Synthesis and Verification of Finite State Machines

Sungho Kang

Yonsei University
Outline

- Minimization of Incompletely Specified Machines
- Binate Covering Problem
- State Encoding
- Decomposition and Encoding
Synthesis of Practical FSMs

• We have learned basic methods for minimizing, encoding, checking equivalence, and synthesizing circuits for realizing completely specified FSMs
• Now we must learn to deal with the more practical case of incomplete specification
• Our goal is thus to find a least cost circuit that satisfies a partial specification
Use don’t-cares to merge states. Merged states must have same output sequences.

Flow Table

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>2</td>
<td>-</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
</tr>
</tbody>
</table>

Compatibility Table

Note each constraint represents pair (incompatibility)
Strategy

- Derive all prime sets of compatible states
- Solve a covering problem to obtain minimum states.
Compatibility Constraints

Compatibility relation: conjunction of constraints (one for each “X”)

Note each constraint represents pair (incompatibility)

\[
C(x) = (x'_1 + x'_3)(x'_1 + x'_4)(x'_2 + x'_4) = (x'_1 + x'_3x'_4)(x'_2 + x'_4) = x'_1x'_2 + x'_3x'_4 + x'_1x'_4
\]

\[
(x'_1 + x'_3) \iff (x'_1 \Rightarrow x'_3) \iff (x'_3 \Rightarrow x'_1)
\]
Computing the Maximal Compatibles

By recursive multiplication method, like computing the Complete Sum:

\[ C(x) = (x'_1 + x'_3)(x'_1 + x'_4)(x'_2 + x'_4) \]
\[ = (x'_1 + x'_3 x'_4)(x'_2 + x'_4) \]
\[ = x'_1 x'_2 + x'_1 x'_3 + x'_2 x'_3 x'_4 + x'_1 x'_4 \]
\[ = x'_1 x'_2 + x'_3 x'_4 + x'_1 x'_4 \]

The (complete) constraint sums are multiplied out, dropping absorbed terms when they arise.
Computing the Maximal Compatibles

\[ x'_1 x'_2 + x'_3 x'_4 + x'_1 x'_4 \]

\[ x'_1 x'_2 \Rightarrow \{ s_3, s_4 \} \]

- Maximal compatibles are “Prime”.

(No superset of these state sets are also pairwise compatible).

\[ e.g., \ x'_1 \Rightarrow \{ s_2, s_3, s_4 \} \text{ but } \{ s_2, s_4 \} \text{ are not compatible} \]
Prime Compatibles

- Unfortunately, some subsets of the maximal compatibles pairs are also prime compatibles.
- Because, selection of one compatible pair may imply selection of other compatible pairs.

\[ \{S_3, S_4\} \implies \{S_1, S_2\} \]
Defining Prime Compatibles

- A compatible $C_s$ is prime if and only if there is no other compatible $C_q$ which contains it or whose class set $\Gamma_q$ contains class set $\Gamma_s$ of $C_s$. That is, $C_s$ is prime if and only if

\[ \neg \exists C_q \text{ such that } \]
\[ (1) \quad C_q \supseteq C_s \quad \text{(Bigger compatible, smaller class set)} \]
\[ (2) \quad \Gamma_s \supseteq \Gamma_q \]

Subsets with smaller class sets are acceptable.
Class Sets and Prime Compatibles

• In minimization, we desire a minimum number of compatible sets that cover all original states. Pick from primes.

• Choice of conditionally compatible set implies choosing all implied pairs.

• Set of implied compatibles pairs is called the class set, e.g., \( \{s_1, s_2\} \) is the class set of \( \{s_3, s_4\} \)

\[
CS_{(s,t)} = \{(s_i, t_i)\}
\]
Update and Strategy

- We just derived maximal compatibles that are prime
- Derive remaining prime compatibles
- Solve a covering problem
Class Sets

\[ \Gamma((a,b)) = \{(a,d)\} \]
\[ \Gamma((b,e)) = \{(d,e), (a,b), (a,e)\} \]

\[
\begin{array}{cccccccc}
  x1 & x2 & x3 & x4 & x5 & x6 & x7 \\
\hline
  a & a,0 & -- & d,0 & e,1 & b,0 & a,-- & -- \\
  b & b,0 & d,1 & a,-- & -- & a,-- & a,1 & -- \\
  c & b,0 & d,1 & a,1 & -- & -- & -- & g,0 \\
  d & -- & e,-- & -- & b,-- & b,0 & -- & a,-- \\
  e & b,-- & e,-- & a,-- & -- & b,-- & e,-- & a,1 \\
  f & b,0 & c,-- & --,1 & h,1 & f,1 & g,0 & -- \\
  g & -- & c,1 & -- & e,1 & -- & g,0 & f,0 \\
  h & a,1 & e,0 & d,1 & b,0 & b,-- & e,-- & a,1 \\
\end{array}
\]
Class Sets and Primes

\[ \Gamma(\{c, f, g\}) = \{(c, d), (e, h)\} \]

\[ \Gamma(\{c, f\}) = \{(c, d)\} \]

Note \(\{c, f\}\) is prime: although \(\{c, f, g\} \supset \{c, f\}\),

\[ \Gamma(\{c, f\}) \subseteq \Gamma(\{c, f, g\}) \]
Class Sets and Primes

\[ \Gamma(\{d,e,h\}) = \{(a,b),(c,d)\} \]

\[ \Gamma(\{e,h\}) = \{(a,b),(c,d)\} \]

Note \( \{e,h\} \) is not prime:

\[ \{d,e,h\} \supset \{e,h\}, \]

\[ \Gamma(\{e,h\}) \supseteq \Gamma(\{d,e,h\}) \]
Class Sets

\[ \Gamma\{a,b\} = \{(a,d)\} \]
\[ \Gamma\{b,e\} = \{(d,e),(a,b),(a,e)\} \]
\[ \Gamma\{a,b,e\} = \{(a,d),(d,e)\} \]
\[ \Gamma\{a,b,d,e\} = \emptyset \]

Note \{c, f\} is prime:
\[ \{c, f, g\} \supset \{c, f\}, \quad \text{but} \]
\[ \Gamma\{c, f\} \subseteq \Gamma\{c, f, g\} \]
### Maximal compatibles are prime

<table>
<thead>
<tr>
<th>Maximal compatibles</th>
<th>Class set</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 {a,b,d,e}</td>
<td>{}</td>
</tr>
<tr>
<td>2 {b,c,d}</td>
<td>{{a,b},{a,g},{d,e}}</td>
</tr>
<tr>
<td>3 {c,f,g}</td>
<td>{{c,d}, {e,h}}</td>
</tr>
<tr>
<td>4 {d,e,h}</td>
<td>{{a,b}, {a,d}}</td>
</tr>
<tr>
<td>11 {a,g}</td>
<td>{}</td>
</tr>
<tr>
<td><strong>Other PCs</strong></td>
<td></td>
</tr>
<tr>
<td>5 {b,c}</td>
<td>{}</td>
</tr>
<tr>
<td>6 {c,d}</td>
<td>{{a,g}, {d,e}}</td>
</tr>
<tr>
<td>7 {c,f}</td>
<td>{{c,d}}</td>
</tr>
<tr>
<td>8 {c,g}</td>
<td>{{c,d}, {f,g}}</td>
</tr>
<tr>
<td>9 {f,g}</td>
<td>{{e,h}}</td>
</tr>
<tr>
<td>10 {d,h}</td>
<td>{}</td>
</tr>
<tr>
<td>12 {f}</td>
<td>{}</td>
</tr>
</tbody>
</table>

Note sub-compatibles \{b,c\} through \{d,h\} are added to the list of prime compatibles before maximal compatible \{a,g\}.
Maximal compatibles are prime

maximal class
compatible set
1  \{a,b,d,e\}  \{\}
2  \{b,c,d\}  \{a,b\}, \{a,g\}, \{d,e\}\}
3  \{c,f,g\}  \{\{c,d\}, \{e,h\}\}
4  \{d,e,h\}  \{\{a,b\}, \{a,d\}\}
11 \{a,g\}  \{\}
other PCs
5  \{b,c\}  \{\}
6  \{c,d\}  \{\{a,g\}, \{d,e\}\}
7  \{c,f\}  \{\{c,d\}\}
8  \{c,g\}  \{\{c,d\}, \{f,g\}\}
9  \{f,g\}  \{\{e,h\}\}
10 \{d,h\}  \{\}
12 \{f\}  \{\}

Note that subsets \{b,d\} and \{d,e\} are not prime because they are contained in \{a,b,d,e\}, which has an empty class set.
Maximal compatibles are prime

Maximal class compatible set
1  {a,b,d,e}   {}  Note that subset \{e,h\}, with
class set \{{a,b},{a,d}\}, is not
prime because it is contained
in \{d,e,h\}, whose class set
is the same.

\exists q \text{ such that}
(1) \ q \supseteq s
(2) \Gamma_s \supseteq \Gamma_q

\n
other PCs
5  {b,c}   {}  \n6  {c,d}   \{{a,g}, {d,e}\}  \n7  {c,f}   \{{c,d}\}  \n8  {c,g}   \{{c,d}, {f,g}\}  \n9  {f,g}   \{{e,h}\}  
10 {d,h}   {}  
12 {f}   {}
Maximal compatibles are prime

<table>
<thead>
<tr>
<th>Maximal Class</th>
<th>Compatible Set</th>
</tr>
</thead>
<tbody>
<tr>
<td>1  {a,b,d,e}</td>
<td>{}</td>
</tr>
<tr>
<td>2  {b,c,d}</td>
<td>{a,b},{a,g},{d,e}</td>
</tr>
<tr>
<td>3  {c,f,g}</td>
<td>{{c,d}, {e,h}}</td>
</tr>
<tr>
<td>4  {d,e,h}</td>
<td>{{a,b}, {a,d}}</td>
</tr>
<tr>
<td>11 {a,g}</td>
<td>{}</td>
</tr>
<tr>
<td>Other PCs</td>
<td></td>
</tr>
<tr>
<td>5  {b,c}</td>
<td>{}</td>
</tr>
<tr>
<td>6  {c,d}</td>
<td>{{a,g}, {d,e}}</td>
</tr>
<tr>
<td>7  {c,f}</td>
<td>{{c,d}}</td>
</tr>
<tr>
<td>8  {c,g}</td>
<td>{{c,d}, {f,g}}</td>
</tr>
<tr>
<td>9  {f,g}</td>
<td>{{e,h}}</td>
</tr>
<tr>
<td>10 {d,h}</td>
<td>{}</td>
</tr>
<tr>
<td>12 {f}</td>
<td>{}</td>
</tr>
</tbody>
</table>

After treating subsets of size 2, we still have to check all subsets of size 1, which have empty class sets.

Note

\{a\}, \{b\}, \{c\}, \{d\}, \{e\}, \{g\}

are all contained in primes with empty class sets, so they are not prime.

But \{f\} is not, so it is prime.
Goal: To Find Prime Compatibles

Maximal Compatibles are prime.

Other prime compatibles are subsets of primes such that:

\[ s \text{ is prime iff its class set does not contain the class set of a larger prime } s' \supset s. \]

\[ e.g., \quad \{e,h\} \rightarrow \{(a,b),(a,d)\} \]

is not prime.
Finding Prime Compatibles

Procedure \( MAXCOMPS,CM \) {
\[ p = \text{LARGEST}(MAXCOMPS); \ k_{\text{max}} = |p| \]

1. \textbf{for} \((k = k_{\text{max}}; k \geq 1; k --) \) {
   \[ Q = \text{SELECT\_BY\_SIZE}(MAXCOMPS,k) \]
   \textbf{for} \((q \in Q) \) ENQUEUE\((P,q)\)

2. \textbf{foreach} \((p \in P; |p| = k) \) {
   \[ CS_p = \text{CLASS\_SET}(CM,p) \]
   \[ S_p = \text{MAX\_SUBSETS}(p) \]
   \textbf{for} \((s \in S_p) \) {
   \[ \text{if} \ (CS_p = \emptyset) \text{ continue} \]
   \[ S_p = \text{MAX\_SUBSETS}(p) \]
   \textbf{for} \((s \in S_p) \) {
   \textbf{if} \ (\text{DONE}(s)) \text{ continue} \]
   \[ CS_s = \text{CLASS\_SET}(CM,s) \]
   \[ \text{prime} = 1 \]
   \textbf{foreach} \((q \in P; |q| \geq k) \) {
   \textbf{if} \ ((s \subseteq q) \) {
   \[ CS_q = \text{CLASS\_SET}(CM,q) \]
   \textbf{if} \ (CS_s \supseteq CS_q) \{ \text{prime} = 0; \text{break} \}
   \}
   \textbf{if} \ (\text{prime} = 1) \text{ ENQUEUE}(P,s) \]
   \text{HASH\_TABLE\_INSERT}(\text{DONE},s)
Finding Prime Compatibles

Procedure \((MAXCOMPS, CM)\) {

\[ p = \text{LARGEST}(MAXCOMPS); \quad k_{\text{max}} = |p| \]

1. \textbf{for} \((k = k_{\text{max}}; k \geq 1; k --)\) {
   \[ Q = \text{SELECT\_BY\_SIZE}(MAXCOMPS, k) \]
   \textbf{for} \((q \in Q)\) \text{ENQUEUE}(P, q) 

2. \textbf{foreach} \((p \in P; |p| = k)\) {
   \[ CS_p = \text{CLASS\_SET}(CM, p) \]

3. \textbf{if} \((CS_p = \emptyset)\) \textbf{continue} 
   \[ S_p = \text{MAX\_SUBSETS}(p) \]

For each value of \(k\), the for-loop of Line 1 puts the maximal compatibles of size \(k\) onto the queue of primes, \(P\).

For \(k = 4\), only \(\{a, b, d, e\}\) is enqueued

For \(k = 3\), \(\{b, c, d\}\), \(\{c, f, g\}\), \(\{d, e, h\}\) are enqueued
Finding Prime Compatibles

For each enqueued prime $p$ (of size $k$), we check every subset of size $k - 1$.

$s$ is a prime compatible if and only if

$\neg \exists q$ such that

1. $q \supseteq s$
2. $\Gamma_s \supseteq \Gamma_q$
### Building the Reduced Machine

<table>
<thead>
<tr>
<th></th>
<th>x1</th>
<th>x2</th>
<th>x3</th>
<th>x4</th>
<th>x5</th>
<th>x6</th>
<th>x7</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>a,0</td>
<td>--</td>
<td>d,0</td>
<td>e,1</td>
<td>b,0</td>
<td>a,--</td>
<td>--</td>
</tr>
<tr>
<td>b</td>
<td>b,0</td>
<td>d,1</td>
<td>a,--</td>
<td>--</td>
<td>a,--</td>
<td>a,1</td>
<td>--</td>
</tr>
<tr>
<td>c</td>
<td>b,0</td>
<td>d,1</td>
<td>a,1</td>
<td>--</td>
<td>--</td>
<td>g,0</td>
<td>--</td>
</tr>
<tr>
<td>d</td>
<td>--</td>
<td>e,--</td>
<td>--</td>
<td>b,--</td>
<td>b,0</td>
<td>--</td>
<td>a,--</td>
</tr>
<tr>
<td>e</td>
<td>b,--</td>
<td>e,--</td>
<td>a,--</td>
<td>--</td>
<td>b,--</td>
<td>e,--</td>
<td>a,1</td>
</tr>
<tr>
<td>f</td>
<td>b,0</td>
<td>c,--</td>
<td>--,1</td>
<td>h,1</td>
<td>f,1</td>
<td>g,0</td>
<td>--</td>
</tr>
<tr>
<td>g</td>
<td>--</td>
<td>c,1</td>
<td>--</td>
<td>e,1</td>
<td>--</td>
<td>g,0</td>
<td>f,0</td>
</tr>
<tr>
<td>h</td>
<td>a,1</td>
<td>e,0</td>
<td>d,1</td>
<td>b,0</td>
<td>b,--</td>
<td>e,--</td>
<td>a,1</td>
</tr>
</tbody>
</table>

\[
\{c_1, c_4, c_5, c_9\}
\]

\[
c_1 = \{a, b, d, e\}
\]

\[
c_4 = \{d, e, h\}
\]

\[
c_5 = \{b, c\}
\]

\[
c_9 = \{f, g\}
\]

<table>
<thead>
<tr>
<th>x1</th>
<th>x2</th>
<th>x3</th>
<th>x4</th>
<th>x5</th>
<th>x6</th>
<th>x7</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1.0</td>
<td>1.1</td>
<td>1.0</td>
<td>1.1</td>
<td>1.0</td>
<td>1.1</td>
</tr>
<tr>
<td>4</td>
<td>1,1</td>
<td>1,0</td>
<td>1,1</td>
<td>1,0</td>
<td>1,0</td>
<td>1,-</td>
</tr>
<tr>
<td>5</td>
<td>1,0</td>
<td>1,1</td>
<td>1,1</td>
<td>-</td>
<td>1,-</td>
<td>1,1</td>
</tr>
<tr>
<td>9</td>
<td>1,0</td>
<td>5,1</td>
<td>-,1</td>
<td>4,1</td>
<td>9,1</td>
<td>9,0</td>
</tr>
</tbody>
</table>

---

**Computer Systems Lab.**
## Reduced Machine

<table>
<thead>
<tr>
<th>x1</th>
<th>x2</th>
<th>x3</th>
<th>x4</th>
<th>x5</th>
<th>x6</th>
<th>x7</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>a,0</td>
<td>--</td>
<td>d,0</td>
<td>e,1</td>
<td>b,0</td>
<td>a,--</td>
</tr>
<tr>
<td>b</td>
<td>b,0</td>
<td>d,1</td>
<td>a,--</td>
<td>--</td>
<td>a,--</td>
<td>a,1</td>
</tr>
<tr>
<td>c</td>
<td>b,0</td>
<td>d,1</td>
<td>a,1</td>
<td>--</td>
<td>--</td>
<td>--</td>
</tr>
<tr>
<td>d</td>
<td>--</td>
<td>e,--</td>
<td>--</td>
<td>b,--</td>
<td>b,0</td>
<td>--</td>
</tr>
<tr>
<td>e</td>
<td>b,--</td>
<td>e,--</td>
<td>a,--</td>
<td>--</td>
<td>b,--</td>
<td>e,--</td>
</tr>
<tr>
<td>f</td>
<td>b,0</td>
<td>c,--</td>
<td>--</td>
<td>1</td>
<td>h,1</td>
<td>f,1</td>
</tr>
<tr>
<td>g</td>
<td>--</td>
<td>c,1</td>
<td>--</td>
<td>e,1</td>
<td>--</td>
<td>g,0</td>
</tr>
<tr>
<td>h</td>
<td>a,1</td>
<td>e,0</td>
<td>d,1</td>
<td>b,0</td>
<td>b,--</td>
<td>e,--</td>
</tr>
</tbody>
</table>

\[
c_1 = \{a, b, d, e\}
\]
\[
c_4 = \{d, e, h\}
\]
\[
c_5 = \{b, c\}
\]
\[
c_9 = \{f, g\}
\]

Where there is a choice, choose 1 (as in x2-successor of compatible 1):
\{d, e\} contained in \(c_1\) or \(c_4\).
Closed Cover

• Closed Cover: Choosing Compatibles

• Every state of the original machine must be covered

• Every implied compatible must be present in the solution
Let's check if the following set of compatibles forms a closed cover: \( \{ c_1, c_4, c_5, c_9 \} \)

### Coverage:

\[
\begin{align*}
\text{a} & \in c_1 \\
\text{b, c} & \in c_5 \\
\text{d, e} & \in c_4 \\
\text{f, g} & \in c_9 \\
\text{h} & \in c_4
\end{align*}
\]

### Closure:

\[
\begin{align*}
\Gamma (c_1): & \quad \{a, b\} \in c_1 \quad \{a, d\} \in c_1 \\
\Gamma (c_4): & \quad \{a, b\} \in c_4 \\
\Gamma (c_5): & \quad \{a, d\} \in c_5 \\
\Gamma (c_9): & \quad \{e, h\} \in c_9
\end{align*}
\]
### Covering Constraints--POS FORM

- **Every state of the original machine must be covered.**

<table>
<thead>
<tr>
<th>Maximal Compatibles</th>
<th>Class Set</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 {a,b,d,e}</td>
<td>{}</td>
</tr>
<tr>
<td>2 {b,c,d}</td>
<td>{{a,b}, {a,g}, {d,e}}</td>
</tr>
<tr>
<td>3 {c,f,g}</td>
<td>{{c,d}, {e,h}}</td>
</tr>
<tr>
<td>4 {d,e,h}</td>
<td>{{a,b}, {a,d}}</td>
</tr>
<tr>
<td>11 {a,g}</td>
<td>{}</td>
</tr>
<tr>
<td>Other PCs</td>
<td></td>
</tr>
<tr>
<td>5 {b,c}</td>
<td>{}</td>
</tr>
<tr>
<td>6 {c,d}</td>
<td>{{a,g}, {d,e}}</td>
</tr>
<tr>
<td>7 {c,f}</td>
<td>{{c,d}}</td>
</tr>
<tr>
<td>8 {c,g}</td>
<td>{{c,d}, {f,g}}</td>
</tr>
<tr>
<td>9 {f,g}</td>
<td>{{e,h}}</td>
</tr>
<tr>
<td>10 {d,h}</td>
<td>{}</td>
</tr>
<tr>
<td>12 {f}</td>
<td>{}</td>
</tr>
</tbody>
</table>

\[
(c_1 + c_{11})(c_1 + c_2 + c_5)
\]

\[
(c_2 + c_3 + c_5 + c_6 + c_7 + c_8)
\]

\[
(c_1 + c_2 + c_4 + c_6 + c_{10})
\]

\[
(c_1 + c_4)(c_3 + c_7 + c_9 + c_{12})
\]

\[
(c_3 + c_8 + c_9 + c_{11})
\]

\[
(c_4 + c_{11}) = 1
\]
### Covering Constraints--POS FORM

<table>
<thead>
<tr>
<th>maximal compatibles</th>
<th>class set</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 {a,b,d,e}</td>
<td>{}</td>
</tr>
<tr>
<td>2 {b,c,d}</td>
<td>{{a,b}, {a,g}, {d,e}}</td>
</tr>
<tr>
<td>3 {c,f,g}</td>
<td>{{c,d}, {e,h}}</td>
</tr>
<tr>
<td>4 {d,e,h}</td>
<td>{{a,b}, {a,d}}</td>
</tr>
<tr>
<td>11 {a,g}</td>
<td>{}</td>
</tr>
<tr>
<td>other PCs</td>
<td></td>
</tr>
<tr>
<td>5 {b,c}</td>
<td>{}</td>
</tr>
<tr>
<td>6 {c,d}</td>
<td>{{a,g}, {d,e}}</td>
</tr>
<tr>
<td>7 {c,f}</td>
<td>{{c,d}}</td>
</tr>
<tr>
<td>8 {c,g}</td>
<td>{{c,d}, {f,g}}</td>
</tr>
<tr>
<td>9 {f,g}</td>
<td>{{e,h}}</td>
</tr>
<tr>
<td>10 {d,h}</td>
<td>{}</td>
</tr>
<tr>
<td>12 {f}</td>
<td>{}</td>
</tr>
</tbody>
</table>

*Every state of the original machine must be covered.*

\[
\begin{align*}
(c_1 + c_{11})(c_1 + c_2 + c_5) \\
(c_2 + c_3 + c_5 + c_6 + c_7) \\
(c_1 + c_2 + c_4 + c_6 + c_{10}) \\
(c_1 + c_4)(c_3 + c_7 + c_9 + c_{12}) \\
(c_3 + c_8 + c_9 + c_{11}) \\
(c_4 + c_{10})
\end{align*}
\]
Finding a Minimum Closed Cover

- Associate a variable $c_i$ to the $i^{th}$ prime compatible
- For each $s \in S$, form the coverage constraint $\prod_{s \in S} \left( \sum_{s \in c_i} c_i \right)$

1. \{a,b,d,e\} \quad \{\}\n2. \{b,c,d\} \quad \{{a,b},{a,g},{d,e}\}\n3. \{c,f,g\} \quad \{{c,d}\}, \{e,h\}\n4. \{d,e,h\} \quad \{{a,b}\}, \{a,d\}\n
\[
\begin{align*}
\text{Minimization} & \\
1 \quad & \{a,b,d,e\} & \{\} \\
2 \quad & \{b,c,d\} & \{{a,b},{a,g},{d,e}\} \\
3 \quad & \{c,f,g\} & \{{c,d}\}, \{e,h\} \\
4 \quad & \{d,e,h\} & \{{a,b}\}, \{a,d\} \\
\end{align*}
\]

\[
\begin{array}{cccccccc}
 & a & b & c & d & e & f & g & h \\
\hline
1 & & & & & & & & \hline
2 & & & & & & & & (c_1 + c_2) \\
3 & & & & & & & & \hline
4 & & & & & & & & \\
\end{array}
\]

\[
c_1(c_1 + c_2)(c_2 + c_3)(c_1 + c_2 + c_3)(c_1 + c_4)c_3c_3c_4
\]

\[
= c_1c_3c_4 \quad \text{This cover is not closed, since } c_2 \text{ is excluded}
\]
Closure Constraints

$C_\Gamma$ is the set of prime compatibles with non-empty class sets

Note $(c_i \Rightarrow c_j) \iff (c'_i + c_j)$

<table>
<thead>
<tr>
<th>$C_\Gamma$</th>
<th>class sets</th>
<th>$c'_2 + c_1$</th>
<th>$c'<em>2 + c</em>{11}$</th>
<th>$c'_2 + c_1 + c_4$</th>
<th>$c'_3 + c_4$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1   {a,b,d,e}</td>
<td>{}</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2   {b,c,d}</td>
<td>{{$a,b$}, {$a,g$}, {$d,e$}}</td>
<td></td>
<td></td>
<td>$(a,g) \subseteq {a, g}$</td>
<td></td>
</tr>
<tr>
<td>3   {c,f,g}</td>
<td>{{$c,d$}, {$e,h$}}</td>
<td>$(c'<em>2 + c</em>{11})$</td>
<td>$(a,g) \subseteq {a, g}$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4   {d,e,h}</td>
<td>{{$a,b$}, {$a,d$}}</td>
<td>$(c'_2 + c_1 + c_4)$</td>
<td>$(d,e) \subseteq {a, b, d, e}$</td>
<td>$(d,e) \subseteq {d, e, h}$</td>
<td>$(c'_3 + c_4)$</td>
</tr>
<tr>
<td>11  {a,g}</td>
<td>{}</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5   {b,c}</td>
<td>{}</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6   {c,d}</td>
<td>{{$a,g$}, {$d,e$}}</td>
<td>$c'_2 + c_1 + c_4$</td>
<td>$(d,e) \subseteq {a, b, d, e}$</td>
<td>$(d,e) \subseteq {d, e, h}$</td>
<td>$(c'_3 + c_4)$</td>
</tr>
<tr>
<td>7   {c,f}</td>
<td>{{$c,d$}}</td>
<td>$(c'_2 + c_1 + c_4)$</td>
<td>$(d,e) \subseteq {a, b, d, e}$</td>
<td>$(d,e) \subseteq {d, e, h}$</td>
<td>$(c'_3 + c_4)$</td>
</tr>
</tbody>
</table>
Covering and Closure Constraints--POS FORM

\( (c_1 + c_{11})(c_1 + c_2 + c_5)(c_2 + c_3 + c_5 + c_6 + c_7 + c_8) \)

\( (c_1 + c_2 + c_4 + c_6 + c_{10})(c_1 + c_4)(c_3 + c_7 + c_9 + c_{12}) \)

\( (c_3 + c_8 + c_9 + c_{11})(c_4 + c_{11}) \)

\( (c'_2 + c_1)(c'_2 + c_{11}) \)

\( (c'_2 + c_1 + c_4)(c'_3 + c_2 + c_6)(c'_3 + c_4)(c'_4 + c_1)(c'_6 + c_{11}) \)

\( (c'_6 + c_1 + c_4)(c'_7 + c_2 + c_6)(c'_8 + c_2 + c_6)(c'_8 + c_3 + c_9) \)

\( (c'_9 + c_4) = 1 \)
Covering Constraints--Matrix FORM

\[(c_1 + c_{11})(c_1 + c_2 + c_5)(c_2 + c_3 + c_5 + c_6 + c_7 + c_8)\]
\[(c_1 + c_2 + c_4 + c_6 + c_{10})(c_1 + c_4)(c_3 + c_7 + c_9 + c_{12})\]
\[(c_3 + c_8 + c_9 + c_{11})(c_4 + c_{11})\]

\[
\begin{pmatrix}
  c_1 & c_2 & c_3 & c_4 & c_5 & c_6 & c_7 & c_8 & c_9 & c_{10} & c_{11} & c_{12} \\
  a & 1 & & & & & & & & & & 1 \\
  b & 1 & 1 & & & & & & & & & & 1 \\
  c & 1 & 1 & 1 & 1 & 1 & 1 & & & & & & 1 \\
  d & 1 & 1 & 1 & & & & & & & & & Row Dominance \\
  e & 1 & & & & & & & & & & 1 \\
  f & 1 & & & 1 & 1 & 1 & & & & & & + \\
  g & 1 & & & 1 & & & 1 & & & & & 1 \\
  h & & 1 & & & & & & & 1 & & & Col Dominance? \\
\end{pmatrix}
\]
(see below)
Closure Constraints--Matrix FORM

class sets

1  \{a,b,d,e\}   {} \\
2  \{b,c,d\}     \{\{a,b\}, \{a,g\}, \{d,e\}\} \\
3  \{c,f,g\}     \{\{c,d\}, \{e,h\}\} \\
4  \{d,e,h\}     \{\{a,b\}, \{a,d\}\} \\
11 \{a,g\}       {} \\
5  \{b,c\}       {} \\
6  \{c,d\}       \{\{a,g\}, \{d,e\}\} \\
7  \{c,f\}       \{\{c,d\}\}

For each pair \(p_j\) in the class set of each compatible \(c_i\), form the clause

\[c'_i + \sum_{k} c_k\]

where \(k\) ranges over the indices of compatibles that contain \(p_j\).

\[
\begin{pmatrix}
(\mathbf{c'}_2 + c_1) \\
(\mathbf{c'}_2 + c_{11}) \\
(\mathbf{c'}_2 + c_1 + c_4)
\end{pmatrix}
\]

\[
\begin{pmatrix}
\mathbf{c_1} & \mathbf{c_2} & \mathbf{c_3} & \mathbf{c_4} & \mathbf{c_5} & \mathbf{c_6} & \mathbf{c_7} & \mathbf{c_8} & \mathbf{c_9} & \mathbf{c_{10}} & \mathbf{c_{11}} & \mathbf{c_{12}} \\
\mathbf{\Gamma_2} & \mathbf{1} & \mathbf{0} \\
\mathbf{\Gamma_2} & \mathbf{0} & \mathbf{1} \\
\mathbf{\Gamma_2} & \mathbf{1} & \mathbf{0} & \mathbf{1}
\end{pmatrix}
\]
Closure Constraints--Matrix FORM

Cover rows by including a 1-col OR excluding a 0-col

\[ c_i' \Rightarrow c_j \]

\[
\begin{array}{cccccccccccc}
& c_1 & c_2 & c_3 & c_4 & c_5 & c_6 & c_7 & c_8 & c_9 & c_{10} & c_{11} & c_{12} \\
\hline
C_2 + C_1 & & & & & 1 & 0 & & & & & & \\
C_2 + C_{11} & & & & & 0 & & & & & & 1 \\
C_2 + C_1 + C_4 & & & & & 1 & 0 & 1 & & & & & \\
C_3 + C_4 & & & & & 1 & 0 & & 1 & & & & \\
C_3 + C_2 + C_6 & & & & & 0 & 1 & & & & & & \\
C_4 + C_1 & & & & & 1 & & 0 & & & & & \\
C_4 + C_1 & & & & & 1 & & 0 & & & & & \\
C_6 + C_{11} & & & & & 0 & & & & 1 & & & \\
C_6 + C_1 + C_4 & & & & & 1 & 1 & 0 & & & & & \\
C_7 + C_2 + C_6 & & & & & 1 & 1 & 0 & & & & & \\
C_8 + C_2 + C_6 & & & & & 1 & 1 & 0 & & & & & \\
C_8 + C_3 + C_9 & & & & & 1 & & 0 & 1 & & & & \\
C_9 + C_4 & & & & & 1 & & 0 & & & & & \\
\end{array}
\]
Closed Covering Problem

Find a minimum set of columns which cover all rows: \{1,4,5,9\}

A row is covered by either including a 1-col or excluding a 0-col.
Binate Covering Problem

- Similar to unate covering
- Matrix
  - Variables on columns
  - Sum expressions on the rows
- Solution may not exist when product is 0
The Binate Covering Problem

- Note: $M$ replaced by $F$ to emphasize POS semantics
- Also there is one addition (for empty solution space)

```plaintext
Procedure BCP(F, U, currentSol)
1   (F, currentSol) = REDUCE(M, currentSol)
   if (terminalCase(F)) {
      if $F \neq 0$ and COST(currentSol) < U) {
         U = COST(currentSol)
      } return (currentSol)
   } else return("no (better) solution (in this subspace)"

2   L = LOWER_BOUND (F, currentSol)
3   if (L ≥ U) return("no (better) solution (in this subspace)"
4   x_i = CHOOSE_VAR(F) \ \\ longest column
5   S^1 = BCP(F_{x_i}, U, currentSol \cup \{x_i\})
6   if (COST(S^1) = L) return(S^1)
7   S^0 = BCP(F_{x_i'}, U, currentSol)
8   return BEST_SOLUTION (S^1, S^0)
```
When $x'_2$ is essential we say that $x_2$ is unacceptable.

When $x'_i$ is essential, we may delete all rows of the matrix which has a zero in the $i^{th}$ column.
Row Dominance

Formally: Row $f_1$ dominates row $f_2$ if $f_1$ is satisfied, in a Boolean sense, whenever $f_2$ is satisfied, that is, $f_1 \leq f_2$.

\[
(x'_3 + x_2)(x'_3 + x_2 + x'_1) = (x'_3 + x_2)
\]

Row 1 ($f_1$) dominates row 2 ($f_2$) since row 2 matches row 1 at all care entries. Row 1 may be deleted.

<table>
<thead>
<tr>
<th>$x_1$</th>
<th>$x_2$</th>
<th>$x_3$</th>
<th>$x_4$</th>
<th>$f_1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td></td>
<td>$f_1$</td>
</tr>
<tr>
<td>-1</td>
<td>1</td>
<td>0</td>
<td>-</td>
<td>$f_2$</td>
</tr>
<tr>
<td>1</td>
<td>-</td>
<td>-</td>
<td>1</td>
<td>$f_3$</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>$f_4$</td>
</tr>
</tbody>
</table>

\[ F = \begin{pmatrix}
0 & 1 & 0 & f_1 \\
-1 & 1 & 0 & - & f_2 \\
1 & - & - & 1 & f_3 \\
1 & 0 & 1 & 0 & f_4
\end{pmatrix} \]
Column Dominance

Let $F_j$ and $F_k$ be two columns of $F$. We say that $F_j$ dominates $F_k$ if, for each row $f_i$ of $F$, one of the following conditions hold:

1. $f_{ij} = 1$
2. $f_{ij} = -1$ and $f_{ik} \neq 1$
3. $f_{ij} = 0$ and $f_{ik} = 0$

Example: reduced column $F_1$ dominates $F_4$

$$F = \begin{bmatrix}
    x_1 & x_2 & x_3 & x_4 \\
    0 & 1 & 0 & \frac{f_1}{f_1}
\end{bmatrix}$$

$$F = \begin{bmatrix}
    - & 1 & 0 & - \\
    1 & - & - & 1 \\
    1 & 0 & 1 & 0
\end{bmatrix}$$

$$F_1 = \begin{bmatrix}
    x_1 & x_2 & x_3 & x_4 \\
    0 & 1 & 0 & \frac{f_1}{f_1}
\end{bmatrix}$$

$$F_4 = \begin{bmatrix}
    x_1 & x_2 & x_3 & x_4 \\
    1 & 0 & 1 & 0
\end{bmatrix}$$
Maximal Independent Set

- Two rows are independent if it is not possible to satisfy both clauses by assigning one variable to 1.
- Thus in finding the MIS, we ignore rows (clauses) that contain 0s, since these are satisfied by assigning variables to 0.

\[
\begin{array}{cccc}
  x_1 & x_2 & x_3 & x_4 \\
  1 & 1 & - & - & f_1 \\
  - & 1 & 1 & - & f_2 \\
  - & 0 & - & 1 & f_3 \\
\end{array}
\]

\[
\begin{array}{cccc}
  x_1 & x_2 & x_3 & x_4 \\
  1 & 0 & - & - & f_1 \\
  0 & 1 & - & - & f_2 \\
  - & 0 & 1 & - & f_3 \\
  - & - & 0 & 1 & f_4 \\
\end{array}
\]

\[MIS = \{ f_1 \}\]

cyclic, \[MIS = \{ \}\]
Infeasible Subproblems

$F = 0$ cannot occur in original problem (first call to the recursive procedure). But it can happen after one or more recursions:

\[
F = \begin{bmatrix}
1 & 1 \\
0 & 1 \\
1 & 0 \\
0 & 0
\end{bmatrix} = (x_1 + x_2)(x'_1 + x_2)(x_1 + x'_2)(x'_1 + x'_2) = 0
\]

This is detected by REDUCTION, which discovers that both $x_2$ and $x'_2$ are essential
### Reduction

<table>
<thead>
<tr>
<th>$f_1$ dominates $f_2$</th>
<th>$F_1$ dominates $F_4$</th>
<th>$x_4 = 0$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x_1$</td>
<td>$x_2$</td>
<td>$x_3$</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>-</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

$x_1$ is essential

<table>
<thead>
<tr>
<th>$x_1$</th>
<th>$x_2$</th>
<th>$x_3$</th>
<th>$x_4$</th>
<th>$F_2$ dominates $F_3$</th>
<th>$x_3 = 0$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>-</td>
<td>$f_1$</td>
<td>Solution: $x = (1,0,0,0,0)$</td>
</tr>
<tr>
<td>-</td>
<td>1</td>
<td>0</td>
<td>-</td>
<td>$f_2$</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>-</td>
<td>-</td>
<td>1</td>
<td>$f_3$</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>$f_4$</td>
<td></td>
</tr>
</tbody>
</table>
State Encoding

• The number of possible assignments is very high

• If one uses \( k \) bits to encode \( p \) states, there are \( \frac{(2^k)!}{(2^k - p)!} \) possible assignments

• If one considers two assignments obtained by permutation or complementation of some of the bits as essentially the same assignment, then there are \( \frac{(2^k - 1)!}{(2^k - p)!} k! \) distinct assignments
Practical Encoding Algorithms

- Mustang tries to identify pairs of states by receiving adjacent pairs
  - Two codes are adjacent if they only differ in one bit
- The first objective is to build a graph representing the attraction between each pair of states
  - Two states that have a strong attraction should be given adjacent codes
- How to build attraction graph
  - In the fanout-oriented algorithm, whenever two states, si and sj have a common fanout state, the weight of the edge (si, sj) of the attraction graph is increased
  - In the fanin-oriented algorithm, if si and sj have a common fanin state, the weight of the edge (si, sj) of the attraction graph is increased
  - Once the graph of the attractions is found, we try to assign codes to pairs of states that have strong attractions
Fanout Oriented Algorithm

- Build two matrices
  - The first with one row for each present state and one column for each next state
  - The second with one row for each present state and one column for each output
Embedding Algorithm

• Assign codes to states
  - Select first the node for which the sum of the weights of the Nb heaviest incident edges is maximum
Fanin Oriented Algorithm

- Build two matrices
  - The first with one row for each next state and one column for each present state
  - The second with one row for each next state and two columns for each output
    - One column is for the true input and the other is for the complement
Decomposition and Encoding

- Rather than aiming directly at minimizing the number of literals in the next-state functions, one may actually try to minimize the support of the functions.
- Reduction of the number of literals and simplification of the interconnections.
Partitions

- A partition $\pi$ is on a set $S$ is a collection of disjoint subsets of $S$ whose set union is $S$, i.e. $\pi = \{ B_a \}$ such that $B_a \cap B_b = \emptyset$ for $a \neq b$
  \[ \text{and } \bigcup \{ B_a \} = S \]
- Each subset is called a block of the partition
- If $\pi_1$ and $\pi_2$ are partitions on $S$, then $\pi_1 \pi_2$ is the partition on $S$ such that $s \equiv t(\pi_1 \pi_2)$ if and only if $s \equiv t(\pi_1)$ and $s \equiv (\pi_2)$, whereas, $\pi_1 + \pi_2$ is the partition on $S$ such that $s \equiv t(\pi_1 + \pi_2)$ if and only if there exists a sequence in $S$ $s=s_0 \ s_1 \ s_2... \ s_n = t$ for which either $s_i \equiv s_{i+1} (\pi_1)$ or $s_i \equiv s_{i+1} (\pi_2)$, $0 \leq i \leq n-1$
Partitions with Substitution Property

- A partition $\pi$ on the set of states of the machine is said to have the substitution property if and only if $s \equiv t(\pi)$ implies that $\delta(s,a) \equiv \delta(t,a) (\pi) \quad \forall a \in I$

- A sequential machine $M$ has a non-trivial parallel decomposition of its state behavior if and only if there exist two nontrivial S.P. partitions $\pi_1$ and $\pi_2$ on $M$ such that $\pi_1 \pi_2 = 0$

- Independent component
- Dependent component
Computation of SP Partitions

- First generate the minimal SP partitions and then sum them until considering all possible sums

- The minimal partitions are those obtained by requiring that two states only are included in a block
General Decomposition and Encoding

- Need to resort to something more general than SP partitions, namely, partition pairs.

- A partition pair \((\pi, \pi')\) on the machine is an ordered pair of partitions on \(S\) such that

\[ s \equiv t(\pi) \text{ implies that } \delta(s,a) \equiv \delta(t,a) (\pi') \quad \forall a \in I \]

- The knowledge of the block of \(\pi\) containing the present state and of the current input allows one to compute the block \(\pi'\) of that will contain the next state.

- It is evident that if \((\pi, \pi)\) is a partition pair, then \(\pi\) has substitution property

" Partition pairs generalize SP partitions \"