We will now consider a method for finding inductive assertions for BDS's. This leads to the verification method of invisible invariants.
Automatic Generation of Invariants

A. Pnueli

Automatic Generation of Auxiliary Invariants

Goal.

1. Compute inductive assertion of the form $\forall i : (?)(\mu)$. 
2. Let $\mu$ be the assertion obtained from $?((\mu)_{i=1})$ by projecting away all the references to variables subscripted by indices other than $i$. 
3. Let $?((\mu)_{i=1})$ be the assertion obtained from $\mu$ by generalizing $i$. The candidate for inductive assertion is $\forall i : (?)(\mu)$. 

Unfortunately, $\forall i : (?)(\mu)$ is not inductive over $\text{MUTEX}(2)$. 

E.g. $\forall i : (?)(\mu)$. 

E.g. $\exists \gamma : (?)(\mu)$. 

E.g. $\forall i : (?)(\mu)$. 

E.g. $\forall i : (?)(\mu)$. 

Let $\gamma = 0 \vdash \gamma = H$ since $\exists i : (?)(\mu)$. 

$H = 0 \vdash \gamma = (0^{N})S$. 

$\exists i : (?)(\mu)$. 

1. Let $\Theta = \exists \gamma : (?)(\mu)$. 

Automatic Generation of Auxiliary Invariants

Lecture 1: Automatic Generation of Invariants
Compute Auxiliary Assertion of the Form $A : \phi$ for $\forall i : \phi$. We take

$$
\bigvee_{0}^{i} \bigwedge_{1}^{i} \phi
$$

index $i$.

1. Let $(\forall i : \phi) \land (i \geq 1)$ be the assertion characterizing all the reachable states of system $S$.

2. Let $d \phi \Theta = (\forall i : \phi)$ be the transition relation characterizing all the reachable states of system $S$.

3. For each $i \in \mathbb{N}$, let $[i] \mathcal{N}$ be the conjunction of boolean variables, and for each array variable $x \land \mathcal{A}$, let $[i] \mathcal{A}$ be the conjunction of boolean variables, for each index variable $k \in \mathbb{N}$ the conjunct $(k = i) \land (k = 1)$.

4. Let $\forall \mathcal{N} \phi$ be the transition abstraction relation, which contains for each finite-domain variable $x$, the conjunct $x = x$. Let $\forall \mathcal{N} \phi$ be the transition relation obtained from reach by preserving all the global variables, and porting all the properties of index $1$ to index $i$.

Since assertion $\phi$ is computed internally and immediately consumed, the user never gets to see it. This is why we refer to this method as verification by invisible invariants.

Deductive Verification of Reactive Systems, NYU, Fall, 2007
Example: MUTEX with 1-Index Assertion

Let \( P_t \) be the event that process \( t \) has acquired the lock.

Local trans: \( \text{next} (\text{loc}) = \text{next}(\text{loc}) \land (\forall t \in \{1, \ldots, N\}) \).
Computed assertion fails to be inductive (Premise 2 fails).

\[ \text{call inv}(\text{exec, phi, 1}); \]
\[ \text{print "in check mutual exclusion";} \]
\[ \text{compute inv}; \]
\[ \text{calc exec}; \]
\[ \text{end -- calc exec;} \]
\[ \text{end -- for i in 1...N;} \]
\[ \text{end -- for j in i+1...N;} \]
\[ \text{let exi = exec \& ((P[i].loc\notin\text{not in 3}) \& (P[j].loc\notin\text{not in 3}))}; \]
\[ \text{for j in i+1...N;} \]
\[ \text{let exi = i }; \]
\[ \text{to calc exec}; \]

File Continued
Compute Auxiliary Assertion of the form $\forall i \neq j : \psi(i, j)$

1. Let $\text{reach} := \Theta \circ \rho^*$ be the assertion characterizing all the reachable states of system $S(N_0)$.

2. Let $\rho_g$ be the transition (abstraction) relation, which contains for each finite-domain variable $x$, the conjunct $x' = x$.

3. For each $i \in 1..N_0$, let $\rho^{[1-i]}$ be the transition relation which contains, for each index variable $k : 1..N$, the conjunct $(k' = k) \equiv (k = 1)$, and for each array variable $y : \text{array}[1..N]$ of boolean the conjunct $y'[i] = y[1]$. Similarly define $\rho^{[2-i]}$.

4. Let $\psi(i, j) = \text{reach} \circ (\rho_g \wedge \rho^{[1-i]} \wedge \rho^{[2-j]})$ be the assertion obtained from $\text{reach}$ by preserving all the global variables, and porting all the properties of indices $1, 2$ to index $i, j$, respectively.

We take

1. $\forall i \neq j : \psi(i, j)$
Lecture 11: Automatic Generation of Invariants

Applying the Algorithm to MUTEX

Consider MUTEX with $N = 3$.

$\exists \tau \exists \nu: (\tau = [\ell] \nu \lor \tau = [i] \nu) \vdash \ell \neq i$.

MUTEX $(N)$. It also implies the property of mutual exclusion:

MUTEX $(5)$, and therefore, over all $\nu$.

Deductive Verification of Reactive Systems, NYU, Fall 2007

278
A. Pnueli

In TLV

In TLV
Example: A Modified Mutex

Local boolean where $x$ 1
Local natural where $N$ 1

$\{N\} I \Rightarrow \text{last} \leq I$

$[N..N] : \text{last} I \Rightarrow x$

$[N] I \Rightarrow 1 = x$ where

$[N] I : \text{last} I$ Release

$\{2\} I \Rightarrow x : \text{last} I$

$\{1\} I : \forall x : \text{last} I$

$\forall x : \text{last} I$ Critical

$\langle 1 \rangle I \Rightarrow \text{last} I$ Request

$x \forall i$ local

Searching for an inductive assertion, we obtained the calculated invariant

$(\forall i : [i] \nu \land 0 = x) \leftrightarrow \{\forall i \in [i] \nu : \forall A : \phi \}$

Exclusion:
The candidate assertion $\phi$ is inductive and also implies the property of mutual exclusion:

$(i = \text{last} \land 0 = x) \leftrightarrow \{\forall i \in [i] \nu : \forall A : \phi \}$

Lecture 11: Automatic Generation of Invariants

A. Pnueli

Searching for a $(i) \phi$ inductive assertion, we obtained the calculated invariant

$(\forall i : [i] \nu \land 0 = x) \leftrightarrow \{\forall i \in [i] \nu : \forall A : \phi \}$

Exclusion:
The candidate assertion $\phi$ is inductive and also implies the property of mutual exclusion:

$(i = \text{last} \land 0 = x) \leftrightarrow \{\forall i \in [i] \nu : \forall A : \phi \}$

Deductive Verification of Reactive Systems, NYU, Fall 2007
This time the auxiliary assertion is inductive:

\[
\text{End -- Func abs(reach', t')!}
\]

\[
\text{Return succ(trans', reach)!}
\]

\[
\begin{align*}
\text{next}(p[t].loc) &= \text{ref}(f).loc \\
\text{next}(tast)=t &= (\text{last}=f) \land (f < g) \land (\forall (f' < g) \text{ Func abs(reach', t')!})
\end{align*}
\]

Local trans : (next(y) = last=f) \land (next(y) = next(t) \land (f < g) \land (\forall (f' < g) \text{ Func abs(reach', t')!})

In TLU:

\[
\text{mutex-modINV1.pf we place:}
\]
Consider the following program Arbiter:

Example: Program Arbiter

\[
\begin{align*}
& (\forall j \neq i. A) : p \\
\end{align*}
\]
In the file arbiter.smv, we place the following:

```smv
MODULE Idle

  process Idle
  {
    VAR
      N : 6;
      DEFINE
        MODULE main

        MODULE Idle

        MODULE main
```
\begin{verbatim}
MODULE MA(k,N,r,g)
DEFINE rk := |for(i=1;i<=N;i=i+1) {i=k?r[i]:0};
VAR loc: 0..4;
ASSIGN init(loc) := 0; init(k) := 1;
next(loc) := case loc=0 & rk: 1;
loc=0: 4;
loc in {1,3,4}: (loc+1) mod 5;
loc=2 & !rk: 3;
1: loc;
endcase;
next(k) := loc=4? (loc mod N) + 1: k;
for(i=1;i<=N;i=i+1) {next(g[i]) := case i=k: g[i];
i=1: 1;
i=3: 0;
i=2: g[i];
i=4: g[i];
endcase;};

JUSTICE
loc != 0, loc != 1, loc != 3, loc != 4, ! (loc=2 & !rk)
end
\end{verbatim}
Lecture 11: Automatic Generation of Invariants

A. Pnueli

MODULE MC(r, g)
VAR loc: 0..5;
ASSIGN init(loc):=0; init(r):=0; init(g):=0;

next(loc):=case
loc in {1, 3, 4}: (loc+1) mod 6;
loc=0: {0, 1};
loc=2 & g: 3;
loc=5 & !g: 0;
1: loc;
esac;

next(r):=case
loc=1: 1;
loc=4: 0;
1: r;
esac;

JUSTICE

loc != 1, !(loc = 2 & g), loc != 3, loc != 4, !(loc = 5 & !g)

Deductive Verification of Reactive Systems, NYU, Fall 2007

285
Applying the Invisible Invariants Method

...
Lecture 11: Automatic Generation of Invariants

A.Pnueli

To compute prop:

Let exc := 1;

For (i in 1...N)

For (j in i+1...N)

Let exc := exc & !(Cl[i].loc=3 & Cl[j].loc=3);

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

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

End--compute_prop;

compute_inv;

compute_prop;

Print "Check mutual exclusion\n";

Call inv(exc, phi, 1);

Deductive Verification of Reactive Systems, NYU, Fall, 2007