

PROGRAM THEORY

state space (memory) [*int*; (0,..20); *char*; *rat*]

state (memory contents) [-2; 15; `A; 3.14]

prestate (initial state) σ

poststate (final state) σ'

$$\sigma = [\sigma_0; \sigma_1; \sigma_2; \sigma_3] = [i; n; c; x]$$

addresses vs. variables

state variables

initial values i, n, c, x

final values i', n', c', x'

For now: prestate, poststate

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

SPECIFICATIONS

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 of computer behavior:

a boolean expression in the initial values x, y, \dots

and final values x', y', \dots of some state variables

We provide initial values. A computation satisfies a specification by computing satisfactory final values.

Specification S is unsatisfiable for prestate σ :

$$\wp(\xi\sigma' \cdot S) = 0$$

Specification S is satisfiable for prestate σ :

$$\wp(\xi\sigma' \cdot S) > 0$$

Specification S is deterministic for prestate σ :

$$\wp(\xi\sigma' \cdot S) \leq 1$$

Specification S is nondeterministic for prestate σ :

$$\wp(\xi\sigma' \cdot S) > 1$$

Specification S is satisfiable for prestate σ :

$$\exists\sigma' \cdot S$$

Specification S is implementable:

$$\forall\sigma \cdot \exists\sigma' \cdot S$$

Examples

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

$$x' > x$$

\top

\perp

$$x \geq 0 \wedge y' = 0$$

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

$$\begin{aligned} ok &= \sigma' = \sigma \\ &= x' = x \wedge y' = y \wedge \dots \end{aligned}$$

$$\begin{aligned} x := e &= (\text{substitute } e \text{ for } x \text{ in } ok) \\ &= x' = e \wedge y' = y \wedge \dots \end{aligned}$$

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

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

Dependent Composition

$S. R$

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

In integer variable x

$$x'=x \vee x'=x+1 . x'=x \vee x'=x+1$$

$$= \exists x'' . (x''=x \vee x''=x+1) \wedge (x'=x'' \vee x'=x''+1)$$

distribute \wedge over \vee

$$= \exists x'' . \quad x''=x \wedge x'=x'' \vee x''=x+1 \wedge x'=x''$$

$$\vee x''=x \wedge x'=x''+1 \vee x''=x+1 \wedge x'=x''+1$$

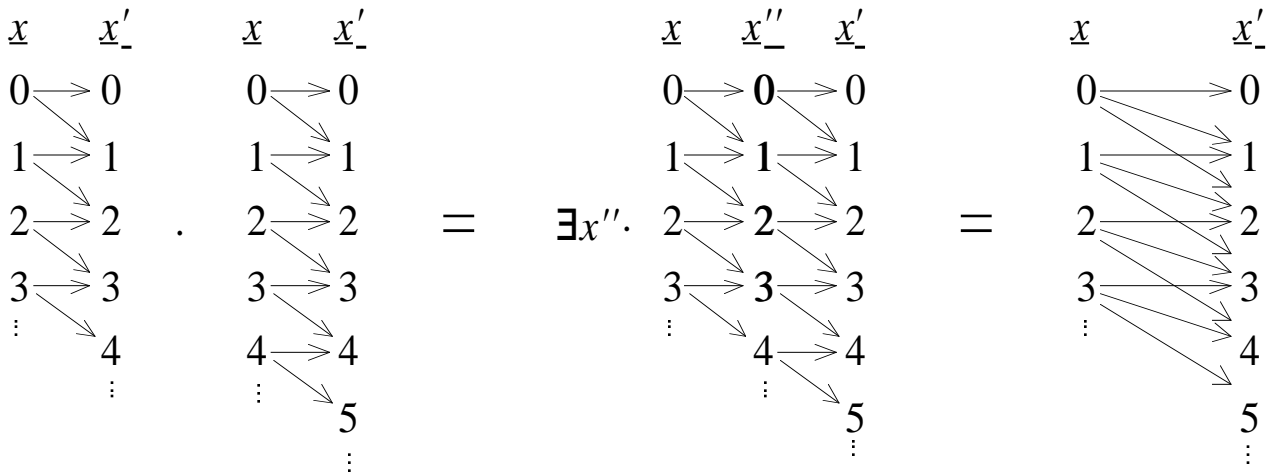
distribute \exists over \vee

$$= (\exists x'' . x''=x \wedge x'=x'') \vee (\exists x'' . x''=x+1 \wedge x'=x'')$$

$$\vee (\exists x'' . x''=x \wedge x'=x''+1) \vee (\exists x'' . x''=x+1 \wedge x'=x''+1)$$

One-Point, 4 times

$$= x'=x \vee x'=x+1 \vee x'=x+2$$



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

$= \exists x'', y'': \text{int}. x'' = 3 \wedge y'' = y \wedge x' = x'' \wedge y' = x'' + y''$

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
if b then P else $P = P$	Idempotent Law
if b then P else $Q =$ if $\neg b$ then Q else P	Case Reversal Law
$P =$ if b then $b \Rightarrow P$ else $\neg b \Rightarrow P$	Case Creation Law
if b then S else $R = b \wedge S \vee \neg b \wedge R$	Case Analysis Law
if b then S else $R = (b \Rightarrow S) \wedge (\neg b \Rightarrow R)$	Case Analysis Law
$P \vee Q. R \vee S = (P. R) \vee (P. S) \vee (Q. R) \vee (Q. S)$	
(if b then P else $Q) \wedge R =$ if b then $P \wedge R$ else $Q \wedge R$	
(if b then P else $Q). R =$ if b then $(P. R)$ else $(Q. R)$	
$x :=$ if b then e else $f =$ if b then $x := e$ else $x := f$	Functional-Imperative Law
$x := e. P =$ (for x substitute e in P)	Substitution Law

Substitution Law Examples

$$x := y+1. y' > x' = y' > x'$$

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

$$x := y+1. y' = 2 \times x = y' = 2 \times (y+1)$$

$$\begin{aligned} x := 1. x \geq 1 \Rightarrow \exists x. y' = 2 \times x &= 1 \geq 1 \Rightarrow \exists x. y' = 2 \times x \\ &= \text{even } y' \end{aligned}$$

$$x := y. x \geq 1 \Rightarrow \exists y. y' = x \times y = y \geq 1 \Rightarrow \exists k. y' = y \times k$$

$$x := 1. ok = x := 1. x' = x \wedge y' = y = x' = 1 \wedge y' = y$$

$$x := 1. y := 2 = x := 1. x' = x \wedge y' = 2 = x' = 1 \wedge y' = 2$$

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

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

REFINEMENT

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

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

Examples:

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

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

$$\begin{aligned} x' \leq x &\Leftarrow \mathbf{if } x=0 \mathbf{ then } x'=x \mathbf{ else } x' < x \\ &= x=0 \wedge x'=x \vee x \neq 0 \wedge x' < x \end{aligned}$$

$$\begin{aligned} x' > y' > x &\Leftarrow y := x + 1. x := y + 1 \\ &= y := x + 1. x' = y + 1 \wedge y' = y \\ &= x' = x + 2 \wedge y' = x + 1 \end{aligned}$$

CONDITIONS

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)

The exact precondition for P to be refined by S is $\forall \sigma'. P \Leftarrow S$

The exact postcondition for P to be refined by S is $\forall \sigma. P \Leftarrow S$

sufficient \Rightarrow exact \Rightarrow necessary

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

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

$$= \forall x'. x' > 5 \Leftarrow x' = x+1$$

One-Point Law

$$= x+1 > 5$$

$$= x > 4$$

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

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

$$= \forall x. x > 4 \Leftarrow (x := x+1)$$

$$= \forall x. x > 4 \Leftarrow x' = x+1$$

One-Point Law

$$= x'-1 > 4$$

$$= x' > 5$$

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

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

$$\begin{aligned}
 P &= \text{“make } x \text{ be farther from } 0 \text{”} \\
 &= \text{abs } x' > \text{abs } x
 \end{aligned}$$

(the exact precondition for P to be refined by $x := x^2$)

$$\begin{aligned}
 &= \forall x'. \text{abs } x' > \text{abs } x \iff x' = x^2 && \text{One-Point Law} \\
 &= \text{abs } (x^2) > \text{abs } x && \text{properties of } \text{abs} \text{ and } 2 \\
 &= x \neq -1 \wedge x \neq 0 \wedge x \neq 1
 \end{aligned}$$

(the exact postcondition for P to be refined by $x := x^2$)

$$\begin{aligned}
 &= \forall x. \text{abs } x' > \text{abs } x \iff x' = x^2 && \text{several omitted steps} \\
 &= x' \neq 0 \wedge x' \neq 1
 \end{aligned}$$

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 .

PROGRAMS

A program is an implemented specification.

ok

$x := e$

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

$P.Q$

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

Recursion is allowed.

$x \geq 0 \Rightarrow x' = 0 \iff \mathbf{if} \ x = 0 \ \mathbf{then} \ ok \ \mathbf{else} \ (x := x - 1. \ x \geq 0 \Rightarrow x' = 0)$

Refinement Laws

Refinement by Steps

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

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 $A \Leftarrow \mathbf{if\ } b \mathbf{\ then\ } C \mathbf{\ else\ } D$
 and $E \Leftarrow \mathbf{if\ } b \mathbf{\ then\ } F \mathbf{\ else\ } G$,
 then $A \wedge E \Leftarrow \mathbf{if\ } b \mathbf{\ then\ } C \wedge F \mathbf{\ else\ } D \wedge G$.

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

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

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

List Summation

List of numbers L ; number variable s .

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

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

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

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

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

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

$$s := s + Ln. \ n := n + 1. \ s' = s + \Sigma L [n;..#L]$$

Compilation

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

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

$$C \Leftarrow \mathit{ok}$$

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

Refinement by Steps = in-line macro-expansion

$$B \Leftarrow \mathbf{if} \ n = \#L \ \mathbf{then} \ \mathit{ok} \ \mathbf{else} \ (s := s + Ln. \ n := n + 1. \ B)$$

Translation

```
void B(void)
```

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

```
s = 0; n = 0; 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 $y' = 2^x$.

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

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

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

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

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

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

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

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

$C \Leftarrow D. \ E$

$D \Leftarrow F. \ A$

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

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

$A \Leftarrow \text{if } x=0 \text{ then } (y:= 1. \ x:= 3)$

$\text{else } (x:= x-1. \ y:= 7. \ A. \ y:= 2 \times y. \ x:= 5)$

```
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

$$\sigma = [t ; x ; y ; \dots]$$

state = [time variable; memory variables]

t is the time at which execution starts

t' is the time at which execution ends

$t, t': xnat$ or $xint$ or $xrat$ or $xreal$

Specification S is implementable if and only if

$$\forall \sigma. \exists \sigma'. S \wedge t' \geq t$$

Real Time

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

$t := t + (\text{the time to evaluate } b \text{ and branch}) . \mathbf{if } b \mathbf{ then } P \mathbf{ else } Q$

$t := t + (\text{the time for the call and return}) . P$

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

$$t' \leq t + f\sigma$$

$$t' \geq t + f\sigma$$

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

is a theorem when

$$P = x' = 0$$

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

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

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

Recursive Time

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

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

is a theorem when

$P = x'=0$

$P = \text{if } x \geq 0 \text{ then } t'=t+x \text{ else } t'=\infty$

$P = \text{if } x \geq 0 \text{ then } x'=0 \wedge t'=t+x \text{ else } t'=\infty$

$P = x'=0 \wedge \text{if } x \geq 0 \text{ then } t'=t+x \text{ else } t'=\infty$

Recursion can be direct or indirect.

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

Prove

$$R \Leftarrow \mathbf{if } x=1 \mathbf{ then } ok \mathbf{ else } (x := \text{div } x \ 2. \ t := t+1. \ R)$$

where

$$\begin{aligned} R &= x'=1 \wedge \mathbf{if } x \geq 1 \mathbf{ then } t' \leq t + \log x \mathbf{ else } t' = \infty \\ &= x'=1 \wedge (x \geq 1 \Rightarrow t' \leq t + \log x) \wedge (x < 1 \Rightarrow t' = \infty) \end{aligned}$$

use Refinement by Parts; prove:

$$\begin{aligned} x'=1 &\Leftarrow \mathbf{if } x=1 \mathbf{ then } ok \\ &\quad \mathbf{else } (x := \text{div } x \ 2. \ t := t+1. \ x'=1) \end{aligned}$$

$$\begin{aligned} x \geq 1 \Rightarrow t' \leq t + \log x &\Leftarrow \\ &\quad \mathbf{if } x=1 \mathbf{ then } ok \\ &\quad \mathbf{else } (x := \text{div } x \ 2. \ t := t+1. \ (x \geq 1) \Rightarrow (t' \leq t + \log x)) \end{aligned}$$

$$\begin{aligned} x < 1 \Rightarrow t' = \infty &\Leftarrow \\ &\quad \mathbf{if } x=1 \mathbf{ then } ok \\ &\quad \mathbf{else } (x := \text{div } x \ 2. \ t := t+1. \ (x < 1) \Rightarrow (t' = \infty)) \end{aligned}$$

use Refinement by Cases; prove:

$$x'=1 \iff x=1 \wedge x'=x \wedge t'=t$$

$$x'=1 \iff x \neq 1 \wedge x'=1$$

$$x \geq 1 \Rightarrow t' \leq t + \log x \iff x=1 \wedge x'=x \wedge t'=t$$

$$x \geq 1 \Rightarrow t' \leq t + \log x \iff$$

$$x \neq 1 \wedge (\text{div } x \ 2 \geq 1 \Rightarrow t' \leq t + 1 + \log (\text{div } x \ 2))$$

$$x < 1 \Rightarrow t' = \infty \iff x=1 \wedge x'=x \wedge t'=t$$

$$x < 1 \Rightarrow t' = \infty \iff x \neq 1 \wedge (\text{div } x \ 2 < 1 \Rightarrow t' = \infty)$$

$$\begin{aligned}
& (\quad x \geq 1 \Rightarrow t' \leq t + \log x \\
& \Leftarrow x \neq 1 \wedge (\text{div } x \ 2 \geq 1 \Rightarrow t' \leq t + 1 + \log (\text{div } x \ 2)) \quad) \\
& \qquad \qquad \qquad \text{portation: } (a \Rightarrow b) \Leftarrow c = b \Leftarrow a \wedge c \\
& = \quad t' \leq t + \log x \\
& \Leftarrow x \geq 1 \wedge x \neq 1 \wedge (\text{div } x \ 2 \geq 1 \Rightarrow t' \leq t + 1 + \log (\text{div } x \ 2)) \\
& \qquad \qquad \qquad \text{simplify and discharge} \\
& = \quad t' \leq t + \log x \\
& \Leftarrow x > 1 \wedge t' \leq t + 1 + \log (\text{div } x \ 2) \qquad \qquad \text{portation} \\
& = (t' \leq t + 1 + \log (\text{div } x \ 2) \Rightarrow t' \leq t + \log x) \Leftarrow x > 1 \\
& \qquad \qquad \qquad \text{Note that } t' \leq a \Rightarrow t' \leq b \Leftarrow a \leq b . \\
& \Leftarrow t + 1 + \log (\text{div } x \ 2) \leq t + \log x \Leftarrow x > 1 \\
& \qquad \qquad \qquad \text{subtract 1 from each side} \\
& = t + \log (\text{div } x \ 2) \leq t + \log x - 1 \Leftarrow x > 1 \\
& = t + \log (\text{div } x \ 2) \leq t + \log (x/2) \Leftarrow x > 1 \\
& \qquad \qquad \qquad \log \text{ and } + \text{ are monotonic for } x > 0 \\
& \Leftarrow \text{div } x \ 2 \leq x/2 \\
& = \top
\end{aligned}$$

$$\begin{aligned}
& (x < 1 \Rightarrow t' = \infty \Leftarrow x = 1 \wedge x' = x \wedge t' = t) && \text{Portation} \\
= & t' = \infty \Leftarrow x < 1 \wedge x = 1 \wedge x' = x \wedge t' = t && \text{generic, base} \\
= & t' = \infty \Leftarrow \perp && \text{base} \\
= & \top
\end{aligned}$$

$$\begin{aligned}
& (x < 1 \Rightarrow t' = \infty \Leftarrow x \neq 1 \wedge (\text{div } x \ 2 < 1 \Rightarrow t' = \infty)) && \text{Portation} \\
= & t' = \infty \Leftarrow x < 1 \wedge x \neq 1 \wedge (\text{div } x \ 2 < 1 \Rightarrow t' = \infty) && \text{Discharge} \\
= & t' = \infty \Leftarrow x < 1 \wedge t' = \infty && \text{Specialization} \\
= & \top
\end{aligned}$$

Termination

$$x'=2 \iff t:=t+1. x'=2$$

complain only if $x' \neq 2$

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

unimplementable

$$x'=2 \wedge (t < \infty \Rightarrow t' < \infty) \iff t:=t+1. x'=2 \wedge (t < \infty \Rightarrow t' < \infty)$$

complain only if $x' \neq 2 \vee t < \infty \wedge t' = \infty$

$$x'=2 \wedge t' \leq t+1 \iff x:=2$$

Linear Search

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

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

Define

$$R = \neg x: L(h, ..h') \wedge (Lh'=x \vee h'=\#L)$$

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

$$h \leq \#L \Rightarrow R \Leftarrow \mathbf{if } h=\#L \mathbf{ then } ok \mathbf{ else } h < \#L \Rightarrow R$$

$$h < \#L \Rightarrow R \Leftarrow \mathbf{if } Lh=x \mathbf{ then } ok \mathbf{ else } (h:=h+1. h \leq \#L \Rightarrow R)$$

Timing:

$$t' \leq t + \#L \iff h := 0. \quad h \leq \#L \Rightarrow t' \leq t + \#L - h$$

$$h \leq \#L \Rightarrow t' \leq t + \#L - h \iff \begin{array}{l} \mathbf{if} \ h = \#L \ \mathbf{then} \ ok \\ \mathbf{else} \ h < \#L \Rightarrow t' \leq t + \#L - h \end{array}$$

$$h < \#L \Rightarrow t' \leq t + \#L - h \iff \begin{array}{l} \mathbf{if} \ Lh = x \ \mathbf{then} \ ok \\ \mathbf{else} \ (h := h + 1. \ t := t + 1. \ h \leq \#L \Rightarrow t' \leq t + \#L - h) \end{array}$$

If L is nonempty,

$$\neg x : L(0, ..h') \wedge (Lh' = x \vee h' = \#L) \iff h := 0. \quad h < \#L \Rightarrow R$$

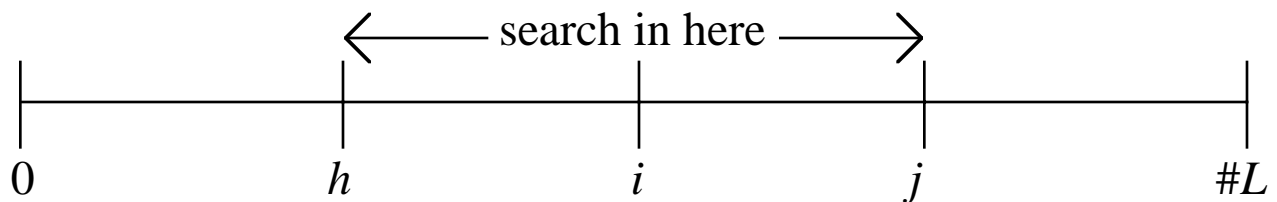
Binary Search

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

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

$$x: L(0, \dots, \#L) = p' \Rightarrow Lh' = x$$

Define $R = (x: L(h, \dots, j) = p' \Rightarrow Lh' = x)$



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

$$h < j \Rightarrow R \Leftarrow \text{if } j - h = 1 \text{ then } p := Lh = x \text{ else } j - h \geq 2 \Rightarrow R$$

$$j - h \geq 2 \Rightarrow R \Leftarrow j - h \geq 2 \Rightarrow h' = h < i' < j = j'.$$

$$\text{if } Li \leq x \text{ then } h := i \text{ else } j := i.$$

$$h < j \Rightarrow R$$

$$j - h \geq 2 \Rightarrow h' = h < i' < j = j' \Leftarrow i := \text{div}(h + j) 2$$

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

$$U \Leftarrow \mathbf{if } j-h = 1 \mathbf{ then } p:=Lh=x \mathbf{ else } V$$

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

$$\mathbf{if } Li \leq x \mathbf{ then } h:=i \mathbf{ else } j:=i.$$

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

$\#L$	=	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

$$T = t' \leq t + \text{ceil}(\log(\#L))$$

$$U = h < j \Rightarrow t' \leq t + \text{ceil}(\log(j-h))$$

$$V = j-h \geq 2 \Rightarrow t' \leq t + \text{ceil}(\log(j-h))$$

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.

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

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

$$P \Leftarrow \text{if } y=0 \text{ then ok else } y>0 \Rightarrow P$$

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

$$y>0 \Rightarrow P \Leftarrow \text{if even } y \text{ then } \text{even } y \wedge y>0 \Rightarrow P \text{ else } \text{odd } y \Rightarrow P$$

$$\text{even } y \wedge y>0 \Rightarrow P \Leftarrow x := x \times x. y := y/2. P$$

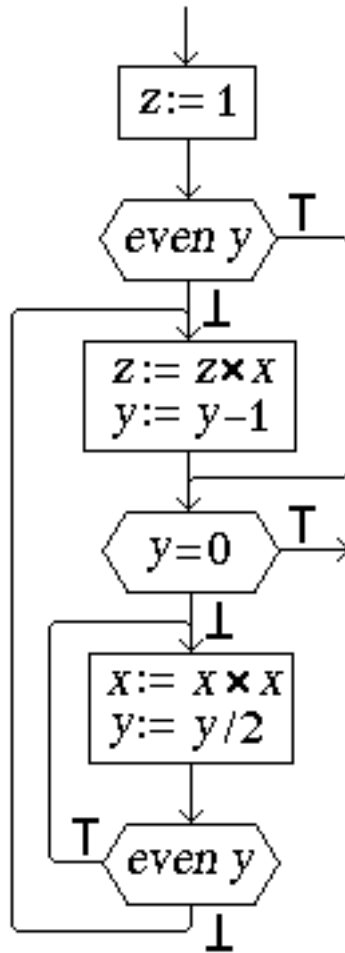
$$\text{odd } y \Rightarrow P \Leftarrow z := z \times x. y := y-1. P$$

$$\text{even } y \wedge y>0 \Rightarrow P \Leftarrow x := x \times x. y := y/2. y>0 \Rightarrow P$$

$$\text{odd } y \Rightarrow P \Leftarrow z := z \times x. y := y-1. \text{even } y \Rightarrow P$$

$$\text{even } y \Rightarrow P \Leftarrow \text{if } y = 0 \text{ then ok else } \text{even } y \wedge y>0 \Rightarrow P$$

$$P \Leftarrow \text{if even } y \text{ then } \text{even } y \Rightarrow P \text{ else } \text{odd } y \Rightarrow P$$



$$\text{even } y \wedge y > 0 \Rightarrow P \iff x := x \times x. \ y := y/2. \ t := t+1. \ y > 0 \Rightarrow P$$

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

if $y=0$ **then** $t'=t$ **else** $t' \leq t + \log y$

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

Fibonacci Numbers

$$f0 = 0$$

$$f1 = 1$$

$$f(n+2) = fn + f(n+1)$$

$$x' = fn \iff P$$

where

$$P = x' = fn \wedge y' = f(n+1)$$

$$P \iff \mathbf{if} \ n=0 \ \mathbf{then} \ (x:=0. \ y:=1) \\ \qquad \qquad \qquad \mathbf{else} \ (n:=n-1. \ P. \ x'=y \wedge y'=x+y)$$

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

$$t'=t+n \iff \mathbf{if} \ n=0 \ \mathbf{then} \ (x:=0. \ y:=1) \\ \qquad \qquad \qquad \mathbf{else} \ (n:=n-1. \ t:=t+1. \ t'=t+n. \ t'=t)$$

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

$$f(2 \times k + 1) = f k^2 + f(k+1)^2$$

$$f(2 \times k + 2) = 2 \times f k \times f(k+1) + f(k+1)^2$$

$P \Leftarrow$ **if** $n=0$ **then** $(x:= 0. y:= 1)$
else if *even* n **then** $\text{even } n \wedge n > 0 \Rightarrow P$
else *odd* $n \Rightarrow P$

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 $n \wedge n > 0 \Rightarrow P \Leftarrow$

$$n := n/2 - 1. P. x' = 2 \times x \times y + y^2 \wedge y' = x^2 + y^2 + x'$$

$x' = x^2 + y^2 \wedge y' = 2 \times x \times y + y^2 \Leftarrow$

$$n := x. x := x^2 + y^2. y := 2 \times n \times y + y^2$$

$x' = 2 \times x \times y + y^2 \wedge y' = x^2 + y^2 + x' \Leftarrow$

$$n := x. x := 2 \times x \times y + y^2. y := n^2 + y^2 + x$$

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

$$n=0 \Rightarrow T \Leftarrow x:=0. y:=1$$

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

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

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

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

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

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.

$n:= n+1$)

Time

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

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

else ($n:= n-1$.

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

$t:= t+1$.

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

$n:= n+1$)

$$\begin{aligned}
& n=0 \wedge ok && \text{expand } ok \\
= & n=0 \wedge n'=n \wedge t'=t && \text{arithmetic} \\
\Rightarrow & t:=t+2^n-1
\end{aligned}$$

$$\begin{aligned}
& n>0 \\
& \wedge (n:=n-1. t:=t+2^n-1. t:=t+1. t:=t+2^n-1. n:=n+1) \\
& \qquad \text{drop conjunct } n>0 ; \text{ expand final assignment} \\
\Rightarrow & n:=n-1. t:=t+2^n-1. t:=t+1. t:=t+2^n-1. n'=n+1 \wedge t'=t \\
& \qquad \text{use substitution law repeatedly from right to left} \\
= & n'=n-1+1 \wedge t'=t+2^{n-1}-1+1+2^{n-1}-1 && \text{simplify} \\
= & n'=n \wedge t'=t+2^{n-1} \\
= & t:=t+2^n-1
\end{aligned}$$

Space

$$s'=s \Leftarrow \begin{array}{l} \text{if } n=0 \text{ then } ok \\ \text{else (} n:=n-1. \\ \quad s:=s+1. \ s'=s. \ s:=s-1. \\ \quad ok. \\ \quad s:=s+1. \ s'=s. \ s:=s-1. \\ \quad n:=n+1 \)} \end{array}$$

Maximum Space

$$m \geq s \Rightarrow (m := \max m (s+n)) \Leftarrow \begin{array}{l} \text{if } n=0 \text{ then } ok \\ \text{else (} n:=n-1. \\ \quad s:=s+1. \ m := \max m \ s. \ m \geq s \Rightarrow (m := \max m (s+n)). \\ \quad s:=s-1. \\ \quad ok. \\ \quad s:=s+1. \ m := \max m \ s. \ m \geq s \Rightarrow (m := \max m (s+n)). \\ \quad s:=s-1. \\ \quad n:=n+1 \)} \end{array}$$

Average Space

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

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

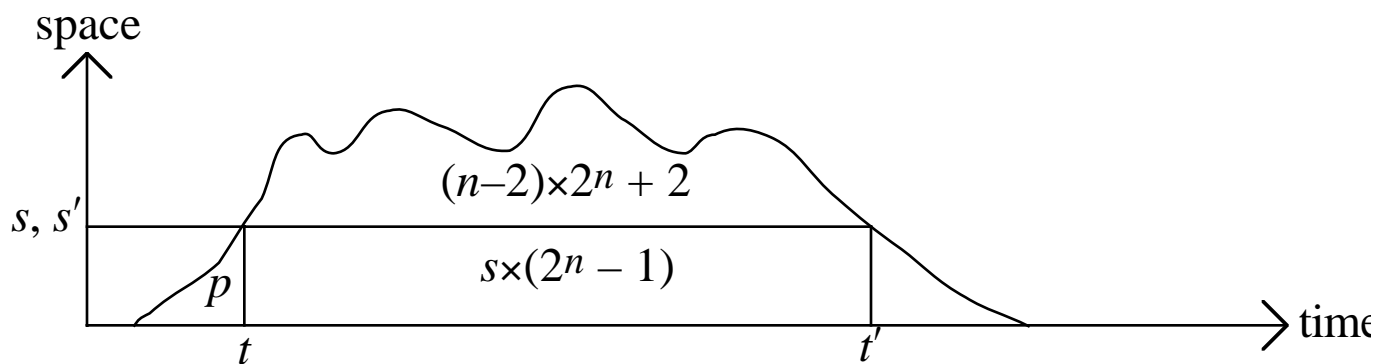
else ($n := n-1$.

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

$$p := p+s.$$

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

$n := n+1$)



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

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

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

average space $\leq n$

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*. $s:=s-1$.

$n:=n+1$)

MovePile = $n'=n$

$\wedge t' = t + 2^n - 1$

$\wedge s' = s$

$\wedge (m \geq s \Rightarrow m' = \max m (s+n))$

$\wedge p' = p + s \times (2^n - 1) + (n-2) \times 2^n + 2$