Construction of Linear Invariants

An integer variable y is called linear if the modification of variable y in each statement has the form $y' = y + c$ for some constant c (possibly 0).

We are looking for invariants of the form

$$\underbrace{\sum_{i=1}^{r} a_i \cdot y_i}_{\text{Body}} + \underbrace{\sum_{\ell \in \mathcal{L}} b_\ell \cdot at\_\ell}_{\text{Compensation Expression}} = \underbrace{K}_{\text{Right Constant}},$$

where $y_1, \ldots, y_r$ are linear variables, a_i, b_j, and K are integer constants.

For a linear variable y and statement $\ell : S$, we define the increment $\Delta(y, \ell) = c$ if the execution of statement S adds the constant c to y.

For a location predicate ℓ_j and statement $\ell_i : S$, we define

$$\Delta(at\_\ell_j, \ell_i) = \begin{cases} +1 & i = j-1 \\ -1 & i = j \\ 0 & i \notin \{j, j-1\} \end{cases}$$

For an expression E and a sequence of consecutive statements $\ell_i : S_i; \ldots; \ell_j : S_j$, we define the accumulated increment

$$\Delta(E, \ell_{i..j}) = \Delta(E, \ell_i) + \cdots + \Delta(E, \ell_j)$$

Linear Invariants Continued

To simplify the presentation, assume that each process has the following structure

$$P_j :: \ell_0 : \textbf{loop forever do } [\ell_1 : S_1; \ldots; \ell_k : S_k]$$

and that there are no nested loops or conditional statements.

Then, for an expression E, we define the process-accumulated increment to be $\Delta(E, P_j) = \Delta(E, \ell_{0..k})$.

Necessary Conditions

Assume that

$$\sum_{i=1}^{r} a_i \cdot y_i + \sum_{\ell \in \mathcal{L}} b_\ell \cdot at\_\ell = K$$

is an invariant of a program consisting of the parallel processes $P_1, \ldots, P_n$. Applying $\Delta(\cdot, P_j)$ to both sides of this equality, we obtain

$$\sum_{i=1}^{r} a_i \cdot \Delta(y_i, P_j) + \sum_{\ell \in \mathcal{L}} b_\ell \cdot \Delta(at\_\ell, P_j) = 0$$

We show now that $\Delta(at\_\ell_i, P_j) = 0$ for all ℓ_i and P_j. If $\ell_i \notin \mathcal{L}_j$, then no statement in P_j can modify ℓ_i. If $\ell_i \in \mathcal{L}_j$, then $\Delta(at\_\ell_i, P_j)$ sums together $\Delta(at\_\ell_i, \ell_{i-1}) = +1$ and $\Delta(at\_\ell_i, \ell_i) = -1$, yielding 0.

We conclude that the coefficients a_i must satisfy the equations

$$\sum_{i=1}^{r} a_i \cdot \Delta(y_i, P_j) = 0$$

for every $j = 1, \ldots, n$.

Computing the Bodies

Solve and find a basis of independent solution to the set of linear equations

$$\sum_{i=1}^{r} a_i \cdot \Delta(y_i, P_j) = 0.$$

Any such solution provides a possible body.

Example: Mutual Exclusion with Two Semaphores

Consider program TWO-SEM:

$$y_1, y_2 : \textbf{natural initially } y_1 = 1, y_2 = 0$$

$$\left[\begin{array}{l} \ell_0 : \textbf{ loop forever do} \\ \quad \left[\begin{array}{l} \ell_1 : \textbf{ Non-critical} \\ \ell_2 : \textbf{ request } y_1 \\ \ell_3 : \textbf{ Critical} \\ \ell_4 : \textbf{ release } y_2 \end{array}\right] \end{array}\right] \parallel \left[\begin{array}{l} m_0 : \textbf{ loop forever do} \\ \quad \left[\begin{array}{l} m_1 : \textbf{ Non-critical} \\ m_2 : \textbf{ request } y_2 \\ m_3 : \textbf{ Critical} \\ m_4 : \textbf{ release } y_1 \end{array}\right] \end{array}\right]$$

This program has the linear variables y_1, y_2. Their process-accumulated increments $\Delta(y_i, P_j)$ are given by

	y_1	y_2
P_1	-1	$+1$
P_2	$+1$	-1

This gives rise to the following set of equations:

$$\begin{array}{rcrcl} -a_1 & + & a_2 & = & 0 \\ a_1 & - & a_2 & = & 0 \end{array}$$

whose solution basis can be given by $a_1 = a_2 = 1$. Thus, any linear invariant for this program will be of the form

$$y_1 + y_2 + \cdots = K$$

Computing the Compensation Expressions

Let ℓ_i^j be a location within process P_j. Assuming that we have already computed a body $B = \sum_{i=1}^{r} a_i \cdot y_i$, then the coefficient b_i is given by

$$b_i = -\Delta(B, \ell^j_{0..i-1})$$

Going back to program TWO-SEM with the body $B = y_1 + y_2$, we compute the accumulated increments $\Delta(y_1 + y_2, \ell^j_{0..i-1})$ as follows:

	$\Delta(y_1 + y_2, \ell_{0..i-1})$	$\Delta(y_1 + y_2, m_{0..i-1})$
$i = 1$	0	0
$i = 2$	0	0
$i = 3$	-1	-1
$i = 4$	-1	-1

It follows that

$$b(\ell_0) = b(m_0) = b(\ell_1) = b(m_1) = b(\ell_2) = b(m_2) = 0$$
$$b(\ell_3) = b(m_3) = b(\ell_4) = b(m_4) = 1$$

Thus, the left-hand side of the linear invariant for program TWO-SEM has the form

$$y_1 + y_2 + at\_\ell_{3,4} + at\_m_{3,4}$$

Computing the Right-Hand-Side Constant

Assume that the initial values of the linear variables $y_1, \ldots, y_r$ are given, respectively, by $\eta_1 \ldots, \eta_r$. Then, the right-hand-side constant K is given by

$$K = \sum_{i=1}^{r} a_i \cdot \eta_i$$

Thus, for program TWO-SEM, the full linear invariant is given by

$$y_1 + y_2 + at\_\ell_{3,4} + at\_m_{3,4} = 1$$

since the initial values are $\eta_1 = 1$ and $\eta_2 = 0$. This together with the obvious invariants $y_1 \geq 0$ and $y_2 \geq 0$ are sufficient in order to establish mutual exclusion.

Example: Producer-Consumer

Consider the following program PROD-CONS:

local r, ne, nf : **natural where** $r = 1, ne = N, nf = 0$
local L : **list of natural where** $L = ()$

$Prod ::$ [**local** x : **natural**
ℓ_0 : **loop forever do** [ℓ_1 : **Produce** x | ℓ_2 : **request** ne | ℓ_3 : **request** r | ℓ_4 : $L := L \circ x$ | ℓ_5 : **release** r | ℓ_6 : **release** nf]]

$\|$

$Cons ::$ [**local** y : **natural**
m_0 : **loop forever do** [m_1 : **request** nf | m_2 : **request** r | m_3 : $(y, L) := (hd(L), tl(L))$ | m_4 : **release** r | m_5 : **release** ne | m_6 : **Consume** y]]

Process $Prod$ produces values and moves them to process $Cons$ for consumption. The values are transferred via the buffer L. We wish to guarantee that the size of the buffer never exceeds the constant N. For that purpose, we maintain the semaphore ne which counts the number of empty slots within L and the semaphore nf which maintains the number of occupied slots within L. Formally, the requirements are

φ_1 :	$\neg(at\_\ell_4 \ \wedge \ at\_m_3)$	Locations ℓ_4 and m_3 are exclusive.
φ_2 :	$at\_\ell_4 \rightarrow \lvert L \rvert < N$	Never attempt to add a value to a full buffer.
φ_3 :	$at\_m_3 \rightarrow \lvert L \rvert > 0$	Never attempt to dequeue an empty buffer.

Computing Linear Invariants for PROD-CONS

As linear variables we take $\{r, ne, nf, |L|\}$. The process-accumulated increments for these four variables are given by

	$v = r$	$v = ne$	$v = nf$	$v = \|L\|$
$\Delta(v, P_1)$	0	-1	$+1$	$+1$
$\Delta(v, P_2)$	0	$+1$	-1	-1

This gives rise to the following set of equations:

$$\begin{array}{llllll} 0 \cdot a_r & -a_{ne} & +a_{nf} & +a_{|L|} & = & 0 \\ 0 \cdot a_r & +a_{ne} & -a_{nf} & -a_{|L|} & = & 0 \end{array}$$

Since we have 4 variables and 1 independent equation, there is a solution basis containing 3 independent solutions. These can be given as

	a_r	a_{ne}	a_{nf}	$a_{\|L\|}$
$\vec{a}_1$	1	0	0	0
$\vec{a}_2$	0	1	0	1
$\vec{a}_3$	0	0	-1	1

Leading to the bodies:

$$\begin{array}{ll} B_1: & r \\ B_2: & ne + |L| \\ B_3: & -nf + |L| \end{array}$$

Computation Continued

To determine the coefficients b_ℓ, we compute the accumulated increments $\Delta(B_i, \ell_{0..j-1})$ and $\Delta(B_i, \ell_{0..j-1})$ as follows:

	$j:2$	$j:3$	$j:4$	$j:5$	$j:6$
$\Delta(B_1, \ell_{0..j-1})$	0	0	-1	-1	0
$\Delta(B_2, \ell_{0..j-1})$	0	-1	-1	0	0
$\Delta(B_3, \ell_{0..j-1})$	0	0	0	1	1

	$j:2$	$j:3$	$j:4$	$j:5$	$j:6$
$\Delta(B_1, m_{0..j-1})$	0	-1	-1	0	0
$\Delta(B_2, m_{0..j-1})$	0	0	-1	-1	0
$\Delta(B_3, m_{0..j-1})$	1	1	0	0	0

After computing the right-hand-constants, we conclude with the following three invariants:

$$\begin{array}{lll} I_1: & r + at\_\ell_{4,5} + at\_m_{3,4} & = 1 \\ I_2: & ne + |L| + at\_\ell_{3,4} + at\_m_{4,5} & = N \\ I_3: & -nf + |L| - at\_\ell_{5,6} - at\_m_{2,3} & = 0 \end{array}$$

Drawing Conclusions

The three obtained linear invariants

$$\begin{array}{lll} I_1: & r + at\_\ell_{4,5} + at\_m_{3,4} & = & 1 \\ I_2: & ne + |L| + at\_\ell_{3,4} + at\_m_{4,5} & = & N \\ I_3: & -nf + |L| - at\_\ell_{5,6} - at\_m_{2,3} & = & 0 \end{array}$$

imply the main safety properties of program PROD-CONS.

- Property $\varphi_1 : \neg(at\_\ell_4 \wedge at\_m_3)$ follows from I_1, because $at\_\ell_4 = at\_m_3 = 1$ implies $r = -1$ which is impossible.

- From I_2, we obtain

$$|L| \;=\; N - ne - at\_\ell_{3,4} - at\_m_{4,5} \;\leq\; N - at\_\ell_4$$

which implies $\varphi_2 : at\_\ell_4 \rightarrow |L| < N$ since, when $at\_\ell_4 = 1$, $|L| \leq N - 1$.

- From I_3, we obtain

$$|L| \;=\; nf + at\_\ell_{5,6} + at\_m_{2,3} \;\geq\; at\_m_3$$

which implies $\varphi_3 : at\_m_3 \rightarrow |L| > 0$ since, when $at\_m_3 = 1$, $|L| \geq 1$.

Proving Liveness Properties

The main liveness property we will be interested in is specified by the response formula

$$p \Rightarrow \diamondsuit\, q$$

claiming that every p is eventually followed by a q.

Rule ABS-CHAIN

For justice requirements $J_1, \ldots, J_m$,
and assertions $p, q = h_0, h_1, \ldots, h_m$

$$\text{C1.} \quad p \;\Rightarrow\; \bigvee_{j=0}^{m} h_j$$

For $i = 1, \ldots, m$

$$\text{C2.} \quad h_i \wedge \rho \;\Rightarrow\; (h_i' \wedge \neg J_i) \;\vee\; \bigvee_{j<i} h_j'$$

$$\overline{\qquad p \Rightarrow \diamondsuit\, q \qquad}$$

Soundness of Rule ABS-CHAIN

Claim 5. *Rule* ABS-CHAIN *is sound for proving the response property* $p \Rightarrow \Diamond q$.

Proof Assume that the premises of rule ABS-CHAIN are valid. Let $\sigma : s_0, s_1, \ldots$ be a computation of $\mathcal{D}$ and let p hold at position j. We have to show that there exists a position $k \geq j$ such that q holds at position k.

Assume to the contrary, that no position beyond j satisfies q. By premise C1, state s_j must satisfy h_i, for some $i \geq 0$. Since q never holds beyond j, we must have $i > 0$. Let us denote by i_j the index $i > 0$ such that h_i holds at state s_j. By premise C2, the successor state s_{j+1} must also satisfy h_i, for some i, $0 \leq i \leq i_j$. Denote this index by i_{j+1}. By argument similar to the above, $i_{j+1} > 0$. In this way we proceed to establish an infinite sequence of indices $i_j \geq i_{j+1} \geq \cdots$ where, for each $k \geq j$, $i_k > 0$ and $s_k \models h_{i_k}$. Since this is an infinite non-increasing sequence, there must exist an index n, such that $i_n = i_{n+1} = \cdots$.

By premise C2, we can have $i_k = i_{k+1}$ only if $s_{k+1} \models \neg J_{i_k}$. Thus, justice requirement J_{i_n} is never satisfied beyond position n. It follows that σ is not a computation, contrary to our original assumption.

We conclude that there must exists a position $k \geq j$ satisfying q. ⌟

Rule CHAIN for Programs

For the case that the considered FDS is derived from a program, it is possible to present a concrete version of rule CHAIN which utilizes the fact that all justice requirements are derived from transitions (statements). Recall that, in such a system, any justice requirement is the negation of an enabling condition of some transition. In the following, we denote the enabling condition of transition t_i by $En(t_i)$.

Rule CHAIN

For just transitions $t_1, \ldots, t_m$,
and assertions $p, q = h_0, h_1, \ldots, h_m$

C1. $p \Rightarrow \bigvee_{j=0}^{m} h_j$

For $i = 1, \ldots, m$

C2. $h_i \wedge \rho_t \Rightarrow \bigvee_{j \leq i} h'_j$ For every $t \neq t_i$

C3. $h_i \wedge \rho_{t_i} \Rightarrow \bigvee_{j < i} h'_j$

C4. $h_i \Rightarrow En(t_i)$

$p \Rightarrow \Diamond q$

It can be shown that rule CHAIN is a special case of rule ABS-CHAIN.

Apply to BAKERY-2

local y_1, y_2 : **natural initially** $y_1 = y_2 = 0$

$$P_1 :: \left[\begin{array}{l} \ell_0 : \textbf{loop forever do} \\ \left[\begin{array}{l} \ell_1 : \textbf{Non-Critical} \\ \ell_2 : y_1 := y_2 + 1 \\ \ell_3 : \textbf{await } y_2 = 0 \ \vee \ y_1 < y_2 \\ \ell_4 : \textbf{Critical} \\ \ell_5 : y_1 := 0 \end{array}\right] \end{array}\right] \ \| \ P_2 :: \left[\begin{array}{l} m_0 : \textbf{loop forever do} \\ \left[\begin{array}{l} m_1 : \textbf{Non-Critical} \\ m_2 : y_2 := y_1 + 1 \\ m_3 : \textbf{await } y_1 = 0 \ \vee \ y_2 \leq y_1 \\ m_4 : \textbf{Critical} \\ m_5 : y_2 := 0 \end{array}\right] \end{array}\right]$$

The desired response properties for program BAKERY-2 are individual accessibility for the two processes

$$\begin{array}{lcll} \psi_1 & : & at\_\ell_2 & \Rightarrow \ \diamondsuit \ at\_\ell_4, \\ \psi_2 & : & at\_m_2 & \Rightarrow \ \diamondsuit \ at\_m_4 \end{array}$$

Let us present a heuristic by which we can systematically derive the auxiliary constructs required by rule CHAIN, in order to prove property ψ_1. Thus, we consider the case that $p = at\_\ell_2$ and $q = at\_\ell_4$.

Identifying t_1 and h_1

Recalling that the condition $f \wedge \rho_t \rightarrow g$ can be rewritten as $f \rightarrow pre(t, g)$, we can summarize premises C2–C4 for the case $i = 1$ into the single implication

$$Imp(h_0): \quad h \quad \rightarrow \quad \neg h_0 \ \wedge \ En(t) \ \wedge \ pre(t, h_0) \ \wedge \ \bigwedge_{\tau \neq t} pre(\tau, h \vee h_0)$$

where we take $h_0 = q = at\_\ell_4$. The conjunct $\neg h_0$ has been added in order to guarantee that all the h_i's will be exclusive.

$Imp(h_0)$ can be viewed as an inequality with the unknown h. For a given t, we can try to solve such an inequality by forming the iteration sequence

$$\begin{array}{lcl} \psi_0 & = & \neg h_0 \ \wedge \ En(t) \ \wedge \ pre(t, h_0), \\ \psi_1 & = & \psi_0 \ \wedge \ \bigwedge_{\tau \neq t} pre(\tau, \psi_0 \vee h_0), \\ \psi_2 & = & \psi_1 \ \wedge \ \bigwedge_{\tau \neq t} pre(\tau, \psi_1 \vee h_0), \\ \ldots & & \end{array}$$

until it converges.

Computation of h_1 Continued

Let us form the recommended iteration sequence for the case that $h_0 = at\_\ell_4$ and $t = \ell_3$. We obtain

$$\begin{aligned}\psi_0 &= \underbrace{\pi_1 = 3 \ \wedge \ (y_2 = 0 \ \vee \ y_1 < y_2)}_{En(\ell_3)} \ \wedge \ \underbrace{En(\ell_3) \rightarrow at\_\ell_4[\pi_1 \mapsto 4]}_{pre(\ell_3, at\_\ell_4)} \\ &= at\_\ell_3 \ \wedge \ (y_2 = 0 \ \vee \ y_1 < y_2)\end{aligned}$$

In principle, we should now compute $\psi_1 = \psi_0 \ \wedge \ \bigwedge_{\tau \neq \ell_3} pre(\tau, \psi_0 \vee h_0)$. However, since we can show that every transition different from ℓ_3 preserves ψ_0, this computation will produce an assertion equivalent to ψ_0. Thus, the iteration sequence converges in a single step, and produce $t_1 = \ell_3$ and

$$h_1 : \quad at\_\ell_3 \ \wedge \ (y_2 = 0 \ \vee \ y_1 < y_2)$$

Why did we choose $t = \ell_3$?

We can try different transitions. However, the computation shows that $\neg h_0 \ \wedge \ En(t) \ \wedge \ pre(t, h_0)$ for any $t \neq \ell_3$ yields 0 (the empty assertion). Therefore, $t_1 = \ell_3$ is the only helpful transition which yields a non-trivial h_1.

Proceeding to t_2 and h_2

Once we identified h_2, the search for h_2 can be based on a solution of the implication $Imp(h_0 \vee h_1)$ for an appropriately chosen $t = t_2$. Repeating the specified procedure, we end up computing the following sequence of h_i and t_i:

i	t_i	h_i
0	–	at_ℓ_4
1	ℓ_3	$at\_\ell_3 \ \wedge \ (y_2 = 0 \ \vee \ y_1 < y_2)$
2	m_5	$at\_\ell_3 \ \wedge \ at\_m_5$
3	m_4	$at\_\ell_3 \ \wedge \ at\_m_4$
4	m_3	$at\_\ell_3 \ \wedge \ at\_m_3 \ \wedge \ (y_1 = 0 \ \vee \ y_2 \leq y_1)$
5	ℓ_2	at_ℓ_2

In the computation of this table, we made free use of the relevant invariants which correlate the values of y_1 and y_1 to the locations of the processes, i.e.

$\square \ (y_1 = 0 \ \leftrightarrow \ at\_\ell_{0..2})$ and $\square \ (y_2 = 0 \ \leftrightarrow \ at\_m_{0..2})$