Refinement

Specification P (the problem) is **refined** by specification S (the solution)

if and only if P is satisfied whenever S is satisfied.

Refinement

Specification P (the problem) is **refined** by specification S (the solution)

if and only if P is satisfied whenever S is satisfied.

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

Refinement

Specification P (the problem) is **refined** by specification S (the solution)

if and only if P is satisfied whenever S is satisfied.

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

$$x' > x \quad \Leftarrow \quad x' = x+1 \wedge y' = y$$

Refinement

Specification P (the problem) is **refined** by specification S (the solution)

if and only if P is satisfied whenever S is satisfied.

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

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

Refinement

Specification P (the problem) is **refined** by specification S (the solution)

if and only if P is satisfied whenever S is satisfied.

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

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

$$x' \leq x \quad \Leftarrow \quad \textbf{if } x=0 \textbf{ then } x'=x \textbf{ else } x'<x \textbf{ fi}$$

Refinement

Specification P (the problem) is **refined** by specification S (the solution)

if and only if P is satisfied whenever S is satisfied.

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

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

$$x' \leq x \quad \Leftarrow \quad \textbf{if } x=0 \textbf{ then } x'=x \textbf{ else } x'<x \textbf{ fi} \quad = \quad x=0 \wedge x'=x \ \vee \ x \neq 0 \wedge x'<x$$

Refinement

Specification P (the problem) is **refined** by specification S (the solution)

if and only if P is satisfied whenever S is satisfied.

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

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

$$x' \leq x \quad \Leftarrow \quad \textbf{if } x=0 \textbf{ then } x'=x \textbf{ else } x'<x \textbf{ fi} \quad = \quad x=0 \wedge x'=x \ \vee \ x \neq 0 \wedge x'<x$$

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

Refinement

Specification P (the problem) is **refined** by specification S (the solution)

if and only if P is satisfied whenever S is satisfied.

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

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

$$x' \leq x \quad \Leftarrow \quad \textbf{if } x=0 \textbf{ then } x'=x \textbf{ else } x'<x \textbf{ fi} \quad = \quad x=0 \wedge x'=x \ \vee \ x \neq 0 \wedge x'<x$$

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

$$= \quad y:= x+1.\ x'=y+1 \wedge y'=y$$

Refinement

Specification P (the problem) is **refined** by specification S (the solution)

if and only if P is satisfied whenever S is satisfied.

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

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

$$x' \leq x \quad \Leftarrow \quad \textbf{if } x=0 \textbf{ then } x'=x \textbf{ else } x'<x \textbf{ fi} \quad = \quad x=0 \wedge x'=x \ \vee \ x \neq 0 \wedge x'<x$$

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

$$= \quad y:= x+1.\ x'=y+1 \wedge y'=y$$

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

A **program** is an implemented specification.

A **program** is an implemented specification.

ok

A **program** is an implemented specification.

ok

$x := e$

A **program** is an implemented specification.

ok	binary values, numbers, characters
$x := e$	bunches, sets, strings, lists
	NOT functions, quantifiers

A **program** is an implemented specification.

ok	binary values, numbers, characters
$x := e$	bunches, sets, strings, lists
if b **then** P **else** Q **fi**	NOT functions, quantifiers

A **program** is an implemented specification.

ok	binary values, numbers, characters
$x := e$	bunches, sets, strings, lists
if b **then** P **else** Q **fi**	NOT functions, quantifiers
$P \,.\, Q$	

A **program** is an implemented specification.

ok	binary values, numbers, characters
$x := e$	bunches, sets, strings, lists
if b **then** P **else** Q **fi**	NOT functions, quantifiers
$P \,.\, Q$	

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

Recursion is allowed.

A **program** is an implemented specification.

ok	binary values, numbers, characters
$x := e$	bunches, sets, strings, lists
if b **then** P **else** Q **fi**	NOT functions, quantifiers
$P \,.\, Q$	

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

Recursion is allowed.

$x \geq 0 \Rightarrow x' = 0$

A **program** is an implemented specification.

ok	binary values, numbers, characters
$x := e$	bunches, sets, strings, lists
if b **then** P **else** Q **fi**	NOT functions, quantifiers
$P \,.\, Q$	

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

Recursion is allowed.

$x \geq 0 \Rightarrow x' = 0 \quad \Leftarrow \quad$ **if** $x = 0$ **then** ok **else** $x := x - 1 \,.\; x \geq 0 \Rightarrow x' = 0$ **fi**

A **program** is an implemented specification.

ok	binary values, numbers, characters
$x := e$	bunches, sets, strings, lists
if b **then** P **else** Q **fi**	NOT functions, quantifiers
$P . Q$	

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

Recursion is allowed.

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

A **program** is an implemented specification.

ok	binary values, numbers, characters
$x := e$	bunches, sets, strings, lists
if b **then** P **else** Q **fi**	NOT functions, quantifiers
$P . Q$	

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

Recursion is allowed.

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

A **program** is an implemented specification.

ok	binary values, numbers, characters
$x := e$	bunches, sets, strings, lists
if b **then** P **else** Q **fi**	NOT functions, quantifiers
$P . Q$	

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

Recursion is allowed.

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

A **program** is an implemented specification.

ok	binary values, numbers, characters
$x := e$	bunches, sets, strings, lists
if b **then** P **else** Q **fi**	NOT functions, quantifiers
$P . Q$	

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

Recursion is allowed.

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

A **program** is an implemented specification.

ok	binary values, numbers, characters
$x := e$	bunches, sets, strings, lists
if b **then** P **else** Q **fi**	NOT functions, quantifiers
$P \,.\, Q$	

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

Recursion is allowed.

$x \geq 0 \Rightarrow x' = 0 \quad \Leftarrow \quad$ **if** $x = 0$ **then** ok **else** $x := x - 1 \,.\, x \geq 0 \Rightarrow x' = 0$ **fi**

A **program** is an implemented specification.

ok	binary values, numbers, characters
$x := e$	bunches, sets, strings, lists
if b **then** P **else** Q **fi**	NOT functions, quantifiers
$P . Q$	

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

Recursion is allowed.

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

A **program** is an implemented specification.

ok	binary values, numbers, characters
$x := e$	bunches, sets, strings, lists
if b **then** P **else** Q **fi**	NOT functions, quantifiers
$P . Q$	

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

Recursion is allowed.

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

refinement by steps

If $A \Leftarrow$ **if** b **then** C **else** D **fi**

refinement by steps

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

refinement by steps

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

then $A \Leftarrow$ **if** b **then** E **else** F **fi** .

refinement by steps

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

then $A \Leftarrow$ **if** b **then** E **else** F **fi** .

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

refinement by steps

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

then $A \Leftarrow$ **if** b **then** E **else** F **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 steps

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

then $A \Leftarrow$ **if** b **then** E **else** F **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 $A \Leftarrow$ **if** b **then** C **else** D **fi** and $E \Leftarrow$ **if** b **then** F **else** G **fi** ,

refinement by steps

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

then $A \Leftarrow$ **if** b **then** E **else** F **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 $A \Leftarrow$ **if** b **then** C **else** D **fi** and $E \Leftarrow$ **if** b **then** F **else** G **fi** ,

then $A \wedge E \Leftarrow$ **if** b **then** $C \wedge F$ **else** $D \wedge G$ **fi** .

refinement by steps

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

then $A \Leftarrow$ **if** b **then** E **else** F **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 $A \Leftarrow$ **if** b **then** C **else** D **fi** and $E \Leftarrow$ **if** b **then** F **else** G **fi** ,

then $A \wedge E \Leftarrow$ **if** b **then** $C \wedge F$ **else** $D \wedge G$ **fi** .

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

refinement by steps

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

then $A \Leftarrow$ **if** b **then** E **else** F **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 $A \Leftarrow$ **if** b **then** C **else** D **fi** and $E \Leftarrow$ **if** b **then** F **else** G **fi** ,

then $A \wedge E \Leftarrow$ **if** b **then** $C \wedge F$ **else** $D \wedge G$ **fi** .

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

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

refinement by steps

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

then $A \Leftarrow$ **if** b **then** E **else** F **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 $A \Leftarrow$ **if** b **then** C **else** D **fi** and $E \Leftarrow$ **if** b **then** F **else** G **fi** ,

then $A \wedge E \Leftarrow$ **if** b **then** $C \wedge F$ **else** $D \wedge G$ **fi** .

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

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

refinement by cases

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

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

List Summation

List of numbers L ; number variable s .

$s' = \Sigma L$

List Summation

List of numbers L ; number variable s .

$s' = \Sigma L \quad \Longleftarrow \quad s := 0.$

List Summation

List of numbers L ; number variable s .

$s' = \Sigma L \quad \Leftarrow \quad s := 0.\ n := 0.$

List Summation

List of numbers L ; number variable s .

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

List Summation

List of numbers L ; number variable s .

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

List Summation

List of numbers L ; number variable s .

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

List Summation

List of numbers L ; number variable s .

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

List Summation

List of numbers L ; number variable s .

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

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

List Summation

List of numbers L ; number variable s .

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

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

if $n = \#L$

List Summation

List of numbers L ; number variable s .

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

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

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

List Summation

List of numbers L ; number variable s .

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

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

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

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

List Summation

List of numbers L ; number variable s .

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

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

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

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

Case Creation Law: a = **if** b **then** $b \Rightarrow a$ **else** $\neg b \Rightarrow a$ **fi**

List Summation

List of numbers L ; number variable s .

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

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

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

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

List Summation

List of numbers L ; number variable s .

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

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

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

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

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

List Summation

List of numbers L ; number variable s .

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

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

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

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

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

List Summation

List of numbers L ; number variable s .

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

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

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

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

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

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

List Summation

List of numbers L ; number variable s .

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

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

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

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

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

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

List Summation

List of numbers L ; number variable s .

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

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

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

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

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

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

List Summation

List of numbers L ; number variable s .

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

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

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

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

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

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

List Summation

List of numbers L ; number variable s .

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

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

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

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

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

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

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

List Summation

List of numbers L ; number variable s .

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

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

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

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

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

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

$s' = s + \Sigma\, L\,[n{+}1;..\#L]$

List Summation

List of numbers L ; number variable s .

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

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

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

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

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

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

$s' = s + L\,n + \Sigma\, L\,[n{+}1;..\#L]$

List Summation

List of numbers L ; number variable s .

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

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

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

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

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

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

$s' = s + L\,n + \Sigma\, L\,[n{+}1;..\#L]$

List Summation

List of numbers L ; number variable s .

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

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

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

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

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

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

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

List Summation

List of numbers L ; number variable s .

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

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

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

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

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

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

compilation

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

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

$C \Leftarrow ok$

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

compilation

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

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

$C \Leftarrow ok$

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

Refinement by Steps = in-line macro-expansion

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

compilation

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

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

$C \Leftarrow ok$

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

Refinement by Steps = in-line macro-expansion

$B \Leftarrow$ **if** $n{=}\#L$ **then** ok **else** $s{:=}s + L\,n.\ n{:=}n{+}1.\ B$ **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( );}}
```

compilation

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

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

$C \Leftarrow ok$

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

Refinement by Steps = in-line macro-expansion

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

translation

```
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 $y' = 2^x$.

Binary Exponentiation

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

$y'=2^x$ $\Leftarrow$

Binary Exponentiation

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

$y'=2^x \Leftarrow$ **if** $x=0$

Binary Exponentiation

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

$y'=2^x \Leftarrow$ **if** $x=0$ **then** $x=0 \Rightarrow y'=2^x$

Binary Exponentiation

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

$y'=2^x \Leftarrow$ **if** $x=0$ **then** $x=0 \Rightarrow y'=2^x$ **else** $x>0 \Rightarrow y'=2^x$ **fi**

Binary Exponentiation

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

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

$x=0 \Rightarrow y'=2^x \quad \Leftarrow$

Binary Exponentiation

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

$y'=2^x \Leftarrow$ **if** $x=0$ **then** $x=0 \Rightarrow y'=2^x$ **else** $x>0 \Rightarrow y'=2^x$ **fi**

$x=0 \Rightarrow y'=2^x \Leftarrow y := 1$

Binary Exponentiation

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

$y'=2^x \Leftarrow$ **if** $x=0$ **then** $x=0 \Rightarrow y'=2^x$ **else** $x>0 \Rightarrow y'=2^x$ **fi**

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

Binary Exponentiation

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

$y'=2^x \Leftarrow$ **if** $x=0$ **then** $x=0 \Rightarrow y'=2^x$ **else** $x>0 \Rightarrow y'=2^x$ **fi**

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

$x>0 \Rightarrow y'=2^x \Leftarrow$

Binary Exponentiation

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

$y'=2^x \Leftarrow$ **if** $x=0$ **then** $x=0 \Rightarrow y'=2^x$ **else** $x>0 \Rightarrow y'=2^x$ **fi**

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

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

Binary Exponentiation

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

$y'=2^x \Leftarrow$ **if** $x=0$ **then** $x=0 \Rightarrow y'=2^x$ **else** $x>0 \Rightarrow y'=2^x$ **fi**

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

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

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

Binary Exponentiation

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

$y'=2^x \Leftarrow$ **if** $x=0$ **then** $x=0 \Rightarrow y'=2^x$ **else** $x>0 \Rightarrow y'=2^x$ **fi**

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

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

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

$y'=2\times y \Leftarrow$

Binary Exponentiation

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

$y'=2^x \Leftarrow$ **if** $x=0$ **then** $x=0 \Rightarrow y'=2^x$ **else** $x>0 \Rightarrow y'=2^x$ **fi**

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

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

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

$y'=2\times y \Leftarrow y:= 2\times y$

Binary Exponentiation

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

$y'=2^x \Leftarrow$ **if** $x=0$ **then** $x=0 \Rightarrow y'=2^x$ **else** $x>0 \Rightarrow y'=2^x$ **fi**

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

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

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

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

Binary Exponentiation

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

$y'=2^x \Leftarrow$ **if** $x=0$ **then** $x=0 \Rightarrow y'=2^x$ **else** $x>0 \Rightarrow y'=2^x$ **fi**

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

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

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

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

$x'=x-1 \Leftarrow$

Binary Exponentiation

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

$y'=2^x \Leftarrow$ **if** $x=0$ **then** $x=0 \Rightarrow y'=2^x$ **else** $x>0 \Rightarrow y'=2^x$ **fi**

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

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

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

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

$x'=x-1 \Leftarrow x:= x-1$

Binary Exponentiation

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

$y'=2^x \Leftarrow$ **if** $x=0$ **then** $x=0 \Rightarrow y'=2^x$ **else** $x>0 \Rightarrow y'=2^x$ **fi**

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

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

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

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

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

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**

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

$C \Leftarrow D.\ E$

$D \Leftarrow F.\ A$

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

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

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** B **else** C **fi**

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** C **fi**

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** $D.\ E$ **fi**

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** $F.\ A.\ E$ **fi**

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** $F.\ A.\ y := 2 \times y.\ x := 5$ **fi**

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**

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;
```

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;}}
```

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);
```

Binary Exponentiation

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

$y'=2^x \Leftarrow$ **if** $x=0$ **then** $x=0 \Rightarrow y'=2^x$ **else** $x>0 \Rightarrow y'=2^x$ **fi**

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

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

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

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

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