

RECURSIVE DATA DEFINITION

construction:

$0: \text{nat}$

$\text{nat}+1: \text{nat}$

$0: \text{nat}$

$\Rightarrow 0+1: \text{nat}+1$

$\Rightarrow 1: \text{nat}$

$\Rightarrow 1+1: \text{nat}+1$

$\Rightarrow 2: \text{nat}$

$\Rightarrow \dots$

induction:

$0: B \wedge B+1: B \Rightarrow \text{nat}: B$



$0, \text{nat}+1: \text{nat}$

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

construction:

$$P0 \wedge \forall n: \text{nat}. Pn \Rightarrow P(n+1) \Leftarrow \forall n: \text{nat}. Pn$$

induction:

$$P0 \wedge \forall n: \text{nat}. Pn \Rightarrow P(n+1) \Rightarrow \forall n: \text{nat}. Pn$$



induction:

$$P0 \wedge \forall n: \text{nat}. Pn \Rightarrow P(n+1) \Rightarrow \forall n: \text{nat}. Pn$$

$$P0 \vee \exists n: \text{nat}. \neg Pn \wedge P(n+1) \Leftarrow \exists n: \text{nat}. Pn$$

$$\forall n: \text{nat}. Pn \Rightarrow P(n+1) \Rightarrow \forall n: \text{nat}. (P0 \Rightarrow Pn)$$

$$\exists n: \text{nat}. \neg Pn \wedge P(n+1) \Leftarrow \exists n: \text{nat}. (\neg P0 \wedge Pn)$$



alternative induction:

$$\forall n: \text{nat}. (\forall m: \text{nat}. m < n \Rightarrow Pm) \Rightarrow Pn \Rightarrow \forall n: \text{nat}. Pn$$

$$\exists n: \text{nat}. (\forall m: \text{nat}. m < n \Rightarrow \neg Pm) \wedge Pn \Leftarrow \exists n: \text{nat}. Pn$$

In philosophy,

induction means guessing the general case from special cases
(an important skill in mathematics)

deduction means proving, via the rules of logic

In mathematics,

induction is an axiom (sometimes presented as a proof rule)
(mathematical induction is philosophical deduction)

Engineering induction means

test a few cases and hope for the best

Define

$$pow = 2^{nat}$$

or

$$pow = \{p: nat \cdot \exists m: nat \cdot p = 2^m\}$$

or

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

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

or

$$P1 \wedge \forall p: pow \cdot Pp \Rightarrow P(2 \times p) = \forall p: pow \cdot Pp$$



Define

$$int = nat, -nat$$

or

$$0, int+1, int-1: int$$

$$0, B+1, B-1: B \Rightarrow int: B$$

or

$$P0 \wedge (\forall i: int \cdot Pi \Rightarrow P(i+1)) \wedge (\forall i: int \cdot Pi \Rightarrow P(i-1)) \\ = \forall i: int \cdot Pi$$

Least Fixed-Points

construction: $0, nat+1: nat$

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

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

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



grammar:

$exp = "x", exp; "+"; exp$

$B = "x", B; "+"; B \Rightarrow exp: B$

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

$$name_\infty = (\text{expression involving } name_\infty)$$

4. Test least fixed-point

$$B = (\text{expression involving } B) \Rightarrow name_\infty : B$$

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

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

$$pow_n = 2^0, \dots, 2^n$$

2. Substitute ∞ for n

$$pow_{\infty} = 2^{0, \dots, \infty} = 2^{nat}$$

3. Test fixed-point.

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

4. Test least fixed-point

$$\begin{aligned} & 2^{nat} : B \\ = & \forall n : nat. 2^n : B && \text{use } nat \text{ induction} \\ \Leftarrow & 2^0 : B \wedge \forall n : nat. 2^n : B \Rightarrow 2^{n+1} : B && \text{change variable} \\ = & 1 : B \wedge \forall m : 2^{nat}. m : B \Rightarrow 2m : B && \text{increase domain} \\ \Leftarrow & 1 : B \wedge \forall m : nat. m : B \Rightarrow 2m : B \\ = & 1 : B \wedge \forall m : nat' B. 2m : B && \text{increase domain} \\ \Leftarrow & 1 : B \wedge \forall m : B. 2m : B \\ \Leftarrow & B = 1, 2 \times B \end{aligned}$$

Alternative step 0: instead of *null* use

$$name_0 = \textit{whatever}$$

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

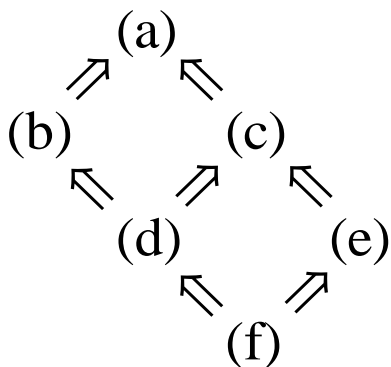
$$\S x \cdot LIM n \cdot x: name_n$$

RECURSIVE PROGRAM SPECIFICATION DEFINITION

$$zap = \text{if } x=0 \text{ then } y:= 0 \text{ else } (x:= x-1. t:= t+1. zap)$$

solutions:

- (a) $x \geq 0 \Rightarrow x'=y'=0 \wedge t' = t+x$
- (b) **if** $x \geq 0$ **then** $x'=y'=0 \wedge t' = t+x$ **else** $t'=\infty$
- (c) $x'=y'=0 \wedge (x \geq 0 \Rightarrow t' = t+x)$
- (d) $x'=y'=0 \wedge$ **if** $x \geq 0$ **then** $t' = t+x$ **else** $t'=\infty$
- (e) $x'=y'=0 \wedge t' = t+x$
- (f) $x \geq 0 \wedge x'=y'=0 \wedge t' = t+x$



$$x \geq 0 \Rightarrow x'=y'=0 \wedge t' = t+x \Leftarrow zap$$

$$zap \Leftarrow \text{if } x=0 \text{ then } y:= 0 \text{ else } (x:= x-1. t:= t+1. zap)$$

Recursive Program Specification Construction

$$zap = \mathbf{if } x=0 \mathbf{ then } y:= 0 \mathbf{ else } (x:= x-1. t:= t+1. zap)$$

$$zap_0 = \top$$

$$\begin{aligned} zap_1 &= \mathbf{if } x=0 \mathbf{ then } y:= 0 \mathbf{ else } (x:= x-1. t:= t+1. zap_0) \\ &= x=0 \Rightarrow x'=y'=0 \wedge t'=t \end{aligned}$$

$$\begin{aligned} zap_2 &= \mathbf{if } x=0 \mathbf{ then } y:= 0 \mathbf{ else } (x:= x-1. t:= t+1. zap_1) \\ &= 0 \leq x < 2 \Rightarrow x'=y'=0 \wedge t' = t+x \end{aligned}$$

$$zap_n = 0 \leq x < n \Rightarrow x'=y'=0 \wedge t' = t+x$$

$$zap_\infty = 0 \leq x < \infty \Rightarrow x'=y'=0 \wedge t' = t+x$$

Alternative step 0: instead of \top use

$$name_0 = \textit{whatever}$$

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

$$\textit{LIM } n \cdot name_n$$

$$zap = \mathbf{if } x=0 \mathbf{ then } y:= 0 \mathbf{ else } (x:= x-1. t:= t+1. zap)$$

$$zap_0 = t' \geq t$$

$$\begin{aligned} zap_1 &= \mathbf{if } x=0 \mathbf{ then } y:= 0 \mathbf{ else } (x:= x-1. t:= t+1. zap_0) \\ &= \mathbf{if } x=0 \mathbf{ then } x'=y'=0 \wedge t'=t \mathbf{ else } t' \geq t+1 \end{aligned}$$

$$\begin{aligned} zap_2 &= \mathbf{if } x=0 \mathbf{ then } y:= 0 \mathbf{ else } (x:= x-1. t:= t+1. zap_1) \\ &= \mathbf{if } x=0 \mathbf{ then } x'=y'=0 \wedge t'=t \\ &\quad \mathbf{else if } x=1 \mathbf{ then } x'=y'=0 \wedge t'=t+1 \mathbf{ else } t' \geq t+2 \\ &= \mathbf{if } 0 \leq x < 2 \mathbf{ then } x'=y'=0 \wedge t' = t+x \mathbf{ else } t' \geq t+2 \end{aligned}$$

$$zap_n = \mathbf{if } 0 \leq x < n \mathbf{ then } x'=y'=0 \wedge t'=t+x \mathbf{ else } t' \geq t+n$$

$$zap_\infty = \mathbf{if } 0 \leq x \mathbf{ then } x'=y'=0 \wedge t'=t+x \mathbf{ else } t'=\infty$$

Loop Definition

$t' \geq t \Leftarrow \text{while } b \text{ do } P$

if b **then** $(P. t := t + inc. \text{while } b \text{ do } P)$ **else** ok

$\Leftarrow \text{while } b \text{ do } P$

$\forall \sigma, \sigma'. (t' \geq t \wedge (\text{if } b \text{ then } (P. t := t + inc. W) \text{ else } ok) \Leftarrow W)$

$\Rightarrow \forall \sigma, \sigma'. (\text{while } b \text{ do } P \Leftarrow W)$



least fixed-points

while b **do** P

$= t' \geq t \wedge (\text{if } b \text{ then } (P. t := t + inc. \text{while } b \text{ do } P) \text{ else } ok)$

$\forall \sigma, \sigma'. (W = t' \geq t \wedge (\text{if } b \text{ then } (P. t := t + inc. W) \text{ else } ok))$

$\Rightarrow \forall \sigma, \sigma'. (\text{while } b \text{ do } P \Leftarrow W)$