The simplest to analyze.

We obtain a cover (I terms rather than 6) which has a simpler common cube decomposition of the functions - as these are encoding space.

However, if we try to assign to maximize the # of common cubes, in particular, the heuristics of interest are those arising from.

We can realize the states of this machine in several ways:

If we minimize the logic cover cardinality:

Better multilevel implementations. Larger cardinality and more common cubes should produce estimates for the size of a multilevel network - but a cover with 

Consider the following machine:

LOGIC IMPLEMENTATION — MUSTANG

STATE ASSIGNMENT FOR MULTILEVEL

"Mustang": State Assignment of Finite State Machines.

"Optimize Area of Multilevel Circuits.

TARGETING MULTILEVEL LOGIC IMPLEMENTATIONS, "IEEE Trans.

CAD v. 7, n. 12, December 1988.

"Mustang": State Assignment of Finite State Machines.

LOGIC IMPLEMENTATION — MUSTANG

STATE ASSIGNMENT FOR MULTILEVEL
Note: If two codes \( c_1 \) and \( c_2 \) represent states which share similar outputs, then \( \left( \sum_{i=1}^{n} d_i \right)^{1/n} \) (one bit change per word)

1. Number of codes at distance \( u = \dim D \).

The number of codes at distance \( k \) from some code \( c \) is a rapidly increasing function of \( k \) for \( k \gg u \).

\[
A^u_A \quad 0 = (c_1, c_2, \ldots, c_u) \\
A^v_A \quad 1 = (c_1, c_2, \ldots, c_v)
\]

Note: The Hamming distance provides a discrete metric on the Boolean space of dimension \( n \).

\[
\text{Distance: } \text{Hamming distance} = 4. \\
\text{Example: } (11010) \text{ and } (10100) \text{ have Hamming distance } 1. \\
\text{where } c_1 \text{ is the } i \text{th bit in the encoding of } c.
\]

\[
d_{c_1 c_2}^{1} = \sum_{i=1}^{n} d_i \quad (0 \neq 1) \\
\]

We wish to capture \( k \) effects in common cube sharing.

Problem: Assign an encoding to states so that the maximum number of common cubes are generated in the output and \# 

Strategy: Between each pair of states set a weight used to determine if the two states should be placed in a local cluster (of Hamming distance) close to each other. Then use this matrix of weights to guide embeddings of the states. Then use this matrix of weights to generate embeddings of all states to produce the same cube from different embeddings. Finally, if multiple states assert similar outputs, then there is a common cube with \( N \) that has a common cube whose size is the weight of \( c_2 \) and \( c_1 \) is the logic for \( c_1 \) and \( c_2 \) both successors to \( c_3 \).

1. If \( c_1 \) and \( c_2 \) both transit to \( c_3 \) and we assign codes \( 0 \) and \( 1 \). If \( c_1 \) and \( c_2 \) both transit to \( c_3 \) and we assign codes \( 0 \) and \( 1 \). We wish to capture \( k \) effects in common cube sharing.

We do not know which cubes will be used, so we must set the heuristic to create as large a set as possible.
CONSTRUCTING THE WEIGHTS
Inputs: Count #S of each state when input is 1, or 0 for each bit.

Output: Count the number of times (edges) in which state k will transition to state e.

We construct sets of similar present states which can be next to each other.

The number of common cubes implies a new common state. Maximize the number of common states and the number of similar states leading to a common state. Then count the number of similar inputs in the present state. We wish to exploit the input similarities in the function.
Hamming distances between the codes.

We are now left with the classical problem of assigning codes to
the states to minimize the sum of the edge weights times the

Edge weights for inputs are counted for next states by first finding
the number of similar inputs for a given next state. Then edge
weights are assigned to states with similar

This fortuitous the encoding to place these states with similar

The edge weight for present states is counted by noting that each

The same example: input sets:

 Edge weights are assigned to states with many common inputs

This time the encoding is slightly different and the

Finally, calculate the edge weights for the graph as before.
We define a cluster as essentially greedy assignment.

Intersect several strongly connected clusters only weakly connected matrices constructed have strong structure, i.e., we use the cluster. This should prove an effective technique since the existence of several heuristics here we shall use Wedge.

This problem is NP-hard, as it contains a specialized but #P-

\[
\begin{array}{c}
\text{Minimize} & \sum_{i=1}^{N} x_i \sum_{j=1}^{N} y_j \\
\text{subject to} & x_i, y_j \in \{0, 1\} \\
\end{array}
\]

The weight matrix.

The Hamming distance between two codes is the edge weight in the induced graph of the \( G \). The induced weight is the induced local weight of the \( G \). The induced weight is the induced local weight of the \( G \). The induced weight is the induced local weight of the \( G \).

\[ G = G \]

Wedge clustering.

Boolean space connected by the Hamming metric.

Consider the Boolean lattice: The elements of an n-dimensional
Results could be either 110 or 011.

If (010) and 110, 111, 011 are free,
we need sf1 to be 1 until distinct from

detect sf2 and edges; now choose sf1 (010)

so is chosen as it has the maximum set of 3 edge weights.

Results:

Example:

N = 3 for 5 states:
Consider the set of edges which, when removed, have been changed. Since only one bit has been changed, the new code is identical to the original code in all but one position, and the change is in the cost of the encoding. Since only a single bit has been changed in the cost of the encoding, the new code is identical to the original code in all but one position, and the change is in the cost of the encoding.

If a move is necessary to complete a move is necessary to complete the move.

1. If the new code is identical to an already existing code, then the move is possible.

2. If the new code (after inversion) is a new unique vector, then the move is possible.

A fast annealing move can be found by modifying the following function:

$$\mathbf{C} = \left( \mathbf{a}^T \mathbf{1}_A \right) \mathbf{H} \left( \mathbf{a}_i \mathbf{1}_A \right)$$

where \( \mathbf{w} \) is the Hamming distance between \( \mathbf{a} \) and \( \mathbf{a}_i \).

Boolean Graph Embedding

Problem (Undirected Graph Embedding): Given a graph (\( G(V, E) \)) and a move and a reasonable smooth cost manifold, find a simple move which produces a much better solution. A simple move is not necessarily a

Where C is a correlation between the values and the cost.

Given a graph (\( G(V, E) \)) and a smooth cost manifold, find a simple move which produces a much better solution. A simple move is not necessarily a.
Applications of Boolean Relations
A sequence is admissible iff each transition is from a well defined state to a well defined state.

Definitions:

- **Output**: A sequence of inputs and outputs can force a machine to be in a reachable state.
- **State**: A reachable state is a state where the machine can be in a state.
- **Initial State**: The initial state is the input as the output function.
- **Next State Function**: The next state function is the output of the function.
- **Input**: The input is the sequence of inputs.
- **Output Function**: The output function is the sequence of outputs.
- **State Machine**: A state machine is defined as a 6-tuple: \( M = (\mathcal{S}, \mathcal{I}, \mathcal{O}, \delta, \mathcal{I}_0) \), where \( \mathcal{S} \) is the set of states, \( \mathcal{I} \) is the set of inputs, \( \mathcal{O} \) is the set of outputs, \( \delta \) is the transition function, and \( \mathcal{I}_0 \) is the initial state.

Let \( f : \mathcal{S} \times \mathcal{I} \rightarrow \mathcal{O} \) be a mapping of states to outputs.

**Definition**: A mapping \( f : \mathcal{S} \times \mathcal{I} \rightarrow \mathcal{O} \) is a **compatibility mapping** if for all states \( s \in \mathcal{S} \) and inputs \( i \in \mathcal{I} \), there exists an output \( o \in \mathcal{O} \) such that \( (s, i) f = o \).

Let \( D \) be the set of symbols and \( \mathcal{I} \) be the set of inputs. Then a mapping \( f : D \times \mathcal{I} \rightarrow \mathcal{O} \) is a **compatibility mapping** if for all symbols \( d \in D \) and inputs \( i \in \mathcal{I} \), there exists an output \( o \in \mathcal{O} \) such that \( (d, i) f = o \).

Let \( \phi \) be a Boolean function defined as \( \phi(x, y) \).

For each symbol \( x \in D \), let \( \mathcal{R}(x) \) be a set of symbols and \( \mathcal{I} \) be the set of inputs. Then a mapping \( f : D \times \mathcal{I} \rightarrow \mathcal{O} \) is a **compatibility mapping** if for all symbols \( d \in D \) and inputs \( i \in \mathcal{I} \), there exists an output \( o \in \mathcal{O} \) such that \( (d, i) f = o \).

Given a function \( \phi : \mathcal{R} \rightarrow \mathcal{O} \), let \( \mathcal{R}(x) \) be the set of symbols and \( \mathcal{I} \) be the set of inputs. Then a mapping \( f : D \times \mathcal{I} \rightarrow \mathcal{O} \) is a **compatibility mapping** if for all symbols \( d \in D \) and inputs \( i \in \mathcal{I} \), there exists an output \( o \in \mathcal{O} \) such that \( (d, i) f = o \).

**Definition**: A mapping \( f : D \times \mathcal{I} \rightarrow \mathcal{O} \) is a **compatibility mapping** if for all symbols \( d \in D \) and inputs \( i \in \mathcal{I} \), there exists an output \( o \in \mathcal{O} \) such that \( (d, i) f = o \).

Let \( \mathcal{R}(x) \) be the set of symbols and \( \mathcal{I} \) be the set of inputs. Then a mapping \( f : D \times \mathcal{I} \rightarrow \mathcal{O} \) is a **compatibility mapping** if for all symbols \( d \in D \) and inputs \( i \in \mathcal{I} \), there exists an output \( o \in \mathcal{O} \) such that \( (d, i) f = o \).
Note that the state of $M_1$ is equivalent to $s_3 = s_4$ of $N_2$.

Consider the following machine:

### State Assignment Revised

\[
\begin{array}{c|cccc}
\text{Input} & 0 & 1 & 0 & 1 \\
\hline
\text{Output} & 0 & 0 & 0 & 0 \\
\end{array}
\]

Note that the state of $M_1$ is equivalent to $s_3 = s_4$ of $N_2$.

However, there is no guarantee that choosing a minimal machine will lead to a smaller design.

When synthesizing a new state machine, we must take equivalent machines into account.

Consider the following machine:

\[
\begin{array}{c|cccc}
\text{Input} & 0 & 1 & 0 & 1 \\
\hline
\text{Output} & 0 & 0 & 0 & 0 \\
\end{array}
\]

Note that the state assignment can be constructed directly from

\[
\begin{array}{c|cccc}
\text{Input} & 0 & 1 & 0 & 1 \\
\hline
\text{Output} & 0 & 0 & 0 & 0 \\
\end{array}
\]

Machine $M_1$ will produce the logic $N_2$ below:

\[
\begin{array}{c|cccc}
\text{Input} & 0 & 1 & 0 & 1 \\
\hline
\text{Output} & 0 & 0 & 0 & 0 \\
\end{array}
\]

Note that no conventional state assignment encoding for the minimal logic is identical to that of $M_2$, given the same inputs. However, $M_2$ is state minimal.
Definition: A cube $C$ is a cube with support set $C = C \subseteq C$ if it is the union of $2$ adjacent cubes $C$ and $C'$, denoted $C \cup C'$. C

define: $\exists x \in A \setminus C \cup C'$.

De: $C = C \subseteq C$ is a cube of $2$ adjacent cubes $C$ and $C'$ denoted $C \cup C'$.

De: $\exists x \in A \setminus C \cup C'$.

De: $C = C \subseteq C$ is a cube of $2$ adjacent cubes $C$ and $C'$ denoted $C \cup C'$.

De: $\exists x \in A \setminus C \cup C'$.

De: $C = C \subseteq C$ is a cube of $2$ adjacent cubes $C$ and $C'$ denoted $C \cup C'$.

De: $\exists x \in A \setminus C \cup C'$.

De: $C = C \subseteq C$ is a cube of $2$ adjacent cubes $C$ and $C'$ denoted $C \cup C'$.

De: $\exists x \in A \setminus C \cup C'$.

De: $C = C \subseteq C$ is a cube of $2$ adjacent cubes $C$ and $C'$ denoted $C \cup C'$.

De: $\exists x \in A \setminus C \cup C'$.
size $s$, or $A^s$ refers to the set of maximal implications of $A^{s-1}$ or $A^s$ are produced.

The cube in $A^s$ ranges from 1 to $t$ after each iteration all maximal implicates

\[ A^s \cap d = d \]

\[ A^s \cap \{ q \} \]

mark all $q \in A^s$ such that $A^s \cap \{ q \}$ is $\forall$.

\[ \forall \in A^s \forall \cap d = d \]

remove all marked vertices from the influence sets of $A^s$.

\[
\begin{cases}
\{ q \} \cap B = B \\
{ } \forall \in A^s \forall \cap d = d
\end{cases}
\]

mark all $q \in A^s$ such that $A^s \cap \{ q \}$ is $\forall$.

\[ \forall \in A^s \forall \cap d = d \]

For each pair $(p, q)$ such that $A^s \cap \{ p, q \}$ is $\forall$, for $i = 1, \ldots, s$

\[ \forall \in A^s \forall \cap d = d \]

[ fundamental implicates of $R$]

output: The set of all prime implicates of $R$.

Input: The set of fundamental implicates of $R$.

Here the $\forall$ is simply bit-wise AND of the elements.

\[
\begin{cases}
\forall \in A^s \forall \cap d = d \\
\forall \in A^s \forall \cap d = d \\
\forall \in A^s \forall \cap d = d
\end{cases}
\]

where \( y \in I, \forall \in I \), $y$ is the greatest lower bound of $B$ such that

\[ B \cap \{ y \} = I \]

and influence set

\[ \forall \in A^s \forall \cap d = d \]
set of c-primes.

There are no more adjacent c-values; so \( \ldots \) are the complete

\[
\begin{align*}
\text{c}_3 \text{ is adjacent to } & \text{c}_4 \iff \text{c}_3 = \text{c}_4 = 11 - 10 \\
\text{c}_2 \text{ is adjacent to } & \text{c}_4 \iff \text{c}_2 = \text{c}_4 = 11 - 10 \\
\text{c}_1 \text{ is adjacent to } & \text{c}_4 \iff \text{c}_1 = \text{c}_4 = 11 - 11 \\
\end{align*}
\]

<table>
<thead>
<tr>
<th></th>
<th>11</th>
<th>11</th>
</tr>
</thead>
<tbody>
<tr>
<td>c₃</td>
<td>01</td>
<td>01</td>
</tr>
<tr>
<td>c₂</td>
<td>01</td>
<td>01</td>
</tr>
<tr>
<td>c₁</td>
<td>01</td>
<td>01</td>
</tr>
</tbody>
</table>

The fundamental implications are:

\[
\begin{align*}
\text{R} \iff 00 & \iff 010 \\
\text{R} \iff 00 & \iff 010 \\
\text{R} \iff 00 & \iff 100 \\
\text{R} \iff 00 & \iff 000 \\
\end{align*}
\]

Consider the following relation:

\[
\text{Ex.} \quad \text{Consider the following relation:}
\]

We can form these constraints in conjunctive form.

However, we must have: \( f(x) \in \mathcal{R}(x), \forall x \in \mathcal{P} \)

In general, \( f = a_1 + a_2 + a_3 \cdots + a_n \)

Note that these primes cannot be used in the conjunctive form:

Each member of \( \mathcal{P} \) is produced by merging the adjacent cubes of \( A-1 \) and each is a maximal cube.

Elements current set are deleted. Thus each prime that is prime to the other is in the adjacency set. This implies that the primes that are covered by the adjacency set are not covered by the adjacency set.
Given $N = 1$, we use $N$ bits to encode the outputs.

The idea is to encode the outputs of symbols as 0-1 hot encodings. The encoding is to encode the outputs of symbols as 0-1 hot encodings.

For encoding functions $f$ and $g$:

$$\forall x \in D, \exists y \in E : f(x) = g(y)$$

For each choice of code choices (encodings) and possibilities of symbolic relations, we must account for all possible combinations of code choices (encodings) and symbolic relations.

To generate all prime implicants for a symbolic relation, we will also base our Binary Cover Problem. We will use the output of all combinations of inputs and outputs. This will allow us to find all combinations of prime implicants for symbolic relations.

We will adopt a similar strategy as before and generate a symbolic prime (or 0-prime) for a symbolic prime.

We can write: $R(A,B) = f$

For which $A \in (\omega \cap \sigma)$ and $B \in (\sigma \cap \omega)$, where $f$ is a prime of $\omega$.

The prime implicants of a symbolic relation, $R$, is a set of all symbolic primes (or 0-primes) of a symbolic relation.
Let $\prod_{i=1}^{n} g_i$ be the output of $g_i$.

The set of all prime implicants which cover $x$: $R(x) = \{d^0\ldots d^n \mid \forall x \in \mathcal{B} \}$

1. Consider a minimum $x \in \mathcal{B}$, let $x$ have no excess.

2. We must insure that each symbol is uniquely function $f$ subject to the output choices of $R$.

I. We must have compatibility to a realizable function $f$.

Given encoding assuming $N > 2^L$.

There are $2^L$ constraints which are needed to validate a relation is a minimum.

Let the number of product terms in the minimized such that $\mathcal{B} \times \mathcal{B}$ is at most $L$. This is $\forall q \in \mathcal{B}$, $q^L = \sum_{q} q^L$.

$R \subseteq \mathcal{B} \times \mathcal{B}$ where $\mathcal{B}$ is the set of all symbols.

Given a binary input symbolic output relation $f$.

Output Encoding

Induction in a compatible function $f$.

Derivation of constraints: Given $c$-primes from above.

Generalization for multiple value inputs.

Recursive encoding for Boolean encoded inputs and can be extended to $c$-prime implicants of the cube is non-prime for final encoding of length $L$.

For each new cube, the number of zeros in its intersection.

For $n > 2^L - 1$, any $n$ distinct cubes have a zero.

Length is $L$ we can remove several cubes since: $\mathcal{B}$.

If we know a prime that the eventual output encoding has a minimal cover. Any output symbols $c$ cannot be a member of a $c$-prime, with all-1 output can (and should) be removed.

I. C-primes with all-1 output can (and should) be removed.

Generation procedure with the following changes:

The fundamental implicants. Then apply the $c$-prime encoding to make the relation as above (output encoding) to make.

For a given implicant:

For Relations, we may have the choice of several outputs.
(next time: static encoding)

Cover:
For this problem, we produce minimal 2-level weight. We shall assume weight (W) = 1, except when consensuses and find the path to 1 which has lost can construct a BDD which represents the set of to build an instance of BGP from these constraints, we

\[ I = \left( \sum_{q=1}^{\sum_{N=1}^{l-1}} \sum_{N=1}^{l-1} \right) \]

Note: the first term ensures no illegal set of simultaneous assignments are made from the set of variables. \( \sum_{q=1}^{\sum_{N=1}^{l-1}} \) is the first bit of the \( i \)-th symbol. Each \( \sum_{N=1}^{l-1} \) represents a boolean selection variable.

\[ \left( x \cdot d \right) e \equiv \left( \left( x \cdot d \right) \bigoplus \sum_{N=1}^{l-1} \sum_{N=1}^{l-1} \right) \]

For \( N \) values, we ensure that the final codes are orthogonal in \( l \) bits.

Equation II:

\[ I = \left( \sum_{q=1}^{\sum_{N=1}^{l-1}} \sum_{N=1}^{l-1} \right) \]

We must have

\[ \{ i \} = \{ I \} \quad \text{for all } i \]

Each \( \sum_{N=1}^{l-1} \) is an output port.

Induce \( \sum_{N=1}^{l-1} \) in its output port.

List of indices for which a prime covering \( x \) also "in" \( x \) is a

Finally, let \( I \).

Finally, let \( I \).
that only prime implications are kept.

This leads to a list of cubes of successively larger size and

that of \( c = 1 \Rightarrow (\text{out } 1 \text{ or out } 2 \text{ or out } 3) \)

where 2.

Then, if the output port of \( \text{out } 1 \) and of

symbolic parts (union of the symbols)

forming union of the cube part and intersecting the

merge of 2 adjacent symbolic output cubes by

1.

Relations we saw last time:

We can construct all generalized prime cubes for this

cover for an encoding of a Boolean function.

So... consider the problem of constructing 2-level

exactly encodings.

Our earlier techniques failed to make use of this

resulting in \( I = \text{out } 1 \).

This is because the case for \( \text{out } 1 \) satisfies both cubes:

\[
\begin{array}{c|ccc}
I & 0 & 1 & 1 \\
\text{out } 1 & 1 & 0 & 1 \\
\text{out } 2 & 1 & 0 & 1 \\
\text{out } 3 & 0 & 1 & 1 \\
\end{array}
\]

we can realize the function in 2 cubes...

However, if we assign

previous technique no reduction is possible.

This machine is output disjoint so that by our

output: 1 1 0 1 0 0 1

Consider the following machine:

These covers:

- Cardinality - there is another way to reduce the size of
- Reduction among the codes to reduce cover

In our earlier study of products we used dominance.

Sideline: Where did those constraints come from?
A valid output encoding.

This simply ensures that any input minterm produces

\[(x'0) \delta = \left( \bigcup_{f' \in \{0, 1\}} \bigcup_{x \in \mathcal{X}} \prod_{f' \in \{0, 1\}} x'0 \bigcup \prod_{x \in \mathcal{X}} \right) \]

Output cover.

Let \( f(x) \) be the Boolean function which selects is

\[\prod_{f' \in \{0, 1\}} \prod_{x \in \mathcal{X}} \]

Output cover.

Let \( p(x) \) be the Boolean function which covers \( x \).

Encoding.

Note that if we encode the symbols as Boolean

Boolean implementation (in 2-levels).

We wish to formulate exact constraints on the primes

Symbols must agree for adjacency.

(Note: 1 - is not adjacent to 0 - don't care)

\[
\begin{array}{c|cccc}
\text{c} & 0000 & 0010 & 0100 & 1000 \\
\text{a} & 0100 & 0010 & 1000 & 0000 \\
\text{q} & 0110 & 1010 & 0100 & 0000 \\
\text{p} & 0010 & 0101 & 1010 & 0100 \\
\end{array}
\]
Finally we can let $p = 1$ and $q' = 0$ and

$$I = \mathbb{Z}_q$$

requires that $q$ and $I$.

Note that $80 = 81 = 82 = 83 = 84 = 85 = 86 = 87 = 88 = 89$.

(difficult output symbols)

plus, we must also insure that the $p$ are disjoint

\[
\begin{align*}
(\mathbb{Z}_q^+ + \mathbb{Z}_q^+ + \mathbb{Z}_q^+ + \mathbb{Z}_q^+ + \mathbb{Z}_q^+ + \mathbb{Z}_q^+ + \mathbb{Z}_q^+ + \mathbb{Z}_q^+) & \iff \text{I} \\
(\mathbb{Z}_q^+ + \mathbb{Z}_q^+ + \mathbb{Z}_q^+ + \mathbb{Z}_q^+ + \mathbb{Z}_q^+) & \iff \text{I} \\
(\mathbb{Z}_q^+ + \mathbb{Z}_q^+ + \mathbb{Z}_q^+ + \mathbb{Z}_q^+ + \mathbb{Z}_q^+) & \iff \text{I}
\end{align*}
\]

the conditions are then: $l = 2$

which is the exact minimal solution.

$01 = (11 \cup 01) = (\ell \cup q)$

$11 - 1$

$I = (11 \cup 10) = (\ell \cup a)$

$I - 1$

This gives the solution:

which is the exact minimal solution.

$01 = (11 \cup 01) = (\ell \cup q)$

$11 - 1$

$I = (11 \cup 10) = (\ell \cup a)$

$I - 1$

We get: $01 = 10$

$01$

Example: For the initial cover: $\ell = 1$

$\ell = 1$

We get: $01 = 10$

$A x 1$

which is the exact minimal solution.

$01 = (11 \cup 01) = (\ell \cup q)$

$11 - 1$

$I = (11 \cup 10) = (\ell \cup a)$

$I - 1$

This gives the solution:

which is the exact minimal solution.
Each such cube prime introduces face constraints:

\[
\text{(S3)}\ \text{cover e(S2),}
\]

smallest cube which covers e(S2), does not

this prime is selected, we must ensure that the

input a constraint on the input encoding. Since it

which has more than 1 \( e.g.: 0 \rightarrow 110\) (S1, S3) (0)

\text{Note: each prime with an input symbolic encoding}

\[
\begin{array}{cccc}
1 & 1 & 1 & 0 \\
1 & 0 & 0 & 1 \\
0 & 0 & 1 & 1 \\
0 & 1 & 0 & 1 \\
0 & 1 & 1 & 0
\end{array}
\]

Cancelation can occur only when:

... symbolic parts

1. Identical input binary parts and

2. Identical input parts and different symbolic parts.

2 cubes can be merged only if:

algorithm as before with the caveat:

... symbolic input into a multiple-valued input (6800).

We will use a photon encoding for the symbolic input.

It has been shown that for a pure symbolic input

Back to Symbolic Relations.

For this problem we have the difficulty that there are

... symbolic inputs as well as outputs and these

... symbolic inputs cause a change in the previous prime

... symbolic inputs as well as outputs and these

... symbolic inputs cause a change in the previous prime

... symbolic inputs as well as outputs and these
\[
I = \frac{1}{N} \left( \sum_{i=1}^{I-1} \prod_{j \neq i} 1 \right) 
\]

and adding new implied merging constraints.

This is done by modifying the Disjoint Constraints.

Equivalent states need not have disjoint codes.

Relax the requirement of unique state codes. If, 

solve the symbolic relation problem in which we we

solve the symbolic relation problem in which we

state equivalent states.

and any transition to a state could transition to any of

and build an implication graph (AY, E) where each

state equivalent and implied pairs.

1. Apply conventional state minimization to identify

defects of freedom.

We can now generalize the state encoding problem by

Joint State Minimization and State Encoding

Note that this is just an exhaustive check of all satisfying

For each constraint

Then:

be the set of states which must not

let 0 \leq \cdot \cdot \cdot \sigma_i

be the set of states in the face and

particular face constraint

select the primes which produce a

following constraint in the encoding

an all-1's input port which is discarded (we add the

for each prime with a merged symbolic input part (except

section of the primes.

defined relation we can find an encoding for some

or absence of a prime in the cover. Thus for any well

compatible.

compatible.

cover was encodable. i.e. p matrices were

In our previous discussion we performed the
containment is very important
over 10,000 variables. So BDD size
with 100 primes (relatively small) we
have
However: We must prune the BDD often as a problem

O(V) on an ROBDD by simply applying Dijkstra's algo.

We can find this shortest path weighted solution in linear
time.

The assignment:

\[ \text{Every path from } 1 \text{ to the root represents a satisfiable assignment.} \]

For an ROBDD representing \( T(x_1, x_2, \ldots, x_n) \), the lowest value of the variable order:

\[ \text{arc is free and a } 1 \text{ arc costs } \omega \text{ for variable } x. \]

Define:

\[ \text{cost of } a \text{ path in a BDD (D) is defined to be the} \]

Solution using BDD's:

Find a minimum cost satisfying assignment.

Let the cost of a negative literal be given by \( \omega \) and the

Let \( \mathcal{L}(X) = 1 \).

If \( \mathcal{L}(X) \neq 1 \), the covering constraints where \( \mathcal{X} \) is satisfiable

Let the Boolean Formula \( \mathcal{L}(X_1, \ldots, X_n) \) represent the

Binate Covering Problem:

\[
I = \left( \left( \gamma^b_1 \oplus \gamma^d_1 \right) \bigcup \frac{1}{2} \right) \bigcup \left( \gamma^b_2 \oplus \gamma^d_2 \right) \bigcup \frac{1}{2} \bigcup \frac{1}{2} \bigcup \frac{1}{2} \bigcup \frac{1}{2} \\
I = \left( \left( \gamma^b_1 \ominus \gamma^d_1 \right) \left( \gamma^b_2 \ominus \gamma^d_2 \right) \left( \gamma^b_3 \ominus \gamma^d_3 \right) \left( \gamma^b_4 \ominus \gamma^d_4 \right) \left( \gamma^b_5 \ominus \gamma^d_5 \right) \left( \gamma^b_6 \ominus \gamma^d_6 \right) \left( \gamma^b_7 \ominus \gamma^d_7 \right) \left( \gamma^b_8 \ominus \gamma^d_8 \right) \left( \gamma^b_9 \ominus \gamma^d_9 \right) \left( \gamma^b_{10} \ominus \gamma^d_{10} \right) \right) \bigcup \frac{1}{2} \bigcup \frac{1}{2} 
\]

Here, the only difference is that we don't check the
Both timed.

No face constraints since either I or 2 possible merges.

Distinct coding constraint: \( q \oplus 2 = 1 \)

Converting M2 to primes leads to the following list:

We can encode this machines state with just 1 bit — but

Example: Consider the state machines below:

\[
\begin{align*}
\text{Initial State:} & \quad 0' \quad q \quad 0\\
\text{Next State:} & \quad 0' \quad q' \quad 0'\\
\end{align*}
\]