Synthesis of Finite State Machines

Chris J. Myers
Lecture 16
Chapter 8

An Incompletely Specified Moore Machine

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Verification/Testing

State Minimization

State Encoding

Logic/Timing Optimization

Simplified FSM Design Flow

Another Incompletely Specified Moore Machine

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3</td>
<td>2</td>
<td>1</td>
</tr>
</tbody>
</table>
Reduced Machine for Previous Example

Machine Obtained by State Splitting

A Flow Table and its Compatibility Table

Minimization of incompletely specified Machines

Solve the unique covering problem.

Find the prime components.

Find the maximal components.

Find all pairs of comparable states.
Finding the Maximal Complements

\[ \frac{x^3}{x + \frac{x^3}{x + \frac{x}{x^4}}} \]

Converting POS formula to SOP formula (e.g., complete sum):

\((x + \frac{x^3}{x + \frac{x}{x^4}})(x + \frac{x^3}{x + \frac{x}{x^4}})\)

Where a POS formula to express complementarity conditions.

If \( y \) and \( z \) are incompatible, then no maximal complement.

Constructing a Complementarity Table

<table>
<thead>
<tr>
<th>( x )</th>
<th>( y )</th>
<th>( z )</th>
<th>( w )</th>
<th>( u )</th>
<th>( v )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Finding the Prime Complements

Exist a \( C \) and \( T \) such that \( C \not=\) and \( T \not=\). A compatible \( C \) with a class set \( T \). Is prime if there does not exist a class set of \( \{ 1, 2 \} \). Example: class set of \( \{ 1, 2 \} \) is prime if there does not exist a class set of \( \{ 1, 2 \} \). Selection of equivalent compatible classes. The set of compatible implied by a compatible is the class set. Some share pairs compatible only if other pairs are needed.
Setting up the Covering Problem

Procedure for the Previous Flow Table

Prime Companions for the Previous Flow Table

Compatible Table for the Previous Flow Table
The image contains a page from a document with mathematical content and tables. Here is a plain text representation of the content:

### Reduced Flow Table

<table>
<thead>
<tr>
<th>0'6</th>
<th>0'6</th>
<th>1'6</th>
<th>1'4</th>
<th>1'2</th>
<th>1'0</th>
<th>0'8</th>
</tr>
</thead>
<tbody>
<tr>
<td>0'6</td>
<td>1'1</td>
<td>1'0</td>
<td>1'1</td>
<td>0'1</td>
<td>0'0</td>
<td>0'8</td>
</tr>
</tbody>
</table>

### Coveting Problem for the Example

\[ I = (a + b + c)(b + c + d)(b + c + d) \]

### Heuristic Choices for Next State Entries

<table>
<thead>
<tr>
<th>0'6</th>
<th>0'6</th>
<th>1'6</th>
<th>1'4</th>
<th>1'2</th>
<th>1'0</th>
<th>0'8</th>
</tr>
</thead>
<tbody>
<tr>
<td>0'6</td>
<td>1'1</td>
<td>1'0</td>
<td>1'1</td>
<td>0'1</td>
<td>0'0</td>
<td>0'8</td>
</tr>
</tbody>
</table>

---

The text appears to be related to computational or algorithmic problems, possibly involving graph theory or constraint satisfaction problems. The tables and equations suggest a focus on flow or coveting in some context, possibly within a larger algorithmic framework.
Given a Boolean formula \( f \) in matrix form, find a subset \( S \) of columns of minimum cost such that for every row \( i \) if either

\[
\left( x_i \in S \cup \{ f \} \right) \vee (0 = f) : f 
\]

there exists a solution.

**Definition 8.2.1** A row index \( i \) is essential for \( f \) if \( i \) is a row index where only one coefficient is different from zero.

**Definition 8.2.2** For a row \( i \) if \( f \) is unsatisfiable, \( f \) if \( \exists! \) is satisfied, i.e., it if \( f \) is unsatisfiable.

**Definition 8.2.3** A row if \( f \) is unsatisfiable otherwise.

**Row Dominance**

\[
\begin{bmatrix}
0 & 1 & 0 & 1 \\
1 & 1 & 1 & 1 \\
0 & 0 & 1 & 1 \\
\end{bmatrix}
\]

\[
\begin{bmatrix}
0 & 1 & 0 & 1 \\
1 & 1 & 1 & 1 \\
0 & 0 & 1 & 1 \\
\end{bmatrix}
\]

\[
\begin{bmatrix}
1 & 0 & 1 & - \\
1 & 1 & 0 & - \\
- & 1 & - & - \\
\end{bmatrix}
\]

\[
\begin{bmatrix}
1 & 0 & 1 & - \\
1 & 1 & 0 & - \\
- & 1 & - & - \\
\end{bmatrix}
\]
\[ \begin{bmatrix} 1 & 0 & - & - \\ \end{bmatrix} \]
\[ \begin{bmatrix} - & - & 1 \\ \end{bmatrix} \]
\[ \begin{bmatrix} 0 & 1 & 0 & 1 \\ \end{bmatrix} \]

\[ \begin{bmatrix} 1 & - & 0 & - \\ - & 1 & I & - \\ - & I & 1 & I \\ \end{bmatrix} \]

In linear case, only consider classes w/o compromised threats.

Maximal Independent Set

\[ \begin{bmatrix} 0 & 0 \\ 0 & 1 \\ 1 & 0 \\ 1 & 1 \\ \end{bmatrix} \]

Infeasible Problems

\[ f'f = 1 \]

Consider only rows with \( f'f = 1 \).

Columns that intersect many short rows are benched.
Therefore, all proposed encoding methods are redundant.

\[ \log (d - \gamma) / \log (1 - \gamma) \]

If permutation or complementation is the same assignment

\[ \log (d - \gamma) / \log (1 - \gamma) \]

If one uses \( d \) bits to encode \( d \) shares, these are

**State Encoding**

An Example of Reductions

**Fanout-Oriented Algorithm**

\[ \text{fanout} \]

\[ \epsilon = \epsilon \hat{x} \hat{y} + \epsilon \hat{x} \hat{y} \]

\[ \text{fanout} = \text{fanout} + \text{fanout} \]

\[ \text{fanout} \]

\[ \text{fanout} \]

\[ \text{fanout} \]

**Fanout-Oriented Algorithm**

\[ \epsilon = \epsilon \hat{x} \hat{y} + \epsilon \hat{x} \hat{y} \]

\[ \text{fanout} = \text{fanout} + \text{fanout} \]

\[ \text{fanout} \]

\[ \text{fanout} \]

**Fanout-Oriented Algorithm**

\[ \epsilon = \epsilon \hat{x} \hat{y} + \epsilon \hat{x} \hat{y} \]

\[ \text{fanout} = \text{fanout} + \text{fanout} \]

\[ \text{fanout} \]

\[ \text{fanout} \]

**Fanout-Oriented Algorithm**

\[ \epsilon = \epsilon \hat{x} \hat{y} + \epsilon \hat{x} \hat{y} \]

\[ \text{fanout} = \text{fanout} + \text{fanout} \]

\[ \text{fanout} \]

\[ \text{fanout} \]
The image contains mathematical expressions and diagrams related to formal language theory and automata theory. The expressions are:

- \( t = [1] \cdot [0] + \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix} = \varepsilon \cdot t \).

- \( f Z' + f S \cdot S' = N \).

- \( 1 \varepsilon = 1 \varepsilon 0 1 \varepsilon \).

- \( 0 \varepsilon = Z \).

- \( 1 X = Z 0 1 1 \varepsilon = S \).

The diagrams illustrate FSMs and their associated graphs.
\[(u^{x_1} \cdots u^{x_n})v^f = \forall x \quad (u^{x_1} \cdots u^{x_n})v^f = \exists x \quad (u^{x_1} \cdots u^{x_n})v^f = \exists x \quad (u^{x_1} \cdots u^{x_n})v^f = \exists x \quad (u^{x_1} \cdots u^{x_n})v^f = \exists x\]

Decomposition and Encoding

\[x^{\forall \theta} = z \quad x^{\exists \theta} = \exists x \quad x^{\forall \theta} + x^{\exists \theta} = \exists x\]

An Assignment Dependent on the Param Algorithm

\[(\tau_{\mu})^1 + s \equiv s \quad \text{or} \quad (1_{\mu}^1 + s) \equiv s \quad \tau_{\mu} = u\]

The variable \(s\) is a subsequence in \(S\) if its length is \(1_{\mu}^1 \equiv s\) and \((1_{\mu}^1) \equiv s\) and \((\tau_{\mu} \cdot 1_{\mu}) \equiv s\)

\[
\{\exists' \cdot z \cdot 1\} \supset \{1 \cdot 3 \cdot z \cdot 1\} \\
\exists_{\mu} \subseteq 1_{\mu} \\
\{1 \cdot 3 \cdot z \cdot 1\} = 0 \\
\{1 \cdot 4 \cdot 3 \cdot z \cdot 1\} = S
\]

Partitions as a Partial Order

\[
\{\exists' \cdot z \cdot 1\} = \{\exists' \cdot z \cdot 1\} \\
= \{\exists' \cdot z \cdot 1\} \\
\text{Example} \quad \{x, y\} \neq 0 \\
\{x, y\} = \{x, y\} \\
\text{A partition } P \text{ on a set } S \text{ is a collection of disjoint subsets of } S
\]
Theorem 8.41 A FSN has a non-trivial parallel decomposition if there exist two non-trivial S.P. partitions

\[ \text{If } s \in S, \text{ then } \exists (s', s) \in \mathcal{S} \text{ such that } s' \cap s = \emptyset. \]

\[ \text{Definition: A partition } \pi \text{ of the set } S \text{ of an FSN is said to be partition with substitution property if } \]

\[ \forall \pi' \subseteq \pi \text{ and } \pi'' \subseteq S \text{ we have } \pi' \cap \pi'' = \emptyset. \]

\[ \{r, 2\} = \Pi \{v, 1\} = \Pi \{s', 0\} = \Pi \{s', 3\} = \Pi \{s, 0\} = \Pi \]

\[ \{s, 0\} = \Pi \{s, 0\} = \Pi \]

\[ \text{Example FSN with Parallel Decomposition} \]

\[ \text{Example FSN with Parallel Decomposition} \]

\[ \text{Example FSN with Parallel Decomposition} \]

\[ \text{Example FSN with Parallel Decomposition} \]
Construction of the Dependent Component

\[
\begin{array}{ccc|ccc|ccc}
\hline
1 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 \\
0 & 1 & 0 & 1 & 1 & 1 & 1 & 0 & 1 \\
0 & 0 & 1 & 1 & 1 & 1 & 0 & 1 & 0 \\
\hline
\end{array}
\]

Construction of the Dependent Component

\[
\begin{array}{ccc|ccc|ccc}
\hline
1 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 \\
0 & 1 & 0 & 1 & 1 & 1 & 1 & 0 & 1 \\
0 & 0 & 1 & 1 & 1 & 1 & 0 & 1 & 0 \\
\hline
\end{array}
\]

Computation of the S.P. Partitions

\[
\begin{align*}
\{1, 4, 2, 3\} &= \nu_5 \\
\{1, 2, 3, 4\} &= \nu_4 \\
\{1, 1, 5, 2, 3\} &= \nu_3 \\
\{1, 1, 2, 4, 5\} &= \nu_2 \\
\{1, 1, 1, 2, 3, 5\} &= \nu_1 \\
\{1, 1, 1, 1, 2, 3, 5\} &= \nu_0 \\
\end{align*}
\]

Computation of the S.P. Partitions

\[
\begin{array}{ccc}
3 & 1 & 0 \\
2 & 3 & 1 \\
1 & 2 & 0 \\
\end{array}
\]

Computation of the S.P. Partitions

\[
\begin{array}{ccc}
3 & 1 & 0 \\
2 & 3 & 1 \\
1 & 2 & 0 \\
\end{array}
\]
If \((\pi,\mu)\) is a partition pair, then \(\pi \neq \mu\) has the S.P.

\(\forall a \in I \implies \phi(a) = s\)

A partition pair \((\pi,\mu)\) is an ordered pair of partitions s.t.

\(\text{Partition Pairs}\)

\[
\begin{align*}
\ell & = z \\
\ell \pi + \ell \mu & = \ell \\
\tau \pi + \tau \mu & = \tau \\
1 & 1 \leftarrow 4 \\
0 & 1 \leftarrow 3 \\
1 & 0 \leftarrow 2 \\
0 & 0 \leftarrow 1 \\
\end{align*}
\]

\((\{1,2,3\},\{1,2,3\}) = (\ell,\tau)\)

\(\text{Encodings Based on Partition Pairs}\)

\(\text{S.P. Partition Lattice}\)

\(\text{Encodings Based on Partition Pairs}\)
Summary

- Described relationship between partitions and encoding
- Introduced an approach to FSR encoding
- Proposed method for minimizing incompletely specified

Scheme for the Decoding of the Machine