Data refinement

A Strength reduction

Recall the division algorithm

```
{ $y > 0 \wedge x \geq 0$ }
$p := 0$
{ $I_0 : p \times y \leq x$ }
while $x \geq (p+1) \times y$ do
    $p := p + 1$
end do
{ $p \times y \leq x \wedge x < (p+1) \times y$ }
```

To save time and/or hardware, it would be nice to eliminate the multiplication.

Data refinement is a technique that changes the coordinate system that we use to represent the state space.

In this case the new coordinate system is the same as the old, but with one variable added

Old coordinates	New coordinates
x	x
y	y
p	p
	t

To eliminate the multiplication, we relate the new coordinate system to the old via a "linking invariant" of

$$t = (p+1) \times y$$

The only place we need this to be true is while evaluating the loop guard. Thus we conjoin the linking invariant with the loop invariant above, to get a new loop invariant.

$$I : p \times y \leq x \wedge t = (p+1) \times y$$

The new algorithm is

```
{ $y > 0 \wedge x \geq 0$ }
$p := 0$
var $t$
initialize $t$ to establish $I$
{ $I : p \times y \leq x \wedge t = (p+1) \times y$ }
while $x \geq t$ do // N.B. No multiplication.
    $p := p + 1$
    change $t$ to reestablish $I$
end do
{ $p \times y \leq x \wedge x < (p+1) \times y$ }
```

and so

```
{ $y > 0 \wedge x \geq 0$ }
$p := 0$
var $t := y$
{ $I : p \times y \leq x \wedge t = (p+1) \times y$ }
while $x \geq t$ do
    $p := p + 1$
    $t := t + y$
end do
{ $p \times y \leq x \wedge x < (p+1) \times y$ }
```

This kind of data refinement is often called a **strength reduction**, as it replaces an expensive operation (multiplication) with a less expensive one (addition).

t is called a **tracking variable**, as it tracks expression $(p+1) \times y$.

In this example the problem did not change as we left alone all variables mentioned in the precondition and the postcondition.

Data Refinement of an abstract problem

In this example, we will data refine an algorithm that solves one problem so that it solves a different problem.

[Note all numeric variables in this slide set are over natural numbers $\mathbb{N}$]

We start with the abstract binary search algorithm

Recall the problem was to find a point p in $\{m, .., n\}$, the domain of a boolean function A, such that $\neg A(p)$ and $A(p+1)$.

To ensure such a point exists, we assume $m < n$ and $\neg A(m)$ and $A(n)$.

$$\{\neg A(m) \wedge A(n) \wedge m < n\} \; ? \; \{\neg A(p) \wedge A(p+1)\}$$

Now consider the problem of designing a divider. Assume $x \geq 0$ and $y > 0$.

$$\{?\} \; ? \; \{p \times y \leq x \wedge x < (p+1) \times y\}$$

We can link these two problems with the following linking invariant

$$L : (A(i) = (i \times y > x)) \wedge (m = 0) \qquad \text{for all } i \geq 0$$

Assuming the linking invariant

$$A(i) = (i \times y > x) \wedge m = 0 \qquad \text{for all } i \geq 0$$

we have for the precondition:

$$\begin{aligned} & \neg A(m) \wedge A(n) \wedge m < n \\ = \; & \text{by the linking in variant} \\ & \neg (0 \times y > x) \wedge n \times y > x \wedge 0 < n \\ = \; & 0 \text{ can't be greater than } x \\ & n \times y > x \wedge 0 < n \\ = \; & \text{since } n \times y > x \text{ implies } n > 0 \\ & n \times y > x \end{aligned}$$

and for the postcondition

$$\begin{aligned} & \neg A(p) \wedge A(p+1) \\ = \; & \text{by the linking invariant} \\ & \neg (p \times y > x) \wedge (p+1) \times y > x \\ = \; & \\ & p \times y \leq x < (p+1) \times y \end{aligned}$$

So, assuming the linking invariant L, the search problem

$$\{\neg A(m) \wedge A(n) \wedge m < n\} \; ? \; \{\neg A(p) \wedge A(p+1)\}$$

is the same as this division problem

$$\{n \times y > x\} \; ? \; \{p \times y \leq x < (p+1) \times y\}$$

We have data refined one problem to another.

Can we do the same for the algorithm?

Data refinement of an abstract algorithm

We know we can use binary search to solve the abstract problem

The abstract algorithm is

```
$\{ \neg A(m) \wedge A(n) \wedge m < n \}$
$p := m$
$r := n$
$\{ I : m \leq p < r \leq n \wedge \neg A(p) \wedge A(r) \}$
while $r \neq p + 1$ do
    var $q := \left\lfloor \frac{p+r}{2} \right\rfloor$
    $\{p < q < r \wedge I \}$
    if $A(q)$ then $r := q$ else $p := q$ end if
end while
$\{ \neg A(p) \wedge A(p + 1) \}$
```

Introduce natural variables x and y using the linking invariant

$$L : A(i) = (i \times y > x) \wedge m = 0 \qquad \text{for all } i \geq 0$$

```
$\{ \neg A(m) \wedge A(n) \wedge m < n \underline{\wedge L} \}$
$p := m$
$r := n$
$\{ I : m \leq p < r \leq n \wedge \neg A(p) \wedge A(r) \underline{\wedge L} \}$
while $r \neq p + 1$ do
    var $q := \left\lfloor \frac{p+r}{2} \right\rfloor$
    $\{p < q < r \wedge I \underline{\wedge L} \}$
    if $A(q)$ then $r := q$ else $p := q$ end if
end while
$\{ \neg A(p) \wedge A(p + 1) \underline{\wedge L} \}$
```

Rewrite in terms of x and y

```
{ $\underline{m \times y \leq x \wedge n \times y > x} \wedge m < n \wedge L$} [NB: simplifies to $n \times y > x \wedge L$]
$p := 0 \qquad r := n$
{ $I0 : \underline{0} \leq p < r \leq n \wedge \underline{p \times y \leq x < q \times y} \wedge L$ }
while $r \neq p + 1$ do
    var $q := \left\lfloor \frac{p+r}{2} \right\rfloor$
    {$p < q < r \wedge I0 \wedge L$ }
    if $\underline{q \times y > x}$ then $r := q$ else $p := q$ end if
end while
{ $\underline{p \times y \leq x < (p + 1) \times y} \wedge L$ }
```

A & m are only mentioned in L. Drop these variables.

```
{ $n \times y > x$ }
$p := 0 \qquad r := n$
{ $I0' : 0 \leq p < r \leq n \wedge p \times y \leq x < r \times y$ }
while $r \neq p + 1$ do
    var $q := \left\lfloor \frac{p+r}{2} \right\rfloor$
    { $p < q < r \wedge I0'$ }
    if $q \times y > x$ then $r := q$ else $p := q$ end if
end while
{ $p \times y \leq x < (p + 1) \times y$ }
```

In summary, we

- Started with an algorithm to search a boolean sequence A
- Introduced new variables x and y using a linking invariant.
- Rewrote to eliminate the need for the sequence A
- Arrived at an algorithm for division that mentions only

numerical variables and arithmetic operations.

Old coordinates	New coordinates
A	x
m	y
n	n
p	p
r	r

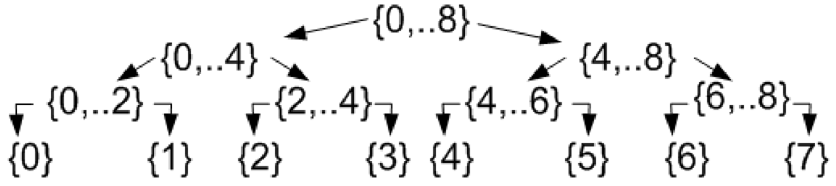
Linking invariant

$$L : A(i) = i \times y > x \land m = 0 \qquad \text{for all } i \geq 0$$

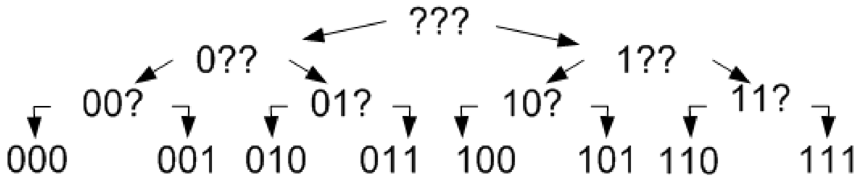
A further data refinement

If n is a power of 2, an interesting thing happens: The algorithm finds one bit per iteration.

Take $n = 2^3$ for example



You can see that, in binary, each step down decides on one more bit of the answer.



At level i above the bottom, each range is of the form $\left\{c2^i, ..\left(c+1\right)2^i\right\}$.

Note that, when $i > 0$,

$$\left\lfloor \frac{c2^i + (c+1)\,2^i}{2} \right\rfloor = \left\lfloor \frac{(2c+1)\,2^i}{2} \right\rfloor = (2c+1)\,2^{i-1}$$

E.g. $\{48, ..56\} = \left\{6 \times 2^3, ..7 \times 2^3\right\}$ is numbers of the form 110???. And this splits in half to give

$$\{48, ..52\} = \left\{12 \times 2^2, ..13 \times 2^2\right\}$$

and

$$\{52, ..56\} = \left\{13 \times 2^2, ..14 \times 2^2\right\}$$

In general, if $i > 0$, $\left\{c2^i, ..\left(c+1\right)2^i\right\}$ is the union of $\left\{2c2^{i-1}, ..\left(2c+1\right)2^{i-1}\right\}$ and $\left\{\left(2c+1\right)2^{i-1}, ..\left(\left(2c+1\right)+1\right)2^{i-1}\right\}$.

Here is the algorithm again, now assuming n is a power of 2. Also, I added a conjunct to the postcondition that I'll need later.

```
{ $n \times y > x \wedge \underline{\left(\exists j \cdot n = 2^j\right)}$ }
$p := 0 \quad r := n$
{ $I1 : p < r \leq n \wedge p \times y \leq x < r \times y \wedge \underline{\left(\exists j \cdot n = 2^j\right)}$ }
while $r \neq p + 1$ do
    var $q := \left\lfloor \frac{p+r}{2} \right\rfloor$
    { $p < q < r \wedge I1$ }
    if $q \times y > x$ then $r := q$ else $p := q$ end if
end while
{ $p \times y \leq x < (p + 1) \times y \wedge \underline{r = p + 1}$ }
```

Replace variable n with variable j, using a linking invariant of $n = 2^j$

```
{ $\underline{2^j} \times y > x$ }
$p := 0 \quad r := \underline{2^j}$
{ $I1' : p < r \leq \underline{2^j} \wedge p \times y \leq x < r \times y$ }
while $r \neq p + 1$ do
    var $q := \left\lfloor \frac{p+r}{2} \right\rfloor$
    { $p < q < r \wedge I1'$ }
    if $q \times y > x$ then $r := q$ else $p := q$ end if
end while
{ $p \times y \leq x < (p + 1) \times y \wedge r = p + 1$ }
```

The next three steps

- Add variables i and c.
- Rewrite the algorithm to use i and c instead of p and r.
- Eliminate variables p and r.

Add variables i and c, using a linking invariant of

$$L1 : p = c2^i \wedge r = (c+1)\,2^i$$

we get

```
{ $2^j \times y > x$ }
$[p, r, c, i] := \left[0, 2^j, 0, j\right]$
{ $I2 : p < r \le 2^j \wedge p \times y \le x < r \times y \wedge L1$ }
while $r \ne p + 1$ do
    var $q := \left\lfloor \frac{p+r}{2} \right\rfloor$
    if $q \times y > x$
    then $\begin{bmatrix} r \\ c \\ i \end{bmatrix} := \begin{bmatrix} q \\ 2c \\ i-1 \end{bmatrix}$ else $\begin{bmatrix} p \\ c \\ i \end{bmatrix} := \begin{bmatrix} q \\ 2c+1 \\ i-1 \end{bmatrix}$
    end if
end while
{ $p \times y \le x < (p+1) \times y \wedge r = p + 1 \wedge L1$ }
```

Rewrite to use i and c in preference to p and r.

```
{ $2^j \times y > x$ }
$[p, r, c, i] := \left[0, 2^j, 0, j\right]$
{ $I2' : c2^i \times y \le x < (c+1)\,2^i \times y \wedge L1$ }
while $\underline{i > 0}$ do
    var $q := \underline{(2c+1)\,2^{i-1}}$
    if $q \times y > x$
    then $\begin{bmatrix} r \\ c \\ i \end{bmatrix} := \begin{bmatrix} q \\ 2c \\ i-1 \end{bmatrix}$ else $\begin{bmatrix} p \\ c \\ i \end{bmatrix} := \begin{bmatrix} q \\ 2c+1 \\ i-1 \end{bmatrix}$
    end if
end while
{ $\underline{c} \times y \le x < (\underline{c} + 1) \times y \wedge \underline{i = 0} \wedge L1$ }
```

Now eliminate variables p and r

```
$\{ 2^j \times y > x \}$
$\begin{bmatrix} c \\ i \end{bmatrix} := \begin{bmatrix} 0 \\ j \end{bmatrix}$
$\{ I2'' : c2^i \times y \le x < (c+1)\, 2^i \times y \}$
while $i > 0$ do
    var $q := (2c+1)\, 2^{i-1}$
    if $q \times y > x$
    then $\begin{bmatrix} c \\ i \end{bmatrix} := \begin{bmatrix} 2c \\ i-1 \end{bmatrix}$ else $\begin{bmatrix} c \\ i \end{bmatrix} := \begin{bmatrix} 2c+1 \\ i-1 \end{bmatrix}$
    end if
end while
$\{ c \times y \le x < (c+1) \times y \wedge i = 0 \}$
```

Old coordinates	New coordinates
n	j
p	c
r	i
x	x
y	y

Linking invariant

$$p = c2^i \wedge r = (c+1)\, 2^i \wedge n = 2^j$$

Another strength reduction

Can we eliminate the multiplication by strength reduction?

$$\begin{aligned} & q \times y \\ = \; & (2c+1)\,2^{i-1}y \\ = \; & 2^i cy + 2^{i-1}y \end{aligned}$$

Introduce tracking variables $d = 2^i cy$ and $e = 2^{i-1}y$, so $q \times y = d + e$

$$L2 : d = 2^i cy \wedge e = 2^{i-1}y$$

After a bit of algebra to find the right assignments to d and e, we get:

```
{ $2^j \times y > x$ }
$\begin{bmatrix} c \\ i \\ d \\ e \end{bmatrix} := \begin{bmatrix} 0 \\ j \\ 0 \\ 2^{j-1}y \end{bmatrix}$
{ $I3 : c2^i \times y \leq x < (c+1)\,2^i \times y \wedge L2$ }
while $i > 0$ do
    if $\underline{d+e} > x$
    then $\begin{bmatrix} c \\ i \\ d \\ e \end{bmatrix} := \begin{bmatrix} 2c \\ i-1 \\ d \\ e/2 \end{bmatrix}$ else $\begin{bmatrix} c \\ i \\ d \\ e \end{bmatrix} := \begin{bmatrix} 2c+1 \\ i-1 \\ d+e \\ e/2 \end{bmatrix}$
    end if
end while
{ $c \times y \leq x < (c+1) \times y$ }
```

Old coordinates	New coordinates
j	j
c	c
i	i
x	x
y	y
	d
	e

Linking invariant

$$d = 2^i cy \text{ and } e = 2^{i-1}y$$

Starting from an algorithm to search a boolean list, we have designed an efficient circuit that requires nothing more complex than an adder.

Here is the data path.