We may use the following basic invariance rule to prove the invariance of assertion $d$. That is, to establish that the formula $\square d$, for an assertion $d$, is $\mathcal{D}$-valid.

**Verication of Invariance Properties**
Consider the following parameterized program coordinating mutual exclusion by semaphores:

```
Example: Program MUX-SEM
```

We use rule \( \text{BINV} \) to verify the invariance of the assertion:

\[
0 < f_i 
\]

This assertion is inductive so the proof succeeds.

The semaphore instructions request \( y \) and release \( y \) respectively stand for:

- **Request**: \( f_i : 2 \)
- **Release**: \( f_i : 4 \)
- **Critical**: \( f_i : 3 \)
- **Non-critical**: \( f_i : 1 \)

Consider the following parameterized programs coordinating mutual exclusion by semaphores.
This attempt fails.

\[ ([\exists z \in \mathbb{A} \cdot \mathcal{A} \cap \mathcal{B} \cdot \mathcal{D} \cdot \mathcal{E}]) : \exists z \]

Next, let us try to verify the property of mutual exclusion which can be specified as the invariance of the assertion

\[ \mathcal{A} \]
As is already explained when one learns mathematical induction, there are valid assertions which cannot be proven by induction, where the induction hypothesis, for example, is the above example, this:

\[ 1 + 3 + 5 + \cdots + (2k - 1) = k^2 \]

is taken to be itself. The sum \( 1 + 3 + 5 + \cdots + (2k - 1) \) is a perfect square for example, the claim is.

\[ y = (1 - y/2) + \cdots + 3 + 1 : \sigma \]

\[ n = (1 - y/2) + \cdots + 3 + 1 : n \in \mathbb{N} : d \]

can be strengthened of \( \sigma \), which implies \( d \) and is inductive. For the above example, this.

To overcome this difficulty, one often has to come up with a
The above considerations lead to the more general INV rule.

\[
(I = 0 + [N]^{\forall \xi \exists y \neg \alpha} + \cdots + [2]^{\forall \xi \exists y \neg \alpha} + [1]^{\forall \xi \exists y \neg \alpha} + (0 \geq \hat{r})) : \phi
\]

Using rule INV with the strengthening

\[
([2]^{\forall \xi \exists y \neg \alpha} \vee [1]^{\forall \xi \exists y \neg \alpha}) : \exists \phi
\]

For example, we can establish the invariance of

For an assertion 'd' implies \( \phi \), it follows that 'd' is also a \( D \)-invariant. Since, by premise I3, \( \phi \) implies 'd' it satisfies \( D \). That is, all reachable states satisfy 'd'. By premises I1 and I2, 'd' is an invariant of the system. That is, all reachable states.

\[
\begin{array}{c}
1. \phi \leftarrow \Theta \\
2. \phi \leftarrow d \lor \phi \\
3. d \leftarrow \phi \\
\hline
\text{For an assertion 'd', 'd' is an invariant of the system.}
\end{array}
\]

Rule INV
Finito confirmation.

If the rule application succeeded, there are good chances (but no guarantee) that the assertion is inductive. This is the time to shift to PVS in order to get the final confirmation.

Inductive! We should strengthen it, and repeat the procedure.

If the rule application produces a counter-example, the assertion is not inductive. We should strengthen it, and apply rule BINV.

To do so, we define a finite-state restriction of the original program, explicitly calculate the candidate assertion, and apply rule BINV.

The TLV tool, developed by Elad Shahar, is a programmable symbolic calculator over finite-state systems. As we will show, it can also be used for incremental development of inductive assertions.

Using TLV for Incremental Strengthening

A Pnueli
Lecture 2: Verification of Invariance Properties

A. Pnueli
Lecture 2: Verification of Invariance Properties

The Input File mux3.smv

MODULE main
DEFINEN := 3; VAR y:boolean;
P: array 1..N of process MP(y);
Id: process Idle;
ASSIGN init(y) := 1;

MODULE Idle
MODULE MP(y)
VAR loc: 0..4;
ASSIGN init(loc) := 0;

next(loc) := case
loc in {0,1,3,4}: (loc + 1) mod 5;
loc = 2 & y: 3;
end;

next(y) := case
loc = 2 & next(loc) = 3: 0;
loc = 4 & next(loc) = 0: 1;
y: 1;
end;

JUSTICE loc != 0, loc != 3, loc != 4
COMPASSION next(loc)

Deductive Verification of Reactive Systems, NYU, Fall, 2007
In file `scr1.pf`, we place the text:

```
Print "\nModel Check mutual exclusion between P[1] and P[2]\n";
```

We then run:

```
tlv mux3.smv
TLV version 3.1

> Load "scr1.pf";
Your wish is my command ...
Model checking ...
*** Property is VALID ***
>```

Deductive Verification of Reactive Systems, NYU, Fall 2007
trying first approximation:
\[
\begin{array}{r}
y = i, 0 \quad p[1] = 0, 0 \quad p[2] = 2, 3 \quad p[3] = 3, 3 \quad p[4] = 3, 3
\end{array}
\]

In file `scr2.pf`, we place

```
Print
Try deductive verification of mutual exclusion
```

To prepare assertion

```
Let ass := 1;
For i in 1...N
  For j in 1...N
    Let ass := ass & (i = j | P[i].loc != 3 | P[j].loc != 3);
  End--For(j in 1...N)
End--For(i in 1...N)
End--prepare_assertion
```

To prepare assertion

```
Call print(ass);
```

To prepare assertion

```
End -- prepare_assertion
```

```
End -- For (i in 1..N)
End -- For (j in 1..N)
```

```
((\exists i \forall j \neg \bigvee [i] \neg \forall j \neg [j] a_{i3}) \land \forall j \neg a_{j3})
```

Trying first approximation:
Strengthening the Assertion

The offending transition captures a situation in which \( \mathcal{P}_3 \) is already at location \( \cdot \).

Consequently, we strengthen \( \mathcal{A}_2 \) into

\[
0 = h \leftarrow [\mathcal{P}_3] \quad \forall \mathcal{A}_1 : \mathcal{A}_1 \mathcal{A}_2 : \mathcal{A}_3
\]

No! because in a real computation, if any process is at \( \mathcal{P}_3 \) then \( h \) must equal 0.

A. Pnueli

Deductive Verification of Reactive Systems, NYU, Fall, 2007
A.Pnueli

Trying Second Approximation:

\[
\begin{align*}
&\exists y \in \mathbb{S}^3 \cdot y = 0, \forall i \exists \mathbb{P}[i] \cdot t_{oc} = 0, \forall i \exists \mathbb{P}[i] \cdot t_{oc} = 4, \forall i \exists \mathbb{P}[i] \cdot t_{oc} = 3
\end{align*}
\]

Premise I2 is not valid. Counter-example = checking Premise I2.

Premise I1 is invalid. Checking Premise I1.

Try deductive verification of mutual exclusion.

In file scr3.pl, we place

\[
\begin{align*}
&\text{Running this script file, we obtain:} \\
&\ldots
\end{align*}
\]

\[
\begin{align*}
&\text{...}
\end{align*}
\]

In file scr3.pl, we place

\[
\begin{align*}
&\text{Trying Second Approximation:}
\end{align*}
\]

\[
\begin{align*}
&\exists y \in \mathbb{S}^3 \cdot y = 0, \forall i \exists \mathbb{P}[i] \cdot t_{oc} = 4, \forall i \exists \mathbb{P}[i] \cdot t_{oc} = 3
\end{align*}
\]

\[
\begin{align*}
&\text{A.Pnueli}
\end{align*}
\]

Lection 2: Verification of Invariance Properties
The offending transition originates at a state in which $P[3]$ is at location $\texttt{1}$, while $P[4]$ is at location $\texttt{3}$. Such a state is unreachable, because the range for which mutual exclusion is ensured includes $\texttt{1}$ together with $\texttt{3}$. Consequently, we strengthen $\varphi$ into:

$$((\texttt{1} \neq \texttt{4}) \land 0 = j \leftarrow [\texttt{3}]i) : \varphi$$

The resulting transition originates at a state in which $P[3]$ is at location $\texttt{3}$, while $P[4]$ is at location $\texttt{4}$. Strengthening $\varphi$
(\lnot \exists i A \land \exists i P[i] \land \mathcal{L} = 3 \rightarrow \exists j P[j] \lor P[i] \lor P[j] \lor \mathcal{L} = 4) \\
\therefore \mathcal{L} = 3 \lor \mathcal{L} = 4

\text{In light of this new counter-example, we strengthen into}
\text{the following.}

\text{Running this version, we obtain...}

\text{Premise I2 is not valid. Counter-example is unreachible because it has }
P[2] \text{ at location } 4 \text{ while } y = 1. \text{ It is thus necessary to extend the range for which } y = 0 \text{ to include also } 4. \text{ Consequently, we strengthen into}

\text{Trying next approximation:}

(\lnot \exists i A \land \exists i P[i] \land \mathcal{L} = 3 \rightarrow \exists j P[j] \lor P[i] \lor P[j] \lor \mathcal{L} = 4) \\
\therefore \mathcal{L} = 3 \lor \mathcal{L} = 4

\text{...
Once more: Try

Running this version, we obtain

Let ass := ass & ((P[i].loc = 3) -> y = 0);

In the `src5.pf` we replace

Once More: Try