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

Incomplete specification and Compatibility

Use don’t-cares to merge states.
Merged states <u>must</u> have same output sequences.

	0	1	
1	4	2	0
2	-	1	-
3	1	-	1
4	2	3	1

2	~		
3	✗	~	
4	✗	✗	1-2
	1	2	3

Flow 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")

	0	1	
1	4	2	0
2	-	1	-
3	1	-	1
4	2	3	1

2	~		
3	✕	~	
4	✕	✕	1-2
	1	2	3

$$C(x) = (x'_1 + x'_3)(x'_1 + x'_4)(x'_2 + x'_4)$$
$$= (x'_1 + x'_3 x'_4)(x'_2 + x'_4)$$
$$= x'_1 x'_2 + x'_3 x'_4 + x'_1 x'_4$$

Note each constraint represents pair (incompatibility)

$$(x'_1 + x'_3) \Leftrightarrow (x'_1 \Rightarrow x'_3) \Leftrightarrow (x'_3 \Rightarrow x'_1)$$

Computing the Maximal Compatibles

By recursive multiplication method,
like computing the Complete Sum:

$$\begin{aligned}
C(x) &= (x_1' + x_3')(x_1' + x_4')(x_2' + x_4') \\
&= (x_1' + x_3' x_4')(x_2' + x_4') \\
&= x_1' x_2' + x_3' x_4' + \underline{x_2' x_3' x_4'} + x_1' x_4' \\
&= x_1' x_2' + x_3' x_4' + x_1' x_4'
\end{aligned}$$

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 c ompatible).

$$e.g.,\ x'_1 \Rightarrow \{s_2, s_3, s_4\}\ \text{but}\ \{s_2, s_4\}\ \text{are not compatible}$$

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.

$$\{S_3, S_4\} \Rightarrow \{S_1, S_2\}$$

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 Γ_q contains class set Γ_s of C_s. That is, C_s is prime if and only if

$$\neg\exists C_q \text{ such that}$$

$$(1)\ C_q \supset C_s$$

$$(2)\ \Gamma_s \supseteq \Gamma_q$$

(Bigger compatible, smaller class set)

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 fr om primes.

- Choice of conditionally compatible set implies choosing *all implied pairs.*

- Set of implied compatibles pairs is called the clas s 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

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

	x1	x2	x3	x4	x5	x6	x7
$\boxed{a}$	a,0	--	$\boxed{d,0}$	e,1	b,0	a,--	--
$\boxed{b}$	b,0	d,1	$\boxed{a,--}$	--	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

$\boxed{b}$	$\boxed{a,d}$						
c	X	~					
d	b,e	a,b d,e	d,e a,g				
e	a,b a,d	d,e a,b a,e	X	~			
f	X	X	c,d	X	X		
g	~	X	c,d f,g	X	X	e,h	
h	X	X	X	~	a,b a,d	X	X
	$\boxed{a}$	b	c	d	e	f	g

Class Sets and Primes

$$\boxed{\Gamma(\{c,f,g\})=\{(c,d),(e,h)\}}$$

$$\boxed{\Gamma(\{c,f\})=\{(c,d)\}}$$

Note $\{c, f\}$ **is prime**: although

$$\{c,f,g\}\supset\{c,f\},$$

$$\Gamma(\{c,f\})\subset\Gamma(\{c,f,g\})$$

b	a,d						
c	X	~					
d	b,e	a,b d,e	d,e a,g				
e	a,b a,d	d,e a,b a,e	X	~			
f	X	X	c,d	X	X		
g	~	X	c,d f,g	X	X	e,h	
h	X	X	X	~	a,b a,d	X	X
	a	b	c	d	e	f	g

Class Sets and Primes

$$\boxed{\Gamma(\{d,e,h\}) = \{(a,b),(c,d)\}}$$

$$\boxed{\Gamma(\{e,h\}) = \{(a,b),(c,d)\}}$$

Note $\{e,h\}$ **is not prime**:

$$\{d,e,h\} \supset \{e,h\},$$

$$\Gamma(\{e,h\}) \supseteq \Gamma(\{d,e,h\})$$

b	a,d						
c	X	~					
d	b,e	a,b d,e	d,e a,g				
e	a,b a,d	d,e a,b a,e	X	~			
f	X	X	c,d	X	X		
g	~	X	c,d f,g	X	X	e,h	
$\boxed{h}$	X	X	X	~	$\boxed{\text{a,b a,d}}$	X	X
	a	b	$\boxed{c}$	d	$\boxed{e}$	f	g

Class Sets

$$\boxed{\Gamma(\{c,f\})=\{(c,d)\}}$$

$$\Gamma(\{a,b\})=\{(a,d)\}$$
$$\Gamma(\{b,e\})=\{(d,e),(a,b),(a,e)\}$$
$$\Gamma(\{a,b,e\})=\{(a,d),(d,e)\}$$
$$\Gamma(\{a,b,d,e\})=\varnothing$$

$$\boxed{\Gamma(\{c,f,g\})=\{(c,d),(e,h)\}}$$

Note $\{c,f\}$ is prime:
$\{c,f,g\}\supset\{c,f\}$, **but**

$$\Gamma(\{c,f\})\subset\Gamma(\{c,f,g\})$$

b	a,d						
c	X	~					
d	b,e	a,b d,e	d,e a,g				
e	a,b a,d	d,e a,b a,e	X	~			
f	X	X	c,d	X	X		
g	~	X	c,d f,g	X	X	e,h	
h	X	X	X	~	a,b a,d	X	X
	a	b	c	d	e	f	g

Maximal compatibles are prime

maximal compatibles		class 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 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 compatible	class 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 compatible	class 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 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.

$\not\exists q$ such that
(1) $q \supset s$
(2) $\Gamma_s \supseteq \Gamma_q$

Maximal compatibles are prime

	maximal compatible	class 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}	{}

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 compatible		class 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}	{}

Maximal Compatibles are **prime.**

Other prime compatibles are **subsets** of primes such that:

s 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( $MAXCOMPS, CM$ ) {
    $p$ = LARGEST( $MAXCOMPS$ ); $k_{\max} = |p|$
1   for( $k = k_{\max}$; $k \geq 1$; $k--$ ) {
        $Q$ = SELECT_BY_SIZE( $MAXCOMPS, k$ )
        for( $q \in Q$ ) ENQUEUE( $P, q$ )
2       foreach( $p \in P$; $|p| = k$ ) {
            $CS_p$ = CLASS_SET( $CM, p$ )
3           if ( $CS_p = \varnothing$ ) continue
            $S_p$ = MAX_SUBSETS( $p$ )
            for( $s \in S_p$ ) {
4               if ( DONE( $s$ ) ) continue
                $CS_s$ = CLASS_SET( $CM, s$ )
                $prime = 1$
5               foreach( $q \in P$; $|q| \geq k$ ) {
                    if ( $s \subset q$ ) {
                        $CS_q$ = CLASS_SET( $CM, q$ )
6                       if ( $CS_s \supseteq CS_q$ ) { $prime = 0$; break }
                    }
                }
7               if ( $prime = 1$ ) ENQUEUE( $P, s$ )
                HASH_TABLE_INSERT( $DONE, s$ )
} } } }
```

Enqueue known primes of size k

Test subcompatibles for primality

Finding Prime Compatibles

```
**Procedure**$(MAXCOMPS, CM)$ {
    $p = \mathrm{LARGEST}(MAXCOMPS);\; k_{\max} = |p|$
1   **for**$(k = k_{\max};\, k \geq 1;\, k--)$ {
        $Q = \mathrm{SELECT\_BY\_SIZE}(MAXCOMPS, k)$
        **for**$(q \in Q)$ $\mathrm{ENQUEUE}(P, q)$
2       **foreach**$(p \in P;\, |p| = k)$ {
            $CS_p = \mathrm{CLASS\_SET}(CM, p)$
3           **if** $(CS_p = \emptyset)$ **continue**
            $S_p = \mathrm{MAX\_SUBSETS}(p)$
```

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

```
      $S_p$ = MAX_SUBSETS($p$)
      **for**($s \in S_p$) {
4       **if** (DONE($s$)) **continue**
        $CS_s$ = CLASS_SET($CM, s$)
        *prime = 1*
5       **foreach**($q \in P$; $|q| \geq k$) {
           **if** ($s \subset q$) {
              $CS_q$ = CLASS_SET($CM, q$)
6             **if** ($CS_s \supseteq CS_q$) {$prime = 0$; **break**}
           }
        }
7       **if** ($prime = 1$) ENQUEUE($P, s$)
        HASH_TABLE_INSERT($DONE, s$)
```

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) $q \supset s$

(2) $\Gamma_s \supseteq \Gamma_q$

Building the Reduced Machine

	x1	x2	x3	x4	x5	x6	x7
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

$$\{c_1, c_4, c_5, c_9\}$$

$$\boxed{c_1 = \{a, b, d, e\}}$$

$$\boxed{c_4 = \{d, e, h\}}$$

$$c_5 = \{b, c\}$$

$$c_9 = \{f, g\}$$

	$x1$	$x2$	$x3$	$x4$	$x5$	$x6$	$x7$
1	1.0	1.1	1.0	1.1	1.0	1.1	1.1
4	1,1	1,0	1,1	1,0	1,0	1,–	1,1
5	1,0	1,1	1,1	–	1,–	1,1	9,0
9	1,0	5,1	–,1	4,1	9,1	9,0	9,0

Reduced Machine

	x1	x2	x3	x4	x5	x6	x7
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

$c_1 = \{a,b,d,e\}$

$c_4 = \{d,e,h\}$

$c_5 = \{b,c\}$

$c_9 = \{f,g\}$

	$x1$	$x2$	$x3$	$x4$	$x5$	$x6$	$x7$
1	1.0	1.1	1.0	1.1	1.0	1.1	1.1
4	1,1	1,0	1,1	1,0	1,0	1,–	1,1
5	1,0	1,1	1,1	–	1,–	1,1	9,0
9	1,0	5,1	–,1	4,1	9,1	9,0	9,0

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

Closed Cover

maximal compatibles	class 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}	{}

Let's check if the following set of compatibles forms a **closed cover:** $\{c_1, c_4, c_5, c_9\}$

Coverage:

$a \in c_1$

$b, c \in c_5$

$d, e \in c_4$

$f, g \in c_9$

$h \in c_4$

Closure:

$\Gamma(c_1)$:

$\Gamma(c_4)$: $\{a,b\} \in c_1 \quad \{a,d\} \in c_1$

$\Gamma(c_5)$:

$\Gamma(c_9)$: $\{e,h\} \in c_4$

Covering Constraints--POS FORM

maximal compatibles	class 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}	{}

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

$$(c_1+c_{11})(c_1+c_2+c_5)$$
$$(c_2+c_3+c_5+c_6+c_7+c_8)$$
$$(c_1+c_2+c_4+c_6+c_{10})$$
$$(c_1+c_4)(c_3+c_7+c_9+c_{12})$$
$$(c_3+c_8+c_9+c_{11})$$
$$(c_4+c_{11})=1$$

Covering Constraints--POS FORM

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

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

$$(c_1 \overset{a}{+} c_{11})(c_1 + c_2 \overset{b}{+} c_5)$$
$$(c_2 + c_3 \boxed{\overset{c}{+} c_5} + c_6 + c_7)$$
$$(c_1 + c_2 \overset{d}{+} c_4 + c_6 + c_{10})$$
$$(c_1 \overset{e}{+} c_4)(c_3 + c_7 \overset{f}{+} c_9 + c_{12})$$
$$(c_3 + c_8 \overset{g}{+} c_9 + c_{11})$$
$$(c_4 \overset{h}{+} c_{10})$$

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}	{}
2	{b,c,d}	{{a,b},{a,g},{d,e}}
3	{c,f,g}	$\boxed{\{\{c,d\},}$ {e,h}}
4	{d,e,h}	{{a,b}, {a,d}}

$$\boxed{\begin{array}{c} b \\ (c_1 + c_2) \end{array}}$$

$a \quad b \quad c \quad d \quad e \quad f \quad g \quad h$

$$c_1(c_1+c_2)(c_2+c_3)(c_1+c_2+c_3)(c_1+c_4)c_3c_3c_4$$

$= c_1c_3c_4$ This cover is not closed, since c_2 is excluded

Closure Constraints

C_Γ is the set of prime compatibles with non-empty class sets

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

	C_Γ	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}}

$(c'_2 + c_1)$ $\{a,b\} \subset \{a,b,d,e\}$

$(c'_2 + c_{11})$ $\{a,g\} \subseteq \{a,g\}$

$(c'_2 + c_1 + c_4)$ $\{d,e\} \subset \{a,b,d,e\}$

$(c'_3 + c_4) \ldots$ $\{d,e\} \subset \{d,e,h\}$

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

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

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

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

$$\begin{array}{c|cccccccccccc}
 & c_1 & c_2 & c_3 & c_4 & c_5 & c_6 & c_7 & c_8 & c_9 & c_{10} & c_{11} & c_{12} \\
a & 1 & & & & & & & & & & 1 & \\
b & 1 & 1 & & & 1 & & & & & & & \\
c & & 1 & 1 & & 1 & 1 & 1 & 1 & & & & \\
d & 1 & 1 & & 1 & & & & & & & & \\
e & 1 & & & 1 & & & & & & & & \\
f & & & 1 & & & & 1 & & 1 & & & \cancel{1} \\
g & & & 1 & & & & & 1 & 1 & & 1 & \\
h & & & & 1 & & & & & & 1 & &
\end{array}$$

Row Dominance

Col Dominance? (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.

$$(c'_2 + c_1)$$

$$(c'_2 + c_{11})$$

$$\boxed{(c'_2 + c_1 + c_4)}$$

$$\begin{array}{c|cccccccccccc}
 & c_1 & c_2 & c_3 & c_4 & c_5 & c_6 & c_7 & c_8 & c_9 & c_{10} & c_{11} & c_{12} \\
\hline
\Gamma_2 & 1 & 0 & & & & & & & & & & \\
\Gamma_2 & & 0 & & & & & & & & & 1 & \\
\Gamma_2 & 1 & 0 & & 1 & & & & & & & &
\end{array}$$

Closure Constraints--Matrix FORM

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

$c_i' \Rightarrow c_j$

$$
\begin{array}{llcccccccccccc}
 & & c_1 & c_2 & c_3 & c_4 & c_5 & c_6 & c_7 & c_8 & c_9 & c_{10} & c_{11} & c_{12} \\
c_2' + c_1 & ☝_2 & 1 & 0 & & & & & & & & & & \\
c_2' + c_{11} & ☝_2 & & 0 & & & & & & & & & 1 & \\
c_2' + c_1 + c_4 & ☝_2 & 1 & 0 & & 1 & & & & & & & & \\
c_3' + c_4 & ☝_3 & & 1 & 0 & & & 1 & & & & & & \\
c_3' + c_2 + c_6 & ☝_3 & & & 0 & 1 & & & & & & & & \\
c_4' + c_1 & ☝_4 & 1 & & & 0 & & & & & & & & \\
c_4' + c_1 & ☝_4 & 1 & & & 0 & & & & & & & & \\
c_6' + c_{11} & ☝_6 & & & & & & 0 & & & & & 1 & \\
c_6' + c_1 + c_4 & ☝_6 & 1 & & & 1 & & 0 & & & & & & \\
\hline
c_7' + c_2 + c_6 & ☝_7 & & 1 & & & & 1 & 0 & & & & & \\
\hline
c_8' + c_2 + c_6 & ☝_8 & & 1 & & & & 1 & & 0 & & & & \\
c_8' + c_3 + c_9 & ☝_8 & & & 1 & & & & & 0 & 1 & & & \\
c_9' + c_4 & ☝_9 & & & & 1 & & & & & 0 & & & \\
\end{array}
$$

Closed Covering Problem

Covering Constraints

	1	2	3	4	5	6	7	8	9	10	11	12
a	1										1	
b	1	1			1							
c		1	1		1	1	1	1				
d	1	1		1		1				1		
e	1			1								
f			1				1		1			1
g			1					1	1		1	
h				1						1		

Closure Constraints

	1	2	3	4	5	6	7	8	9	10	11	12
☝ 2	1	0										
☝ 2		0									1	
☝ 2	1	0		1								
☝ 3		1	0			1						
☝ 3			0	1								
☝ 4	1			0								
☝ 4	1			0								
☝ 6						0					1	
☝ 6	1			1		0						
☝ 7		1				1	0					
☝ 8		1				1		0				
☝ 8			1					0	1			
☝ 9				1					0			

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)

```
  Procedure BCP($F, U, currentSol$){
1     $(F, currentSol) = \text{REDUCE}(M, currentSol)$
      if ($terminalCase(F)$){                        \\ $||F|| = 0$
          if($\boxed{F \neq 0}$ and $\text{COST}(currentSol) < U$){
              $U = \text{COST}(currentSol)$
2             return ($currentSol$)
          }
3         else return("no (better) solution (in this subspace)")
      }
4     $L = \text{LOWER\_BOUND}(F, currentSol)$
      if($L \geq U$) return("no (better) solution (in this subspace)")
5     $x_i = \text{CHOOSE\_VAR}(F)$                    \\ longest column
6     $S^1 = \text{BCP}(F_{x_i}, U, currentSol \cup \{x_i\})$
7     if($COST(S^1) = L$) return($S^1$)
      $S^0 = \text{BCP}(F_{x_i'}, U, currentSol)$
8     return  BEST_SOLUTION ($S^1, S^0$)
  }
```

Unacceptable (anti-essential)Variables

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

$$(x'_3 + x_2)(x'_3 + x_2 + x'_1)$$
$$= (x'_3 + x_2)$$

$$F = \begin{array}{|cccc|c|} x_1 & x_2 & x_3 & x_4 & \\ \hline \cancel{0} & \cancel{1} & \cancel{0} & \cancel{-} & f_1 \\ - & 1 & 0 & - & f_2 \\ 1 & - & - & 1 & f_3 \\ 1 & 0 & 1 & 0 & f_4 \end{array}$$

Row 1 (f_1) **dominates** row 2 (f_2) since row 2 matches row 1 at all care entries.

Row 1 may be deleted.

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

. $f_1 \leq f_2$

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) $f_{ij} = -$ **and** $f_{ik} \neq 1$

(3) $f_{ij} = 0$ **and** $f_{ik} = 0$

Example: reduced column F_1 dominates F_4

$$F = \begin{array}{|cccc|c|} \hline x_1 & x_2 & x_3 & x_4 & \\ \hline \cancel{0} & \cancel{1} & \cancel{0} & & f_1 \\ - & 1 & 0 & - & f_2 \\ 1 & - & - & 1 & f_3 \\ 1 & 0 & 1 & 0 & f_4 \\ \hline \end{array}$$

Maximal Independent Set

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

x_1	x_2	x_3	x_4	
1	1	–	–	f_1
–	1	1	–	f_2
–	0	–	1	f_3

$MIS = \{f_1\}$

x_1	x_2	x_3	x_4	
1	0	–	–	f_1
0	1	–	–	f_2
–	0	1	–	f_3
–	–	0	1	f_4

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:

$$F = \begin{array}{|cc|} x_1 & x_2 \\ \hline 1 & 1 \\ 0 & 1 \\ 1 & 0 \\ 0 & 0 \\ \hline \end{array} \equiv (x_1 + x_2)(x_1' + x_2)(x_1 + x_2')(x_1' + x_2') = 0$$

This is detected by REDUCTION, which discovers that both x_2 and x_2' are essential

Reduction

f_1 dominates f_2

x_1	x_2	x_3	x_4	
0	1	0	−	f_1
−	1	0	−	f_2
1	−	−	1	f_3
1	0	1	0	f_4

F_1 dominates F_4

x_1	x_2	x_3	x_4	
0	1	0	−	f_1
−	1	0	−	f_2
1	−	−	1	f_3
1	0	1	0	f_4

$x_4 = 0$

x_1	x_2	x_3	x_4	
0	1	0	−	f_1
−	1	0	−	f_2
1	−	−	1	f_3
1	0	1	0	f_4

x_1 is essential

x_1	x_2	x_3	x_4	
0	1	0	−	f_1
−	1	0	−	f_2
1	−	−	1	f_3
1	0	1	0	f_4

F_2 dominates F_3

x_1	x_2	x_3	x_4	
0	1	0	−	f_1
−	1	0	−	f_2
1	−	−	1	f_3
1	0	1	0	f_4

$x_3 = 0$

Solution:

$x = (1,0,0,0)$

State Encoding

- **The number of possible assignments is very high**
- **If one uses k bits to encode p states, there are $(2^k)!/(2^k - p)!$ 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 $(2^k - 1)! / (2^k - p)!\, k!$ 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 π 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

  $B_a \cap B_b = \Phi$ for $a \neq b$

  and $\cup \{ B_a \} = S$
- Each subset is called a block of the partition
- If π_1 and π_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

  $s=s_0\ s_1\ s_2 \ldots\ s_n = t$

  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 π 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 π_1 and π_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 (π, π') on the machine is an ordered pair of partitions on S such that**

  s $\equiv$ t(π) implies that δ(s,a) $\equiv$ δ(t,a) (π') $\forall$a $\in$ I

- **The knowledge of the block of π containing the present state and of the current input allows one to compute the block π' of that will contain the next state.**
- **It is evident that if (π, π) is a partition pair, then π has substitution property**
  - **Partition pairs generalize SP partitions**