# Programming Languages:
# Imperative Program Construction
# 6. Loop Construction III: Using Associativity

Shin-Cheng Mu

Autumn. 2021

## 1 General Use of Associativity

### Tail Recursion

- A function $f$ is *tail recursive* if it looks like:

$$f\ x\ = h\ x, \qquad \text{if } b\ x;$$
$$f\ x\ = f\ (g\ x), \quad \text{if } \neg(b\ x).$$

- Tail recursive functions can be naturally computed in a loop. To derive a program that computes $f\ X$ for given $X$:

$$\textbf{con } X;\ \textbf{var } r, x;$$

$$x := X$$
$$\{f\ x = f\ X\}$$
$$\textbf{do } \neg(b\ x) \rightarrow x := g\ x\ \textbf{od}$$
$$r := h\ x$$
$$\{r = f\ X\}$$

provided that the loop terminates.

### Using Associativity

- What if the function to be computed is not tail recursive?

- Consider function $k$ such that:

$$k\ x\ = a, \qquad\qquad\ \text{if } b\ x;$$
$$k\ x\ = h\ x \oplus k\ (g\ x), \quad \text{if } \neg(b\ x).$$

  where $\oplus$ is associative with identity $e$.

- Note that $k$ is not tail recursive.

- Goal: establish $r = k\ X$ for given $X$.

- Trick: use an invariant $r \oplus k\ x = k\ X$.

  - 'computed' $\oplus$ 'to be computed' $= k\ X$.
  - Strategy: keep shifting stuffs from right hand side of $\oplus$ to the left, until the right is $e$.

### Constructing the Loop Body

If $b\ x$ holds:

$$r \oplus k\ x = k\ X$$
$$\equiv\ \ \{\ b\ x\ \}$$
$$r \oplus a = k\ X.$$

Otherwise:

$$r \oplus k\ x = k\ X$$
$$\equiv\ \ \{\ \neg(b\ x)\ \}$$
$$r \oplus (h\ x \oplus k\ (g\ x)) = k\ X$$
$$\equiv\ \ \{\ \oplus \text{ associative }\}$$
$$(r \oplus h\ x) \oplus k\ (g\ x) = k\ X$$
$$\equiv\ (r \oplus k\ x = k\ X)[r, x \backslash r \oplus h\ x, g\ x].$$

### The Program

$$\textbf{con } X;\ \textbf{var } r, x;$$

$$r, x := e, X$$
$$\{r \oplus k\ x = k\ X\}$$
$$\textbf{do } \neg(b\ x) \rightarrow r, x := r \oplus h\ x, g\ x\ \textbf{od}$$
$$\{r \oplus a = k\ X\}$$
$$r := r \oplus a$$
$$\{r = k\ X\}$$

if the loop terminates.

## 2 Example: Exponentation

### Exponentation Again

- Consider again computing $A^B$.

$$\textbf{con } A, B : Int \ \{0 \leqslant B\}$$
$$\textbf{var } r : Int$$
$$?$$
$$\{r = A^B\}$$

- Notice that:

$$x^0 = 1$$
$$x^y = 1 \times (x \times x)^{y \ \textbf{div} \ 2} \quad \text{if } even \ y,$$
$$= x \times x^{y-1} \quad \text{if } odd \ y.$$

- How does it fit the pattern above? (Hint: $k$ now has type $(Int \times Int) \rightarrow Int$.)

- To be concrete, let us look at this specialised case in more detail.

## Invariant and Initialisation

- To achieve $r = A^B$, introduce variables $a$, $b$ and choose invariant $r \times a^b = A^B$.

- To satisfy the invariant, initialise with $r, a, b := 1, A, B$.

- If $b = 0$ we have $r = A^B$. Therefore the strategy would be use $b$ as bound and decrease $b$.

## Linear-Time Exponentation

- How to decrease $b$? One might try $b := b - 1$. We calculate:

$$(r \times a^b = A^B)[b \backslash b - 1]$$
$$= r \times a^{b-1} = A^B \ .$$

- To fullfill the spec below

$$\{r \times a^b = A^B\}$$
$$r := ?$$
$$\{r \times a^{b-1} = A^B\}$$

One may choose $r := r \times a$.

- That results in the program (omitting the assertions):

$$\textbf{con } A, B : Int \ \{0 \leqslant B\}$$
$$\textbf{var } r : Int$$
$$r, a, b := 1, A, B$$
$$\textbf{do } b \neq 0 \rightarrow r := r \times a; b := b - 1 \textbf{ od}$$
$$\{r = A^B\}$$

- This program use $O(B)$ multiplications. But we wish to do better this time.

## Try to Decrease Faster

- Or, we try to decrease $b$ faster by halfing it (let $(/)$ denote integer division).

$$(r \times a^b = A^B)[b \backslash b \ / \ 2]$$
$$= r \times a^{b/2} = A^B \ .$$

- How to fullfill the spec below?

$$\{r \times a^b = A^B\}$$
$$?$$
$$\{r \times a^{b/2} = A^B\}$$

- If we choose $a := a \times a$:

$$(r \times a^{b/2})[a \backslash a \times a]$$
$$= r \times (a \times a)^{b/2}$$
$$= r \times (a^2)^{b/2}$$
$$= r \times a^{2 \times (b/2)}$$
$$= \quad \{ \ even \ b \ \}$$
$$r \times a^b \ .$$

- But wait! For the last step to be valid we need $even \ b$!

- That means the program fragment has to be put under a guarded command:

$$even \ b \rightarrow$$
$$\{r \times a^b = A^B \wedge even \ b\}$$
$$a := a \times a$$
$$\{r \times a^{b/2} = A^B\}$$
$$b := b \ / \ 2$$
$$\{r \times a^b = A^B\}$$

- For that we need to introduce an **if** in the loop body.

## Fast Exponentiation

- We can put the $b := b - 1$ choice under an $odd \ b$ guard, resulting in the following program:

$$\textbf{con } A, B : Int \ \{0 \leqslant B\}$$
$$\textbf{var } r : Int$$
$$r, a, b := 1, A, B$$
$$\{r \times a^b = A^B \wedge 0 \leqslant b, bnd : b\}$$
$$\textbf{do } b \neq 0 \rightarrow$$
$$\quad \textbf{if} \quad odd \ b \ \rightarrow r := r \times a$$
$$\qquad\qquad\qquad\qquad b := b - 1$$
$$\quad \mid even \ b \rightarrow a := a \times a$$
$$\qquad\qquad\qquad\qquad b := b \ / \ 2$$
$$\quad \textbf{fi}$$
$$\textbf{od}$$
$$\{r = A^B\}$$

- This program uses $O(\log B)$ multiplications.

## Fast Exponentiation

- There is no reason, however, that you have to put the $b := b - 1$ choice under an $odd\ b$ guard.

- You might come up with something like this:

$$\textbf{con}\ A, B : Int\ \{0 \leqslant B\}$$
$$\textbf{var}\ r : Int$$

$$r, a, b := 1, A, B$$
$$\{r \times a^b = A^B \wedge 0 \leqslant b, bnd : b\}$$
$$\textbf{do}\ b \neq 0 \rightarrow$$
$$\quad r := r \times a$$
$$\quad b := b - 1$$
$$\quad \textbf{if}\quad True\quad \rightarrow skip$$
$$\quad\quad |\ even\ b \rightarrow a := a \times a$$
$$\quad\quad\quad\quad\quad\quad b := b\ /\ 2$$
$$\quad \textbf{fi}$$
$$\textbf{od}$$
$$\{r = A^B\}$$

- This program would be correct! Every pieces of proofs we need has been constructed.

- But you do not get a faster program this way.

## Side Note: Constructing Branches

- How do we construct branches?

- If a program fragment needs a side condition to work, we know that we need a guard.

- We keep constructing branches until the disjunction of all the guards can be satisfied.