the axiom, $0: nat$
$\Rightarrow$	$0: nat$	add 1 to each side
$\Rightarrow$	$0+1: nat+1$	by arithmetic, $0+1 = 1$; by the axiom, $nat+1: nat$
$\Rightarrow$	$1: nat$	add 1 to each side
$\Rightarrow$	$1+1: nat+1$	by arithmetic, $1+1 = 2$; by the axiom, $nat+1: nat$
$\Rightarrow$	$2: nat$	and so on

Recursive Data Definition

example: *nat*

can be constructed by starting with 0 and repeatedly adding 1

construction axiom 0: *nat*

construction axiom *nat*+1: *nat*

Recursive Data Definition

example: *nat*

can be constructed by starting with 0 and repeatedly adding 1

construction axiom $\quad 0: nat$

construction axiom $\quad nat+1: nat$

$nat \;=\; 0, 1, 2, 3, 4, 5, \ldots$ **?**

Recursive Data Definition

example: *nat*

can be constructed by starting with 0 and repeatedly adding 1

construction axiom $0: nat$

construction axiom $nat+1: nat$

$nat = 0, 1, 2, 3, 4, 5, ...$ **?**

$nat = ..., -3, -2, -1, 0, 1, 2, 3, ...$ **?**

Recursive Data Definition

example: *nat*

can be constructed by starting with 0 and repeatedly adding 1

construction axiom   0: *nat*

construction axiom   *nat*+1: *nat*

nat = 0, 1, 2, 3, 4, 5, ... ?

nat = ..., -3, -2, -1, 0, 1, 2, 3, ... ?

nat = the rationals ?

Recursive Data Definition

example: *nat*

can be constructed by starting with 0 and repeatedly adding 1

construction axiom 0: *nat*

construction axiom *nat*+1: *nat*

nat = 0, 1, 2, 3, 4, 5, ... ?

nat = ..., -3, -2, -1, 0, 1, 2, 3, ... ?

nat = the rationals ?

nat = the reals ?

Recursive Data Definition

example: *nat*

can be constructed by starting with 0 and repeatedly adding 1

construction axiom   0: *nat*

construction axiom   *nat*+1: *nat*

nat = 0, 1, 2, 3, 4, 5, ... **?**

nat = ..., -3, -2, -1, 0, 1, 2, 3, ... **?**

nat = the rationals **?**

nat = the reals **?**

nat = 0, 0.5, 1, 1.5, 2, 2.5, 3, 3.5, ... **?**

Recursive Data Definition

example: *nat*

can be constructed by starting with 0 and repeatedly adding 1

construction axiom	$0: nat$
construction axiom	$nat+1: nat$
induction axiom	$0: B \ \land\ B+1: B \ \Rightarrow\ nat: B$

Recursive Data Definition

example: *nat*

can be constructed by starting with 0 and repeatedly adding 1

construction axiom	$0{:}\ nat$
construction axiom	$nat{+}1{:}\ nat$
induction axiom	$0{:}\ B\ \land\ B{+}1{:}\ B\ \Rightarrow\ nat{:}\ B$

construction axiom	$0,\ nat{+}1{:}\ nat$
induction axiom	$0,\ B{+}1{:}\ B\ \Rightarrow\ nat{:}\ B$

Recursive Data Definition

example: *nat*

can be constructed by starting with 0 and repeatedly adding 1

construction axiom	$0\text{: } nat$
construction axiom	$nat\text{+}1\text{: } nat$
induction axiom	$0\text{: } B \ \land \ B\text{+}1\text{: } B \ \Rightarrow \ nat\text{: } B$

construction axiom	$0,\ nat\text{+}1\text{: } nat$
induction axiom	$0,\ B\text{+}1\text{: } B \ \Rightarrow \ nat\text{: } B$

construction axiom	$P\,0 \land \forall n\text{: } nat\cdot\ P\,n \Rightarrow P(n\text{+}1) \ \Leftarrow \ \forall n\text{: } nat\cdot\ P\,n$
induction axiom	$P\,0 \land \forall n\text{: } nat\cdot\ P\,n \Rightarrow P(n\text{+}1) \ \Rightarrow \ \forall n\text{: } nat\cdot\ P\,n$

Recursive Data Definition

nat induction

$P\,0 \land \forall n\colon nat\cdot P\,n \Rightarrow P(n+1) \quad\Longrightarrow\quad \forall n\colon nat\cdot P\,n$

$P\,0 \lor \exists n\colon nat\cdot \neg\, P\,n \land P(n+1) \quad\Longleftarrow\quad \exists n\colon nat\cdot P\,n$

$\forall n\colon nat\cdot P\,n \Rightarrow P(n+1) \quad\Longrightarrow\quad \forall n\colon nat\cdot (P\,0 \Rightarrow P\,n)$

$\exists n\colon nat\cdot \neg\, P\,n \land P(n+1) \quad\Longleftarrow\quad \exists n\colon nat\cdot (\neg\, P\,0 \land P\,n)$

$\forall n\colon nat\cdot (\forall m\colon nat\cdot m{<}n \Rightarrow P\,m) \Rightarrow P\,n \quad\Longrightarrow\quad \forall n\colon nat\cdot P\,n$

$\exists n\colon nat\cdot (\forall m\colon nat\cdot m{<}n \Rightarrow \neg\, P\,m) \land P\,n \quad\Longleftarrow\quad \exists n\colon nat\cdot P\,n$

Recursive Data Definition

nat induction

$P\,0 \land \forall n{:}\ nat\cdot\ P\,n \Rightarrow P(n+1) \implies \forall n{:}\ nat\cdot\ P\,n$ ⟵

$P\,0 \lor \exists n{:}\ nat\cdot\ \neg\, P\,n \land P(n+1) \impliedby \exists n{:}\ nat\cdot\ P\,n$

$\forall n{:}\ nat\cdot\ P\,n \Rightarrow P(n+1) \implies \forall n{:}\ nat\cdot\ (P\,0 \Rightarrow P\,n)$

$\exists n{:}\ nat\cdot\ \neg\, P\,n \land P(n+1) \impliedby \exists n{:}\ nat\cdot\ (\neg\, P\,0 \land P\,n)$

$\forall n{:}\ nat\cdot\ (\forall m{:}\ nat\cdot\ m<n \Rightarrow P\,m) \Rightarrow P\,n \implies \forall n{:}\ nat\cdot\ P\,n$

$\exists n{:}\ nat\cdot\ (\forall m{:}\ nat\cdot\ m<n \Rightarrow \neg\, P\,m) \land P\,n \impliedby \exists n{:}\ nat\cdot\ P\,n$

Recursive Data Definition

nat induction

$P\,0 \land \forall n{:}\ nat\cdot P\,n \Rightarrow P(n+1) \quad\Longrightarrow\quad \forall n{:}\ nat\cdot P\,n$

$P\,0 \lor \exists n{:}\ nat\cdot \neg P\,n \land P(n+1) \quad\Longleftarrow\quad \exists n{:}\ nat\cdot P\,n$

$\forall n{:}\ nat\cdot P\,n \Rightarrow P(n+1) \quad\Longrightarrow\quad \forall n{:}\ nat\cdot (P\,0 \Rightarrow P\,n)$ ⟵

$\exists n{:}\ nat\cdot \neg P\,n \land P(n+1) \quad\Longleftarrow\quad \exists n{:}\ nat\cdot (\neg P\,0 \land P\,n)$

$\forall n{:}\ nat\cdot (\forall m{:}\ nat\cdot m<n \Rightarrow P\,m) \Rightarrow P\,n \quad\Longrightarrow\quad \forall n{:}\ nat\cdot P\,n$

$\exists n{:}\ nat\cdot (\forall m{:}\ nat\cdot m<n \Rightarrow \neg P\,m) \land P\,n \quad\Longleftarrow\quad \exists n{:}\ nat\cdot P\,n$

Recursive Data Definition

nat induction

$P\,0 \wedge \forall n{:}\ nat\cdot\ P\,n \Rightarrow P(n+1) \quad \Longrightarrow \quad \forall n{:}\ nat\cdot\ P\,n$

$P\,0 \vee \exists n{:}\ nat\cdot\ \neg\ P\,n \wedge P(n+1) \quad \Longleftarrow \quad \exists n{:}\ nat\cdot\ P\,n$

$\forall n{:}\ nat\cdot\ P\,n \Rightarrow P(n+1) \quad \Longrightarrow \quad \forall n{:}\ nat\cdot\ (P\,0 \Rightarrow P\,n)$

$\exists n{:}\ nat\cdot\ \neg\ P\,n \wedge P(n+1) \quad \Longleftarrow \quad \exists n{:}\ nat\cdot\ (\neg\ P\,0 \wedge P\,n)$

$\forall n{:}\ nat\cdot\ (\forall m{:}\ nat\cdot\ m<n \Rightarrow P\,m) \Rightarrow P\,n \quad \Longrightarrow \quad \forall n{:}\ nat\cdot\ P\,n$ ⟵

$\exists n{:}\ nat\cdot\ (\forall m{:}\ nat\cdot\ m<n \Rightarrow \neg\ P\,m) \wedge P\,n \quad \Longleftarrow \quad \exists n{:}\ nat\cdot\ P\,n$

Recursive Data Definition

nat induction

$$P\,0 \land \forall n{:}\ nat\cdot\ P\,n \Rightarrow P(n+1) \quad \Longrightarrow \quad \forall n{:}\ nat\cdot\ P\,n$$

$$P\,0 \lor \exists n{:}\ nat\cdot\ \neg\ P\,n \land P(n+1) \quad \Longleftarrow \quad \exists n{:}\ nat\cdot\ P\,n$$

$$\forall n{:}\ nat\cdot\ P\,n \Rightarrow P(n+1) \quad \Longrightarrow \quad \forall n{:}\ nat\cdot\ (P\,0 \Rightarrow P\,n)$$

$$\exists n{:}\ nat\cdot\ \neg\ P\,n \land P(n+1) \quad \Longleftarrow \quad \exists n{:}\ nat\cdot\ (\neg\ P\,0 \land P\,n)$$

$$\forall n{:}\ nat\cdot\ (\forall m{:}\ nat\cdot\ m<n \Rightarrow P\,m) \Rightarrow P\,n \quad \Longrightarrow \quad \forall n{:}\ nat\cdot\ P\,n$$

$$\exists n{:}\ nat\cdot\ (\forall m{:}\ nat\cdot\ m<n \Rightarrow \neg\ P\,m) \land P\,n \quad \Longleftarrow \quad \exists n{:}\ nat\cdot\ P\,n \qquad \longleftarrow$$

philosophical induction: guessing the general case from special cases

(an important skill in mathematics)

philosophical induction: guessing the general case from special cases

(an important skill in mathematics)

philosophical deduction: proving, using the rules of logic

philosophical induction: guessing the general case from special cases

(an important skill in mathematics)

philosophical deduction: proving, using the rules of logic

mathematical induction: an axiom (sometimes presented as a proof rule)

(mathematical induction is part of philosophical deduction)

philosophical induction: guessing the general case from special cases

(an important skill in mathematics)

philosophical deduction: proving, using the rules of logic

mathematical induction: an axiom (sometimes presented as a proof rule)

(mathematical induction is part of philosophical deduction)

engineering induction:

If it works for $n = 1, 2,$ and 3 then that's good enough for me.

philosophical induction: guessing the general case from special cases

(an important skill in mathematics)

philosophical deduction: proving, using the rules of logic

mathematical induction: an axiom (sometimes presented as a proof rule)

(mathematical induction is part of philosophical deduction)

engineering induction:

If it works for $n = 1, 2,$ and 3 then that's good enough for me.

military induction:

philosophical induction: guessing the general case from special cases

(an important skill in mathematics)

philosophical deduction: proving, using the rules of logic

mathematical induction: an axiom (sometimes presented as a proof rule)

(mathematical induction is part of philosophical deduction)

engineering induction:

If it works for $n = 1, 2,$ and 3 then that's good enough for me.

military induction:

tax deduction:

Recursive Data Definition

example: *int*

Recursive Data Definition

example: int

Define $\quad int \;=\; nat, -nat$

Recursive Data Definition

example: *int*

Define $int \;=\; nat, -nat$

or $0, int{+}1, int{-}1{:}\ int$

$0, B{+}1, B{-}1{:}\ B \;\Rightarrow\; int{:}\ B$

Recursive Data Definition

example: *int*

Define $int \;=\; nat, -nat$

or $0, int{+}1, int{-}1: int$

$0, B{+}1, B{-}1: B \;\Rightarrow\; int: B$

or $P\,0 \land (\forall i: int\cdot P\,i \Rightarrow P(i{+}1)) \land (\forall i: int\cdot P\,i \Rightarrow P(i{-}1)) \;=\; \forall i: int\cdot P\,i$

Recursive Data Definition

example: *pow*

Recursive Data Definition

example: *pow*

Define $pow = 2^{nat}$

Recursive Data Definition

example: *pow*

Define $pow = 2^{nat}$

or $pow = \S p: nat \cdot \exists m: nat \cdot p = 2^m$

Recursive Data Definition

example: *pow*

Define $pow = 2^{nat}$

or $pow = \S p: nat \cdot \exists m: nat \cdot p = 2^m$

or $1, 2 \times pow: pow$

$1, 2 \times B: B \Rightarrow pow: B$

Recursive Data Definition

example: *pow*

Define $pow = 2^{nat}$

or $pow \;=\; \S p: nat \cdot \exists m: nat \cdot p = 2^m$

or $1, 2 \times pow: pow$

$1, 2 \times B: B \;\Rightarrow\; pow: B$

or $P\,1 \;\wedge\; \forall p: pow \cdot P\,p \Rightarrow P(2 \times p) \;=\; \forall p: pow \cdot P\,p$

Least Fixed-Points

nat construction: $0, nat{+}1 \colon nat$

nat induction: $0, B{+}1 \colon B \;\Rightarrow\; nat \colon B$

Least Fixed-Points

nat construction: $0, nat{+}1: nat$

nat induction: $0, B{+}1: B \ \Rightarrow\ nat: B$

nat fixed-point construction: $nat \ = \ 0, nat{+}1$

nat fixed-point induction: $B = 0, B{+}1 \ \Rightarrow\ nat: B$

Least Fixed-Points

nat construction:	$0, nat+1 \colon nat$ ⟵
nat induction:	$0, B+1 \colon B \;\Rightarrow\; nat \colon B$
nat fixed-point construction:	$nat \;=\; 0, nat+1$ ⟵
nat fixed-point induction:	$B = 0, B+1 \;\Rightarrow\; nat \colon B$

Least Fixed-Points

nat construction: $0, nat{+}1 \colon nat$

nat induction: $0, B{+}1 \colon B \;\Rightarrow\; nat \colon B$ ⟵

nat fixed-point construction: $nat \;=\; 0, nat{+}1$

nat fixed-point induction: $B = 0, B{+}1 \;\Rightarrow\; nat \colon B$ ⟵

Least Fixed-Points

nat construction: $0, nat{+}1 \colon nat$

nat induction: $0, B{+}1 \colon B \;\Rightarrow\; nat \colon B$

nat fixed-point construction: $nat \;=\; 0, nat{+}1$

nat fixed-point induction: $B = 0, B{+}1 \;\Rightarrow\; nat \colon B$

x is a fixed-point of f

Least Fixed-Points

nat construction:	$0, nat{+}1 \colon nat$
nat induction:	$0, B{+}1 \colon B \;\Rightarrow\; nat \colon B$
nat fixed-point construction:	$nat \;=\; 0, nat{+}1$
nat fixed-point induction:	$B = 0, B{+}1 \;\Rightarrow\; nat \colon B$
x is a fixed-point of f	$x \;=\; f\,x$

Least Fixed-Points

nat construction: $0, nat{+}1 : nat$

nat induction: $0, B{+}1 : B \;\Rightarrow\; nat : B$

nat fixed-point construction: $nat \;=\; 0, nat{+}1$ ←

nat fixed-point induction: $B = 0, B{+}1 \;\Rightarrow\; nat : B$

x is a fixed-point of f $x \;=\; f\,x$ ←

Least Fixed-Points

nat construction:	$0, nat{+}1 : nat$
nat induction:	$0, B{+}1 : B \;\Rightarrow\; nat : B$
nat fixed-point construction:	$nat \;=\; 0, nat{+}1$
nat fixed-point induction:	$B = 0, B{+}1 \;\Rightarrow\; nat : B$ ⟵
x is a fixed-point of f	$x \;=\; f\,x$

Least Fixed-Points

nat construction: $0, nat{+}1 : nat$

nat induction: $0, B{+}1 : B \;\Rightarrow\; nat : B$

nat fixed-point construction: $nat \;=\; 0, nat{+}1$

nat fixed-point induction: $B = 0, B{+}1 \;\Rightarrow\; nat : B$

x is a fixed-point of f $\quad x \;=\; f\,x$

grammar: $exp \;=\;$ “x”, exp; “+”; exp

Least Fixed-Points

nat construction:	$0, nat{+}1 : nat$
nat induction:	$0, B{+}1 : B \;\Rightarrow\; nat : B$
nat fixed-point construction:	$nat \;=\; 0, nat{+}1$
nat fixed-point induction:	$B = 0, B{+}1 \;\Rightarrow\; nat : B$
x is a fixed-point of f	$x \;=\; f\, x$
grammar:	$exp \;=\;$ “x”, exp; “+”; exp
	$B \;=\;$ “x”, B; “+”; $B \;\Rightarrow\; exp : B$

Recursive Data Construction

Recursive Data Construction

name = (expression involving *name*)

Recursive Data Construction

$name$ = (expression involving $name$)

0. Construct $name_0 = null$

$name_{n+1}$ = (expression involving $name_n$)

Recursive Data Construction

$name = (\text{expression involving } name)$

0. Construct $\quad name_0 = null$

   $name_{n+1} = (\text{expression involving } name_n)$

1. Guess $\quad name_n = (\text{expression involving } n \text{ but not } name)$

Recursive Data Construction

$name$ = (expression involving $name$)

0. Construct — $name_0 = null$

   $name_{n+1}$ = (expression involving $name_n$)

1. Guess — $name_n$ = (expression involving n but not $name$)

2. Substitute ∞ for n — $name_\infty$ = (expression involving neither n nor $name$)

Recursive Data Construction

$name = (\text{expression involving } name)$

0. Construct — $name_0 = null$

   $name_{n+1} = (\text{expression involving } name_n)$

1. Guess — $name_n = (\text{expression involving } n \text{ but not } name)$

2. Substitute ∞ for n — $name_\infty = (\text{expression involving neither } n \text{ nor } name)$

3. Test fixed-point — $name_\infty = (\text{expression involving } name_\infty)$

Recursive Data Construction

$name$ = (expression involving $name$)

0. Construct $name_0 = null$

   $name_{n+1}$ = (expression involving $name_n$)

1. Guess $name_n$ = (expression involving n but not $name$)

2. Substitute ∞ for n $name_\infty$ = (expression involving neither n nor $name$)

3. Test fixed-point $name_\infty$ = (expression involving $name_\infty$)

4. Test least fixed-point B = (expression involving B) $\Rightarrow$ $name_\infty$: B

Recursive Data Construction

example: pow

$$pow \;=\; 1, 2{\times}pow$$

Recursive Data Construction

example: *pow*

$$pow \;\; = \;\; 1, 2{\times}pow$$

0. Construct

Recursive Data Construction

example: *pow*

$$pow \;=\; 1, 2{\times}pow$$

0. Construct

$$pow_0 \;=\; null$$

Recursive Data Construction

example: *pow*

$$pow \;=\; 1, 2{\times}pow$$

0. Construct

$$pow_0 \;=\; null$$

$$pow_1 \;=\; 1, 2{\times}pow_0$$

Recursive Data Construction

example: *pow*

$$pow \;=\; 1, 2\times pow$$

0. Construct

$$pow_0 \;=\; null$$

$$pow_1 \;=\; 1, 2\times pow_0 \;=\; 1, 2\times null$$

Recursive Data Construction

example: *pow*

$$pow \;=\; 1, 2{\times}pow$$

0. Construct

$$pow_0 \;=\; null$$

$$pow_1 \;=\; 1, 2{\times}pow_0 \;=\; 1, 2{\times}null \;=\; 1, null$$

Recursive Data Construction

example: *pow*

$$pow \;=\; 1, 2{\times}pow$$

0. Construct

$$pow_0 \;=\; null$$

$$pow_1 \;=\; 1, 2{\times}pow_0 \;=\; 1, 2{\times}null \;=\; 1, null \;=\; 1$$

Recursive Data Construction

example: *pow*

$$pow \;=\; 1, 2{\times}pow$$

0. Construct

$$pow_0 \;=\; null$$

$$pow_1 \;=\; 1, 2{\times}pow_0 \;=\; 1, 2{\times}null \;=\; 1, null \;=\; 1$$

$$pow_2 \;=\; 1, 2{\times}pow_1$$

Recursive Data Construction

example: *pow*

$$pow \;=\; 1,2\times pow$$

0. Construct

$$pow_0 \;=\; null$$

$$pow_1 \;=\; 1,2\times pow_0 \;=\; 1,2\times null \;=\; 1,null \;=\; 1$$

$$pow_2 \;=\; 1,2\times pow_1 \;=\; 1,2\times 1$$

Recursive Data Construction

example: *pow*

$$pow \;=\; 1, 2{\times}pow$$

0. Construct

$$pow_0 \;=\; null$$

$$pow_1 \;=\; 1, 2{\times}pow_0 \;=\; 1, 2{\times}null \;=\; 1, null \;=\; 1$$

$$pow_2 \;=\; 1, 2{\times}pow_1 \;=\; 1, 2{\times}1 \;=\; 1, 2$$

Recursive Data Construction

example: *pow*

$$pow \;=\; 1, 2{\times}pow$$

0. Construct

$$pow_0 \;=\; null$$

$$pow_1 \;=\; 1, 2{\times}pow_0 \;=\; 1, 2{\times}null \;=\; 1, null \;=\; 1$$

$$pow_2 \;=\; 1, 2{\times}pow_1 \;=\; 1, 2{\times}1 \;=\; 1, 2$$

$$pow_3 \;=\; 1, 2{\times}pow_2$$

Recursive Data Construction

example: *pow*

$$pow \;=\; 1, 2{\times}pow$$

0\. Construct

$$pow_0 \;=\; null$$

$$pow_1 \;=\; 1, 2{\times}pow_0 \;=\; 1, 2{\times}null \;=\; 1, null \;=\; 1$$

$$pow_2 \;=\; 1, 2{\times}pow_1 \;=\; 1, 2{\times}1 \;=\; 1, 2$$

$$pow_3 \;=\; 1, 2{\times}pow_2 \;=\; 1, 2{\times}(1, 2)$$

Recursive Data Construction

example: *pow*

$$pow \;=\; 1, 2{\times}pow$$

0. Construct

$$pow_0 \;=\; null$$

$$pow_1 \;=\; 1, 2{\times}pow_0 \;=\; 1, 2{\times}null \;=\; 1, null \;=\; 1$$

$$pow_2 \;=\; 1, 2{\times}pow_1 \;=\; 1, 2{\times}1 \;=\; 1, 2$$

$$pow_3 \;=\; 1, 2{\times}pow_2 \;=\; 1, 2{\times}(1, 2) \;=\; 1, 2, 4$$

Recursive Data Construction

example: *pow*

$$pow \;=\; 1, 2{\times}pow$$

0. Construct

$$pow_0 \;=\; null$$

$$pow_1 \;=\; 1, 2{\times}pow_0 \;=\; 1, 2{\times}null \;=\; 1, null \;=\; 1$$

$$pow_2 \;=\; 1, 2{\times}pow_1 \;=\; 1, 2{\times}1 \;=\; 1, 2$$

$$pow_3 \;=\; 1, 2{\times}pow_2 \;=\; 1, 2{\times}(1, 2) \;=\; 1, 2, 4$$

1. Guess

Recursive Data Construction

example: *pow*

$$pow \;=\; 1, 2{\times}pow$$

0. Construct

$$pow_0 \;=\; null$$

$$pow_1 \;=\; 1, 2{\times}pow_0 \;=\; 1, 2{\times}null \;=\; 1, null \;=\; 1$$

$$pow_2 \;=\; 1, 2{\times}pow_1 \;=\; 1, 2{\times}1 \;=\; 1, 2$$

$$pow_3 \;=\; 1, 2{\times}pow_2 \;=\; 1, 2{\times}(1, 2) \;=\; 1, 2, 4$$

1. Guess $\qquad pow_n \;=\; 2^{0,..n}$

Recursive Data Construction

example: pow

$$pow \;=\; 1, 2{\times}pow$$

0. Construct

$$pow_0 \;=\; null$$

$$pow_1 \;=\; 1, 2{\times}pow_0 \;=\; 1, 2{\times}null \;=\; 1, null \;=\; 1$$

$$pow_2 \;=\; 1, 2{\times}pow_1 \;=\; 1, 2{\times}1 \;=\; 1, 2$$

$$pow_3 \;=\; 1, 2{\times}pow_2 \;=\; 1, 2{\times}(1, 2) \;=\; 1, 2, 4$$

1. Guess $\qquad pow_n \;=\; 2^{0,..n}$

2. Substitute ∞ for n

Recursive Data Construction

example: *pow*

$$pow \;=\; 1, 2 \times pow$$

0. Construct

$$pow_0 \;=\; null$$

$$pow_1 \;=\; 1, 2 \times pow_0 \;=\; 1, 2 \times null \;=\; 1, null \;=\; 1$$

$$pow_2 \;=\; 1, 2 \times pow_1 \;=\; 1, 2 \times 1 \;=\; 1, 2$$

$$pow_3 \;=\; 1, 2 \times pow_2 \;=\; 1, 2 \times (1, 2) \;=\; 1, 2, 4$$

1. Guess $\qquad pow_n \;=\; 2^{0,..n}$

2. Substitute ∞ for n $\qquad pow_\infty \;=\; 2^{0,..\infty}$

Recursive Data Construction

example: *pow*

$$pow \;=\; 1, 2{\times}pow$$

0. Construct

$$pow_0 \;=\; null$$

$$pow_1 \;=\; 1, 2{\times}pow_0 \;=\; 1, 2{\times}null \;=\; 1, null \;=\; 1$$

$$pow_2 \;=\; 1, 2{\times}pow_1 \;=\; 1, 2{\times}1 \;=\; 1, 2$$

$$pow_3 \;=\; 1, 2{\times}pow_2 \;=\; 1, 2{\times}(1, 2) \;=\; 1, 2, 4$$

1. Guess $\quad pow_n \;=\; 2^{0,..n}$

2. Substitute ∞ for n $\quad pow_\infty \;=\; 2^{0,..\infty} \;=\; 2^{nat}$

Recursive Data Construction

example: *pow*

$$pow \;=\; 1, 2{\times}pow$$

3. Test fixed-point.

Recursive Data Construction

example: *pow*

$$pow \;=\; 1, 2{\times}pow$$

3. Test fixed-point.

$$2^{nat} \;=\; 1, 2{\times}2^{nat}$$

Recursive Data Construction

example: *pow*

$$pow \;=\; 1, 2{\times}pow$$

3. Test fixed-point.

$$2^{nat} \;=\; 1, 2{\times}2^{nat}$$

$$= \qquad 2^{nat} \;=\; 2^0, 2^1{\times}2^{nat}$$

Recursive Data Construction

example: *pow*

$$pow \;=\; 1, 2{\times}pow$$

3. Test fixed-point.

$$2^{nat} \;=\; 1, 2{\times}2^{nat}$$

$$= \qquad 2^{nat} \;=\; 2^0, 2^1{\times}2^{nat}$$

$$= \qquad 2^{nat} \;=\; 2^0, 2^{1+nat}$$

Recursive Data Construction

example: *pow*

$$pow \;=\; 1, 2{\times}pow$$

3. Test fixed-point.

$$2^{nat} \;=\; 1, 2{\times}2^{nat}$$

$$= \quad 2^{nat} \;=\; 2^0, 2^1{\times}2^{nat}$$

$$= \quad 2^{nat} \;=\; 2^0, 2^{1+nat}$$

$$= \quad 2^{nat} \;=\; 2^{0,\, 1+nat}$$

Recursive Data Construction

example: *pow*

$$pow \;=\; 1, 2{\times}pow$$

3. Test fixed-point.

$$2^{nat} = 1, 2{\times}2^{nat}$$

$$= \quad 2^{nat} = 2^0, 2^1{\times}2^{nat}$$

$$= \quad 2^{nat} = 2^0, 2^{1+nat}$$

$$= \quad 2^{nat} = 2^{0,\ 1+nat}$$

$$\Leftarrow \quad nat = 0, nat{+}1$$

Recursive Data Construction

example: *pow*

$$pow \;=\; 1,\, 2{\times}pow$$

3. Test fixed-point.

$$\begin{aligned}
& && 2^{nat} = 1,\, 2{\times}2^{nat} \\
&= && 2^{nat} = 2^0,\, 2^1{\times}2^{nat} \\
&= && 2^{nat} = 2^0,\, 2^{1+nat} \\
&= && 2^{nat} = 2^{0,\, 1+nat} \\
&\Leftarrow && nat = 0,\, nat{+}1 \\
&= && \top
\end{aligned}$$

Recursive Data Construction

example: *pow*

$$pow \;=\; 1, 2{\times}pow$$

Recursive Data Construction

example: *pow*

$$pow = 1, 2 \times pow$$

4. Test least fixed-point

Recursive Data Construction

example: *pow*

$$pow \;=\; 1, 2{\times}pow$$

4. Test least fixed-point

$$2^{nat}\colon B$$

$$\Leftarrow \qquad B = 1, 2{\times}B$$

Recursive Data Construction

example: *pow*

$$pow \;=\; 1, 2{\times}pow$$

4. Test least fixed-point

$$\begin{aligned} & 2^{nat}: B \\ = \quad & \forall n: nat\cdot\ 2^n: B \\ \Leftarrow \quad & B = 1, 2{\times}B \end{aligned}$$

Recursive Data Construction

example: *pow*

$pow \;=\; 1, 2{\times}pow$

4. Test least fixed-point

$2^{nat}: B$

$=\quad \forall n: nat\cdot\ 2^n: B$ use *nat* induction with $P\ n = 2^n: B$

$\Leftarrow\quad 2^0: B \;\land\; \forall n: nat\cdot\ 2^n: B \;\Rightarrow\; 2^{n+1}: B$

$\Leftarrow\quad B = 1, 2{\times}B$

Recursive Data Construction

example: *pow*

$pow \;=\; 1, 2{\times}pow$

4. Test least fixed-point

	$2^{nat}: B$	
$=$	$\forall n: nat\cdot\ 2^n: B$	use *nat* induction with $P\ n = 2^n: B$
$\Leftarrow$	$2^0: B \;\wedge\; \forall n: nat\cdot\ 2^n: B \;\Rightarrow\; 2^{n+1}: B$	change variable
$=$	$1: B \;\wedge\; \forall m: 2^{nat}\cdot\ m: B \;\Rightarrow\; 2{\times}m: B$	increase domain
$\Leftarrow$	$1: B \;\wedge\; \forall m: nat\cdot\ m: B \;\Rightarrow\; 2{\times}m: B$	domain change law
$=$	$1: B \;\wedge\; \forall m: nat`B\cdot\ 2{\times}m: B$	increase domain
$\Leftarrow$	$1: B \;\wedge\; \forall m: B\cdot\ 2{\times}m: B$	
$\Leftarrow$	$B = 1, 2{\times}B$	

Recursive Data Construction

Recursive Data Construction

Alternative step 0: instead of *null* use

$name_0 = whatever$

Recursive Data Construction

Alternative step 0: instead of *null* use

$name_0 = whatever$

Alternative step 2: instead of $name_\infty$ use

$§x \cdot ⇕n \cdot x: name_n$