DECOMPOSITION OF RELATIONS:
A NEW APPROACH TO CONSTRUCTIVE INDUCTION IN MACHINE LEARNING AND DATA MINING - AN OVERVIEW

Marek Perkowski
Portland State University
Data Mining Application for Epidemiologists

Control of a robot

FPGA

Machine Learning from Medical databases

VLSI Layout
This is a review paper that presents work done at Portland State University and associated groups in years 1989 - 2001 in the area of functional decomposition of multi-valued functions and relations, as well as some applications of these methods.
Group Members

**Current Students:**
- Anas Al-Rabadi
- Marek Perkowski
- Alan Mishchenko

**Faculty**
- Bernd Steinbach
- Lech Jozwiak
- Martin Zwick

**Previous Students:**
- Stanislaw Grygiel, Ph.D., Intel
- Craig Files, Ph.D., Agilent
- Paul Burkey, Intel
- Rahul Malvi, Synopsys
- Michael Burns, Vlsi logic
- Timothy Brandis, OrCAD
- Tu Dinh
- Michael Levy, Georgia Tech
Essence of logic synthesis approach to learning
Example of Logical Synthesis

John
Mark
Dave
Jim
Alan
Mate
Nick
Robert
A - size of hair
B - size of nose
C - size of beard
D - color of eyes

Good guys
- Dave
- Jim

Bad guys
- Alan
- Mate
- Nick
- Robert

John
Mark
### Good guys

<table>
<thead>
<tr>
<th></th>
<th>00</th>
<th>01</th>
<th>11</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>AB</td>
<td>00</td>
<td>-</td>
<td>-</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>01</td>
<td>-</td>
<td>-</td>
<td>1</td>
</tr>
<tr>
<td>11</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>10</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

- **A** - size of hair
- **B** - size of nose
- **C** - size of beard
- **D** - color of eyes
<table>
<thead>
<tr>
<th>CD</th>
<th>AB</th>
<th>00</th>
<th>01</th>
<th>11</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>1</td>
</tr>
<tr>
<td>01</td>
<td>0</td>
<td>0</td>
<td>-</td>
<td>1</td>
<td>-</td>
</tr>
<tr>
<td>11</td>
<td>-</td>
<td>-</td>
<td>0</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>10</td>
<td>-</td>
<td>0</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

Legend:
- **A** - size of hair
- **B** - size of nose
- **C** - size of beard
- **D** - color of eyes

**Bad guys**
- Alan
- Mate
- Nick
- Robert
Generalization 1:
Bald guys with beards are good

Generalization 2:
All other guys are no good

A - size of hair
B - size of nose
C - size of beard
D - color of eyes

A’C
Short Introduction: multiple-valued logic

Signals can have values from some set, for instance \{0,1,2\}, or \{0,1,2,3\}

\{0,1\} - binary logic (a special case)
\{0,1,2\} - a ternary logic
\{0,1,2,3\} - a quaternary logic, etc
Types of Logical Synthesis

- Sum of Products
- Decision Diagrams
- Functional Decomposition
Sum of Products

AND gates, followed by an OR gate that produces the output. (Also, use Inverters as needed.)
A Decision diagram breaks down a Karnaugh map into set of decision trees.

A decision diagram ends when all of branches have a yes, no, or do not care solution.

This diagram can become quite complex if the data is spread out as in the following example.

**Example Karnaugh Map**

<table>
<thead>
<tr>
<th>00</th>
<th>01</th>
<th>10</th>
<th>11</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>1</td>
<td>-</td>
<td>1</td>
</tr>
<tr>
<td>01</td>
<td>-</td>
<td>1</td>
<td>-</td>
</tr>
<tr>
<td>10</td>
<td>1</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>11</td>
<td>0</td>
<td>1</td>
<td>-</td>
</tr>
</tbody>
</table>
Decision Tree for Example
Karnaugh Map
Incompletely specified function
BDD Representation of function

Completely specified function
Functional Decomposition

Evaluates the data function and attempts to decompose into simpler functions.

\[
F(X) = H( G(B), A ), \quad X = A \cup B
\]

- If \( A \cap B = \emptyset \), it is disjoint decomposition.
- If \( A \cap B \neq \emptyset \), it is non-disjoint decomposition.

\( X \) - bound set

\( B \) - bound set

\( A \) - free set
Pros and cons

In generating the final combinational network, BDD decomposition, based on multiplexers, and SOP decomposition, trade flexibility in circuit topology for time efficiency.

Generalized functional decomposition sacrifices speed for a higher likelihood of minimizing the complexity of the final network.
Overview of data mining
What is Data Mining?

Databases with millions of records and thousands of fields are now common in business, medicine, engineering, and the sciences.

To extract useful information from such data sets is an important practical problem.

Data Mining is the study of methods to find useful information from the database and use data to make predictions about the people or events the data was developed from.
Some Examples of Data Mining

1) Stock Market Predictions

2) Large companies tracking sales

3) Military and intelligence applications
Epidemiologists track the spread of infectious disease and try to determine the disease's original source.

Often times Epidemiologists only have an initial suspicion about what is causing an illness. They interview people to find out what those people that got sick have in common.

Currently they have to sort through this data by hand to try and determine the initial source of the disease.

A data mining application would speed up this process and allow them to quickly track the source of an infectious disease.
Types of Data Mining

Data Mining applications use, among others, three methods to process data:

1) Neural Nets
2) Statistical Analysis
3) Logical Synthesis

*The method we are using*
### A Standard Map of function ‘z’

<table>
<thead>
<tr>
<th></th>
<th>Bound Set</th>
<th>Free Set</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>a b \ c</td>
<td>0 0</td>
<td>0 1</td>
<td>0 2</td>
</tr>
<tr>
<td>0 0</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>0 1</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>0 2</td>
<td>1</td>
<td>0 , 1</td>
<td>-</td>
</tr>
<tr>
<td>1 0</td>
<td>-</td>
<td>-</td>
<td>2</td>
</tr>
<tr>
<td>1 1</td>
<td>-</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>1 2</td>
<td>-</td>
<td>1</td>
<td>-</td>
</tr>
<tr>
<td>2 0</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>2 1</td>
<td>-</td>
<td>-</td>
<td>0</td>
</tr>
<tr>
<td>2 2</td>
<td>-</td>
<td>2 , 3</td>
<td>-</td>
</tr>
</tbody>
</table>

Columns 0 and 1 and columns 0 and 2 are compatible. Column compatibility = 2
Decomposition of Multi-Valued Relations

If $A \cap B = \emptyset$, it is disjoint decomposition.

If $A \cap B \neq \emptyset$, it is non-disjoint decomposition.

\[ F(X) = H( G(B), A ), \quad X = A \cup B \]
### Forming a CCG from a K-Map

#### Bound Set

<table>
<thead>
<tr>
<th></th>
<th>a</th>
<th>b</th>
<th>c</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>0</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>-</td>
<td>-</td>
<td>2</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>-</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>-</td>
<td>1</td>
<td>-</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>-</td>
<td>-</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>-</td>
<td>2</td>
<td>3</td>
</tr>
</tbody>
</table>

Columns 0 and 1 and columns 0 and 2 are compatible.

**Column Compatibility Graph**

- C₀
- C₁
- C₂

**Column Compatibility Index** = 2
Forming a CIG from a K-Map

Columns 1 and 2 are incompatible

Chromatic number = 2

Column Incompatibility Graph
CCG and CIG are complementary

Maximal clique covering clique partitioning

Graph coloring graph multi-coloring

Compatibility Graph

Column Incompatibility Graph
clique partitioning
example.
Maximal clique covering example.
$g = a$ high pass filter whose acceptance threshold begins at $c > 1$
Cost Function

*Decomposed Function Cardinality* is the total cost of all blocks.

Cost is defined for a single block in terms of the block’s $n$ inputs and $m$ outputs

$$\text{Cost} := m \times 2^n$$
DFC = Decomposed Function Cardinality

\[ C_x(f) = \log_2 \min \{ \text{cost of } \Gamma : \Gamma \text{ simulates } f \} \]

\[ \text{cost}(f) = 2^{|X|}|Y| \]
Example of DFC calculation

Cost(B1) = $2^4 \times 1 = 16$

Cost(B2) = $2^3 \times 2 = 16$

Cost(B3) = $2^2 \times 1 = 4$

Total DFC = 16 + 16 + 4 = 36

Other cost functions
New Complexity Measures

\[ C_x = \log_2 \left( \prod_{x_i \in X} |x_i| \prod_{y_j \in Y} |y_j| \right) \]

where:  
\( |x_i| \) is cardinality of variable \( x_i \in X \),  
\( |y_j| \) is cardinality of variable \( y_j \in Y \).

\[ C_x = \log_2 \left( \prod_{y_j \in Y} |y_j| \right)^{\prod_{x_i \in X} |x_i|} = \prod_{x_i \in X} |x_i| \log_2 \prod_{y_j \in Y} |y_j| \]
Comparison of RC before and after decomposition

\[ RC_{\text{before}} = (3 \times 3 \times 3) \times (\log_2 4) = 54 \]

\[ RC_{\text{after}} = [(3) \times (\log_2 2)] + [(2 \times 3 \times 3) \times (\log_2 4)] = 3 + 36 = 39 \]
Two-Level Curtis Decomposition

F(X) = H( G(B), A ),  X = A ∪ B

if A ∩ B = ∅, it is disjoint decomposition
if A ∩ B ≠ ∅, it is non-disjoint decomposition
Decomposition Algorithm

- Find a set of partitions \((A_i, B_i)\) of input variables \((X)\) into free variables \((A)\) and bound variables \((B)\)
- For each partitioning, find decomposition \(F(X) = H_i(G_i(B_i), A_i)\) such that column multiplicity is minimal, and calculate DFC
- Repeat the process for all partitioning until the decomposition with minimum DFC is found.
Algorithm Requirements

- Since the process is iterative, it is of high importance that minimization of the column multiplicity index is done as fast as possible.
- At the same time, for a given partitioning, it is important that the value of the column multiplicity is as close to the absolute minimum value.
### Column Multiplicity

#### Bound Set

<table>
<thead>
<tr>
<th></th>
<th>00</th>
<th>01</th>
<th>11</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>0</td>
<td>0</td>
<td>−</td>
<td>1</td>
</tr>
<tr>
<td>01</td>
<td>−</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>11</td>
<td>1</td>
<td>−</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>10</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

#### Free Set

- 00
- 01
- 11
- 10

#### Node Connections

- 1 connects to 3
- 2 connects to 3, 4
- 3 connects to 1, 2
- 4 connects to 2
Column Multiplicity-other example

X = G(C, D)
X = C in this case

But how to calculate function H?
Decomposition of multiple-valued relation

Karnaugh Map

Compatibility Graph for columns

Kmap of block G

Kmap of block H

One level of decomposition
Discovering new concepts

- Discovering concepts useful for purchasing a car
Variable ordering

- **Uncertainty (Shannon):**
  \[ u(a) = -\sum_i p(a = a_i) \log_2 p(a = a_i) \]

- **Conditional Uncertainty (Shannon):**
  \[ u(a|b) = u(ab) - u(b) \]

Diagram:

- Variables: a, b, c, d, u(y)
- Conditional probabilities: u(y | b), u(y | bd), u(y | bda), u(y | bdac)
Vacuous variables removing

- Variables b and d reduce uncertainty of y to 0 which means they provide all the information necessary for determination of the output y.
- Variables a and c are vacuous.
Example of removing inessential variables (a) original function (b) variable a removed (c) variable b removed, variable c is no longer inessential.
Generalization of the Ashenhurst-Curtis decomposition model
Compatibility graph construction for data with noise

<table>
<thead>
<tr>
<th>ab</th>
<th>cd</th>
<th>B1_{00}</th>
<th>B1_{01}</th>
<th>B1_{11}</th>
<th>B1_{10}</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>1,3</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>1</td>
<td>-</td>
<td>0,1</td>
<td>8</td>
</tr>
<tr>
<td>01</td>
<td>1</td>
<td>2</td>
<td>5</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>10</td>
<td>0</td>
<td>3</td>
<td>6</td>
<td>-</td>
<td>4</td>
</tr>
<tr>
<td>10</td>
<td>0</td>
<td>0</td>
<td>0.25</td>
<td>0.25</td>
<td>0.25</td>
</tr>
</tbody>
</table>

Kmap

Compatibility Graph for Threshold 0.75

Compatibility Graph for Threshold 0.25
Compatibility graph for metric data

Kmap

Compatibility Graph for nominal data

Compatibility Graph for metric data

Difference of 1
MV relations can be created from contingency tables

![Contingency tables](image)

**Figure 1: Contingency tables**
Example of decomposing a Curtis non-decomposable function.