Laws of Programming

We will look at various general laws that are helpful in deriving programs.

‘Universally true’ and ‘Stronger Than’

If a boolean expression $\mathcal{A}$ is true regardless of the values of its free variables, it is said to be **universally true.**

Here are some examples of universally true expressions:

$$\mathfrak{true}$$
$$x \geq x$$
$$x + 42 > x$$
$$x \in \{x, y, z\}$$
$$p \wedge q \Rightarrow p$$

A boolean expression $\mathcal{B}$ is considered to be **stronger than** a boolean expression $\mathcal{A}$ if

$\mathcal{B} \Rightarrow \mathcal{A}$, is universally true

For example

$$0 < x < y$$

is stronger than

$$0 \leq x \leq y$$

If $\mathcal{A}$ is stronger than $\mathcal{B}$, we say $\mathcal{B}$ is **weaker than** $\mathcal{A}$.

Some examples

$\mathcal{A}$ is stronger than $\mathcal{A} \vee \mathcal{B}$

$\mathcal{A}$ is stronger than $\mathcal{B} \Rightarrow \mathcal{A}$

$\mathcal{A} \wedge \mathcal{B}$ is stronger than $\mathcal{A}$

Monotonicity properties: If $\mathcal{B}$ is stronger than $\mathcal{A}$ then

$\mathcal{B} \wedge \mathcal{C}$ is stronger than $\mathcal{A} \wedge \mathcal{C}$

$\mathcal{B} \vee \mathcal{C}$ is stronger than $\mathcal{A} \vee \mathcal{C}$

$\mathcal{C} \Rightarrow \mathcal{B}$ is stronger than $\mathcal{C} \Rightarrow \mathcal{A}$

Anti-monotonicity properties: If $\mathcal{B}$ is stronger than $\mathcal{A}$ then

$\neg\mathcal{A}$ is stronger than $\neg\mathcal{B}$

$\mathcal{A} \Rightarrow \mathcal{C}$ is stronger than $\mathcal{B} \Rightarrow \mathcal{C}$

(Perhaps we should say “stronger than or the same as”, but this is a mouthful.)

Strengthening laws

The strengthening law says: If $\mathcal{B}$ is stronger than $\mathcal{A}$ then

$$\langle\mathcal{A}\rangle \sqsubseteq \langle\mathcal{B}\rangle$$

Some examples

$$\begin{aligned}\langle\mathcal{A} \vee \mathcal{B}\rangle \ &\sqsubseteq\ \langle\mathcal{A}\rangle \\ \langle\mathcal{B} \Rightarrow \mathcal{A}\rangle \ &\sqsubseteq\ \langle\mathcal{A}\rangle \\ \langle\mathcal{A}\rangle \ &\sqsubseteq\ \langle\mathcal{A} \wedge \mathcal{B}\rangle\end{aligned}$$

Monotonicity properties: If $\langle\mathcal{A}\rangle \sqsubseteq \langle\mathcal{B}\rangle$ then

$$\begin{aligned}\langle\mathcal{A} \wedge \mathcal{C}\rangle \ &\sqsubseteq\ \langle\mathcal{B} \wedge \mathcal{C}\rangle \\ \langle\mathcal{A} \vee \mathcal{C}\rangle \ &\sqsubseteq\ \langle\mathcal{B} \vee \mathcal{C}\rangle \\ \langle\mathcal{C} \Rightarrow \mathcal{A}\rangle \ &\sqsubseteq\ \langle\mathcal{C} \Rightarrow \mathcal{B}\rangle\end{aligned}$$

Anti-monotonicity properties: If $\langle\mathcal{A}\rangle \sqsubseteq \langle\mathcal{B}\rangle$ then

$$\begin{aligned}\langle\neg\mathcal{B}\rangle \ &\sqsubseteq\ \langle\neg\mathcal{A}\rangle \\ \langle\mathcal{B} \Rightarrow \mathcal{C}\rangle \ &\sqsubseteq\ \langle\mathcal{A} \Rightarrow \mathcal{C}\rangle\end{aligned}$$

Monotonicity Laws

With the natural numbers, $\mathbb{N}$, the operations of addition and multiplication are *monotonic* with respect to $\leq$: For example, if p, q, and r are natural numbers, then if $p \leq q$, we have $p + r \leq q + r$ and $p \cdot r \leq q \cdot r$.

Similarly we can say that our programming operators are monotonic with respect to refinement.

In particular, if f, g, and h are specifications such that $f \sqsubseteq g$, we have

- $f \wedge h \sqsubseteq g \wedge h$
- $f \vee h \sqsubseteq g \vee h$
- $h \Rightarrow f \sqsubseteq h \Rightarrow g$
- $f; h \sqsubseteq g; h$
- $h; f \sqsubseteq h; g$
- $\mathbf{if}\ \mathcal{A}\ \mathbf{then}\ f\ \mathbf{else}\ h \sqsubseteq \mathbf{if}\ \mathcal{A}\ \mathbf{then}\ g\ \mathbf{else}\ h$
- $\mathbf{if}\ \mathcal{A}\ \mathbf{then}\ h\ \mathbf{else}\ f \sqsubseteq \mathbf{if}\ \mathcal{A}\ \mathbf{then}\ h\ \mathbf{else}\ g$
- $\mathbf{while}\ \mathcal{A}\ \mathbf{do}\ f \sqsubseteq \mathbf{while}\ \mathcal{A}\ \mathbf{do}\ g$

Skip laws

The following laws follow from the definition of skip and the strengthening laws

$$\begin{aligned} \langle x' = x \rangle &\sqsubseteq \mathbf{skip} \\ \langle x' = x \wedge y' = y \rangle &\sqsubseteq \mathbf{skip} \\ \langle x' = x \wedge y' = y \wedge z' = z \rangle &\sqsubseteq \mathbf{skip} \end{aligned}$$

Assignment laws

The following laws follow from the definition of assignment and the strengthening law

$$\begin{aligned} \langle x' = \mathcal{E} \rangle &\sqsubseteq x := \mathcal{E} \\ \langle x' = \mathcal{E} \wedge y' = y \rangle &\sqsubseteq x := \mathcal{E} \\ \langle x' = \mathcal{E} \wedge y' = y \wedge z' = z \rangle &\sqsubseteq x := \mathcal{E} \end{aligned}$$

Erasure laws

Erasure law for skip

The above laws for skip and assignment can be generalized.

Consider $x' \geq x$, this is weaker than $x' = x$, we have

$$\langle x' \geq x\rangle \sqsubseteq \langle x' = x\rangle \sqsubseteq \mathbf{skip}$$

More generally any expression $\mathcal{A}$ will be weaker than $x' = x$ if replacing every x' in $\mathcal{A}$ with an x gives a universally true expression. (This is the one-point law.).

We'll use the notation $\widetilde{\mathcal{A}}$ to mean the expression $\mathcal{A}$ with all primes removed.

E.g. $\widetilde{x' \geq x}$ is $x \geq x$.

In general we have an

Erasure law for skip. $\langle\mathcal{A}\rangle \sqsubseteq \mathbf{skip}$ exactly if $\widetilde{\mathcal{A}}$ is universally true.

Example: $\langle x > 0 \Rightarrow x' \geq 0\rangle \sqsubseteq \mathbf{skip}$ since $x > 0 \Rightarrow x \geq 0$ is universally true.

Erasure law for assigment

Consider a state space with integer variables x and y. We have

$$x' = x + 42 \wedge y' = y$$

stronger than

$$x' > x \wedge y' \geq y$$

since $x + 42 > x \wedge y \geq y$ is universally true.

In general we have the following

Erasure law for assignment $\langle \mathcal{A} \rangle \sqsubseteq \mathcal{V} := \mathcal{E}$ exactly if $\mathcal{A}[\mathcal{V}' : \mathcal{E}]$ is universally true.

Example $\langle x' = x \wedge y' = t \rangle \sqsubseteq y := t$ since $(x' = x \wedge y' = t)[y' : t]$ is $x' = x \wedge t = t$ and since $x' = x \wedge t = t$ is $x = x \wedge t = t$ which is universally true.

Example $\langle x' = y \wedge y' = x \rangle \sqsubseteq x, y := y, x$ since $(x' = y \wedge y' = x)[x', y'; y, x]$ is $y = y \wedge x = x$, which is universally true.

The forward substitution law

The following **forward substitution law** is very useful for introducing assignment statements into programs

The forward substitution law $\langle \mathcal{A}[\mathcal{V} : \mathcal{E}] \rangle = (\mathcal{V} := \mathcal{E}; \langle \mathcal{A} \rangle)$

Example:

Consider refining $\langle x' = 3x + 42 \wedge y' = 3x + 41 \rangle$

$$\begin{array}{ll} & \langle x' = 3x + 42 \wedge y' = 3x + 41 \rangle \\ \sqsubseteq & \text{“rewrite } 41 \text{ as } 42 - 1\text{”} \\ & \langle x' = \underline{3x + 42} \wedge y' = \underline{3x + 42} - 1 \rangle \\ \sqsubseteq & \text{“forward substitution”} \\ & x := 3x + 42; \langle x' = x \wedge y' = x - 1 \rangle \end{array}$$

Example:

We can use parallel assignment. Consider $g = \left\langle i \leq n \Rightarrow s' = s + \sum_{k \in \{i,..n\}} a(k) \right\rangle$

Then

$$\begin{array}{ll} & \left\langle \underline{i + 1} \leq n \Rightarrow s' = \underline{s + a(i)} + \sum_{k \in \{\underline{i+1},..n\}} a(k) \right\rangle \\ = & \text{Substitution law} \\ & i, s := i + 1, s + a(i)\ ; g \end{array}$$

Example: Swap

Consider the following specification

$$\langle x' = y \wedge y' = x \rangle$$

We will assume that multiple assignments are not allowed. We'll also assume that there is a variable t of appropriate type.

Can we derive a sequential composition of single assignments that does the job?

$$\begin{array}{ll}
 & \underline{\langle x' = y \wedge y' = x \rangle} \\
= & \text{Forward substitution} \\
 & t := x\,;\underline{\langle x' = y \wedge y' = t \rangle} \\
= & \text{Forward substitution} \\
 & t := x\,; x := y\,;\underline{\langle x' = x \wedge y' = t \rangle} \\
\sqsubseteq & \text{Erasure law for assignment} \\
 & t := x\,; x := y\,; y := t
\end{array}$$

Note how the last step also uses a monotonicity law. We generally won't call attention to uses of monotonicity laws. They are used implicitly.

Backward Substitution

We can also introduce an assignment as the final statement, using the **backward substitution law**.

Let $\mathcal{E}'$ be an expression identical to $\mathcal{E}$ except with a prime added to each variable.

The backward substitution law $\langle\mathcal{A}\rangle \sqsubseteq (\langle\mathcal{A}[\mathcal{V}' : \mathcal{E}']\rangle\,; \mathcal{V} := \mathcal{E})$

Example. Consider swapping again. Again, we'll assume there is a variable t that we can use.

$$\underline{\langle x' = y \land y' = x\rangle}$$

$\sqsubseteq$ "Backward substitution"

$$\underline{\langle x' = y \land t' = x\rangle}; y := t$$

$\sqsubseteq$ "Backward substitution"

$$\underline{\langle y' = y \land t' = x\rangle}; x := y; y := t$$

$\sqsubseteq$ "Erasure law"

$$t := x; x := y; y := t$$

Alternation law

Once we have checked a condition, it can become a precondition. This idea is captured in the alternation law

$$f = \textbf{if } \mathcal{A} \textbf{ then } (\langle\mathcal{A}\rangle \Rightarrow f) \textbf{ else } (\neg\langle\mathcal{A}\rangle \Rightarrow f)$$

Example: Find the minimum

We know that

$$\min(a,b) = a, \text{ if } a \le b \tag{1}$$

$$\min(a,b) = b, \text{ if } b \le a \tag{2}$$

Suppose we wish to implement

$$f = \langle a' = \min(a,b)\rangle$$

f

$=$ Alternation law

$\textbf{if } a \le b \textbf{ then } (\langle a \le b\rangle \Rightarrow f) \textbf{ else } (\langle a > b\rangle \Rightarrow f)$

We can implement the first case as follows

$\langle a \le b\rangle \Rightarrow f$

$=$ Defn of f

$\langle a \le b \Rightarrow a' = \min(a,b)\rangle$

$=$ By (1)

$\langle a \le b \Rightarrow a' = a\rangle$

$\sqsubseteq$ Erasure law

$\textbf{skip}$

The second case is implemented by

$$
\begin{aligned}
& \langle a > b \rangle \Rightarrow f \\
= \quad & \text{Defn of } f \\
& \langle \underline{a > b} \Rightarrow a' = \min(a, b) \rangle \\
\sqsubseteq \quad & \text{Strengthening} \\
& \left\langle a \geq b \Rightarrow a' = \underline{\min(a, b)} \right\rangle \\
= \quad & (2) \\
& \langle a \geq b \Rightarrow a' = b \rangle \\
\sqsubseteq \quad & \text{Erasure law} \\
& a := b
\end{aligned}
$$

Now we have

$$
\begin{aligned}
& f \\
= \quad & \text{Alternation law} \\
& \textbf{if } a \leq b \textbf{ then } \underline{(\langle a \leq b \rangle \Rightarrow f)} \textbf{ else } \underline{(\langle a > b \rangle \Rightarrow f)} \\
\sqsubseteq \quad & \text{Above results} \\
& \textbf{if } a \leq b \textbf{ then skip else } a := b
\end{aligned}
$$

While law (incomplete version)

One property of the while loop is the following. Let

$$w = \textbf{while } \mathcal{A} \textbf{ do } h$$

then

$$w = \textbf{if } \mathcal{A} \textbf{ then } (h; w) \textbf{ else skip}$$

While law (incomplete version): For any g, h, and $\mathcal{A}$, such that ..., if

$$g \sqsubseteq \textbf{if } \mathcal{A} \textbf{ then } (h; g) \textbf{ else skip} \text{ ,}$$

then

$$g \sqsubseteq \textbf{while } \mathcal{A} \textbf{ do } h$$

[Later we will complete this law (fill in the "...") with additional conditions that ensure it is valid. In the mean time we will blithely ignore the "such that ...".]

Summation of an array

For this problem, we calculate the sum of all the elements in an array of integers a of size n (a natural number)

$$f = \left\langle s' = \sum_{k \in \{0,..n\}} a(k) \right\rangle$$

We'll assume a natural number variable i is in the state space.

The strategy is to *find a generalization* of the problem g that can serve as the specification of a loop:

$$\begin{array}{ll} & f \\ \sqsubseteq & \text{Substitution law} \\ & i, s := 0, 0 \,;\, g \end{array}$$

where

$$g = \left\langle i \le n \Rightarrow s' = s + \sum_{k \in \{i,..n\}} a(k) \right\rangle$$

Now the problem remaining is to derive a program for g.

In the case where $i = n$ the problem is easy to solve

$$\begin{array}{l} \quad g \\ \sqsubseteq \\ \quad\quad \mathbf{if}\ i \ne n \\ \quad\quad \mathbf{then}\ \langle i \ne n \rangle \Rightarrow g \\ \quad\quad \mathbf{else}\ \langle i = n \rangle \Rightarrow g \end{array}$$

Tackling the second problem first we have

$$\left\langle i = n \Rightarrow \left(i \le n \Rightarrow s' = s + \sum_{k \in \{i,..n\}} a(k) \right) \right\rangle$$

$=$ One point law

$$\left\langle i = n \Rightarrow \left(\underline{n \le n} \Rightarrow s' = s + \sum_{k \in \{i,..n\}} a(k) \right) \right\rangle$$

$=$ Since $n \le n$ is true and $\mathfrak{true} \Rightarrow p$ is p

$$\left\langle i = n \Rightarrow s' = s + \sum_{k \in \underline{\{n,..n\}}} a(k) \right\rangle$$

$=$ Since $\{n, ..n\} = \emptyset$

$$\left\langle i = n \Rightarrow s' = s + \underline{\sum_{k \in \emptyset} a(k)} \right\rangle$$

$=$ The sum over an empty set is 0

$$\langle i = n \Rightarrow s' = s \rangle$$

$\sqsubseteq$ Erasure law

$\mathbf{skip}$

In the second case

$$\left\langle i \neq n \Rightarrow \left(i \leq n \Rightarrow s' = s + \sum_{k \in \{i,..n\}} a(k) \right) \right\rangle$$

$=$ Shunting

$$\left\langle \underline{i \neq n \wedge i \leq n} \Rightarrow s' = s + \sum_{k \in \{i,..n\}} a(k) \right\rangle$$

$=$ Simplify

$$\left\langle i < n \Rightarrow s' = s + \sum_{k \in \underline{\{i,..n\}}} a(k) \right\rangle$$

$=$ If $i < n$ we can rewrite $\{i, ..n\}$ as $\{i\} \cup \{i+1, ..n\}$

$$\left\langle i < n \Rightarrow s' = s + \underline{\sum_{k \in \{i\} \cup \{i+1,..n\}} a(k)} \right\rangle$$

$=$ Split the summation

$$\left\langle \underline{i < n} \Rightarrow s' = s + a(i) + \sum_{k \in \{i+1,..n\}} a(k) \right\rangle$$

$=$ Rewrite the antecedant

$$\left\langle \underline{i + 1} \leq n \Rightarrow s' = \underline{s + a(i)} + \sum_{k \in \{\underline{i+1},..n\}} a(k) \right\rangle$$

$=$ Substitution law

$$i, s := i + 1, s + a(i) \,;\, g$$

Putting these results together (with monotonicity) we get that

$$
\begin{array}{ll}
 & g \\
\sqsubseteq & \\
 & \mathbf{if}\ i \neq n \\
 & \mathbf{then}\ \langle i \neq n \rangle \Rightarrow g \\
 & \mathbf{else}\ \langle i = n \rangle \Rightarrow g \\
\sqsubseteq & \text{Above calculations} \\
 & \mathbf{if}\ i \neq n \\
 & \mathbf{then}\ (i, s := i + 1, s + a(i); g) \\
 & \mathbf{else\ skip}
\end{array}
$$

Now we apply the while law

$$g \sqsubseteq \mathbf{while}\ i \neq n\ \mathbf{do}\ i, s := i + 1, s + a(i)$$

and thus (by monotonicity)

$$
\begin{array}{ll}
f \sqsubseteq & i, s := 0, 0; \\
 & \mathbf{while}\ i \neq n\ \mathbf{do} \\
 & \qquad i, s := i + 1, s + a(i)
\end{array}
$$

Greatest Common Denominator

$a \mid b$ iff natural number a divides natural number b. I.e. there exists a $q \in \mathbb{N}$ such that $aq = b$

The **greatest common divisor** of two natural numbers a and b is a natural number $\gcd(a, b)$ with the following properties.

$\gcd(a, b) \mid a$, for all natural numbers a, b

$\gcd(a, b) \mid b$, for all natural numbers a, b

if $c \mid a$ and $c \mid b$ then $c \mid \gcd(a, b)$, for all natural numbers a, b, c

From these properties we can derive the following facts (proof left as exercise)

$$\gcd(a, 0) = a, \quad \text{for all natural numbers } a\text{, where } a \neq 0 \tag{3}$$

$$\gcd(a, b) = \gcd(b, a \bmod b), \quad \text{for all natural numbers } a, b \text{ where } b \neq 0 \tag{4}$$

$$g = \langle a \neq 0 \lor b \neq 0 \Rightarrow a' = \gcd(a, b) \rangle$$

$$\begin{array}{ll} & g \\ = & \text{Alternation} \\ & \quad \mathbf{if}\ b \neq 0 \\ & \quad \mathbf{then}\ \langle b \neq 0 \rangle \Rightarrow g \\ & \quad \mathbf{else}\ \langle b = 0 \rangle \Rightarrow g \end{array}$$

In the second case we have (after shunting)

$$\langle b = 0 \wedge (a \neq 0 \vee b \neq 0) \Rightarrow a' = \gcd(a, b)\rangle$$
$=$ One point and identity law for $\vee$
$$\langle b = 0 \wedge a \neq 0 \Rightarrow a' = \gcd(a, 0)\rangle$$
$=$ Fact (3)
$$\langle b = 0 \wedge a \neq 0 \Rightarrow a' = a\rangle$$
$\sqsubseteq$ Erasure law for **skip**
$$\mathbf{skip}$$

In the first case we have (after shunting)

$$\langle b \neq 0 \wedge (a \neq 0 \vee b \neq 0) \Rightarrow a' = \gcd(a, b)\rangle$$
$=$ Domination law for $\vee$
$$\langle b \neq 0 \wedge \mathfrak{true} \Rightarrow a' = \gcd(a, b)\rangle$$
$=$ Identity law for $\wedge$
$$\langle b \neq 0 \Rightarrow a' = \gcd(a, b)\rangle$$
$=$ Fact (4)
$$\langle b \neq 0 \Rightarrow a' = \gcd(b, a \bmod b)\rangle$$
$\sqsubseteq$ Strengthening (by weakening the antecedent)
$$\langle b \neq 0 \vee a \bmod b \neq 0 \Rightarrow a' = \gcd(b, a \bmod b)\rangle$$
$=$ Substitution law
$$a, b := b, a \bmod b \; ; g$$

Now putting the two cases together we get

$$g$$
$$\sqsubseteq$$
$$\begin{array}{l} \mathbf{if}\ b \neq 0 \\ \mathbf{then}\ \ a, b := b, a \bmod b \; ; g \\ \mathbf{else\ skip} \end{array}$$

So by the while loop law we have

$$g \sqsubseteq \textbf{while } b \neq 0 \textbf{ do } a, b := b, a \bmod b$$