The immediate predecessor transformer can be iterated to yield the eventual:

\[ (1 + x = x \lor 0 = x : \mathcal{F}) = (t = x) \phi (1 + x = x) \]

For example:

\[ \{ s \text{ is an } \psi \text{-predecessor of } a \text{-state} \} = \| \phi \| \]

Obviously:

\[ (\lambda \phi \lor (\lambda \psi) \phi : \mathcal{F}) = \phi \]

Successor predecessor transformer:

\[ (1 + x = x \lor 0 = x : \mathcal{F}) = (t = x) \phi (1 + x = x) \]

For example:

\[ \{ s \text{ is an } \psi \text{-successor of } a \text{-state} \} = \| \phi \| \]

Obviously:

\[ (\lambda \phi \lor (\lambda \psi) \phi : \mathcal{F}) = \phi \]

\[ (\lambda \phi \lor (\lambda \psi) \phi : \mathcal{F}) = (t = x) \phi (1 + x = x) \]

For example:

\[ \{ s \text{ is an } \psi \text{-successor of } a \text{-state} \} = \| \phi \| \]

Obviously:

\[ (\lambda \phi \lor (\lambda \psi) \phi : \mathcal{F}) = \phi \]

\[ (\lambda \phi \lor (\lambda \psi) \phi : \mathcal{F}) = (t = x) \phi (1 + x = x) \]

For example:

\[ \{ s \text{ is an } \psi \text{-successor of } a \text{-state} \} = \| \phi \| \]

Obviously:

\[ (\lambda \phi \lor (\lambda \psi) \phi : \mathcal{F}) = \phi \]

\[ (\lambda \phi \lor (\lambda \psi) \phi : \mathcal{F}) = (t = x) \phi (1 + x = x) \]

For example:

\[ \{ s \text{ is an } \psi \text{-successor of } a \text{-state} \} = \| \phi \| \]

Obviously:

\[ (\lambda \phi \lor (\lambda \psi) \phi : \mathcal{F}) = \phi \]

\[ (\lambda \phi \lor (\lambda \psi) \phi : \mathcal{F}) = (t = x) \phi (1 + x = x) \]
Algorithm INV(d,p); Check that P d s satisfies

as in example, we can use the following algorithm:

The algorithm returns an assertion characterizing all the states from which there exists a finite path leading to a violation of \( \phi \), if \( \models \phi \) is satisfied. If there exists a finite path leading to a violation of \( \phi \), the algorithm returns the empty set.

4. return \( \phi \)
   \( \wedge \) \( \models \phi \).
3. if \( \models \phi \) then return \( \phi \).
2. \( \models \phi \); \( \models \phi \).
1. \( \models \phi \).

As in example, we can use the following algorithm:

Claim 4.4 (Invariance) Property A.Pnueli

\( \phi \)
\( \wedge \)
\( \models \phi \).

Using fix-point notation, we can represent the following:

Using fix-point notation, the situation is much better.

A set function is called monotonic if \( f \subseteq \mathcal{P}(S) \subseteq \mathcal{P}(S) \) implies the function is monotonic. For every set \( f \subseteq \mathcal{P}(S) \) there is a unique maximal solution which, for a finite set of naturals into the set of their successors. Similarly, we can define \( f \) as a set function.

\( \models \phi \).
If we intersect with the initial condition \( I \), we obtain
\[
\begin{align*}
\mu & \sim \left( 0 = \overline{N} \land \overline{C} \land \overline{L} \right) \\
& = \left( \overline{N} \land \left( \overline{C} \land \overline{L} \right) \right) \\
& = \left( \overline{C} \land \overline{L} \right) \\
& = \left( \overline{C} \right) \\
& = 0
\end{align*}
\]

The last equivalence is due to the general property

\[
\begin{align*}
C & = 0 \\
& = \overline{C} \\
& = 0
\end{align*}
\]


We iterate as follows:

Similarly backwards exploration on MUX-SEM

The semaphore instructions `request` and `release` respectively stand for

Below, we present a simpler version of program MUX-SEM.

**Example: A Simpler MUX-SEM**
**F-Sets Imply Feasibility**

A reachable state \(s\) is feasible if it has a path leading to some \(F\)-set.

**Claim 5. [F-sets]**

A reachable state \(s\) is feasible if it has a path leading to some \(F\)-set.

**Proof:**

Assume that \(s\) is a feasible state. Then it participates in some computation \(\sigma\). Let \(S\) be the (finite) set of all states that appear infinitely many times in \(\sigma\). We will show that \(S\) is an \(F\)-set.

Since \(\sigma\) is a computation, \(S\) contains all the states that appear at positions beyond \(t\). Obviously all states appearing in \(\sigma\) are reachable. If \(s \in S\) appears in \(\sigma\) at position \(i > t\), then \(i\) is a successor of \(s\) and \(j \in F\) is some justice requirement.

Since \(\sigma\) is a computation, \(S\) contains infinitely many \(F\)-positions. Let \(k \geq i\) be one of the \(F\)-positions appearing later than \(i\). Then the path \(s_0, s_1, \ldots, s_k\) is an \(S\)-path from \(s\) to a \(J\)-state.

We iterate as follows:

**Iteration 1:**

We start with the initial state \(s_0\).

**Iteration 2:**

We update the state \(s_1\) by applying the next state function.

**Iteration 3:**

We continue updating the state \(s_2\) and \(s_3\).

**Iteration 4 (Convergent):**

Since last iteration does not intersect \(C_1 \wedge C_2\), we conclude \(\neg(C_1 \wedge C_2)\).
We conclude that the above FDS is feasible. No reason is to remove any unique states from \( \{ q \} \), so this is our final set.

\[
\{ q \} : \{ q \} = \{ q \}
\]

Removing from \( \{ q \} \) all states which do not have a \( q \)-path leading to an \( \epsilon \)-state, we are left with

\[
\{ q \} : \{ q \} = \{ q \}
\]

\( 1 \neq x : \{ q \}
\]

We set \( g \) and then proceed as follows:

\[
(1 = x : \{ q \} = \{ q \})
\]

\( (g = x : \{ q \} = \{ q \})
\]

\( (1 \neq x : \{ q \}
\]

As an example, consider the following FDS:

**Example:**

\[
\begin{align*}
&\{ b \lor \phi \} \lor (d \lor \phi) \lor d_1 \lor (f \lor \phi) \\
&\lor \phi \lor d \lor (g \lor \phi) \lor \phi \\
\end{align*}
\]

Since we are interested in a maximal F-set, the computation can be expressed

\[
\begin{align*}
&\{ b \lor \phi \} \lor (d \lor \phi) \lor d_1 \lor (f \lor \phi) \\
&\lor \phi \lor d \lor (g \lor \phi) \lor \phi \\
\end{align*}
\]

This can be summarized as

\[
\begin{align*}
&\text{For every } b \lor \phi \lor d \lor f \lor g \\
&\text{and all reachable states } \phi \lor d \lor (g \lor \phi) \lor \phi
\end{align*}
\]

Next, we obtain the following requirements

Assume an assertion \( \phi \) which characterizes an F-set. Translating the requirements into formulas, we obtain the following F-sets:

**Computing F-sets:**

\[
\begin{align*}
&\text{For every } (b \lor \phi) \lor (d \lor \phi) \lor d_1 \lor (f \lor \phi) \\
&\lor \phi \lor d \lor (g \lor \phi) \lor \phi
\end{align*}
\]

\[
\begin{align*}
&\text{For every } (b \lor \phi) \lor (d \lor \phi) \lor d_1 \lor (f \lor \phi) \\
&\lor \phi \lor d \lor (g \lor \phi) \lor \phi
\end{align*}
\]

**Algorithmic Interpretation**

It is not difficult to see that the infinite sequence constructed in this way is a

**Computation:**

\[
\begin{align*}
&\text{For every } (b \lor \phi) \lor (d \lor \phi) \lor d_1 \lor (f \lor \phi) \lor d_1 \lor (f \lor \phi) \\
&\lor \phi \lor d \lor (g \lor \phi) \lor \phi
\end{align*}
\]

**Proof Continued**
Let $D$: $h; V; J; Ci$ be a response property we wish to verify over $D$. Let $D^r$ be the assertion characterizing all the reachable $D$-reachable states which do not have a path leading to a $\lnot \psi$-state. This leaves us with nothing. We conclude that $\psi$. Next, we eliminate all states which do not have a path leading to a $\psi$-state. This leaves us with $\psi$. First, we eliminate all $\psi$-states which do not have a path leading to a $\psi$-state. This leaves us with $\psi$. Next, we eliminate all states which do not have a path leading to a $\psi$-state. This leaves us with $\psi$. Finally, we eliminate all states which do not have a path leading to a $\psi$-state. This leaves us with $\psi$. We conclude that $D_j = \psi$ if and only if $D^r$ is unreachable.

Claim 6: [Model Checking Response]

If $D_j = \psi$, then $D^r$ is unreachable.

Proof:
The claim is justified by the observation that every computation that reaches a $\psi$-state is unreachable.

Example: $\text{MUX-SEM}$

Following is the set of all reachable states of program $\text{MUX-SEM}$.