THEORY OF PROGRAMMING

and its application to the design of correct and efficient computer programs.

A COURSE OF LECTURES

given by Professor C.A.R. Hoare

Professor of Computation at Oxford University

BASED ON THE TEXT

a discipline of programming

Edsger W. Dijkstra

Prentice Hall, 1976

BASIC DEFINITIONS

A number is denoted by a string of digits

e.g. 0 3 03 703

A variable is denoted by a string of letters

e.g. X Y ALPHA QUOT

A machine state is a finite mapping between variables and values.

e.g. \[
\begin{array}{cc|c|c|}
\text{X} & \text{Y} & \text{REM} & \text{QUOT} \\
\hline
37 & 7 & & 0 \\
\end{array}
\]

m(X) = 37, m(Y) = 7, m(QUOT) = 0
m(REM) is undefined. m is also undefined for all other variables.
A command $C$ is a relation between machine states, i.e. the states of the machine before and after the command is obeyed. (we define a relation as a set of ordered pairs $(m, m') \in C$).

The **skip** command is defined:

\[
\text{skip} = \{ (m, m') \mid m = m' \}
\]

i.e. the identity relation.

This command is obeyed by doing nothing!

The **abort** command is defined

\[
\text{abort} = \emptyset
\]

i.e. the empty relation.

A machine which attempts to obey this command will simply fail (break)!

An expression is constructed from variables, values, operators, and brackets.

**e.g.** $X \ 17 \ X + 17 \ Y(X - 3)$

Given machine state $m$ and expression $e$, we define $m^*(e)$ - as the value taken by $e$, when evaluated in machine state $m$; i.e. when its variables are replaced by the values given by $m$.

**e.g.** if $m(X) = 3$ then $m^*(X + 17) = 20$

\[
m^*(17) = 17
\]

if $m(X)$ is not defined, then nor is

\[
m^*(Y(X + 3))
\]
An assignment command takes the form

\[ x := e \quad (x \text{ becomes } y) \]

where \( x \) is a variable
and \( e \) is an expression

e.g. \( Y := 17 \quad Y := X - 1 \quad X := X + 1 \).

\[ x := e \triangleq \{ (m,m') \mid m'(x) = m^*(e) \} \]
\[ \forall y \not= x \quad m'(y) = m(y) \]

It is obeyed by evaluating \( e \) in the initial machine state \( m \), and then changing \( m(x) \) to have this value instead of its old one. If \( e \) is undefined in \( m \), the machine breaks.

The composition of commands \( c1 \) and \( c2 \) is:

\[ c1; c2 \triangleq \{ (m,m') \mid \exists m'' \quad (m,m'') \subseteq c1 \land (m'',m') \subseteq c2 \} \]

It is obeyed by first obeying \( c1 \) and then obeying \( c2 \). \( m'' \) is the final machine state of \( c1 \) and the initial machine state of \( c2 \).

Theorem. \( c1;(c2; c3) = (c1; c2); c3 \)

The associativity of \( ; \) will justify omission of brackets.

Theorem. \( \text{skip}; c = c; \text{skip} = c \)
\[ \text{abort}; c = c; \text{abort} = \text{abort} \]

Compare: \( 1^*c = c^*1 = c \)
\( 0^*c = c^*0 = 0 \)
A condition \( b \) is an expression which is either true \( \top \) or false \( \bot \).

\[
e.g. \quad X \quad 0 \quad X = Y \quad \top \quad \bot
\]

It defines a "command"

\[
\left\{ (m, m') \mid m'(b) = \top \text{ and } m' = m \right\}
\]

i.e. the identity relation restricted to those machine states in which \( b \) is true.

It is obeyed by evaluating \( b \); if this is true, skip; otherwise abort.

A conditional command is defined

\[
\text{if } b \rightarrow \text{cl}, \text{c2 fi } = b; \text{ cl } \cup \overline{b}; \text{ c2}
\]

It is obeyed by first evaluating \( b \); if the value is true, cl is obeyed and c2 omitted. If the value is false, c2 is obeyed and cl omitted.

\[
\text{if } b \rightarrow \text{skip, abort fi } = b
\]

Theorems

\[
\text{if } b \rightarrow \text{cl}, \text{c2 fi } = \text{cl} \rightarrow (\text{cl}; \text{c3}), (\text{c2}; \text{c3}) \text{ fi}
\]

\[
\text{if } b \rightarrow \text{cl}, \text{c2 fi } = \text{ch} \rightarrow \text{c2, cl fi}
\]

\[
\text{if } \bot \rightarrow \text{cl}, \text{c2 fi } = \text{cl fi } \rightarrow \text{c2, cl fi } = \text{cl}
\]

The repetitive command is defined

\[
\text{do } b \rightarrow c \text{ od } = \bigcup_{n=0}^{\infty} c_n \quad \text{(while } b \text{ do } c)\]

where \( c_0 = \overline{b} \)

\[
c_{n+1} = b; \ c_n \left[ \text{u } b \right]
\]

It is obeyed by first evaluating \( b \). If this is false, the task is finished.

If it is true, \( c \) is next obeyed, and then the whole command is repeated.

Theorem. \( \text{do } b \rightarrow c \text{ od } = \text{if } b \rightarrow (c; \text{do } b \rightarrow c \text{ od}), \text{skip fi} \)

\[
\text{do } \bot \rightarrow c \text{ od } = \text{skip}
\]

\[
\text{do } \top \rightarrow c \text{ od } = \text{abort}
\]

\[
\text{do } b \rightarrow c \text{ od } = \overline{b}, b; c; \overline{b}, b; c; b; c; \overline{b}, ...
\]

\[
c_n \in c_{n+1}
\]

\[
c_n = b; c; b; c; \overline{b} \quad \leq n \text{ times.}
\]
EXAMPLE

\[ x := X; \ y := Y; \]
\[ \text{do } x \neq y \rightarrow \text{if } x < y \rightarrow y := y - x, \ x := x - y \]
\[ \text{fi} \]
\[ \text{od} \]
\[ = x := X; \ y := Y; \]
\[ \{ x = y \cup (x \neq y; x < y; \ y := y - x; \ x = y) \]
\[ \cup (x \neq y; \overline{x < y}; \ x := x - y; \ x = y) \]
\[ \ldots \]
\[ \cup x \neq y; x < y; \ y := y - x; \ x \neq y; \overline{x < y}; \ x := x - y; \ x = y \]
\[ \ldots \]

**EXECUTION TRACES**

<table>
<thead>
<tr>
<th></th>
<th>X</th>
<th>Y</th>
<th></th>
<th>X</th>
<th>Y</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>initial m/s</strong></td>
<td>111</td>
<td>259</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>x := X</td>
<td>111</td>
<td>259</td>
<td></td>
<td>111</td>
<td></td>
</tr>
<tr>
<td>y := Y</td>
<td>&quot;</td>
<td>&quot;</td>
<td>&quot;</td>
<td>259</td>
<td></td>
</tr>
<tr>
<td>x \neq y</td>
<td>√</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>x &lt; y</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>y := y - x</td>
<td></td>
<td></td>
<td></td>
<td>14.8</td>
<td></td>
</tr>
<tr>
<td>x \neq y; x &lt; y</td>
<td>√</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>y := y - x</td>
<td></td>
<td></td>
<td></td>
<td>37</td>
<td></td>
</tr>
<tr>
<td>x \neq y; \overline{x &lt; y}</td>
<td>√</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>x := x - y</td>
<td></td>
<td></td>
<td></td>
<td>74</td>
<td></td>
</tr>
<tr>
<td>x \neq y; \overline{x &lt; y}</td>
<td>√</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>x := x - y</td>
<td></td>
<td></td>
<td></td>
<td>37</td>
<td></td>
</tr>
<tr>
<td>x = y</td>
<td>√</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>final m/s.</strong></td>
<td>111</td>
<td>259</td>
<td>37</td>
<td>37</td>
<td></td>
</tr>
</tbody>
</table>
A. Obey the following commands, giving their execution traces.

1. \[ y := 0; \ x := 3; \ \textbf{do} \ y + 1 \ < \ x \ \rightarrow \ y := y + 1 \ \textbf{od} \]
2. \[ y := 2; \ x := 3; \ \textbf{do} \ y > 0 \ \rightarrow \ x := x + 1; \ y := y - 1 \ \textbf{od} \]
3. \[ y := 2; \ x := 3; \ s := 0; \]
   \[ \textbf{do} \ y > 0 \ \rightarrow \ y := y - 1; \ s := s + x \ \textbf{od} \]
4. \[ y := 3; \ x := 7; \ q := 0; \]
   \[ \textbf{do} \ x > y \ \rightarrow \ x := x - y; \ q := q + 1 \ \textbf{od} \]

B. In each example above, replace the repetitive command by a pair of assignments which would have the same effect for any initial assignment to \( x, y, q, s \).

\textit{e.g.} the answer to example 4 is \[ q := q + x \div y; \ x := x \mod y \]

\textbf{WEAKEST PRECONDITIONS}

\( c \ \textbf{achieves} \ r \ \text{is defined:} \)

\[ c \ \textbf{achieves} \ r = \left\{ (m, m') \mid \exists m' \ (m, m') \in c \ \& \ m'^*(r) = \mathbb{Z} \right\} \]

i.e. the inverse image of \( r \) under \( c \). It is the condition satisfied by exactly those initial machine states in which \( c \) can successfully be obeyed, and can end in a machine state satisfying \( r \).

Theorem \( b \ \land \ r = b \land r \)

where \( b \) is a condition.

Proof. LHS = \( \left\{ (m, m') \mid (m, m') \in b \ \& \ m'^*(r) = \mathbb{Z} \right\} \)

= \( \left\{ (m, m) \in b \ \& \ m(r) = \mathbb{Z} \right\} \)

= \( \left\{ (m, m) \in B \ \& \ (m, m) \in \mathbb{Z} \right\} \)

= \( b \land r \)

We usually identify a condition with the set of machine state pairs in which the condition is true.
\[ T \text{ a } r = r \]

Strict \quad c \text{ a } F = F, \quad F \text{ a } r = F

Distributive \quad c \text{ a } (\bigcup \limits_n r_n) = \bigcup \limits_n (c \text{ a } r_n)

(Additive) \quad (\bigcup \limits_n c_n) \text{ a } r = \bigcup \limits_n (c_n \text{ a } r)

A command \( c \) is **deterministic** if for each initial state \( m \),
there is at most one final state \( m' \) such that \((m, m') \in c\). All
commands defined so far are deterministic.

If \( c \) is deterministic.

Multiplicative \quad c \text{ a } \bigcap \limits_n r_n = \bigcap \limits_n (c \text{ a } r_n)

This is **not** true if \( c \) is nondeterministic,

\[ e.g. \quad c = x := 0 \cup x := 1 \]

then \quad c \text{ a } x = 0 = T \quad \& \quad x = 1 = T

\[ c \text{ a } (x = 0 \& x = 1) = c \text{ a } F = F \]
Theorem of assignment

\[ x := e \triangleleft b(x) = b(e) \]

where \( b(e) \) is the result of replacing all occurrences of \( x \) in \( b(x) \) by \( e \).

\[ (X := 37 \triangleleft X > 12) \equiv 37 > 12 \equiv T \]
\[ (X := Y^5 \triangleleft X > 12) \equiv Y^5 > 12 \equiv Y > 4 \]
\[ (X := X - 1 \triangleleft X > 12) \equiv (X - 1 > 12) \equiv (X > 13) \]

Proof. The value of \( x \) after the assignment is by definition equal to the value of \( e \) before the assignment. So \( b(x) \) is true (of \( x \)) after the assignment if any only if \( b(e) \) is true (of the value of \( e \)) before the assignment. The values of all other variables of \( b \) remain unchanged by the assignment.

Theorem of composition

\[ (c_1; c_2) \not\Delta r = c_1 \not\Delta (c_2 \not\Delta r) \]
\[ (x := X; y := Y) \not\Delta (\text{GCD}(x, y) = \text{GCD}(X, Y)) \]
\[ \equiv (x := X \not\Delta \text{GCD}(x, y) = \text{GCD}(X, Y)) \]
\[ \equiv \text{GCD}(X, Y) = \text{GCD}(X, Y) \]
\[ \equiv X > 0 \& Y > 0 \]

Proof. \( c_1; c_2 \) arrives at a state satisfying \( r \) if \( c_1 \) arrives at a state from which \( c_2 \) achieves \( r \), and conversely.

\[ \text{LHS} = \exists m \exists m' \exists m'' (m, m') \in c_1 \not\Delta (m, m'') \in c_2 \not\Delta m' (m'') = T \]

\[ = \exists m \exists m' \exists m'' (m, m'') \in c_1 \& (m', m'') \in c_2 \not\Delta (m'') = T \]

\[ = \exists m \exists m' \exists m'' (m, m'') \in c_1 \& \exists (m'', m') \in c_2 \not\Delta (m') = T \]

\[ = \exists m \exists m'' (m, m'') \in c_1 \& m'' \not\Delta (m, m') = \text{RHS} \]
Theorem

\[ \text{if } b \Rightarrow c_1, c_2 \ a r = \text{if } b \Rightarrow (c_1 \ a r), (c_2 \ a r) \ a r \]
\[ = b \cap (c_1 \ a r) \cup \overline{b} \cap (c_2 \ a r) \]
\[ \text{e.g.} \quad \text{if } x \prec y \Rightarrow y_1 = y - x, x := x - y_1 \ a \ GCD(x, y) = K \]
\[ \quad \text{if } x \prec y \Rightarrow GCD(x, y - x) = K, GCD(x - y, y) = K \ a r \]
\[ \quad GCD(x, y) = K \ & \ x \neq y \]

Proof. LHS = (b; c_1 \ a r; b; c_2) \ a r
\[ = (b; c_1) \ a r \cup (\overline{b}; c_2) \ a r \]
\[ = b \ a (c_1 \ a r) \cup \overline{b} \ a (c_2 \ a r) \]
\[ = b \cap (c_1 \ a r) \cup \overline{b} \cap (c_2 \ a r) \]
\[ = \text{RHS}. \]

Theorem of repetition.

\[ \text{do } b \Rightarrow c \ a r = \bigcup_{n=0}^{\infty} P_n \]

where \( P_0 = \overline{b} \cap r \)
\[ P_{n+1} = b \cap c_n \ a r \cup (\overline{b}) \cap r \]
\[ \text{e.g.} \quad \text{do } y + 1 \prec x \Rightarrow y := y + 1 \ a y = x - 1 = \bigcup_{n=0}^{\infty} P_n \]

where \( P_0 \equiv y + 1 \prec x \cap y = x - 1 \equiv y = x - 1 \)
\[ P_1 \equiv y + 1 \prec x \cap (y := y + 1 \ a P_0) \cup y = x - 1 \]
\[ \equiv y = x - 2 \lor y = x - 1 \]
\[ \equiv x - 1 - 1 \leq y < x \]
\[ \vdots \]
\[ P_n \equiv x - n - 1 \leq y < x \]
\[ \vdots \]
\[ \text{where } \bigcup_{n=0}^{\infty} P_n = \exists n \quad x - n - 1 \leq y < x \]
\[ \equiv y < x \]

LHS = \( \bigcup_n c_n \ a r \) where \( c_0 = \overline{b}, \ c_{n+1} = b; c_n \cup \overline{b} \)
\[ = \bigcup_n (c_n \ a r) \) where \( c_0 \ a r = \overline{b} \cap r, c_{n+1} \ a r = (b; c_n \ a r) \cup \overline{b} \cap r \]
\[ = b \cap (c_n \ a r) \cup \overline{b} \cap r \]
\[ = \bigcup_n (P_n) \]
EXERCISES

1. Derive and simplify the following preconditions:

a. \((y := 0; \text{do } y \leftarrow y + 1 \land x \rightarrow y := y + 1 \land d) \land (y = x - 1)\)

b. \((x := X; y := Y; \text{do } y \leftarrow 0 \rightarrow x := x + 1; y := y - 1 \land d \land x = X + Y + Z)\)

c. \((y := Y; s := 0; \text{do } y \leftarrow 0 \rightarrow y := y - 1; s := s + y \land d \land s = X^y)\)

d. \((x := X; q := 0; \text{do } x \leftarrow y \rightarrow x := x - y; q := q + 1 \land d \land x = q^y \land x < y)\)

(all variables are nonnegative integers).

We can now solve problems of the form:

\[ ? \land S \land r \land \] (19)

given command \(S\) and postcondition \(r\),

But programmers must solve a different problem:

given postcondition \(r\) and precondition \(p\),

\[ p \Rightarrow ? \land r \land \] (20)

e.g. using only \(+1\) and \(<\), write a command \(c\) which does not change \(x\),

and which satisfies:

\[ 0 < x \Rightarrow c \land y = x - 1 \]

This is usually more difficult! We shall need some more theory.

TRIVIALITIES

If \(p\) and \(b\) are conditions

\[ p \land b = p; b = p \land b = b \land p = b \land p \]

\(c; (c_1 \cup c_2) = c; c_1 \cup c; c_2\)

\((c_1 \cup c_2); c = c_1; c \cup c_2; c\)

\(p \land (b; c \land r) = p \land (b; c \land r) = (p; b; c) \land r\)

\[ = (p \land b; c) \land r = b \land (p \land b; (c \land r)) \]

if \(p \land b \Rightarrow c_1 \land r\)

and \(p \land b \Rightarrow c_2 \land r\)

then \(p \Rightarrow \text{if } b \rightarrow c_1, c_2, \text{ } \land \land r\)
THEOREM OF INVARIANCE

Let $c$ be defined as $b \rightarrow cl$; $od$ (for deterministic $cl$)

and let $b \cap p \subseteq cl \cap p$

(i.e., $p$ is an invariant of $c$)

then $p \cap (c \cap T) \subseteq c \cap (b \cap p)$

Proof: define $c_0 = \overline{b}, \; c_{n+1} = b; c; \; c_n \cup \overline{b}$

so $c = \bigcup_n c_n$

LHS = $p \cap (\bigcup_n c_n) \cap T = p \cap (\bigcup_n (c_n \cap T))$

distribution

$= \bigcup_n p \cap (c_n \cap T) \subseteq \bigcup_n (c_n \cap (\overline{b} \cap p))$

(by lemma)

$= (\bigcup_n c_n) \cap (\overline{b} \cap p) = \text{RHS}$

LEMMA

(1)

If $c$ is deterministic

and $p \cap b \subseteq c \cap p$

and $c_0 = \overline{b}$ and $c_{n+1} = b; c; c_n \cup \overline{b}$

then $\forall n \; p \cap (c_n \cap T) \subseteq c_n \cap (\overline{b} \cap p)$

Proof: case $n = 0$: $p \cap (\overline{b} \cap T) = p \cap \overline{b} = \overline{b} \cap (p \cap \overline{b}) = \overline{b} \cap (\overline{b} \cap p)$

$p \cap (c_{n+1} \cap T) = p \cap (b; c; c_n) \cap T \cup p \cap (\overline{b} \cap T)$

by (3)

$\subseteq b \cap (c \cap p) \cap (c \cap (c_n \cap T)) \cup p \cap \overline{b}$

by (2)

$= b \cap (c \cap (p \cap (c_n \cap T))) \cup p \cap \overline{b}$

by (1)

$\subseteq b \cap (c \cap (c_n \cap (\overline{b} \cap p))) \cap \overline{b} \cap (\overline{b} \cap p)$

by (4)

$= (b; c; c_n) \cap (\overline{b} \cap p) \cup \overline{b} \cap (\overline{b} \cap p)$

$= ((b; c; c_n) \cup \overline{b}) \cap (\overline{b} \cap p)$

(3)
VARIANT EXPRESSIONS

Let \( t \) be an expression, (always defined)
\[
c \text{ dec } t \overset{\text{df}}{=} (k := t; c) \text{ a } (0 \leq t < k)
\]
(\text{where } k \text{ is a fresh variable}).

i.e. the weakest precondition under which the command \( c \) decreases value of \( t \).

Theorem 2. Let \( c = \text{ do } b \rightarrow c_1 \text{ od } \) (deterministic)
\[
p \land b \subseteq c_1 \land p
\]
\[
p \land b \subseteq c_1 \text{ dec } t
\]
then \( p \subseteq c \land \neg(b \land p) \)

Proof define \( c_0 = \overline{b} \), \( c_{n+1} = b; c; c_n \cup \overline{b} \)
by Lemma 2 \( p \land (t \leq n) \subseteq c_n \land T \)

\[
\therefore p = \bigcup_{n} (p \land (t \leq n)) \subseteq \bigcup_{n} (c_n \land T) = p \land c \land T
\]
\[
= c \land \neg(b \land p)
\]
by theorem (1)

\text{LEMMA}

If \( p \land b \subseteq (k := t; c) \land (p \land t < k) \) for fresh \( k \)
and \( c_0 = \overline{b} \), \( c_{n+1} = b; c; c_n \cup \overline{b} \)
then \( \forall n \ p \land t \leq n \subseteq c_n \land T \)

Proof. \( p \land b \subseteq (k := t; c) \land t < k \subseteq t > 0 \)
\[
\therefore p \land t \leq 0 \subseteq \overline{b} = c_0.
\]

induction step.
\[
p \land t \leq n+1 = p \land t = n+1
\]
\[
\subseteq p \land b \land t = n+1 \cup \overline{b}
\]
\[
\subseteq b \land t = n+1 \land (k := t; c) \land (p \land t < k) \cup \overline{b}
\]
\[
\subseteq b \land c \land (p \land t < n) \cup \overline{b}
\]
\[
\subseteq b \land c \land (c_n \land T) \cup \overline{b} \land T
\]
\[
= c_{n+1} \land T
\]
\[
= c_{n+1} \land T
\]
since \( c_n \subseteq c_{n+1} \)
EXAMPLE 1

Using only $\prec$, successor as operators, find $c$ s.t.

$$0 \prec x \implies c \land y = x - 1 \& c \text{ changes only } y.$$  

Solution: reformulate postcondition as

$$\overline{\bar{y} + 1 \prec x \cap y \prec x} = \bar{b} \land p$$

and find $c_1, t$ s.t.

$$y + 1 \prec x \cap y \prec x \implies c_1 \land \text{dec } t$$

$$c_1 \land y \prec x$$

try $c_1 = y := y + 1$

$$t = x - y$$

check $y + 1 \prec x \implies (k := x - y; y := y + 1) \land (x - y \prec k \cap y \prec x)$

$$\text{RHS } = x - y - 1 \prec x - y$$

... by theorem 2.

$$y \prec x \implies \text{do } y + 1 \prec x \implies y := y + 1 \land \text{a } y = x - 1$$

It remains to find $c_0$ s.t.

$$0 \prec x \implies c_0 \land y \prec x.$$  

Using theorem of assignment, $c_0$ is $y := 0$.

EXAMPLE 2

Using only $\prec$, successor, and predecessor as operators find $c$ s.t.

$$c \land x = X + Y \text{ where } c \text{ changes only } x \text{ and } y.$$  

Solution: reformulate postcondition as

$$\overline{b \cap p} = 0 \prec y \cap x + y = X + Y$$

check that $x := X; y := Y \land p$.  

... (1)

we need to find $c_1, t$ s.t.

$$0 \prec (y \cap x + y = X + Y \implies (k := t; c_1) \land (t \prec k \cap p)$$

an obvious choice is $t = y$.

$$c_1 = c_2; y := y - 1$$

... (2)
where \( 0 \langle y \cap P \Rightarrow c2 \ a \ (y := y - 1 \ a \ x + y = x + y) \)

\[ i.e. \ 0 \langle y \cap (x + y = x + y) \Rightarrow c2 \ a \ (x + y - 1 = x + y) \]

obviously \( c2 = x := x + 1 \)  \( \ldots \) (5)

collecting (1), (2), (3), we get:

\[ x := X; \ y := Y; \ do \ 0 \langle y \Rightarrow x := x + 1; \ y := y - 1 \ od \]

**EXAMPLE 3**

Using only \( \langle, +, - \), find \( c \) s.t.

\( Y > 0 \Rightarrow c \ a \ q = X - Y \cap r = X \mod Y. \ c \) changes only \( q \) and \( r \)

Reformulate postcondition as

\[ Y \langle r \cap X = q * Y + r \quad (= b \cap p) \]

(1)

\[ Y \rangle r \Rightarrow r := r - Y \text{ deco } r \]

(2)

we need to find c1 changing only \( q \), s.t.

\[ b \cap p \Rightarrow c1 \ a \ (X = q * Y + r - Y) \]

this is solved by \( c1 = q := q + 1. \)

\[ c = q := 0; \ r := X; \ do \ Y \langle r \Rightarrow q := q + 1; \ r := r - y \ od \]

Exercise: using only \( \langle, +, - \), find \( c \) s.t.

\( c \ a \ s = X * Y \), and \( c \) changes only \( s \) and \( y \).