Specification

state space	memory	*int*; (0,..20); *char*; *rat*
state	memory contents	–2; 15; "A"; 3.14
prestate	initial state	$\sigma \ = \ \sigma_0; \sigma_1; \sigma_2; \sigma_3 \ = \ i \ ; \ n \ ; \ c \ ; \ x$
poststate	final state	$\sigma' \ = \ \sigma'_0; \sigma'_1; \sigma'_2; \sigma'_3 \ = \ i' \ ; \ n' \ ; \ c' \ ; \ x'$
addresses	low level	0 , 1 , 2 , 3
state variables	high level	$i \ , \ n \ , \ c \ , \ x$
	initial values	$i \ , \ n \ , \ c \ , \ x$
	final values	$i' \ , \ n' \ , \ c' \ , \ x'$

For now: prestate, poststate

Later: time (termination = finite time), space, interaction, communication, ...

Specification

specification of computer behavior: a boolean expression

in variables σ and σ'

We provide a prestate as input.

A computation satisfies a specification by computing a satisfactory poststate as output.

The given prestate and computed poststate must make the specification true.

Specification

specification of computer behavior: a boolean expression

in the initial values x , y , ... and final values x' , y' , ... of some state variables

We provide initial values as input.

A computation satisfies a specification by computing satisfactory final values as output.

The given initial values and computed final values must make the specification true.

Specification

Specification S is **unsatisfiable** for prestate σ : $¢(§\sigma' \cdot S) < 1$

Specification S is **satisfiable** for prestate σ : $¢(§\sigma' \cdot S) \geq 1$

Specification S is **deterministic** for prestate σ : $¢(§\sigma' \cdot S) \leq 1$

Specification S is **nondeterministic** for prestate σ : $¢(§\sigma' \cdot S) > 1$

Specification S is **satisfiable** for prestate σ : $\exists\sigma' \cdot S$

Specification S is **implementable**: $\forall\sigma \cdot \exists\sigma' \cdot S$

Specification

examples

$x' = x+1 \;\land\; y' = y$	implementable, deterministic
$x' > x$	implementable, nondeterministic
$\top$	implementable, extremely nondeterministic
$\bot$	unimplementable, overly deterministic
$x\geq 0 \;\land\; y'=0$	unimplementable, overly deterministic
$x\geq 0 \;\Rightarrow\; y'=0$	implementable, nondeterministic

ok	$=$	$\sigma'=\sigma$	$=$	$x'=x \land y'=y \land \ldots$
$x:= e$	$=$	$\sigma' \;=\; \sigma \triangleleft address\text{ “}x\text{”} \triangleright e$	$=$	$x'=e \land y'=y \land \ldots$
$x:= x+y$			$=$	$x'=x+1 \land y'=y$

if $x=y$ **then** $x:= x+1$ **else** $x'+y' = 3$ **fi**

dependent composition

$$S . R \quad = \quad \exists x'', y'', ...\cdot \quad (\text{substitute } x'', y'', ... \text{ for } x', y', ... \text{ in } S)$$
$$\land \quad (\text{substitute } x'', y'', ... \text{ for } x, y, ... \text{ in } R)$$

In integer variable x

$$x'=x \lor x'=x+1 \;.\; x'=x \lor x'=x+1$$

$$= \quad \exists x''\cdot \; (x''=x \lor x''=x+1) \land (x'=x'' \lor x'=x''+1) \qquad \text{distribute } \land \text{ over } \lor$$

$$= \quad \exists x''\cdot \quad x''=x \land x'=x'' \;\lor\; x''=x+1 \land x'=x''$$
$$\lor \quad x''=x \land x'=x''+1 \;\lor\; x''=x+1 \land x'=x''+1 \qquad \text{distribute } \exists \text{ over } \lor$$

$$= \quad (\exists x''\cdot\; x''=x \land x'=x'') \lor (\exists x''\cdot\; x''=x+1 \land x'=x'')$$
$$\lor \; (\exists x''\cdot\; x''=x \land x'=x''+1) \lor (\exists x''\cdot\; x''=x+1 \land x'=x''+1) \qquad \text{One-Point Law 4 times}$$

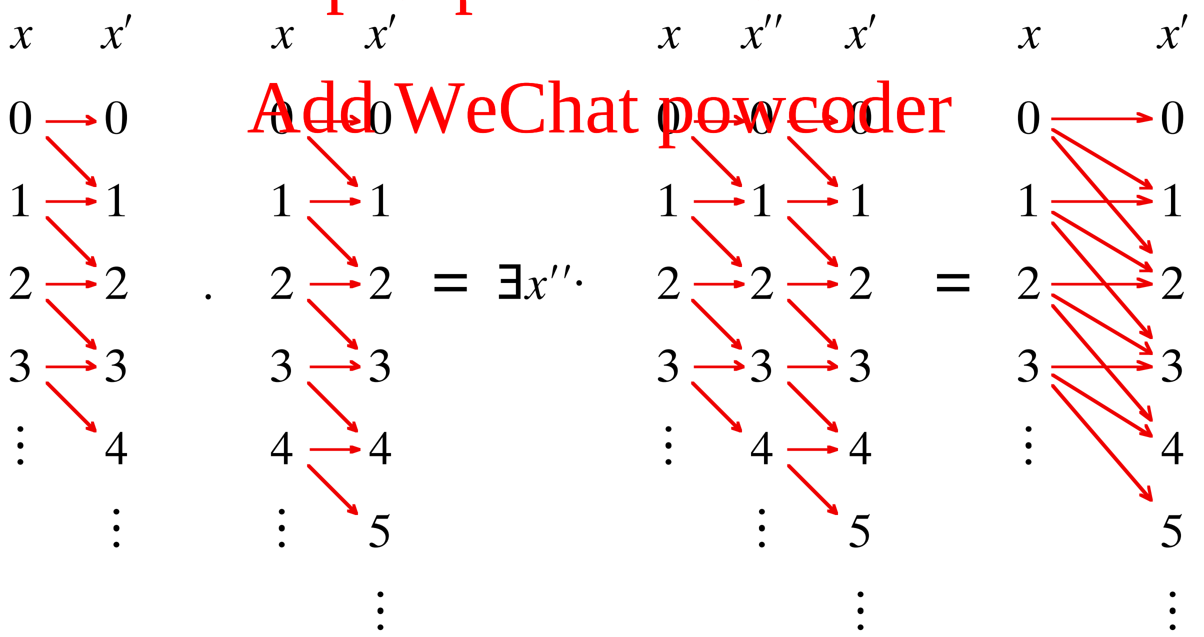
$$= \quad x'=x \lor x'=x+1 \lor x'=x+2$$

dependent composition

$$S.R \quad = \quad \exists x'', y'', ...\cdot \quad (\text{substitute } x'', y'', ... \text{ for } x', y', ... \text{ in } S)$$
$$\land \quad (\text{substitute } x'', y'', ... \text{ for } x, y, ... \text{ in } R)$$

In integer variable x

$$x'=x \lor x'=x+1 \;.\; x'=x \lor x'=x+1$$



dependent composition

$$S.\,R \;=\; \exists x'', y'', ...\cdot \quad (\text{substitute } x'', y'', ... \text{ for } x', y', ... \text{ in } S\,)$$
$$\wedge \quad (\text{substitute } x'', y'', ... \text{ for } x, y, ... \text{ in } R\,)$$

In integer variables x and y

	$x{:=}\ 3.\ \ y{:=}\ x{+}y$	eliminate assignments first
$=$	$x'{=}3 \wedge y'{=}y.\ \ x'{=}x \wedge y'{=}x{+}y$	then eliminate dependent composition
$=$	$\exists x'', y'': int\cdot\ x''{=}3 \wedge y''{=}y \ \wedge\ x'{=}x'' \wedge y'{=}x''{+}y''$	use One-Point Law twice
$=$	$x'{=}3 \ \wedge\ y' = 3{+}y$	

specification laws

$ok.\ P \;=\; P.\ ok \;=\; P$	Identity Law
$P.\ (Q.\ R) \;=\; (P.\ Q).\ R$	Associative Law
$\textbf{if } b \textbf{ then } P \textbf{ else } P \textbf{ fi} \;=\; P$	Idempotent Law
$\textbf{if } b \textbf{ then } P \textbf{ else } Q \textbf{ fi} \;=\; \textbf{if } \neg b \textbf{ then } Q \textbf{ else } P \textbf{ fi}$	Case Reversal Law
$P \;=\; \textbf{if } b \textbf{ then } b \Rightarrow P \textbf{ else } \neg b \Rightarrow P \textbf{ fi}$	Case Creation Law
$\textbf{if } b \textbf{ then } S \textbf{ else } R \textbf{ fi} \;=\; b{\wedge}S \vee \neg b{\wedge}R$	Case Analysis Law
$\textbf{if } b \textbf{ then } S \textbf{ else } R \textbf{ fi} \;=\; (b{\Rightarrow}S) \wedge (\neg b{\Rightarrow}R)$	Case Analysis Law
$P{\vee}Q.\ R{\vee}S \;=\; (P.\ R) \vee (P.\ S) \vee (Q.\ R) \vee (Q.\ S)$	Distributive Law
$\textbf{if } b \textbf{ then } P \textbf{ else } Q \textbf{ fi} \wedge R \;=\; \textbf{if } b \textbf{ then } P{\wedge}R \textbf{ else } Q{\wedge}R \textbf{ fi}$	Distributive Law
$\textbf{if } b \textbf{ then } P \textbf{ else } Q \textbf{ fi}.\ R \;=\; \textbf{if } b \textbf{ then } P.\ R \textbf{ else } Q.\ R \textbf{ fi}$	Distributive Law
$x{:=}\ \textbf{if } b \textbf{ then } e \textbf{ else } f \textbf{ fi} \;=\; \textbf{if } b \textbf{ then } x{:=}\ e \textbf{ else } x{:=}\ f \textbf{ fi}$	Functional-Imperative Law
$x{:=}\ e.\ P \;=\; (\text{for } x \text{ substitute } e \text{ in } P\,)$	Substitution Law

substitution law

$$x{:=}\, e.\, P \;=\; (\text{for } x \text{ substitute } e \text{ in } P\,)$$

$$x{:=}\, y{+}1.\;\; y' > x' \;=\; y' > x'$$

$$x{:=}\, x{+}1.\;\; y'{>}x \land x'{>}x \;=\; y' > x{+}1 \;\land\; x' > x{+}1$$

$$x{:=}\, y{+}1.\;\; y' = 2{\times}x \;=\; y' = 2{\times}(y{+}1)$$

$$x{:=}\, 1.\;\; x{\ge}1 \Rightarrow \exists x\cdot\;\; y' = 2{\times}x \;=\; 1{\ge}1 \Rightarrow \exists x\cdot\;\; y' = 2{\times}x \;=\; \mathit{even}\; y'$$

$$x{:=}\, y.\;\; x{\ge}1 \Rightarrow \exists y\cdot\;\; y' = x{\times}y \;=\; x{:=}\, y.\;\; x{\ge}1 \Rightarrow \exists k\cdot\;\; y' = x{\times}k$$

$$=\; y{\ge}1 \Rightarrow \exists k\cdot\;\; y' = y{\times}k$$

$$x{:=}\, 1.\;\; ok \;=\; x{:=}\, 1.\;\; x'{=}x \land y'{=}y \;=\; x'{=}1 \land y'{=}y$$

$$x{:=}\, 1.\;\; y{:=}\, 2 \;=\; x{:=}\, 1.\;\; x'{=}x \land y'{=}2 \;=\; x'{=}1 \land y'{=}2$$

substitution law

$$x{:=}\,e.\,P \;=\; (\text{for } x \text{ substitute } e \text{ in } P\,)$$

$$
\begin{aligned}
& x{:=}\,1.\;\; y{:=}\,2.\;\; x{:=}\,x{+}y \\
= \quad & x{:=}\,1.\;\; y{:=}\,2.\;\; x' = x{+}y \;\wedge\; y'{=}y \\
= \quad & x{:=}\,1.\;\; x' = x{+}2 \;\wedge\; y'{=}2 \\
= \quad & x'{=}3 \wedge y'{=}2
\end{aligned}
$$

$$
\begin{aligned}
& x{:=}\,1.\;\; x' > x.\;\; x' = x{+}1 \\
= \quad & x' > 1.\;\; x' = x{+}1 \\
= \quad & \exists x'', y''\cdot\;\; x''{>}1 \;\wedge\; x' = x''{+}1 \\
= \quad & \exists x''\cdot\;\; x''{>}1 \;\wedge\; x' = x''{+}1 \\
= \quad & \exists x''\cdot\;\; x''{>}1 \;\wedge\; x'' = x'{-}1 \\
= \quad & x'{-}1 > 1 \\
= \quad & x'{>}2
\end{aligned}
$$

Refinement

Specification P (the problem) is **refined** by specification S (the solution) if and only if P is satisfied whenever S is satisfied.

$$\forall \sigma, \sigma' \cdot P \Leftarrow S$$

$$x'>x \quad \Leftarrow \quad x'=x+1 \land y'=y \quad = \quad x:= x+1$$

$$x' \le x \quad \Leftarrow \quad \textbf{if } x=0 \textbf{ then } x'=x \textbf{ else } x'<x \textbf{ fi} \quad = \quad x=0 \land x'=x \;\lor\; x \ne 0 \land x'<x$$

$$x'>y'>x \quad \Leftarrow \quad y:= x+1.\; x:= y+1$$

$$= \quad y:= x+1.\; x'=y+1 \land y'=y$$

$$= \quad x'=x+2 \land y'=x+1$$

condition: specification that refers to (at most) one state

initial condition, precondition: refers to (at most) the initial state (prestate)

final condition, postcondition: refers to (at most) the final state (poststate)

exact precondition for P to be refined by S : $\forall\sigma'\cdot P \Leftarrow S$

exact postcondition for P to be refined by S : $\forall\sigma\cdot P \Leftarrow S$

sufficient $\Rightarrow$ exact $\Rightarrow$ necessary

(the exact precondition for $x'>5$ to be refined by $x:=x+1$)

$=\quad \forall x' \cdot x'>5 \Leftarrow (x:=x+1)$

$=\quad \forall x' \cdot x'>5 \Leftarrow x'=x+1$ One-Point Law

$=\quad x+1>5$

$=\quad x>4$

$x>4 \Rightarrow x'>5 \quad \Leftarrow \quad x:=x+1$

one-point laws

$\exists v \cdot v=e \land P \quad = \quad$ (replace v with e in P)

$\forall v \cdot v=e \Rightarrow P \quad = \quad$ (replace v with e in P)

(the exact postcondition for $x{>}4$ to be refined by $x{:=}\ x{+}1$)

$=\quad \forall x\cdot\ x{>}4 \Leftarrow (x{:=}\ x{+}1)$

$=\quad \forall x\cdot\ x{>}4 \Leftarrow x'{=}x{+}1$

$=\quad \forall x\cdot\ x{>}4 \Leftarrow x{=}x'{-}1$ One-Point Law

$=\quad x'{-}1 > 4$

$=\quad x' > 5$

$x'{>}5 \Rightarrow x{>}4 \quad \Leftarrow \quad x{:=}\ x{+}1$

$x{\le}4 \Rightarrow x'{\le}5 \quad \Leftarrow \quad x{:=}\ x{+}1$

contrapositive law

$a{\Rightarrow}b \quad = \quad \neg b{\Rightarrow}\neg a$

condition laws

$$C \wedge (P.\ Q) \Leftarrow C\wedge P.\ Q$$

$$C \Rightarrow (P.Q) \Leftarrow C\Rightarrow P.\ Q$$

$$(P.Q) \wedge C' \Leftarrow P.\ Q\wedge C'$$

$$(P.Q) \Leftarrow C' \Leftarrow P.\ Q\Leftarrow C'$$

$$P.\ C\wedge Q \Leftarrow P\wedge C'.\ Q$$

$$P.\ Q \Leftarrow P\wedge C'.\ C\Rightarrow Q$$

precondition law

C is a sufficient precondition for P to be refined by S

if and only if $C\Rightarrow P$ is refined by S .

postcondition law

C' is a sufficient postcondition for P to be refined by S

if and only if $C'\Rightarrow P$ is refined by S .

A **program** is an implemented specification.

ok

$x{:=}\ e$

if b **then** P **else** Q **fi**

$P.Q$

binaries, numbers, characters

bunches, sets, strings, lists

NOT functions, quantifiers

An implementable specification that is refined by a program is a program.

Recursion is allowed.

$x{\ge}0 \Rightarrow x'{=}0 \quad \Leftarrow \quad$ **if** $x{=}0$ **then** ok **else** $x{:=}\ x{-}1.\ \ x{\ge}0 \Rightarrow x'{=}0$ **fi**

refinement by steps

If $A \Leftarrow \textbf{if } b \textbf{ then } C \textbf{ else } D \textbf{ fi}$ and $C \Leftarrow E$ and $D \Leftarrow F$,

then $A \Leftarrow \textbf{if } b \textbf{ then } E \textbf{ else } F \textbf{ fi}$.

If $A \Leftarrow B.C$ and $B \Leftarrow D$ and $C \Leftarrow E$, then $A \Leftarrow D.E$.

If $A \Leftarrow B$ and $B \Leftarrow C$, then $A \Leftarrow C$.

refinement by parts

If $A \Leftarrow \textbf{if } b \textbf{ then } C \textbf{ else } D \textbf{ fi}$ and $E \Leftarrow \textbf{if } b \textbf{ then } F \textbf{ else } G \textbf{ fi}$,

then $A \land E \Leftarrow \textbf{if } b \textbf{ then } C \land F \textbf{ else } D \land G \textbf{ fi}$.

If $A \Leftarrow B.C$ and $D \Leftarrow E.F$, then $A \land D \Leftarrow B \land E.\ C \land F$.

If $A \Leftarrow B$ and $C \Leftarrow D$, then $A \land C \Leftarrow B \land D$.

refinement by cases

$$P \Leftarrow \textbf{if } b \textbf{ then } Q \textbf{ else } R \textbf{ fi}$$

if and only if $\quad P \Leftarrow b \land Q$ and $P \Leftarrow \neg b \land R$

List Summation

List of numbers L ; number variable s .

$s' = \Sigma L \quad \Leftarrow \quad s{:=}\ 0.\ \ n{:=}\ 0.\ \ s' = s + \Sigma\, L\, [n;..\#L]$

$s' = s + \Sigma\, L\, [n;..\#L] \quad \Leftarrow$

if $n{=}\#L$ **then** $n{=}\#L \ \Rightarrow\ s' = s + \Sigma\, L\, [n;..\#L]$

else $n{\ne}\#L \ \Rightarrow\ s' = s + \Sigma\, L\, [n;..\#L]$ **fi**

$n{=}\#L \ \Rightarrow\ s' = s + \Sigma\, L\, [n;..\#L] \quad \Leftarrow \quad ok$

$n{\ne}\#L \ \Rightarrow\ s' = s + \Sigma\, L\, [n;..\#L] \quad \Leftarrow \quad s{:=}\ s{+}Ln.\ \ n{:=}\ n{+}1.\ \ s' = s + \Sigma\, L\, [n;..\#L]$

compilation

$$A \Leftarrow s{:=}\,0.\ \ n{:=}\,0.\ \ B$$

$$B \Leftarrow \textbf{if } n{=}\#L \textbf{ then } C \textbf{ else } D \textbf{ fi}$$

$$C \Leftarrow ok$$

$$D \Leftarrow s{:=}\, s{+}Ln.\ \ n{:=}\, n{+}1.\ \ B$$

Refinement by Steps = in-line macro-expansion

$$B \Leftarrow \textbf{if } n{=}\#L \textbf{ then } ok \textbf{ else } s{:=}\, s{+}Ln.\ \ n{:=}\, n{+}1.\ \ B \textbf{ fi}$$

translation

```
void A(void) {s = 0;  n = 0;  B( );}

void B(void) {if (n==sizeof(L)/sizeof(L[0])); else {s+=L[n];  n++;  B( );}}
```

```
s = 0;  n = 0;

B: if (n==sizeof(L)/sizeof(L[0])); else {s+=L[n];  n++;  goto B;}
```

Binary Exponentiation

Given natural variables x and y, write a program for $y' = 2^x$.

$y'=2^x \quad \Leftarrow \quad \textbf{if } x=0 \textbf{ then } x=0 \Rightarrow y'=2^x \textbf{ else } x>0 \Rightarrow y'=2^x \textbf{ fi}$

$x=0 \Rightarrow y'=2^x \quad \Leftarrow \quad y{:=}\ 1.\ \ x{:=}\ 3$

$x>0 \Rightarrow y'=2^x \quad \Leftarrow \quad x>0 \Rightarrow y'=2^{x-1}.\ \ y'=2\times y$

$x>0 \Rightarrow y'=2^{x-1} \quad \Leftarrow \quad x'=x-1.\ \ y'=2^x$

$y'=2\times y \quad \Leftarrow \quad y{:=}\ 2\times y.\ \ x{:=}\ 5$

$x'=x-1 \quad \Leftarrow \quad x{:=}\ x-1.\ \ y{:=}\ 7$

Binary Exponentiation

Given natural variables x and y, write a program for $y' = 2^x$.

$A \Leftarrow$ **if** $x{=}0$ **then** B **else** C **fi**

$B \Leftarrow y{:=}\ 1.\ \ x{:=}\ 3$

$C \Leftarrow D.\ E$

$D \Leftarrow F.\ A$

$E \Leftarrow y{:=}\ 2{\times}y.\ \ x{:=}\ 5$

$F \Leftarrow x{:=}\ x{-}1.\ \ y{:=}\ 7$

$A \Leftarrow$ **if** $x{=}0$ **then** $y{:=}\ 1.\ \ x{:=}\ 3$ **else** $x{:=}\ x{-}1.\ \ y{:=}\ 7.\ \ A.\ \ y{:=}\ 2{\times}y.\ \ x{:=}\ 5$ **fi**

```
int x, y;

void A (void) {if (x==0) {y = 1;  x = 3;} else {x = x–1;  y = 7;  A( );  y = 2*y;  x = 5;}}

x = 5;  A( );  printf ("%i", y);
```

Time

$\sigma \;=\; t\,;\; x\,;\; y\,;\; ...$

state = time variable; memory variables

t is the time at which execution starts

t' is the time at which execution ends

extended natural or extended nonnegative real

Specification S is **implementable** if and only if

$\forall\sigma\cdot\ \exists\sigma'\cdot\ S \land t' \ge t$

real time

$t:= t+$(the time to evaluate and store e). $x:= e$

$t:= t+$(the time to evaluate b and branch). **if** b **then** P **else** Q **fi**

$t:= t+$(the time for the call and return). P

$t' = t + f\,\sigma$

$t' \le t + f\,\sigma$

$t' \ge t + f\,\sigma$

real time

$P \Leftarrow t:= t+1.\ \mathbf{if}\ x=0\ \mathbf{then}\ ok\ \mathbf{else}\ t:= t+1.\ x:= x-1.\ t:= t+1.\ P\ \mathbf{fi}$

is a theorem when

$P = x'=0$

$P = \mathbf{if}\ x\ge 0\ \mathbf{then}\ t'=t+3\times x+1\ \mathbf{else}\ t'=\infty\ \mathbf{fi}$

$P = \mathbf{if}\ x\ge 0\ \mathbf{then}\ x'=0 \land t'=t+3\times x+1\ \mathbf{else}\ t'=\infty\ \mathbf{fi}$

$P = x'=0 \land \mathbf{if}\ x\ge 0\ \mathbf{then}\ t'=t+3\times x+1\ \mathbf{else}\ t'=\infty\ \mathbf{fi}$

recursive time

- Each recursive call costs time 1.
- All else is free.

$P \Leftarrow$ **if** $x=0$ **then** ok **else** $x:=x-1.\ t:=t+1.\ P$ **fi**

is a theorem when

$P = x'=0$

$P =$ **if** $x\ge 0$ **then** $t'=t+x$ **else** $t'=\infty$ **fi**

$P =$ **if** $x\ge 0$ **then** $x'=0 \wedge t'=t+x$ **else** $t'=\infty$ **fi**

$P = x'=0 \wedge$ **if** $x\ge 0$ **then** $t'=t+x$ **else** $t'=\infty$ **fi**

Recursion can be direct or indirect.

In every loop of calls, there must be a time increment of at least one time unit.

Prove $R \Leftarrow$ **if** $x{=}1$ **then** ok **else** $x{:=}\ div\ x\ 2.\ \ t{:=}\ t{+}1.\ \ R$ **fi**

where $R{=}\ \ x'{=}1 \wedge$ **if** $x{\ge}1$ **then** $t' \le t + log\ x$ **else** $t'{=}\infty$ **fi**

$$= \ \ x'{=}1 \wedge (x{\ge}1 \ \Rightarrow\ t' \le t + log\ x) \wedge (x{<}1 \Rightarrow t'{=}\infty)$$

use Refinement by Parts; prove:

$x'{=}1 \ \Leftarrow$ **if** $x{=}1$ **then** ok **else** $x{:=}\ div\ x\ 2.\ \ t{:=}\ t{+}1.\ \ x'{=}1$ **fi**

$x{\ge}1 \ \Rightarrow\ t' \le t + log\ x \ \Leftarrow$ **if** $x{=}1$ **then** ok **else** $x{:=}\ div\ x\ 2.\ \ t{:=}\ t{+}1.\ \ x{\ge}1 \Rightarrow t' \le t + log\ x$ **fi**

$x{<}1 \ \Rightarrow\ t'{=}\infty \ \Leftarrow$ **if** $x{=}1$ **then** ok **else** $x{:=}\ div\ x\ 2.\ \ t{:=}\ t{+}1.\ \ x{<}1 \Rightarrow t'{=}\infty$ **fi**

Prove $R \Leftarrow$ **if** $x{=}1$ **then** ok **else** $x{:=}\ div\ x\ 2.\ \ t{:=}\ t{+}1.\ \ R$ **fi**

where $R =\ x'{=}1 \land$ **if** $x{\ge}1$ **then** $t' \le t + log\ x$ **else** $t'{=}\infty$ **fi**

$$=\ x'{=}1 \land (x{\ge}1 \Rightarrow t' \le t + log\ x) \land (x{<}1 \Rightarrow t'{=}\infty)$$

use Refinement by Parts and Cases; prove:

$x'{=}1 \Leftarrow x{=}1 \land ok$

$x'{=}1 \Leftarrow x{\ne}1 \land (x{:=}\ div\ x\ 2.\ \ t{:=}\ t{+}1.\ \ x'{=}1)$

$x{\ge}1 \Rightarrow t' \le t + log\ x \Leftarrow x{=}1 \land ok$

$x{\ge}1 \Rightarrow t' \le t + log\ x \Leftarrow x{\ne}1 \land (x{:=}\ div\ x\ 2.\ \ t{:=}\ t{+}1.\ \ x{\ge}1 \Rightarrow t' \le t + log\ x)$

$x{<}1 \Rightarrow t'{=}\infty \Leftarrow x{=}1 \land ok$

$x{<}1 \Rightarrow t'{=}\infty \Leftarrow x{\ne}1 \land (x{:=}\ div\ x\ 2.\ \ t{:=}\ t{+}1.\ \ x{<}1 \Rightarrow t'{=}\infty)$

$(x{\ge}1 \;\Rightarrow\; t' \le t + log\ x \;\;\Leftarrow\;\; x{=}1 \wedge x'{=}x \wedge t'{=}t)$ context $x{=}1$ and $t'{=}t$

$= \quad (1{\ge}1 \;\Rightarrow\; t \le t + log\ 1 \;\;\Leftarrow\;\; x{=}1 \wedge x'{=}x \wedge t'{=}t)$ simplify

$= \quad (\top \;\;\Leftarrow\;\; x{=}1 \wedge x'{=}x \wedge t'{=}t)$ base law

$= \quad \top$

$$(x \ge 1 \;\Rightarrow\; t' \le t + log\ x \;\;\Leftarrow\;\; x \ne 1 \wedge (div\ x\ 2 \ge 1 \;\Rightarrow\; t' \le t + 1 + log\ (div\ x\ 2)))$$

portation

$$= \quad x \ne 1 \wedge (div\ x\ 2 \ge 1 \;\Rightarrow\; t' \le t + 1 + log\ (div\ x\ 2)) \wedge x \ge 1 \;\;\Rightarrow\;\; t' \le t + log\ x$$

simplify

$$= \quad x > 1 \wedge (x > 1 \;\Rightarrow\; t' \le t + 1 + log\ (div\ x\ 2)) \;\;\Rightarrow\;\; t' \le t + log\ x$$

discharge

$$= \quad x > 1 \;\wedge\; t' \le t + 1 + log\ (div\ x\ 2) \;\;\Rightarrow\;\; t' \le t + log\ x$$

portation

$$= \quad x > 1 \;\;\Rightarrow\;\; (t' \le t + 1 + log\ (div\ x\ 2) \;\Rightarrow\; t' \le t + log\ x)$$

Connection Law $t' \le a \Rightarrow t' \le b \;\;\Leftarrow\;\; a \le b$

$$\Leftarrow \quad x > 1 \;\;\Rightarrow\;\; t + 1 + log\ (div\ x\ 2) \;\le\; t + log\ x$$

subtract $t+1$ from each side

$$= \quad x > 1 \;\;\Rightarrow\;\; log\ (div\ x\ 2) \;\le\; log\ x - 1$$

property of log

$$= \quad x > 1 \;\;\Rightarrow\;\; log\ (div\ x\ 2) \;\le\; log\ (x/2)$$

log is monotonic for $x > 0$

$$\Leftarrow \quad div\ x\ 2 \le x/2$$

$$= \quad \top$$

$$(x{<}1 \Rightarrow t'{=}\infty \quad \Leftarrow \quad x{=}1 \;\wedge\; x'{=}x \;\wedge\; t'{=}t) \qquad \text{portation}$$

$$= \quad x{<}1 \wedge x{=}1 \;\wedge\; x'{=}x \;\wedge\; t'{=}t \quad \Rightarrow \quad t'{=}\infty \qquad \text{generic, base}$$

$$= \quad \bot \quad \Rightarrow \quad t'{=}\infty \qquad \text{base}$$

$$= \quad \top$$

$$
\begin{array}{lll}
 & (x<1 \Rightarrow t'=\infty \;\;\Leftarrow\;\; x\ne 1 \;\wedge\; (div\ x\ 2 < 1 \;\Rightarrow\; t'=\infty)) & \text{portation} \\
= & x<1 \wedge x\ne 1 \wedge (div\ x\ 2 < 1 \;\Rightarrow\; t'=\infty) \;\;\Rightarrow\;\; t'=\infty & \text{discharge} \\
= & x<1 \wedge t'=\infty \;\;\Rightarrow\;\; t'=\infty & \text{specialization} \\
= & \top &
\end{array}
$$

Termination

$x'=2 \quad \Leftarrow \quad t:= t+1.\ \ x'=2$

complain only if $x' \neq 2$

$x'=2 \ \land \ t'<\infty$

unimplementable

$x'=2 \land (t<\infty \Rightarrow t'<\infty) \quad \Leftarrow \quad t:= t+1.\ \ x'=2 \land (t<\infty \Rightarrow t'<\infty)$

complain only if $x' \neq 2 \ \lor \ t<\infty \land t'=\infty$

$x'=2 \land t' \le t+1 \quad \Leftarrow \quad t:= t+1.\ \ x'=2 \land t' \le t+1$ ✗

$x'=2 \land t' \le t+1 \quad \Leftarrow \quad x:= 2$

Linear Search

Find the first occurrence of item x in list L. The execution time must be linear in $\#L$.

$\neg\, x\colon L\,(0,..h') \;\wedge\; (Lh'=x \vee h'=\#L)$

$$[\; ? \;;\; ? \;;\; ? \;;\; ? \;;\; x \;;\; ? \;;\; ? \;;\; ? \;;\; ? \;]$$

$0 \longleftarrow$ x is not in this part $\longrightarrow h'$ $\qquad\qquad \#L$

Linear Search

Find the first occurrence of item x in list L. The execution time must be linear in $\#L$.

$$\neg\ x{:}\ L\,(0,..h') \ \wedge\ (Lh'{=}x \vee h'{=}\#L) \ \Leftarrow\ h{:=}\,0.\ h{\le}\#L \Rightarrow\ \neg\ x{:}\ L\,(h,..h') \ \wedge\ (Lh'{=}x \vee h'{=}\#L)$$

$$h{\le}\#L \Rightarrow \neg\ x{:}\ L\,(h,..h') \ \wedge\ (Lh'{=}x \vee h'{=}\#L) \ \Leftarrow$$

$$\textbf{if } h{=}\#L \textbf{ then } ok \textbf{ else } h{<}\#L \Rightarrow\ \neg\ x{:}\ L\,(h,..h') \ \wedge\ (Lh'{=}x \vee h'{=}\#L) \textbf{ fi}$$

$$h{<}\#L \Rightarrow \neg\ x{:}\ L\,(h,..h') \ \wedge\ (Lh'{=}x \vee h'{=}\#L) \ \Leftarrow$$

$$\textbf{if } Lh{=}x \textbf{ then } ok \textbf{ else } h{:=}\,h{+}1.\ h{\le}\#L \Rightarrow\ \neg\ x{:}\ L\,(h,..h') \ \wedge\ (Lh'{=}x \vee h'{=}\#L) \textbf{ fi}$$

Linear Search

timing

$t' \le t+\#L \;\;\Leftarrow\;\; h:=0.\;\; h\le\#L \;\Rightarrow\; t' \le t+\#L-h$

$h\le\#L \;\Rightarrow\; t' \le t+\#L-h \;\;\Leftarrow\;\;$ **if** $h=\#L$ **then** ok **else** $h<\#L \;\Rightarrow\; t' \le t+\#L-h$ **fi**

$h<\#L \;\Rightarrow\; t' \le t+\#L-h \;\;\Leftarrow\;\;$ **if** $L\,h=x$ **then** ok **else** $h:=h+1.\;\; t:=t+1.\;\; h\le\#L \;\Rightarrow\; t' \le t+\#L-h$ **fi**

	$h:=h+1.\;\; t:=t+1.\;\; h\le\#L \;\Rightarrow\; t' \le t+\#L-h$	substitution law
$=$	$h:=h+1.\;\; h\le\#L \;\Rightarrow\; t' \le t+1+\#L-h$	substitution law
$=$	$h+1\le\#L \;\Rightarrow\; t' \le t+1+\#L-h-1$	simplify
$=$	$h<\#L \;\Rightarrow\; t' \le t+\#L-h$	

Linear Search

Find the first occurrence of item x in list L. The execution time must be linear in $\#L$.

nonempty

$\neg\, x\colon L\,(0,..h') \;\wedge\; (Lh'{=}x \vee h'{=}\#L) \;\Leftarrow\; h:=0.\;\; h{\le}\#L \Rightarrow\; \neg\, x\colon L\,(h,..h') \;\wedge\; (Lh'{=}x \vee h'{=}\#L)$

<

$h{\le}\#L \Rightarrow \neg\, x\colon L\,(h,..h') \;\wedge\; (Lh'{=}x \vee h'{=}\#L) \;\Leftarrow$

if $h{=}\#L$ **then** ok **else** $h{<}\#L \Rightarrow\; \neg\, x\colon L\,(h,..h') \;\wedge\; (Lh'{=}x \vee h'{=}\#L)$ **fi**

$h{<}\#L \Rightarrow \neg\, x\colon L\,(h,..h') \;\wedge\; (Lh'{=}x \vee h'{=}\#L) \;\Leftarrow$

if $Lh{=}x$ **then** ok **else** $h:=h{+}1.\;\; h{\le}\#L \Rightarrow\; \neg\, x\colon L\,(h,..h') \;\wedge\; (Lh'{=}x \vee h'{=}\#L)$ **fi**

Binary Search

Find an occurrence of item x in nonempty sorted list L .

The execution time must be logarithmic in $\#L$.

$(x\colon L\,(0,..\#L) \;=\; p' \;\;\Rightarrow\;\; Lh' = x) \;\;\Leftarrow\;\; h:= 0.\;\; j:= \#L.\;\; h<j \Rightarrow R$

$h<j \Rightarrow R \;\;\Leftarrow\;\;$ **if** $j{-}h = 1$ **then** $p:= Lh{=}x$ **else** $j{-}h{\ge}2 \Rightarrow R$ **fi**

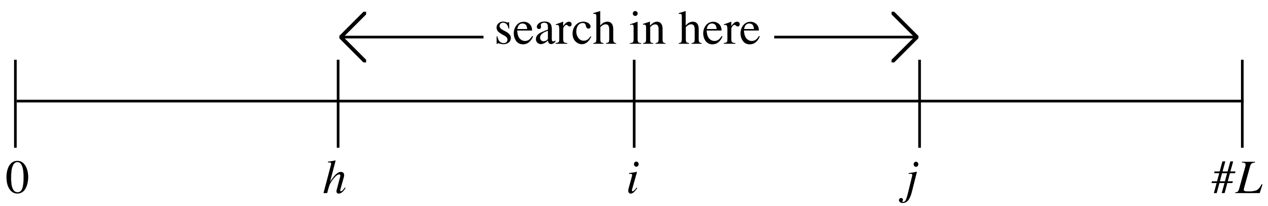
$j{-}h{\ge}2 \Rightarrow R \;\;\Leftarrow\;\; j{-}h{\ge}2 \;\Rightarrow\; h'{=}h{<}i'{<}j{=}j'$.

if $Li{\le}x$ **then** $h:= i$ **else** $j:= i$ **fi**.

$h<j \Rightarrow R$

$j{-}h{\ge}2 \;\Rightarrow\; h'{=}h{<}i'{<}j{=}j' \;\;\Leftarrow\;\; i:= div\,(h{+}j)\;2$

Define $R \;=\; (x\colon L\,(h,..j) \;=\; p' \;\;\Rightarrow\;\; Lh' = x)$



$\top \Leftarrow h := 0.\ \ j := \#L.\ \ U$

$U \Leftarrow$ **if** $j–h = 1$ **then** $p := Lh=x$ **else** V **fi**

$V \Leftarrow$ $i := div\ (h+j)\ 2.$

if $Li \le x$ **then** $h := i$ **else** $j := i$ **fi**.

$t := t+1.\ \ U$

$\#L$	=	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
$t'–t$	=	0	1	2	2	3	3	3	3	4	4	4	4	4	4	4	4	5	5

$\top = t' \le t + ceil\ (log\ (\#L))$

$U = h<j \Rightarrow t' \le t + ceil\ (log\ (j–h))$

$V = j–h\ge 2 \Rightarrow t' \le t + ceil\ (log\ (j–h))$

Three Levels of Care

highest write all specifications

prove all refinements (an automated theorem prover can help)

middle write all specifications

but don't prove the refinements (just argue them informally)

lowest don't bother with specifications

don't bother with refinements

just write code

Fast Exponentiation

Given rational variables x and z and natural variable y, write a program for $z' = x^y$ that runs fast without using exponentiation.

$z'=x^y \quad \Leftarrow \quad z:= 1.\ z' = z \times x^y$

Proof:	$z:= 1.\ z' = z \times x^y$	Substitution Law
$=$	$z' = 1 \times x^y$	1 is identity for $\times$
$=$	$z' = x^y$	

Fast Exponentiation

Given rational variables x and z and natural variable y, write a program for $z' = x^y$ that runs fast without using exponentiation.

$z'=x^y \quad \Leftarrow \quad z := 1.\ z' = z \times x^y$

$z' = z \times x^y \quad \Leftarrow \quad$ **if** $y=0$ **then** ok **else** $y>0 \Rightarrow z' = z \times x^y$ **fi**

Proof: $y=0 \wedge ok$ expand ok

$=$ $y=0 \wedge x'=x \wedge y'=y \wedge z'=z$ specialize, 1 is identity for $\times$

$\Rightarrow$ $y=0 \wedge z' = z \times 1$ $x^0=1$

$=$ $y=0 \wedge z' = z \times x^0$ context $y=0$ and specialize

$\Rightarrow$ $z' = z \times x^y$

Fast Exponentiation

Given rational variables x and z and natural variable y, write a program for $z' = x^y$ that runs fast without using exponentiation.

$z'=x^y \Leftarrow z:= 1.\ z' = z \times x^y$

$z' = z \times x^y \Leftarrow$ **if** $y=0$ **then** ok **else** $y>0 \Rightarrow z' = z \times x^y$ **fi**

$y>0 \Rightarrow z' = z \times x^y \Leftarrow z:= z \times x.\ y:= y-1.\ z' = z \times x^y$

Proof: $(y>0 \Rightarrow z' = z \times x^y \Leftarrow z:= z \times x.\ y:= y-1.\ z' = z \times x^y)$ portation

$= \quad z' = z \times x^y \Leftarrow y>0 \land (z:= z \times x.\ y:= y-1.\ z' = z \times x^y)$ Substitution Law twice

$= \quad z' = z \times x^y \Leftarrow y>0 \land z' = z \times x \times x^{y-1}$ Law of Exponents

$= \quad z' = z \times x^y \Leftarrow y>0 \land z' = z \times x^y$ specialize

$= \quad \top$

Fast Exponentiation

Given rational variables x and z and natural variable y, write a program for $z' = x^y$ that runs fast without using exponentiation.

$z'=x^y \quad \Leftarrow \quad z:= 1.\ z' = z\times x^y$

$z' = z\times x^y \quad \Leftarrow \quad$ **if** $y=0$ **then** ok **else** $y>0 \Rightarrow z' = z\times x^y$ **fi**

$y>0 \Rightarrow z' = z\times x^y \quad \Leftarrow \quad \cancel{z:= z\times x.\ y:= y-1.\ z' = z\times x^y}$

if $even\ y$ **then** $even\ y \wedge y>0 \Rightarrow z' = z\times x^y$ **else** $odd\ y \Rightarrow z' = z\times x^y$ **fi**

$even\ y \wedge y>0 \Rightarrow z' = z\times x^y \quad \Leftarrow \quad x:= x\times x.\ y:= y/2.\ z' = z\times x^y$

Proof: $(even\ y \wedge y>0 \Rightarrow z' = z\times x^y \quad \Leftarrow \quad x:= x\times x.\ y:= y/2.\ z' = z\times x^y)$ portation

$=$ $z' = z\times x^y \Leftarrow even\ y \wedge y>0 \wedge (x:= x\times x.\ y:= y/2.\ z' = z\times x^y)$ Substitution Law twice

$=$ $z' = z\times x^y \Leftarrow even\ y \wedge y>0 \wedge z' = z\times(x\times x)^{y/2}$ Law of Exponents

$=$ $z' = z\times x^y \Leftarrow even\ y \wedge y>0 \wedge z' = z\times x^y$ specialize

$=$ $\top$

Fast Exponentiation

Given rational variables x and z and natural variable y, write a program for $z' = x^y$ that runs fast without using exponentiation.

$z'=x^y \quad \Leftarrow \quad z:= 1.\ z' = z\times x^y$

$z' = z\times x^y \quad \Leftarrow \quad$ **if** $y=0$ **then** *ok* **else** $y>0 \Rightarrow z' = z\times x^y$ **fi**

$y>0 \Rightarrow z' = z\times x^y \quad \Leftarrow \quad \cancel{z:= z\times x.\ y:= y-1.\ z' = z\times x^y}$

if *even* y **then** *even* $y \wedge y>0 \Rightarrow z' = z\times x^y$ **else** *odd* $y \Rightarrow z' = z\times x^y$ **fi**

even $y \wedge y>0 \Rightarrow z' = z\times x^y \quad \Leftarrow \quad x:= x\times x.\ y:= y/2.\ z' = z\times x^y$

odd $y \Rightarrow z' = z\times x^y \quad \Leftarrow \quad z:= z\times x.\ y:= y-1.\ z' = z\times x^y$

Fast Exponentiation

Given rational variables x and z and natural variable y, write a program for $z' = x^y$ that runs fast without using exponentiation.

$z'=x^y \quad \Leftarrow \quad z:= 1.\ z' = z \times x^y$

$z' = z \times x^y \quad \Leftarrow \quad$ **if** $y=0$ **then** *ok* **else** $y>0 \Rightarrow z' = z \times x^y$ **fi**

$y>0 \Rightarrow z' = z \times x^y \quad \Leftarrow \quad \cancel{z:= z \times x.\ y:= y-1.\ z' = z \times x^y}$

if *even* y **then** *even* $y \wedge y>0 \Rightarrow z' = z \times x^y$ **else** *odd* $y \Rightarrow z' = z \times x^y$ **fi**

even $y \wedge y>0 \Rightarrow z' = z \times x^y \quad \Leftarrow \quad x:= x \times x.\ y:= y/2.\ \cancel{z' = z \times x^y}\ y>0 \Rightarrow z' = z \times x^y$

odd $y \Rightarrow z' = z \times x^y \quad \Leftarrow \quad z:= z \times x.\ y:= y-1.\ z' = z \times x^y$

Fast Exponentiation

Given rational variables x and z and natural variable y, write a program for $z' = x^y$ that runs fast without using exponentiation.

$z'=x^y \quad \Leftarrow \quad z:= 1.\ z' = z \times x^y$

$z' = z \times x^y \quad \Leftarrow \quad \textbf{if } y=0 \textbf{ then } ok \textbf{ else } y>0 \Rightarrow z' = z \times x^y \textbf{ fi}$

$y>0 \Rightarrow z' = z \times x^y \quad \Leftarrow \quad \cancel{z:= z \times x.\ y:= y-1.\ z' = z \times x^y}$

$\textbf{if } even\ y \textbf{ then } even\ y \wedge y>0 \Rightarrow z' = z \times x^y \textbf{ else } odd\ y \Rightarrow z' = z \times x^y \textbf{ fi}$

$even\ y \wedge y>0 \Rightarrow z' = z \times x^y \quad \Leftarrow \quad x:= x \times x.\ y:= y/2.\ \cancel{z' = z \times x^y}\ y>0 \Rightarrow z' = z \times x^y$

$odd\ y \Rightarrow z' = z \times x^y \quad \Leftarrow \quad z:= z \times x.\ y:= y-1.\ \cancel{z' = z \times x^y}\ even\ y \Rightarrow z' = z \times x^y$

$even\ y \Rightarrow z' = z \times x^y \quad \Leftarrow \quad \textbf{if } y = 0 \textbf{ then } ok \textbf{ else } even\ y \wedge y>0 \Rightarrow z' = z \times x^y \textbf{ fi}$

Fast Exponentiation

Given rational variables x and z and natural variable y, write a program for $z' = x^y$ that runs fast without using exponentiation.

$z'=x^y \quad \Leftarrow \quad z:= 1.\ \ z' = z\times x^y$

$z' = z\times x^y \quad \Leftarrow \quad \cancel{\textbf{if } y=0 \textbf{ then } ok \textbf{ else } y>0 \Rightarrow z' = z\times x^y \textbf{ fi}}$

$\qquad\qquad \textbf{if } even\ y \textbf{ then } even\ y \Rightarrow z' = z\times x^y \textbf{ else } odd\ y \Rightarrow z' = z\times x^y \textbf{ fi}$

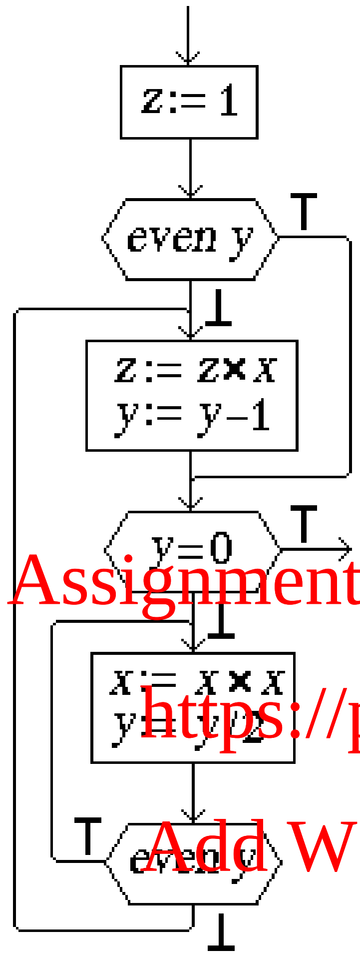
$y>0 \Rightarrow z' = z\times x^y \quad \Leftarrow \quad \cancel{z:= z\times x.\ \ y:= y-1.\ \ z' = z\times x^y}$

$\qquad\qquad \textbf{if } even\ y \textbf{ then } even\ y \land y>0 \Rightarrow z' = z\times x^y \textbf{ else } odd\ y \Rightarrow z' = z\times x^y \textbf{ fi}$

$even\ y \land y>0 \Rightarrow z' = z\times x^y \quad \Leftarrow \quad x:= x\times x.\ \ y:= y/2.\ \ \cancel{z' = z\times x^y}\ \ y>0 \Rightarrow z' = z\times x^y$

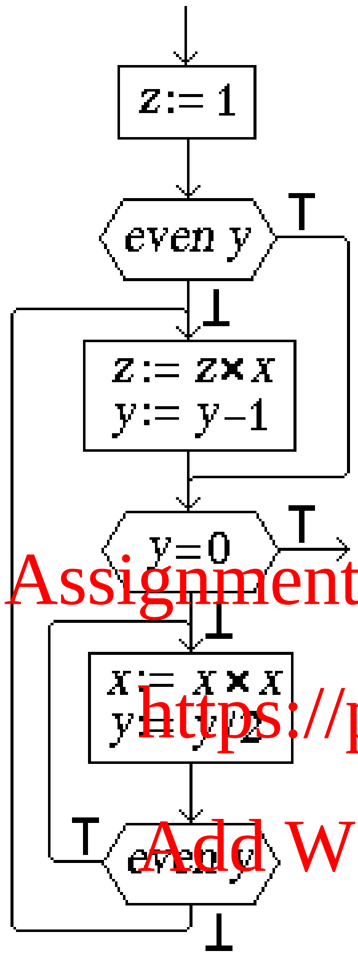
$odd\ y \Rightarrow z' = z\times x^y \quad \Leftarrow \quad z:= z\times x.\ \ y:= y-1.\ \ \cancel{z' = z\times x^y}\ \ even\ y \Rightarrow z' = z\times x^y$

$even\ y \Rightarrow z' = z\times x^y \quad \Leftarrow \quad \textbf{if } y = 0 \textbf{ then } ok \textbf{ else } even\ y \land y>0 \Rightarrow z' = z\times x^y \textbf{ fi}$



$$even\ y \land y>0 \Rightarrow z' = z\times x^y \quad \Leftarrow \quad x:= x\times x.\ \ y:= y/2.\ \ t:= t+1.\ \ y>0 \Rightarrow z' = z\times x^y$$

y	=	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
$t'-t$	=	0	0	1	1	2	2	2	2	3	3	3	3	3	3	3	3	4	4	4



$$even\ y \wedge y{>}0 \Rightarrow\ z' = z{\times}x^y \quad \Leftarrow \quad x{:=}\ x{\times}x.\ \ y{:=}\ y/2.\ \ t{:=}\ t{+}1.\ \ y{>}0 \Rightarrow\ z' = z{\times}x^y$$

if $y{=}0$ **then** $t'{=}t$ **else** $t' = t + floor\ (log\ y)$ **fi**

if $y{=}0$ **then** $t'{=}t$ **else** $t' \le t + log\ y$ **fi**

Fibonacci Numbers

$fib\ 0 = 0$

$fib\ 1 = 1$

$fib\ (n+2) = fib\ n + fib\ (n+1)$

$fib = 0 \to 0 \mid 1 \to 1 \mid \langle n: nat+2 \to fib(n-2) + fib(n-1) \rangle$

$fib = \langle n: nat \to \textbf{if } n<2 \textbf{ then } n \textbf{ else } fib(n-2) + fib(n-1) \textbf{ fi} \rangle$

Fibonacci Numbers

$$x' = fib\ n \ \Leftarrow\ x' = fib\ n \ \wedge\ y' = fib\ (n+1) \ = \ P$$

$P \ \Leftarrow$ **if** $n{=}0$ **then** $x{:=}\ 0.\ \ y{:=}\ 1$ **else** $n{:=}\ n{-}1.\ \ P.\ \ x'{=}y \ \wedge\ y'{=}x{+}y$ **fi**

we have these $x\ \ y$

$\downarrow\ \downarrow$

$\ldots\ f\ f\ f\ f\ f\ f\ f\ f\ f\ f\ f\ f\ f\ f\ f\ f\ \ldots$

$\uparrow\ \uparrow$

we want these $x'\ y'$

Fibonacci Numbers

$x' = fib\ n \ \Leftarrow\ x' = fib\ n \ \wedge\ y' = fib\ (n+1) \ = \ P$

$P \ \Leftarrow\ $ **if** $n=0$ **then** $x:= 0.\ \ y:= 1$ **else** $n:= n-1.\ \ P.\ \ x'=y \ \wedge\ y'=x+y$ **fi**

$x'=y \ \wedge\ y'=x+y \ \Leftarrow\ n:= x.\ \ x:= y.\ \ y:= n+y$

$t'=t+n \ \Leftarrow\ $ **if** $n=0$ **then** $x:= 0.\ \ y:= 1$ **else** $n:= n-1.\ \ t:= t+1.\ \ t'=t+n.\ \ t'=t$ **fi**

$t'=t \ \Leftarrow\ n:= x.\ \ x:= y.\ \ y:= n+y$

Fibonacci Numbers

$$fib(2\times k+1) \;=\; (fib\ k)^2 + (fib(k+1))^2$$

$$fib(2\times k+2) \;=\; 2 \times fib\ k \times fib(k+1) \;+\; (fib(k+1))^2$$

$P \;\Leftarrow\;$ **if** n=0 **then** x:= 0. y:= 1

else if *even n* **then** $even\ n \wedge n>0 \Rightarrow P$ **fi**

else $odd\ n \Rightarrow P$ **fi**

$$odd\ n \Rightarrow P \;\Leftarrow\; n:= (n-1)/2.\ P.\ x' = x^2 + y^2 \;\wedge\; y' = 2\times x\times y + y^2$$

$$even\ n \wedge n>0 \Rightarrow P \;\Leftarrow\; n:= n/2 - 1.\ P.\ x' = 2\times x\times y + y^2 \;\wedge\; y' = x^2 + y^2 + x'$$

$$x' = x^2 + y^2 \;\wedge\; y' = 2\times x\times y + y^2 \;\Leftarrow\; n:= x.\ x:= x^2 + y^2.\ y:= 2\times n\times y + y^2$$

$$x' = 2\times x\times y + y^2 \wedge\; y' = x^2 + y^2 + x' \;\Leftarrow\; n:= x.\ x:= 2\times x\times y + y^2.\ y:= n^2 + y^2 + x$$

Fibonacci Numbers

$T \;=\; t' \le t + log\ (n+1)$

$T \;\Leftarrow\;$ **if** $n=0$ **then** $x:= 0.\ \ y:= 1$

else if $even\ n$ **then** $even\ n \wedge n>0 \Rightarrow T$ **fi**

else $odd\ n \Rightarrow T$ **fi**

$odd\ n \Rightarrow T \;\Leftarrow\; n:= (n-1)/2.\ \ t:= t+1.\ \ T.\ \ t'=t$

$even\ n \wedge n>0 \Rightarrow T \;\Leftarrow\; n:= n/2 - 1.\ \ t:= t+1.\ \ T.\ \ t'=t$

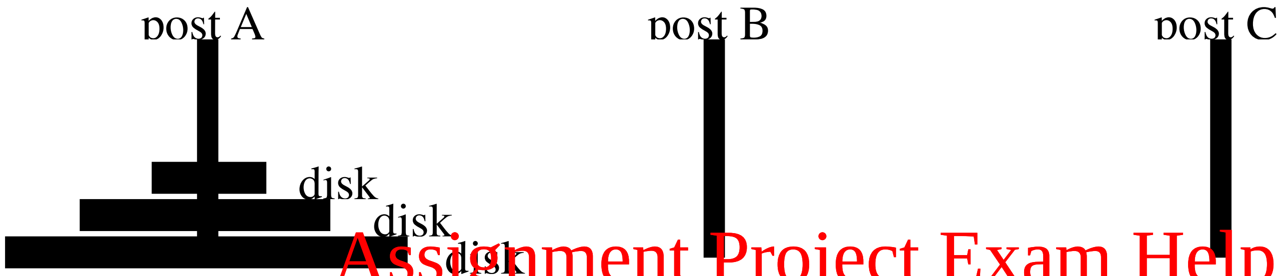
$t'=t \;\Leftarrow\; n:= x.\ \ x:= x^2 + y^2.\ \ y:= 2\times n\times y + y^2$

$t'=t \;\Leftarrow\; n:= x.\ \ x:= 2\times x\times y + y^2.\ \ y:= n^2 + y^2 + x$

Fibonacci Numbers

```
void P (void)

{       if (n==0) {x = 0;  y = 1;}

        else if (n%2==0) {n = n / 2 – 1;  P();  n = x;  x = 2*x*y + y*y;  y = n*n + y*y + x;}

        else {n = (n–1) / 2;  P();  n = x;  x = x*x + y*y;  y = 2*n*y + y*y;}

}
```

Towers of Hanoi



Towers of Hanoi

MovePile from to using $\Leftarrow$

if n=0 **then** *ok*

else n:= n–1.

MovePile from using to.

MoveDisk from to.

MovePile using to from.

n:= n+1 **fi**

Towers of Hanoi — time

$t := t + 2^n - 1 \Leftarrow$

if $n=0$ **then** ok

else $n := n-1$.

$t := t + 2^n - 1$.

$t := t+1$.

$t := t + 2^n - 1$.

$n := n+1$ **fi**

Towers of Hanoi — space

$s'{=}s \Leftarrow$

if $n{=}0$ **then** ok

else $n{:=}\ n{-}1.$

$s{:=}\ s{+}1.\ \ s'{=}s.\ \ s{:=}\ s{-}1.$

$ok.$

$s{:=}\ s{+}1.\ \ s'{=}s.\ \ s{:=}\ s{-}1.$

$n{:=}\ n{+}1$ **fi**

Towers of Hanoi — maximum space

$m \geq s \Rightarrow (m := max\ m\ (s+n)) \Leftarrow$

if $n=0$ **then** ok

else $n := n-1.$

$s := s+1.\ \ m := max\ m\ s.\ \ m \geq s \Rightarrow (m := max\ m\ (s+n)).\ \ s := s-1.$

$ok.$

$s := s+1.\ \ m := max\ m\ s.\ \ m \geq s \Rightarrow (m := max\ m\ (s+n)).\ \ s := s-1.$

$n := n+1$ **fi**

Towers of Hanoi — average space

$p := p + s\times(2^n - 1) + (n-2)\times 2^n + 2 \;\Leftarrow$

if $n=0$ **then** ok

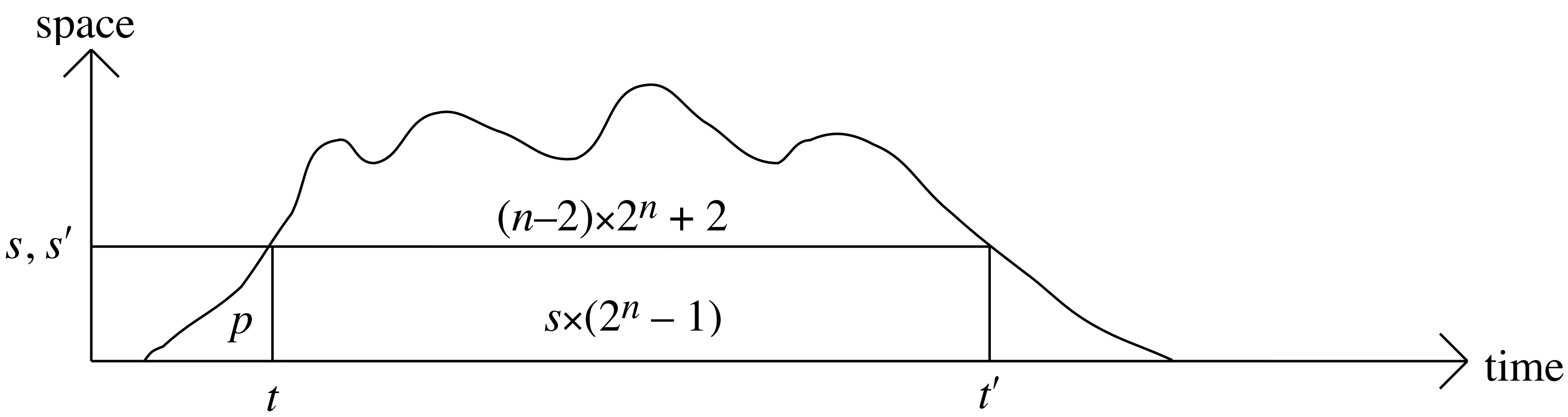
else $n := n-1$.

$s := s+1.\;\; p := p + s\times(2^n - 1) + (n-2)\times 2^n + 2.\;\; s := s-1.$

$p := p+s.$

$s := s+1.\;\; p := p + s\times(2^n - 1) + (n-2)\times 2^n + 2.\;\; s := s-1.$

$n := n+1$ **fi**



Towers of Hanoi — average space

$p := p + s\times(2^n - 1) + (n-2)\times 2^n + 2 \ \Leftarrow$

if $n=0$ **then** ok

else $n := n-1$.

$s := s+1$. $p := p + s\times(2^n - 1) + (n-2)\times 2^n + 2$. $s := s-1$.

$p := p+s$.

$s := s+1$. $p := p + s\times(2^n - 1) + (n-2)\times 2^n + 2$. $s := s-1$.

$n := n+1$ **fi**

average space $= ((n-2)\times 2^n + 2) / (2^n - 1)$

$= n + n/(2^n - 1) - 2$

Easier: $p' \le p + (s+n)\times(2^n-1)$

average space $\le n$

Towers of Hanoi

$MovePile \Leftarrow$

if $n=0$ **then** ok

else $n:= n-1.$

$s:= s+1.\ \ m:= max\ m\ s.\ \ MovePile.\ \ s:= s-1.$

$t:= t+1.\ \ p:= p+s.\ \ ok.$

$s:= s+1.\ \ m:= max\ m\ s.\ \ MovePile.\ \ s:= s-1.$

$n:= n+1$ **fi**

$$\begin{aligned} MovePile \ = \quad & n'=n \\ \wedge \quad & t' = t + 2^n - 1 \\ \wedge \quad & s'=s \\ \wedge \quad & (m \ge s \Rightarrow m' = max\ m\ (s+n)) \\ \wedge \quad & p' = p + s \times (2^n - 1) + (n-2) \times 2^n + 2 \end{aligned}$$