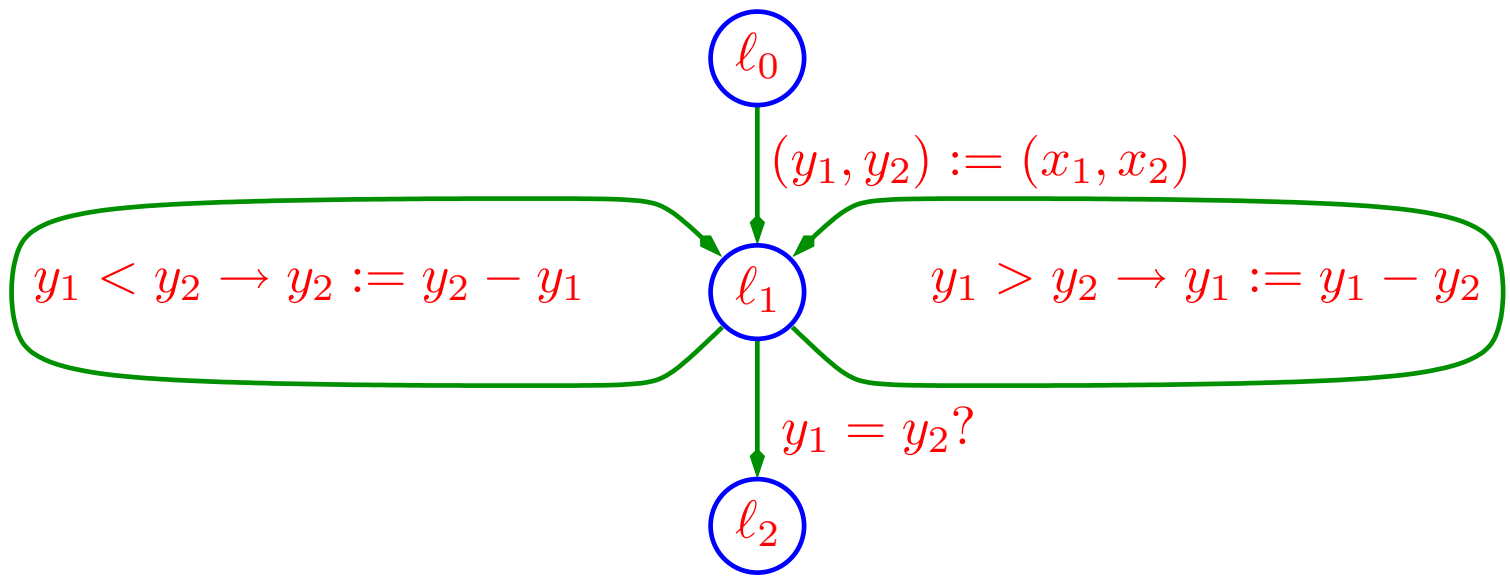
Additional Examples

Consider the following program, based on Euclid's algorithm, for finding the greatest common divisor (gcd) of two positive integers:



The specification for this program is given by

$$p: \quad x_1 > 0 \ \wedge \ x_2 > 0 \qquad\qquad q: \quad y_1 = \gcd(x_1, x_2)$$

As a cut-set we take ℓ_0, ℓ_1, ℓ_2. For the assertion network, we take

$$\varphi_0 : p, \qquad \varphi_1 : y_1 > 0 \ \wedge \ y_2 > 0 \ \wedge \ \gcd(y_1, y_2) = \gcd(x_1, x_2), \qquad \varphi_2 : q$$

The verification conditions are given by:

$$VC_{01} : x_1 > 0 \ \wedge \ x_2 > 0 \quad \rightarrow \quad x_1 > 0 \ \wedge \ x_2 > 0 \ \wedge \ \gcd(x_1, x_2) = \gcd(x_1, x_2)$$

$$VC_{1l1} : y_1 > 0 \ \wedge \ y_2 > 0 \ \wedge \ \gcd(y_1, y_2) = \gcd(x_1, x_2) \ \wedge \ y_1 > y_2 \ \rightarrow$$
$$y_1 - y_2 > 0 \ \wedge \ y_2 > 0 \ \wedge \ \gcd(y_1 - y_2, y_2) = \gcd(x_1, x_2)$$

$$VC_{1r1} : y_1 > 0 \ \wedge \ y_2 > 0 \ \wedge \ \gcd(y_1, y_2) = \gcd(x_1, x_2) \ \wedge \ y_1 < y_2 \ \rightarrow$$
$$y_1 > 0 \ \wedge \ y_2 - y_1 > 0 \ \wedge \ \gcd(y_1, y_2 - y_1) = \gcd(x_1, x_2)$$

$$VC_{12} : y_1 > 0 \ \wedge \ y_2 > 0 \ \wedge \ \gcd(y_1, y_2) = \gcd(x_1, x_2) \ \wedge \ y_1 = y_2 \ \rightarrow$$
$$y_1 = \gcd(x_1, x_2)$$

Basic Properties

The proof is based on the following basic properties of the gcd function:

$$P1. \quad y_1 \neq y_2 \quad \rightarrow \quad \gcd(y_1 - y_2, y_2) = \gcd(y_1, y_2 - y_1) = \gcd(y_1, y_2)$$
$$P2. \quad y > 0 \quad \rightarrow \quad \gcd(y, y) = y$$

Raising a to Integer Power b

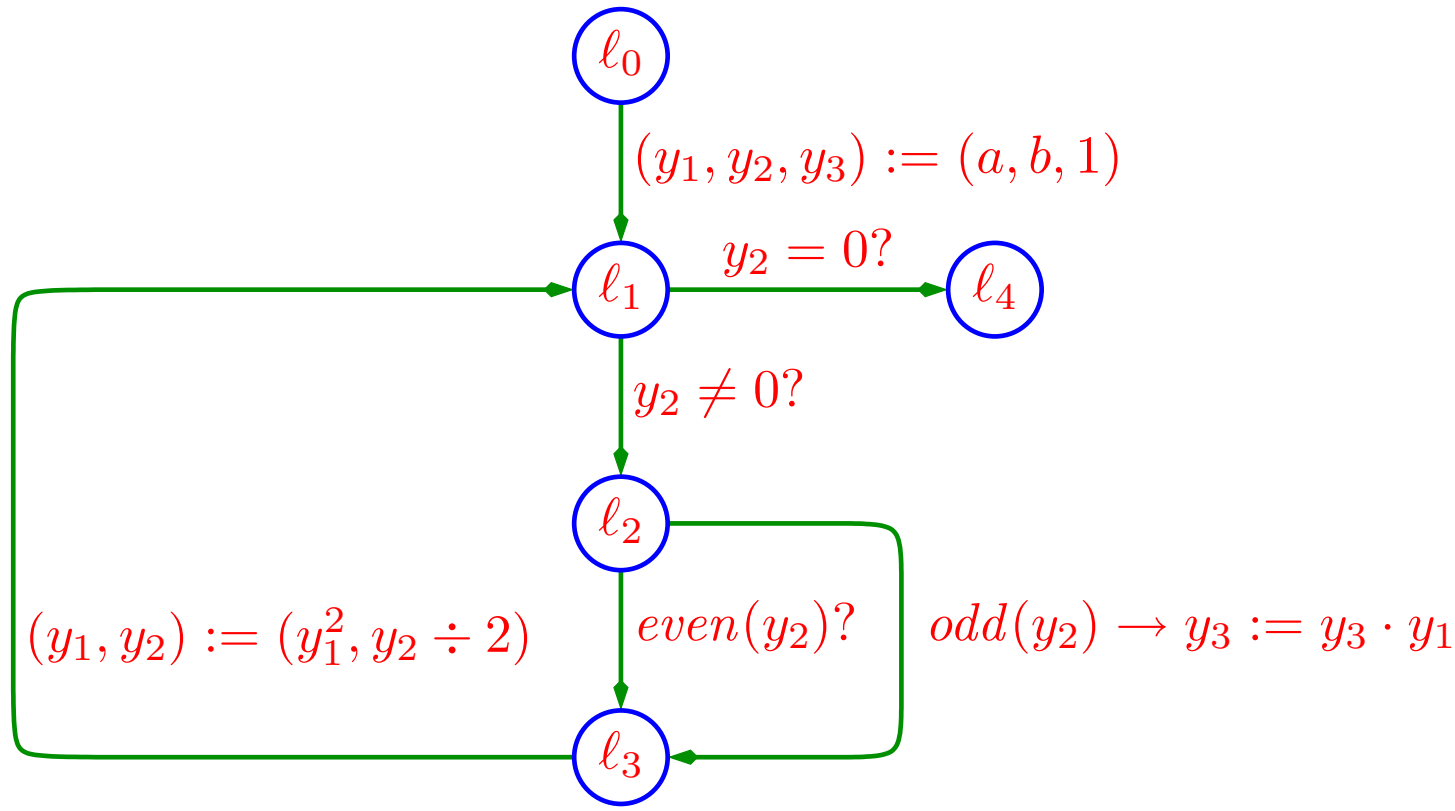
As additional example, we consider a program for computing integer powers of an arbitrary real number. The algorithm is based on the binary representation of integers as follows:
Let $b = \sum_{i=0}^{k} b_i 2^i$ be the binary representation of the natural number b. Then,

$$a^b = a^{\sum_{i=0}^{k} b_i 2^i} = \prod_{i=0}^{k} a^{(2^i b_i)}$$

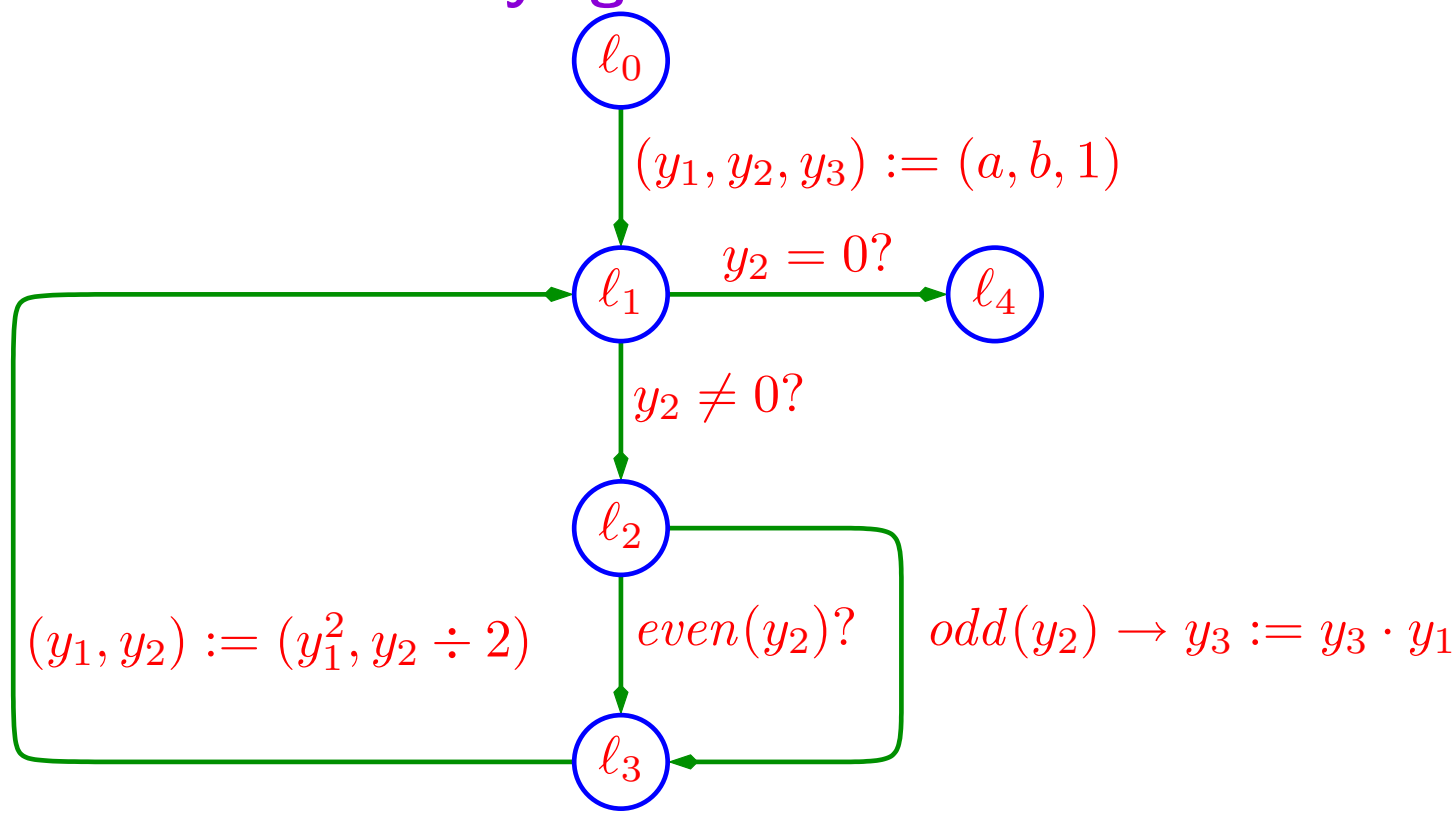
The sequence of a^{2^i}, $i = 0, \ldots, k$ can be obtained by starting with a and then successively squaring the previous element:

$$a, a^2, a^4, a^8, \ldots, a^{2^i}$$

Also, since b_i is either 0 or 1, the product above can be interpreted as multiply all a^{2^i} for which $b_i = 1$. This leads to the following program:



Verifying a^b



The specification for this program is given by:

$$p: \quad b \geq 0 \qquad\qquad q: \quad y_3 = a^b$$

As the cut-set, we choose ℓ_0, ℓ_1, ℓ_4. For the assertion network, we take $\varphi_0 = p$, $\varphi_4 = q$, and $\varphi_1 : y_2 \geq 0 \ \wedge \ y_3 \cdot y_1^{y_2} = a^b$.

Forming the verification conditions,we obtain:

$$\begin{array}{lll}
VC_{01}: & b \geq 0 & \rightarrow \quad b \geq 0 \ \wedge \ 1 \cdot a^b = a^b \\
VC_{1l1}: & y_2 \geq 0 \ \wedge \ y_3 \cdot y_1^{y_2} = a^b \ \wedge \ y_2 \neq 0 \ \wedge \ even(y_2) & \rightarrow \\
 & & y_2 \div 2 \geq 0 \ \wedge \ y_3 \cdot (y_1^2)^{y_2 \div 2} = a^b \\
VC_{1r1}: & y_2 \geq 0 \ \wedge \ y_3 \cdot y_1^{y_2} = a^b \ \wedge \ y_2 \neq 0 \ \wedge \ odd(y_2) & \rightarrow \\
 & & y_2 \div 2 \geq 0 \ \wedge \ (y_3 \cdot y_1) \cdot (y_1^2)^{y_2 \div 2} = a^b \\
VC_{14}: & y_2 \geq 0 \ \wedge \ y_3 \cdot y_1^{y_2} = a^b \ \wedge \ y_2 = 0 & \rightarrow \quad y_3 = a^b
\end{array}$$

Dependence on the Cut-Set

Interested in exploring the limitations of the inductive assertion method, we first check whether the method is sensitive to the choice of the cut-set $\mathcal{C}$. A special case is that of a full cut-set $\mathcal{C} = \mathcal{L}$ in which the cut-set includes all the locations in the program.

The following claim shows that any inductive assertions which is not full, can be extended to a bigger inductive network.

Claim 4. [Inductive networks can be extended] *Let $\mathcal{N} = \langle \mathcal{C}, \{\varphi_\ell \mid \ell \in \mathcal{C}\}\rangle$ be an inductive assertion network, and $\widetilde{\ell} \notin \mathcal{C}$ a location not in $\mathcal{C}$. There exists an inductive assertion network over the extended cut-set $\widetilde{\mathcal{C}} = \mathcal{C} \cup \{\widetilde{\ell}\}$ which agrees with $\mathcal{N}$ on the assertions φ_ℓ for all $\ell \in \mathcal{C}$.*

Proof: The extended network has the form $\mathcal{N} = \langle \mathcal{C}, \{\psi_\ell \mid \ell \in \mathcal{C}\}\rangle$, where $\widetilde{\mathcal{C}} = \mathcal{C} \cup \{\widetilde{\ell}\}$. For all $\ell \in \mathcal{C}$, we take $\psi_\ell = \varphi_\ell$. We will present two different constructions for the computation of $\psi_{\widetilde{\ell}}$, one based on backwards propagation while the other is based on forward propagation.

Backwards Propagation

Let $\Pi_{\widetilde{\ell},\mathcal{C}}$ be the set of paths connecting $\widetilde{\ell}$ to a location in $\mathcal{C}$ without passing through any other cut-point. For each path $\pi \in \Pi_{\widetilde{\ell},\mathcal{C}}$, let $dest(\pi)$, c_π, and f_π denote, respectively, the cut-point at the end of path π, the summary traversal condition, and data transformation associated with π. We define the pre-condition for $\mathcal{N}$ at $\widetilde{\ell}$ which is given by:

$$pre(\widetilde{\ell},\mathcal{N}) : \bigwedge_{\pi\in\Pi_{\widetilde{\ell},\mathcal{C}}} \Big(c_\pi(V) \quad \rightarrow \quad \varphi_{dest(\pi)}(f_\pi(V))\Big)$$

Formula $pre(\widetilde{\ell},\mathcal{N})$ is the condition at $\widetilde{\ell}$ which guarantees that if execution continues to reach a location $\ell \in \mathcal{C}$, then it will reach it with a data state satisfying φ_ℓ.

We will now show that, under the assumption that $\mathcal{N}$ is inductive, the extended network obtained by taking $\psi_{\widetilde{\ell}} = pre(\ell,\mathcal{N})$ is also inductive. Clearly we only have to consider new verification paths, i.e. verification paths which appear in $\widetilde{\mathcal{N}}$ but did not exist in $\mathcal{N}$. There are two such classes of paths, which we consider separately.

Backwards Propagation: Paths from $\widetilde{\ell}$ to $\ell \in \mathcal{C}$

Let π be a $\widetilde{\mathcal{N}}$-path connecting $\widetilde{\ell}$ to some cut-point $\ell_2 \in \mathcal{C}$. The verification condition for such a path is given by:

$$\psi_{\widetilde{\ell}}(V) \ \wedge \ c_\pi(V) \quad \rightarrow \quad \varphi_{\ell_2}(f_\pi(V))$$

which is equivalent to

$$\psi_{\widetilde{\ell}}(V) \quad \rightarrow \quad (c_\pi(V) \rightarrow \varphi_{\ell_2}(f_\pi(V)))$$

As π connects $\widetilde{\ell}$ to a location in $\mathcal{C}$, this path is one of the members of the set $\Pi_{\widetilde{\ell},\mathcal{C}}$ over which the conjunction defining $\psi_{\widetilde{\ell}} = \mathit{pre}(\widetilde{\ell}, \mathcal{N})$ is taken. Therefore, $c_\pi(V) \rightarrow \varphi_{\ell_2}(f_\pi(V))$ is one of these conjuncts and is implied by $\psi_{\widetilde{\ell}}(V)$.

Backwards Propagation: Paths from $\ell \in \mathcal{C}$ to $\widetilde{\ell}$

Next, consider a path π_1 connecting some location $\ell_1 \in \mathcal{C}$ to $\widetilde{\ell}$ and not passing through any other cut-point.

Let $\pi_2 \in \Pi_{\widetilde{\ell},\mathcal{C}}$ be an arbitrary verification path connecting $\widetilde{\ell}$ to some location $\ell_2 = dest(\pi_2)$. Let $\pi = \pi_1 \circ \pi_2$ be the path obtained by following π_1 first and then continuing along π_2 (fusion of π_1 and π_2). Path π was a verification path for the network $\mathcal{N}$. Since $\mathcal{N}$ was inductive, we know that the verification condition

$$\varphi_1(V) \ \wedge \ c_\pi(V) \quad \rightarrow \quad \varphi_2(f_\pi(V)) \tag{1}$$

is valid. It is not difficult to relate the traversal condition and data transformation of π to these of its constituents. These are given by

$$c_\pi(V) = c_{\pi_1}(V) \ \wedge \ c_{\pi_2}(f_{\pi_1}(V)) \qquad \text{and} \qquad f_\pi(V) = f_{\pi_2}(f_{\pi_1}(V)) \tag{2}$$

With these relation, Formula (1) can be rewritten as

$$\varphi_1(V) \ \wedge \ c_{\pi_1}(V) \quad \rightarrow \quad (c_{\pi_2}(f_{\pi_1}(V)) \rightarrow \varphi_2(f_{\pi_2}(f_{\pi_1}(V))))$$

or, equivalently, as

$$\varphi_1(V) \ \wedge \ c_{\pi_1}(V) \quad \rightarrow \quad \textbf{let } \overline{V} = f_{\pi_1}(V) \textbf{ in } \left(c_{\pi_2}(\overline{V}) \rightarrow \varphi_2(f_{\pi_2}(\overline{V}))\right)$$

Taking the conjunction of this implication over all paths $\pi_2 \in \Pi_{\widetilde{\ell},\mathcal{C}}$, we obtain

$$\varphi_1(V) \ \wedge \ c_{\pi_1}(V) \quad \rightarrow \quad \textbf{let } \overline{V} = f_{\pi_1}(V) \textbf{ in } \bigwedge_{\pi_2 \in \Pi_{\widetilde{\ell},\mathcal{C}}} \left(c_{\pi_2}(\overline{V}) \rightarrow \varphi_2(f_{\pi_2}(\overline{V}))\right)$$

or, equivalently,

$$\varphi_1(V) \ \wedge \ c_{\pi_1}(V) \quad \rightarrow \quad \psi_{\widetilde{\ell}}(f_{\pi_1}(V))$$

Forward Propagation

An alternate definition of $\psi_{\widetilde{\ell}}$ is based on the consideration of paths from $\mathcal{C}$ to $\widetilde{\ell}$.

Let $\Pi_{\mathcal{C},\widetilde{\ell}}$ be the set of paths connecting a location in $\mathcal{C}$ to $\widetilde{\ell}$ without passing through any other cut-point. For each path $\pi \in \Pi_{\mathcal{C},\widetilde{\ell}}$, let $srce(\pi)$, c_π, and f_π denote, respectively, the cut-point at the beginning of path π, the summary traversal condition, and data transformation associated with π. We define the post-condition for $\mathcal{N}$ at $\widetilde{\ell}$ which is given by:

$$post(\mathcal{N},\widetilde{\ell}): \quad \exists V_0 \bigvee_{\pi\in\Pi_{\mathcal{C},\widetilde{\ell}}} \left(\varphi_{srce(\pi)}(V_0) \ \wedge \ c_\pi(V_0) \ \wedge \ V = f_\pi(V_0)\right)$$

Formula $post(\mathcal{N},\widetilde{\ell})$ characterize the states which can be reached at location $\widetilde{\ell}$ by an execution whose previous visit to a location $\ell \in \mathcal{C}$ was with a data state satisfying φ_ℓ.

We will now show that, under the assumption that $\mathcal{N}$ is inductive, the extended network obtained by taking $\psi_{\widetilde{\ell}} = post(\mathcal{N},\widetilde{\ell})$ is also inductive. Clearly we only have to consider new verification paths, i.e. verification paths which appear in $\widetilde{\mathcal{N}}$ but did not exist in $\mathcal{N}$. As before, there are two such classes of paths, which we consider separately.

Forward Propagation: Paths from $\ell \in \mathcal{C}$ to $\widetilde{\ell}$

Let π be a $\widetilde{\mathcal{N}}$-path connecting a location $\ell_1 \in \mathcal{C}$ to $\widetilde{\ell}$. The verification condition for such a path is given by:

$$\varphi_{\ell_1}(V_0) \ \wedge \ c_\pi(V_0) \quad \rightarrow \quad \psi_{\widetilde{\ell}}(f_\pi(V_0))$$

which is validity-equivalent to

$$\varphi_{srce(\pi)}(V_0) \ \wedge \ c_\pi(V_0) \ \wedge \ V = f_\pi(V_0) \quad \rightarrow \quad \psi_{\widetilde{\ell}}(V)$$

As π connects a location in $\mathcal{C}$ to $\widetilde{\ell}$, this path is one of the members of the set $\Pi_{\mathcal{C},\widetilde{\ell}}$ over which the disjunction defining $\psi_{\widetilde{\ell}} = post(\mathcal{N}, \widetilde{\ell})$ is taken. Therefore, $\varphi_{srce(\pi)}(V_0) \ \wedge \ c_\pi(V_0) \ \wedge \ V = f_\pi(V_0)$ is one of these disjuntcs and hence implies $\psi_{\widetilde{\ell}}(V)$.

Forward Propagation: Paths from $\widetilde{\ell}$ to $\ell \in \mathcal{C}$

Next, consider a path π_2 connecting $\widetilde{\ell}$ to some location $\ell_2 \in \mathcal{C}$ and not passing through any other cut-point.

Let $\pi_1 \in \Pi_{\widetilde{\ell},\mathcal{C}}$ be an arbitrary verification path connecting some location $\ell_1 = src e(\pi_1)$ to $\widetilde{\ell}$. Let $\pi = \pi_1 \circ \pi_2$ be the fusion of paths π_1 and π_2. Path π was a verification path for the network $\mathcal{N}$. Since $\mathcal{N}$ was inductive, we know that the verification condition

$$\varphi_1(V) \ \wedge \ c_\pi(V) \quad \rightarrow \quad \varphi_2(f_\pi(V)) \tag{3}$$

is valid. Using the relations between the traversal condition and data transformation of π to these of its constituents, Formula (3) can be rewritten as

$$\varphi_1(V) \ \wedge \ c_{\pi_1}(V) \quad \rightarrow \quad (c_{\pi_2}(f_{\pi_1}(V)) \rightarrow \varphi_2(f_{\pi_2}(f_{\pi_1}(V))))$$

which is validity equivalent to

$$(\exists V_0 : \varphi_1(V_0) \ \wedge \ c_{\pi_1}(V_0) \ \wedge \ V = f_{\pi_1}(V_0)) \quad \rightarrow \quad (c_{\pi_2}(V) \rightarrow \varphi_2(f_{\pi_2}(V)))$$

Taking the conjunction of this implication over all paths $\pi_1 \in \Pi_{\mathcal{C},\widetilde{\ell}}$, we obtain

$$\left[\bigvee_{\pi_1 \in \Pi_{\mathcal{C},\widetilde{\ell}}} \exists V_0 : \varphi_1(V_0) \ \wedge \ c_{\pi_1}(V_0) \ \wedge \ V = f_{\pi_1}(V_0)\right] \quad \rightarrow \quad (c_{\pi_2}(V) \rightarrow \varphi_2(f_{\pi_2}(V)))$$

or, equivalently,

$$\psi_{\widetilde{\ell}}(V) \ \wedge \ c_{\pi_2}(V) \quad \rightarrow \quad \varphi_2(f_{\pi_2}(V))$$

Removing Cut-Points

In the previous discussion we have shown that it is always possible to add more cut-points to an inductive network, while maintaining inductivity. We will now show that it is also possible to remove cut-points, provided the remaining set is still a cut-set.

Claim 5. *Let $\mathcal{N} = \langle \mathcal{C}, \{\varphi_\ell \mid \ell \in \mathcal{C}\}\rangle$ be an inductive network. Let $\widetilde{\ell} \in \mathcal{C}$ be a location in $\mathcal{C}$ such that $\overline{\mathcal{C}} = \mathcal{C} - \{\widetilde{\ell}\}$ is a cut-set. Then the network $\overline{\mathcal{N}} = \langle \overline{\mathcal{C}}, \{\varphi_\ell \mid \ell \in \overline{\mathcal{C}}\}\rangle$, obtained by removing $\widetilde{\ell}$ and $\varphi_{\widetilde{\ell}}$ from $\mathcal{N}$, is also inductive.*

Proof: We only need to consider "new" verification paths, i.e. paths which exist in $\overline{\mathcal{N}}$ but not in $\mathcal{N}$. Such a path π connecting $\ell_1 \in \overline{\mathcal{C}}$ to $\ell_2 \in \overline{\mathcal{C}}$ must be the fusion $\pi = \pi_1 \circ \pi_2$ of two paths, where path π_1 connects ℓ_1 to $\widetilde{\ell}$, while π_2 connects $\widetilde{\ell}$ to ℓ_2.

Since both π_1 and π_2 are verification paths in the inductive network $\mathcal{N}$, we know that the following implications are valid:

$$\begin{array}{rcl} \varphi_{\ell_1}(V_1) \ \wedge \ c_{\pi_1}(V_1) & \rightarrow & \varphi_{\widetilde{\ell}}\,(f_{\pi_1}(V_1)) \\ \varphi_{\widetilde{\ell}}\,(V_2) \ \wedge \ c_{\pi_2}(V_2) & \rightarrow & \varphi_{\ell_2}(f_{\pi_2}(V_2)) \end{array}$$

Substituting $f_{\pi_1}(V_1)$ for V_2 in the second implication, and combining the two together yields

$$\varphi_{\ell_1}(V) \ \wedge \ c_{\pi_1}(V) \ \wedge \ c_{\pi_2}(f_{\pi_1}(V)) \quad \rightarrow \quad \varphi_{\ell_2}(f_{\pi_2}(f_{\pi_1}(V)))$$

which, using the relations in Formula (2), can be rewritten as

$$\varphi_{\ell_1}(V) \ \wedge \ c_{\pi}(V) \quad \rightarrow \quad \varphi_{\ell_2}(f_{\pi}(V))$$

Method is Independent of the Choice of the Cut-Set

The preceding discussions can be summarized by the statement that the success or failure of an application of the inductive assertion method is independent of the particular choice of the cut-set $\mathcal{C}$.

Technically, this can be summatized by the following corollary:

Corollary 6. *Let $\langle p, q\rangle$ be a specification for program P, and let $\mathcal{C}_1$ and $\mathcal{C}_2$ be two cut-sets. Then, there exists an inductive network $\mathcal{N}_1$ based on $\mathcal{C}_1$ and entailing $\langle p, q\rangle$ iff there exists an inductive network $\mathcal{N}_2$ based on $\mathcal{C}_2$ and entailing $\langle p, q\rangle$.*

The proof of this statement can be obtained by starting with a $\mathcal{C}_1$-based inductive network $\mathcal{N}_1$ and incrementally adding missing locations, using Claim 4, until we obtain a full network. Then we start removing locations which do not belong to $\mathcal{C}_2$, relying on Claim 5, until we obtain a network $\mathcal{N}_2$ based on $\mathcal{C}_2$. Since both ℓ_0 and ℓ_t belong to both $\mathcal{N}_1$ and $\mathcal{N}_2$, their associated assertions are preserved throughout the entire procecss. Therefore, if $\mathcal{N}_1$ entails $\langle p, q\rangle$ then so does $\mathcal{N}_2$.

Completeness of the Method

An important question which arises whenever a proof method is introduced is that of completeness. Namely, is it the case that, whenever a program is partially correct w.r.t a specification $\langle p, q \rangle$, this fact can be proven, using the inductive assertion method?

In our case, the answer is positive, and we will prepare the necessary constructs for proving this fact.

Let ℓ_i and ℓ_j be two locations in the program which are connected by a direct edge, labeled by the guaraded command $c_{ij} \rightarrow [V := f_{ij}(V)]$. We define the formula

$$\rho_{ij}(V_1, V_2): \quad c_{ij}(V_1) \ \wedge \ V_2 = f_{ij}(V_1)$$

Obviously, ρ_{ij} is satisfied by the two data states $V_1 = d_1$ and $V_2 = d_2$ iff there exists a computation step $\langle \ell_i, d_1 \rangle \rightarrow \langle \ell_j, d_2 \rangle$ leading from the execution state $\langle \ell_i, d_1 \rangle$ to the execution state $\langle \ell_j, d_2 \rangle$.

Let E denote the set of direct edges in the program, i.e. set of pairs (ℓ_i, ℓ_j), such that there is a direct edge from ℓ_i to ℓ_j. Assume that we consider a fixed specification $\langle p, q \rangle$ for program P. For simplicity, we assume first that the program has a single data variable (V) which ranges over domain D.

The Minimal Predicate at Location ℓ

For each location ℓ in the program, we define the minimal predicate $M_\ell(V)$. It is intended that a data state d satisfies M_ℓ iff

There exists a p-computation reaching the execution state $\langle \ell, d \rangle$.

This can be formalized by the following extended predicate logic formula:

$$M_\ell(V): \left(\begin{array}{l} \exists k \geq 0 : \exists loc : [0..k] \mapsto [0..t], A : [0..k] \mapsto D : \\ \quad loc[0] = 0 \ \wedge \ p(A[0]) \ \wedge \ \ell_{loc[k]} = \ell \ \wedge \ V = A[k] \ \wedge \\ \quad \forall r : [0..k) : \bigvee_{(i,j) \in E} loc[r] = i \ \wedge \ loc[r{+}1] = j \ \wedge \ \rho_{ij}(A[r], A[r{+}1]) \end{array} \right)$$

The formula states the existence of two arrays of size $k+1$ for some $k \geq 0$. The array $loc[0..k]$ encodes the indices of the locations traversed during the computation from ℓ_0 to ℓ, while the array $A[0..k]$ encodes the sequence of data states encoutered during this computation. The conjunction $loc[0] = 0 \ \wedge \ p(A[0])$ ensures that the exeuction state $\langle \ell_{loc[0]}, A[0] \rangle$ is an initial state. The conjunction $\ell_{loc[k]} = \ell \ \wedge \ V = A[k]$ ensure that the last exeuction state encoded by these two arrays is of the form $\langle \ell, d \rangle$ where d equals the current value of V. The conjunction under the $\forall r$ quantification guarantees that the sequence evolve according to the rules of the program P and is, therefore, a computation of P.

The Claim of Completeness

Claim 7. [Completeness of the inductive assertions method] *Let P be a program which is partially correct w.r.t the specification $\langle p, q\rangle$. Then there exists an inductive assertion network which entails $\langle p, q\rangle$.*

Proof: For the cut-set we take $\mathcal{C} = \mathcal{L}$, i.e. a full cut-set. As the aseertion associated with location ℓ we take the minimal predicate M_ℓ. It remains to show that this assertion network is inductive and that it entails $\langle p, q\rangle$.

To show inductiveness, let ℓ_i and ℓ_j be two locations which are connected by a direct edge. We have to show that if a data state d satisfies $\varphi_i \ \wedge \ c_{ij}$, where $\varphi_i = M_i$, then $f_{ij}(d)$ satisfies $\varphi_j = M_j$. By definition of M_i, $d \models M_i$ implies that there exists a p-computation segment $\sigma_i : s_0, \ldots, \langle \ell_i, d\rangle$ reaching location ℓ_i with data state d. Since $d \models c_{ij}$, we can extend σ_i by one more step to obtain the p-computation segment $\sigma_j : s_0, \ldots, \langle \ell_i, d\rangle, \langle \ell_j, f_{ij}(d)\rangle$, i.e., a computation segment reaching location ℓ_j with data state $f_{ij}(d)$. By the definition of M_j, it follows that $f_{ij}(d) \models M_j$.

Next, we have to show that the defined network entails the specification $\langle p, q\rangle$. Since location ℓ_0 has no incoming edges, the only computation segments reaching ℓ_0 must be singleton sequences of the form $\langle \ell_0, d\rangle$, where $d \models p$. It follows that the minimal predicate M_{ℓ_0} equals p, and therefore is trivially implied by p.

For the terminal location, we have to show that $M_{\ell_t} \rightarrow q$. Let d be a data state satisfying M_{ℓ_t}. This implies that d is a possible final result of a terminating p-computation. Since Claim 7 assumes that P is partially correct w.r.t $\langle p, q\rangle$, d must satisfy q. It follows that M_{ℓ_t} implies q. ⌟

What we Have Not Proven

It is not very difficult to remove the restriction that V consists of a single data variable. Assume, instead, that $V = \{y_1, \ldots, y_m\}$. Then, the only difference is that the array A will have to be a two-dimensional array of the form $A[1..m, 0..k]$.

Also, allowing the formula for M_ℓ to quantify over arrays, is not a very significant deviation from conventional first-order logic. If the data domain D is the naturals or integers or, for that matter, any other recursive data structure, we can use Gödel encoding, in order to reduce arrays of any dimension and sequences to natural numbers.

On the other hand, the reader should notice that the ony thing we proved is the existence of an assertion network whose verification conditions are valid. Nowhere did we claim that these conditions are provable in any formal system. Rather, all the available undecidability and imcompleteness results for first-order logic imply that that there does not exist a single formal system in which these verification conditions can always be proven.

Furthermore, the "construction" of inductive network as outlined in the completeness proof, should not be interpreted to mean that such construction is useful for any actual application. The main reason why this construction is not useful is that its soundness relies on the a priori assumption that the program is partially correct w.r.t $\langle p, q \rangle$. If we are already ensured of this fact, there is no remaining motivation for applying the inductive assertion method.