As previously stated, we would like to separate the two main activities (and challenges) associated with deductive verification:

- Establishing the validity of the premises of the relevant rule.
- The invention of auxiliary constructs.

In this lecture, we will present one of the simplest methods for deciding validity of the premises — the small model theory.
A parameterized system is a parallel composition, such as

\[ S(N) = P[1] \parallel \cdots \parallel P[N] \]

where \( N > 1 \) is a parameter and \( P[1], \ldots, P[N] \) are finite-state processes.

\[ [N]p \parallel \cdots \parallel [1]p \parallel \mathcal{O} = (N)S \]

or

\[ [N]p \parallel \cdots \parallel [1]p = (N)S \]

Uniform Verification: Establish in one verification effort that all instances of the system satisfy a property. That is, for every \( \phi \), for every \( \phi \),

\[ \phi \models (N)S \]
So far, only partial results have been obtained by Emerson, Ip, and Namjoshi.

$\phi \models (0^N S) \iff \phi \models (N S : N ^ A)$

That much research has been expended in trying to identify a cutoff value for $N$ such that the absolute correctness of the system, this is insufficient.

Much research has been uniquely uncovered most of the existing bugs. However, if we wish to establish

To the extent that formal verification is viewed as a debugging tool, then this

they all come out valid. What can we conclude?

Of course, we can check separately $\phi \models (\forall S), \phi \models (\exists S), \phi \models (2 S)$, etc. Suppose

Non-Mathematical Induction
It only remains to check the validity of the premises of rule INV.

\[ \models (x + [\ell]_{\exists y} y \rightarrow a \ell + [\ell]_{\exists y} y \rightarrow a \ell) : \ell \neq \ell \land A \]

Therefore it into the inductive assertion:

The semaphore instructions "request x" and "release x" appearing in the program stand, respectively, for when x = 0 do x = x and when x = 1 do x := 0 and x := 1.

Suppose we wish to establish the invariance of the assertion

\[ (0 = x) \leftarrow [1]_{\exists y} y \rightarrow a \ell : p \]

The example invariants stand, respectively, for when x = 0 do x = x and when x = 1 do x := 0 and x := 1.

\[ [N]_{\exists y} y \rightarrow a \ell : p \]
Checking the Premises for MUX-SEM

We can check the validity of the premises by using BDD techniques, e.g. using TLV in the `mux5 pf` file. We write:

```
Tocl phi;
let phi := 1;
for (i in 1...N)
  for (j in i+1...N)
    let phi := phi & ((P[i].loc in 2..3) + (P[j].loc in 2..3) + (P[i].loc in 2..3) + (P[j].loc in 2..3));
end--for (j in i+1...N)
end--for (i in 1...N)
end--Tocl phi;
```

We can check the validity of the premises by using BDD techniques, e.g. using TLV.
Premise Checking for MUX-SEM

Continued

Let \( p := ((P[1].loc=2) \rightarrow (x=0)); \)

\[
\text{calc-phi; Let counter:=} \Theta & !\phi; \text{-- Check Premise 3 } \\
\text{end -- If (counter) } \\
\text{Get counter := } \phi & \rho & !\text{next(\phi}); \text{-- Check Premise 2 } \\
\text{end -- If (counter) } \\
\text{Get counter := } \Theta & \eta & l \phi; \text{-- Check Premise 1 } \\
\text{calc-\phi;} \\
\text{Get p := (p)[1].loc=2} \rightarrow (x=0)?
An $R$-assertion is a Boolean combination of atomic formulas.

An $R$-assertion is built out of Boolean $R$-assertions to which we apply further Boolean operations and quantification over index variables. For example:

$$[i]f_i \neq [f]f_i \lor N \geq i \lor i \geq j : i \in (N \geq j \geq i) : ?A$$

Note that the first formula can be viewed as an abbreviation of the second formula.

We define an index term to be an index variable or one of the constants $1, N$.

An atomic formula is a Boolean variable $x_i$, where $x_i$ is an array variable, or a Boolean term to which we apply further Boolean operations.

We define an index term to be an index variable or an array $[N \ldots I]$, where $N$ and $I$ are considered to be constants of type index.

Consider the following signature:

- Boolean arrays $[N \ldots I]$, where $N$ and $I$ are considered to be constants of type index.
- Boolean variables $x_i$, where $i$ is an index variable.
- Boolean $R$-assertions $\phi$, where $\phi$ is a Boolean $R$-assertion.
- Boolean $R$-assertions are $\forall x_i : \phi$.
By every model \( \phi \) satisfies \( \mathcal{R} \)-assertion \( (u) \mathcal{W}_n \). To find out whether \( \phi \) is said to be valid if it is satisfied given an \( \mathcal{R} \)-assertion and a model \( \mathcal{W}_n \), we can evaluate over \( \mathcal{W}_n \), (1, 1, 0) \( = \) \( x \). \( x \) = \( n \).

For example, a \( 3 \)-model for this formula can be specified by the assignment:

- Each boolean variable \( x \) is assigned a value from \( \{0, 1\} \).
- Each array variable \( a \) is assigned a boolean array of size \( n \).
- Each index variable \( i \) is assigned a value from \( \{1 : n\} \).
- The constants \( 1 \) and \( n \).

Thus, an adequate model over the vocabulary of the \( \mathbb{AER} \)-assertion bounded in \( n \). In general, it is not necessary to assign values to index variables which appear following interpretation to the elements appearing in \( \phi \) over the vocabulary of \( \mathcal{W}_n \). An \( n \)-assertion. An \( \mathcal{R} \)-assertion \( (u) \mathcal{W}_n \).
A Small Model Theorem

Let \( i_1, \ldots, i_k \) and \( j_1, \ldots, j_m \) be a closed AEFR-assertion.

Claim 1.2 [Small Model Theorem]

Let \( \phi \) be a closed AEFR-assertion.

For each index term \( t \), we let

\[
(\forall i_1 > \cdots > i_k) N \begin{cases}
I & \text{if } \forall i \in [1..N] \\
I - u = (N) & \text{else}
\end{cases}
\]

Proof Sketch:

Assertion is valid iff it is satisfied by all models \( N(\nu) \) for \( \nu \geq k + 2 \).
For each array $\mathcal{A}$ we let $A$. Pnueli

$\text{it is not too difficult to show that } \mathcal{M} \text{ is also a satisfying model.}$

$[1 - n] (\forall) \mathcal{M} \text{ else } [n] (\forall) \mathcal{M} \text{ then } \mu > n : \forall \mathcal{M} = (\forall) \mathcal{M}$
Example
Typically, a bounded-data parameterized system is a parallel composition

\[ [N]P \parallel \cdots \parallel [1]P \]

of a bounded-datadiscrete system. We consider BDS.

A bounded-datadiscrete system (BDS) is an FDS whose system variables have the form:

1. \[ i_1, \ldots, i_b : [1 \cdots N] \]
   - The index variables.

2. \[ y_1, \ldots, y_c : \text{array}[1 \cdots N][1 \cdots N] \]
   - The index variables.

3. \[ x_1, \ldots, x_a : \text{boolean} \]
   - Any finite-domain type can be encoded by boolean.

4. \[ N : \text{integer} \text{ where } 0 < N \]
   - A bounded-data discrete system (BDS) is an FDS whose system variables have the form:

   The systems we consider: BDS.
Lecture 10: Small Model Theory

Example: MUTEX

\[
\begin{align*}
\text{release } x & : 3 \\
\text{Critical } x & : 2 \\
\text{request } x & : 1 \\
\text{Non-Critical} & : 0 \\
\text{loop forever do} & \\
1 = x & : \text{boolean where } I \\
1 < N & : \text{natural where } I \\
\end{align*}
\]

Deductive Verification of Reactive Systems, NYU, Fall 2007
BDS usually contain the system array variable
\[ \text{[N]} \] which represents the program counter in each of the processes.

\[
\begin{align*}
\nu [t] & = [t], \nu : y \neq t \wedge \\
0 = [t], \nu & \lor \exists \ z = \nu [y] \nu \lor \exists \ z = \nu [y] \nu \\
0 = \nu & \lor \exists \ z = \nu [y] \nu \lor \exists \ z = \nu [y] \nu \\
\nu & \land \nu [y] \nu = \nu [y] \nu \\
0 = \nu [y] \nu & : y \in \Omega : d \\
\nu & : \Theta \\
\end{align*}
\]

Array of \( [N] \) where \( I = x \) and natural where \( I < N \).
Establishing the validity of the premises.

For BDS’s, this solves the 2nd task associated with deductive verification —

it is sufficient to check the validity of all premises on models of size $n \leq 5$.

Thus, this premise has 3 universally quantified variables. According to Claim 12,

$$ (\forall, ?) \phi \leftarrow ((\forall, i) R \lor (\forall, n) \phi) : \forall, ? A, \forall, n A $$

Moving quantifiers to the front, this can be rewritten as

$$ (\forall, i) \phi : \forall, ? A \leftarrow ((\forall, i) R : \forall, A \forall A) \lor ((\forall, n) \phi : \forall, n A) $$

The most complex verification condition, 12, is then:

where $R$-assertion.

$$ (\forall, i) \phi : \forall, ? A = \phi $$

Consider the case that property $d$ and the auxiliary assertion both have the form

Deciding the Verification Conditions
To verify mutual exclusion of program MUTEX, we take

\( L \geq \text{mutex.pf} \)
Consider the following program: ARBITER:

\( \text{Example: Program Arbiter:} \)

\[
(\exists i \in \{0, 1\} : i \neq 1) \quad : \mathcal{P}
\]

for which we wish to prove

\[
(\exists i \in \{0, 1\} : i \neq 1) \quad : \mathcal{P}
\]

\[
\begin{align*}
0 &= \begin{array}{ll}
\text{Critical} & : i \\
\text{Non-Critical} & : i \\
\text{wait} & : i \\
\text{wait} & : i \\
\text{critical} & : i \\
\text{critical} & : i \\
\text{loop forever} & : i \\
\end{array}
\end{align*}
\]

\[
\begin{align*}
1 \oplus \varphi &= \varphi : \mathcal{N} \\
1 &= \begin{array}{ll}
\text{wait} & : \varphi \\
\text{wait} & : \varphi \\
\text{if then} & : \varphi \\
\text{loop forever} & : \varphi \\
\text{array of boolean where} & : \mathcal{N} \\
\end{array}
\end{align*}
\]
An appropriate auxiliary inductive assertion is given by:

$$
\begin{align*}
&\forall i : [i] \nu \quad r[i] = g[i] = 0 \\
&\forall i : [i] \nu \quad r[i] = g[i] = 0 \quad \forall i : [i] \nu \quad r[i] = g[i] = 0 \quad \forall i : [i] \nu \quad r[i] = g[i] = 0
\end{align*}
$$
Progressing to Richer Signatures

Up to now, we only considered systems with array signatures

The method can be extended to signatures

Declarative verification of Reactive Systems, NYU, Fall, 2007

A. Pnueli
Example: A Finitary Version of the Bakery Algorithm
In the file bakery.smv we place:

The SMV Representation of Bakery

A. Pnueli
Lecture 10: Small Model Theory

A.Pnueli

loc: 0..4;

ASSIGN

init(loc) := 0;
init(y[i]) := 0;
next(loc) := case
loc = 0: {0, 1};
loc = 1 & cond1: 2;
loc = 2 & cond2: 3;
loc = 3, 4: (loc + 1) mod 5;
1: loc;
esac;

for (j = 1; j <= N; j = j + 1) {
next(y[j]) := case
j != i: y[j];
loc = 1 & cond1: u;
loc = 4: 0;
1: y[i];
esac;
}

JUSTICE

loc != 1, !(loc = 2 & cond2), loc != 3, loc != 4

Deductive Verification of Reactive Systems, NYU, Fall, 2007

264
MODULE CR(i, y, u, N)
DEFINE
cond: = (u < y[i]) & (\(\forall j \leq N \), y[j] < u | y[i] \leq y[j])
ASSIGN
for (j = 1; j \leq N; j++)
next(y[j]) := case
\(j = i \land \text{cond} \) : u;
1 : y[j];
endcase
for
ASSIGN

\(\{[\varphi] \Box : \quad I\}
\}
\(\forall n : \text{cond}
next = ([\varphi] \Box (i + 1) \Rightarrow [\varphi] \Box i)
\) for \(\forall \varphi \land ([\varphi] \Box \Rightarrow n) = : \text{cond}
\)
DEFINE
MODUL (i, y, \Box, n, N)
Similarly, for $\eta = N$.

$\forall h^0 \exists h^0 \eta : h^0 \leq \eta$

Can be rewritten as

$(\exists h^0 \eta \leq \eta) \land (\exists h^1 \eta \geq \eta) : h^0 = 0 \land h^1 = 1$

Additional extensions are introduced by allowing

$((1H + 1I + q)\epsilon + 2H + 2I + p = \epsilon \frac{N}{0} \land 1H + 1I + q = \frac{N}{0})$

where $(\frac{\epsilon \frac{N}{0}}{0} \frac{\epsilon \frac{N}{0}}{0}) \supseteq (\frac{\epsilon \frac{N}{0}}{\epsilon \frac{N}{0}})$ for all $\frac{\epsilon \frac{N}{0}}{\epsilon \frac{N}{0}}$ are valid over $\frac{\epsilon \frac{N}{0}}{\epsilon \frac{N}{0}}$ $\frac{\epsilon \frac{N}{0}}{\epsilon \frac{N}{0}}$ $\frac{\epsilon \frac{N}{0}}{\epsilon \frac{N}{0}}$ $\frac{\epsilon \frac{N}{0}}{\epsilon \frac{N}{0}}$ $\frac{\epsilon \frac{N}{0}}{\epsilon \frac{N}{0}}$ $\frac{\epsilon \frac{N}{0}}{\epsilon \frac{N}{0}}$

Claim 13. The premises of rule INV are valid over $\frac{\epsilon \frac{N}{0}}{\epsilon \frac{N}{0}}$ $\frac{\epsilon \frac{N}{0}}{\epsilon \frac{N}{0}}$ $\frac{\epsilon \frac{N}{0}}{\epsilon \frac{N}{0}}$ $\frac{\epsilon \frac{N}{0}}{\epsilon \frac{N}{0}}$ $\frac{\epsilon \frac{N}{0}}{\epsilon \frac{N}{0}}$ $\frac{\epsilon \frac{N}{0}}{\epsilon \frac{N}{0}}$

Adjusting the Bounds for the Signature

$\langle \text{type} \ 1 \ bo \text{ol}, \ \text{type} \ 1 \ \text{type} \ 2 \rangle$

A. Pnueli
Applying bottom-up techniques, we propose the following assertion:

\[(0 = \bar{i} \leftrightarrow \exists i \forall \exists j \neg \bar{a} t) : \bar{i} \neq \bar{j} \]

Applying bottom-up techniques, we propose the following assertion:

\[([\ell] \bar{i} > [\ell] \bar{i} \land 0 = [\ell] \bar{i}) \leftarrow \exists i \forall \exists j \neg \bar{a} t \]

Using the affirmed invariant heuristic and propagation techniques we derive the following assertion:

\[A \quad \text{Deductive Verification of BAKERY Using Small Model Theory} \]
Consequently, we place in the badly named "check mutual exclusion"
\begin{verbatim}
calc_phi;
    phi := 1;
    exc := 1;
    for (i in 1...N-1)
        phi := phi & (C[i].loc in 0..1 <-> y[i] = 0);
    for (j in 1...N-1)
        if (j != i)
            phi := phi & (C[i].loc > 2 -> (y[j] = 0 | y[i] < y[j]));
            exc := exc & ((C[i].loc not in 3) | (C[j].loc not in 3));
    end if (j != i)
    end for (j in 1...N-1)
    end for (i in 1...N-1)
end calc_phi;
\end{verbatim}

Check mutual exclusion.

Call inv(exc, phi, 1);
It is possible to extend the approach to any stratified signature, allowing type $\text{type}_1; \ldots; \text{type}_k$ as long as $i < j$. This leads to:

\[
\begin{align*}
\text{Further Extensions}
\end{align*}
\]
In some cases, we have to move to unstratified systems. This is the case of

Unstratified Systems
<table>
<thead>
<tr>
<th>System</th>
<th>$T_1$</th>
<th>$T_2$</th>
<th>$T_3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>3-stages Pipeline</td>
<td>6</td>
<td>20.66</td>
<td>29.59</td>
</tr>
<tr>
<td>Cache</td>
<td>4</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>S. German's Cache</td>
<td>5</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>Szymanski</td>
<td>4</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>Token-Ring</td>
<td>5</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>Bakery</td>
<td>5</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>Peterson (N)</td>
<td>6</td>
<td>79</td>
<td>1211</td>
</tr>
<tr>
<td>MUX-SEM</td>
<td>7</td>
<td>7</td>
<td>7</td>
</tr>
<tr>
<td>Illinois's Cache</td>
<td>133.04</td>
<td>1.47</td>
<td>1.01</td>
</tr>
<tr>
<td>4</td>
<td>10.72</td>
<td>10.21</td>
<td>0.01</td>
</tr>
<tr>
<td>5</td>
<td>9</td>
<td>1.01</td>
<td>0.01</td>
</tr>
<tr>
<td>0</td>
<td>1211</td>
<td>79</td>
<td>6</td>
</tr>
<tr>
<td>2</td>
<td>20.66</td>
<td>29.59</td>
<td>5</td>
</tr>
<tr>
<td>3</td>
<td>133.04</td>
<td>1.47</td>
<td>0.01</td>
</tr>
<tr>
<td>4</td>
<td>10.72</td>
<td>10.21</td>
<td>0.01</td>
</tr>
<tr>
<td>5</td>
<td>9</td>
<td>1.01</td>
<td>0.01</td>
</tr>
<tr>
<td>6</td>
<td>79</td>
<td>1211</td>
<td>6</td>
</tr>
<tr>
<td>7</td>
<td>20.66</td>
<td>29.59</td>
<td>5</td>
</tr>
<tr>
<td>8</td>
<td>133.04</td>
<td>1.47</td>
<td>0.01</td>
</tr>
<tr>
<td>9</td>
<td>10.72</td>
<td>10.21</td>
<td>0.01</td>
</tr>
<tr>
<td>10</td>
<td>9</td>
<td>1.01</td>
<td>0.01</td>
</tr>
</tbody>
</table>

Summary of Results

Here is a table summarizing the results:

- Time to check premises over $S^0(N)$:
- Time to compute candidate assertion:
- Time to compute reach:

Here are the results for various systems:

<table>
<thead>
<tr>
<th>System</th>
<th>$T_1$</th>
<th>$T_2$</th>
<th>$T_3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>3-stages Pipeline</td>
<td>6</td>
<td>20.66</td>
<td>29.59</td>
</tr>
<tr>
<td>Cache</td>
<td>4</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>S. German's Cache</td>
<td>5</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>Szymanski</td>
<td>4</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>Token-Ring</td>
<td>5</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>Bakery</td>
<td>5</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>Peterson (N)</td>
<td>6</td>
<td>79</td>
<td>1211</td>
</tr>
<tr>
<td>MUX-SEM</td>
<td>7</td>
<td>7</td>
<td>7</td>
</tr>
<tr>
<td>Illinois's Cache</td>
<td>133.04</td>
<td>1.47</td>
<td>1.01</td>
</tr>
<tr>
<td>4</td>
<td>10.72</td>
<td>10.21</td>
<td>0.01</td>
</tr>
<tr>
<td>5</td>
<td>9</td>
<td>1.01</td>
<td>0.01</td>
</tr>
<tr>
<td>6</td>
<td>79</td>
<td>1211</td>
<td>6</td>
</tr>
<tr>
<td>7</td>
<td>20.66</td>
<td>29.59</td>
<td>5</td>
</tr>
<tr>
<td>8</td>
<td>133.04</td>
<td>1.47</td>
<td>0.01</td>
</tr>
<tr>
<td>9</td>
<td>10.72</td>
<td>10.21</td>
<td>0.01</td>
</tr>
<tr>
<td>10</td>
<td>9</td>
<td>1.01</td>
<td>0.01</td>
</tr>
</tbody>
</table>

Deductive Verification of Reactive Systems, NYU, Fall, 2007