
# Introduction

By Marek Perkowski

# THE MULTIPLE-VALUED LOGIC.

- What is it?
- WHY WE BELIEVE IT HAS A ***BRIGHT FUTURE.***
- Research topics (not circuit-design oriented)
- New research areas
- The need of unification

# Is this whole a nonsense?

- When you ask an average engineer from industry, he will tell you ***"multi-valued logic is useless because nobody builds circuits with more than two values"***
- **First**, it is not true, there are such circuits built by top companies (*Intel Flash Strata*)
- **Second**, MV logic is used in some top EDA tools as mathematical technique to ***minimize binary*** logic (Synopsys, Cadence, Lattice)
- **Thirdly**, MV logic can be ***realized in software*** and as such is used in Machine Learning, Artificial Intelligence, Data Mining, and Robotics

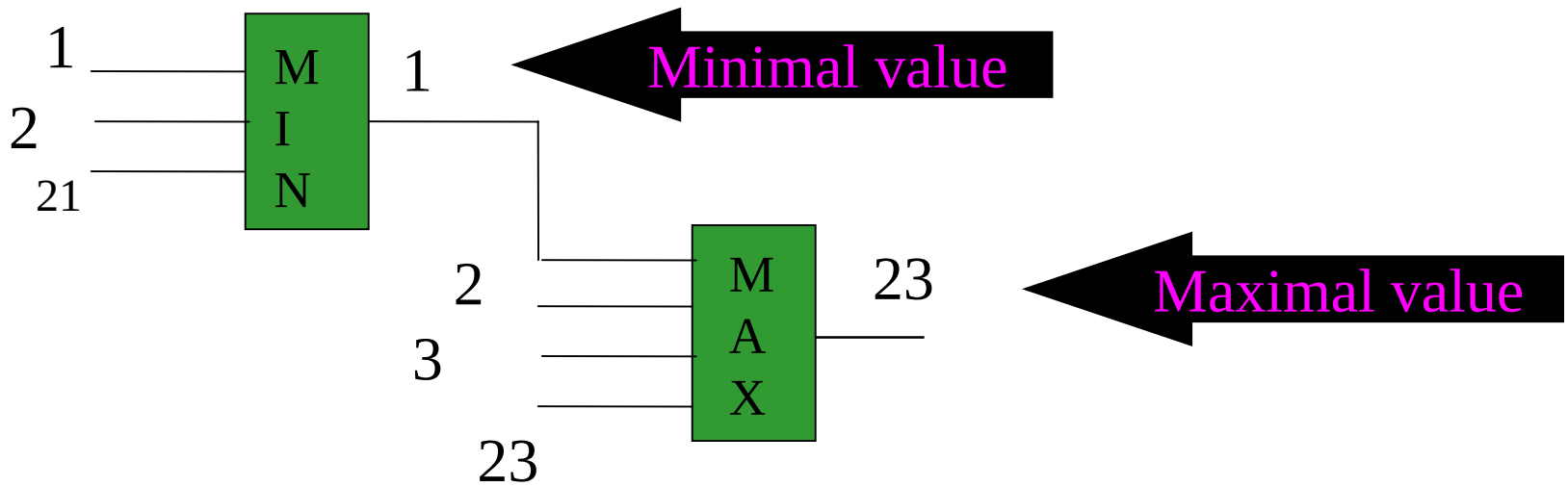
# Short Introduction: **multiple-valued logic**

**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**



# Binary logic is doomed

- It dominates hardware since 1946
- Many researchers and analysts believe that the binary logic is ***already doomed*** - because of Moore's Law
- You cannot shrink sizes of transistors ***indefinitely***
- We will be not able to use binary logic **alone** in the generation of computer products that will ***start to appear*** around 2020.

# Quantum phenomena

- They will have to be considered in one way or another
- It is not sure if standard binary logic will be still a reasonable choice in new generation computing
- Biological models

# Future "Edge" of MVL

- Chip size and performance are increasingly related to **number of wires, pins**, etc., rather than to the devices themselves.
- Connections will occupy **higher and higher percentage** of future binary chips, hampering future progress around year 2020.
- In principle, MVL can provide a means of increasing data processing capability per unit chip area.
- **MVL can create automatically efficient programs from data**

# From **two** values to **more** values

- The researchers in MV logic propose to **abandon Boolean principles entirely**
- They proceed bravely to ***another kind of logic***, such as **multi-valued**, **fuzzy**, **continuous**, **set** or **quantum.**
- It seems very probable, that this approach will be used in at least some future ***calculating*** products.

# Multi-Valued Logic **Synthesis**(cont)

- The MVL research investigates
  - ***Possible gates***,
  - Regular gate ***connection structures*** (MVL PLA),
  - **Representations** - generalizations of **cube calculus** and **binary decision diagrams** (used in binary world to represent Boolean functions),
  - Application of design/minimization **algorithms**
  - **General problem-solving approaches** known from binary logic such as:
    - generalizations of **satisfiability**, **graph algorithms** or **spectral** methods,
    - application of **simulated annealing**, **genetic algorithms** and **neural networks** in the synthesis of multiple valued functions.

# **Binary versus MV** Logic Synthesis Research

- There is ***less research interest in MVL*** because such circuits are not yet widely used in industrial products
- MV logic synthesis is not much used in industry.
- Researchers in hundreds
- Only big companies, military, government. IBM
- The research is more theoretical and fundamental.
- **You can become a pioneer** – it is like Quine and McCluskey algorithm in 1950
- **Breakthroughs are still possible** and there are many open research problems
- ***Similarity*** to binary logic is helpful.

# However……..,

- if some day **MV gates** were introduced to practical applications, the markets for them will be so large that it will stimulate **exponential growth** of research and development in MV logic.
- and then, the **accumulated 50 years of research** in MV logic will prove to be very practical.

# Applications

- Image Processing
- New transforms for encoding and compression
- Encoding and State Assignment
- Representation of discrete information
- New types of decision diagrams
- Generalized algebra
- Automatic Theorem Proving

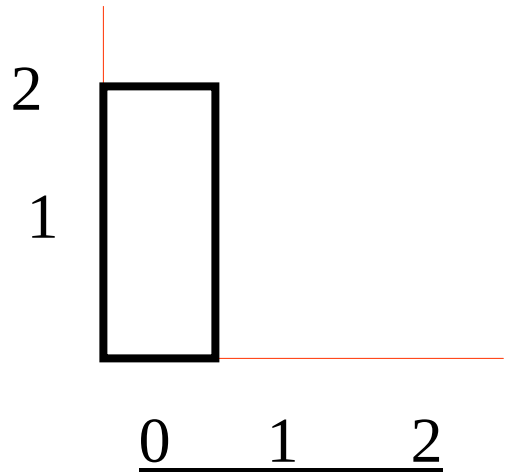
# The Shortest History of Multi-Valued Logic

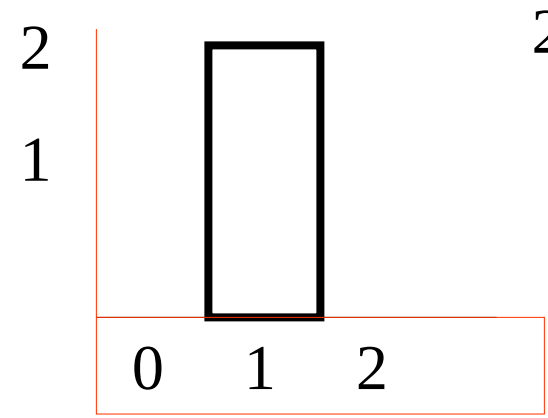
# Jan Lukasiewicz (1878-1956)

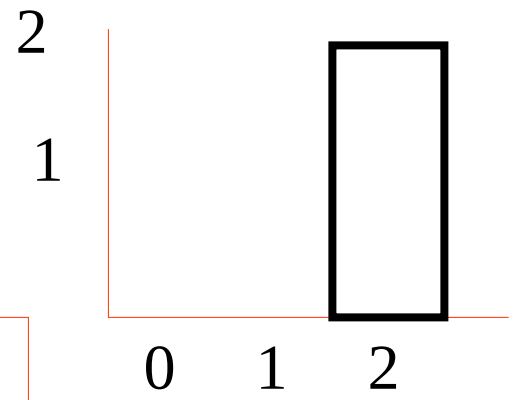
- Polish minister of Education 1919
- Developed first **ternary predicate calculus** in 1920
- Many fundamental works on ***multiple-valued*** logic
- Followed by **Emil Post,**
- American logician born in Bialystok, Poland

# MV functions of single variable

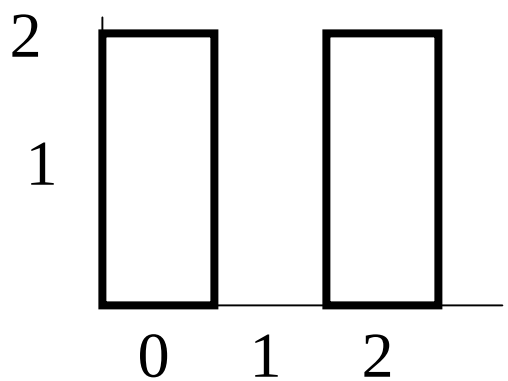
- Post Literals:

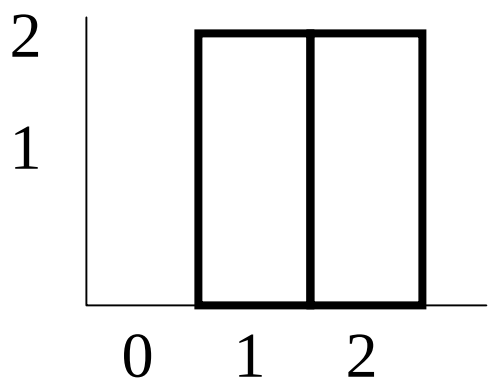


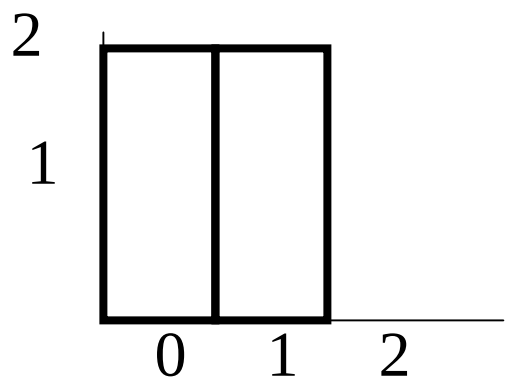




- Generalized Post Literals:

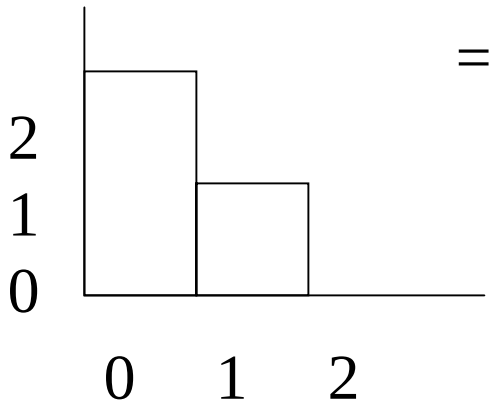


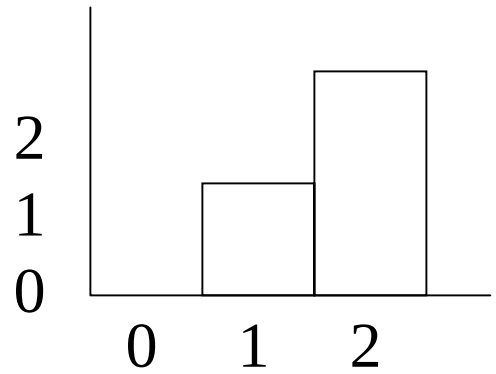



# MV functions of single variable (cont)

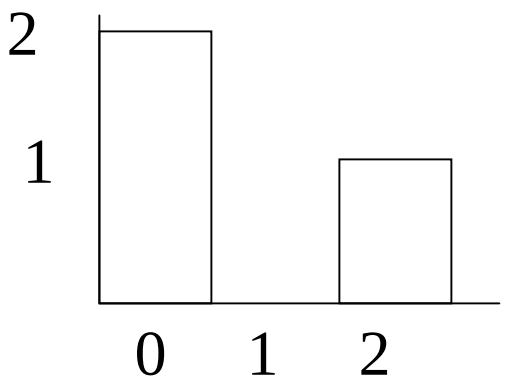
- Universal Literals:

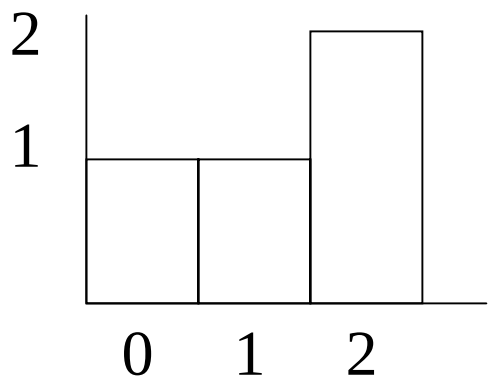



= negation




= wire



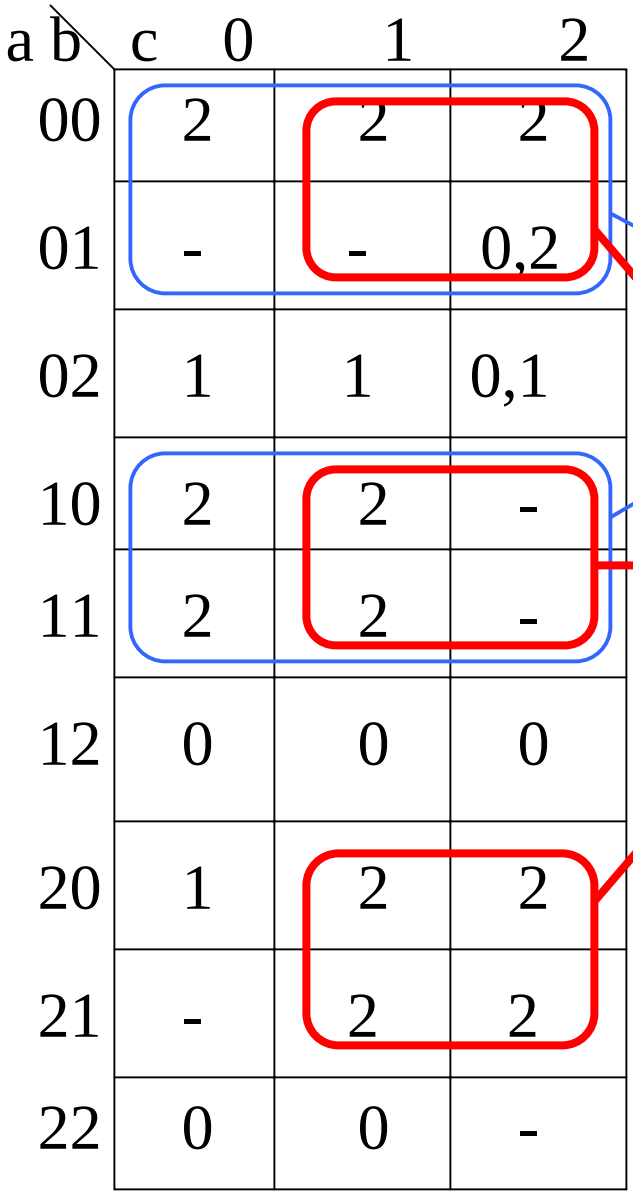
# Multiple-Valued Logic

- Let us start with an example that will help to understand,

Suppose that we have the following table, and we need to build a circuit with MV-Gates, (**MAX** & **MIN**).

As we can see, this is ***ternary logic.***

| a b \ c | 0 | 1 | 2 |
|---|---|---|---|
| 00 | 2 | 2 | 2 |
| 01 | - | - | 0,2 |
| 02 | 1,2 | 1,2 | 0,1 |
| 10 | 2 | 2 | - |
| 11 | 2 | 2 | - |
| 12 | 0 | 0 | 0 |
| 20 | 1 | 2 | 2 |
| 21 | - | 2 | 2 |
| 22 | 0 | 0 | - |

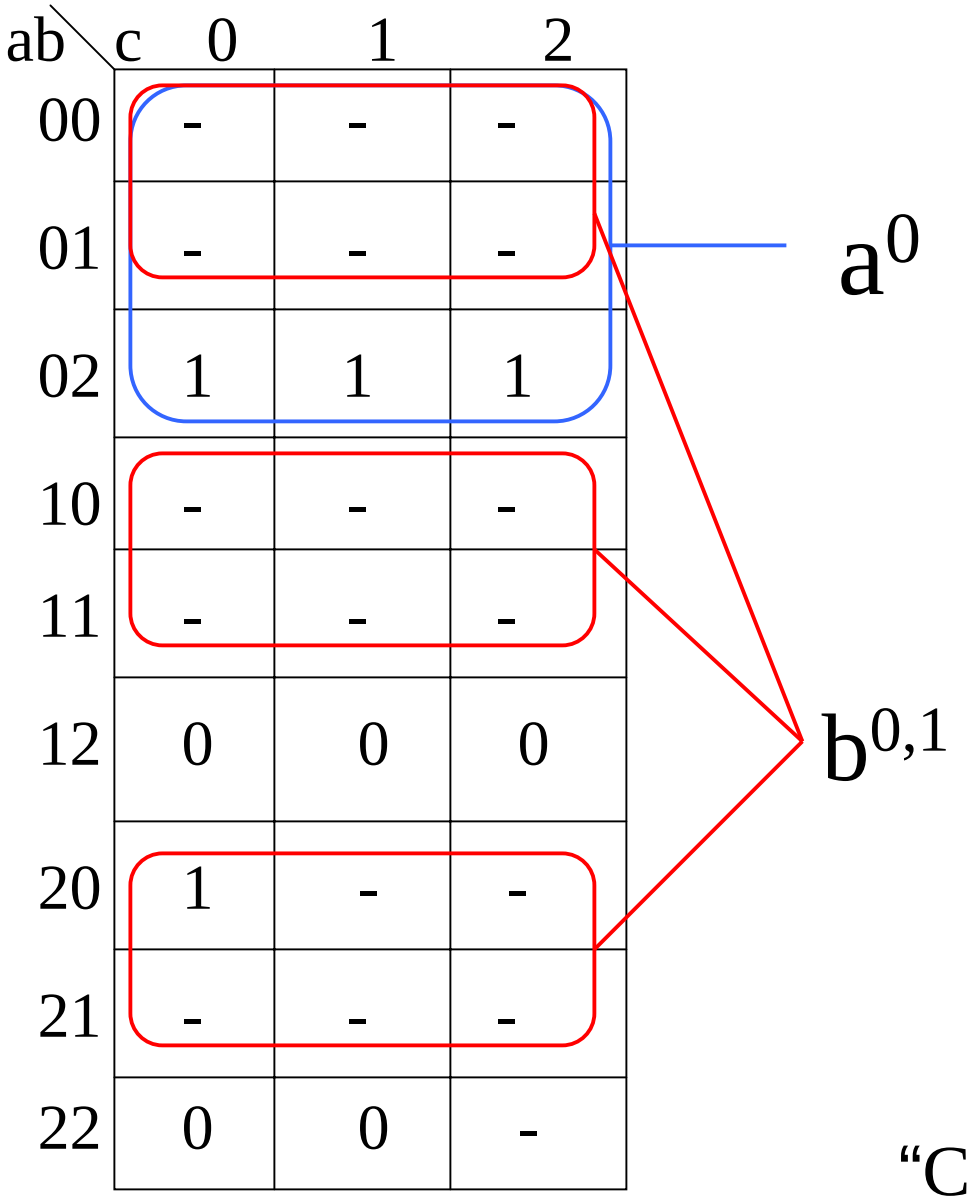



$a^{0,1}\ b^{0,1}$

$b^{0,1}\ c^{1,2}$

$a^{0,1}\ b^{0,1} + b^{0,1}\ c^{1,2}$

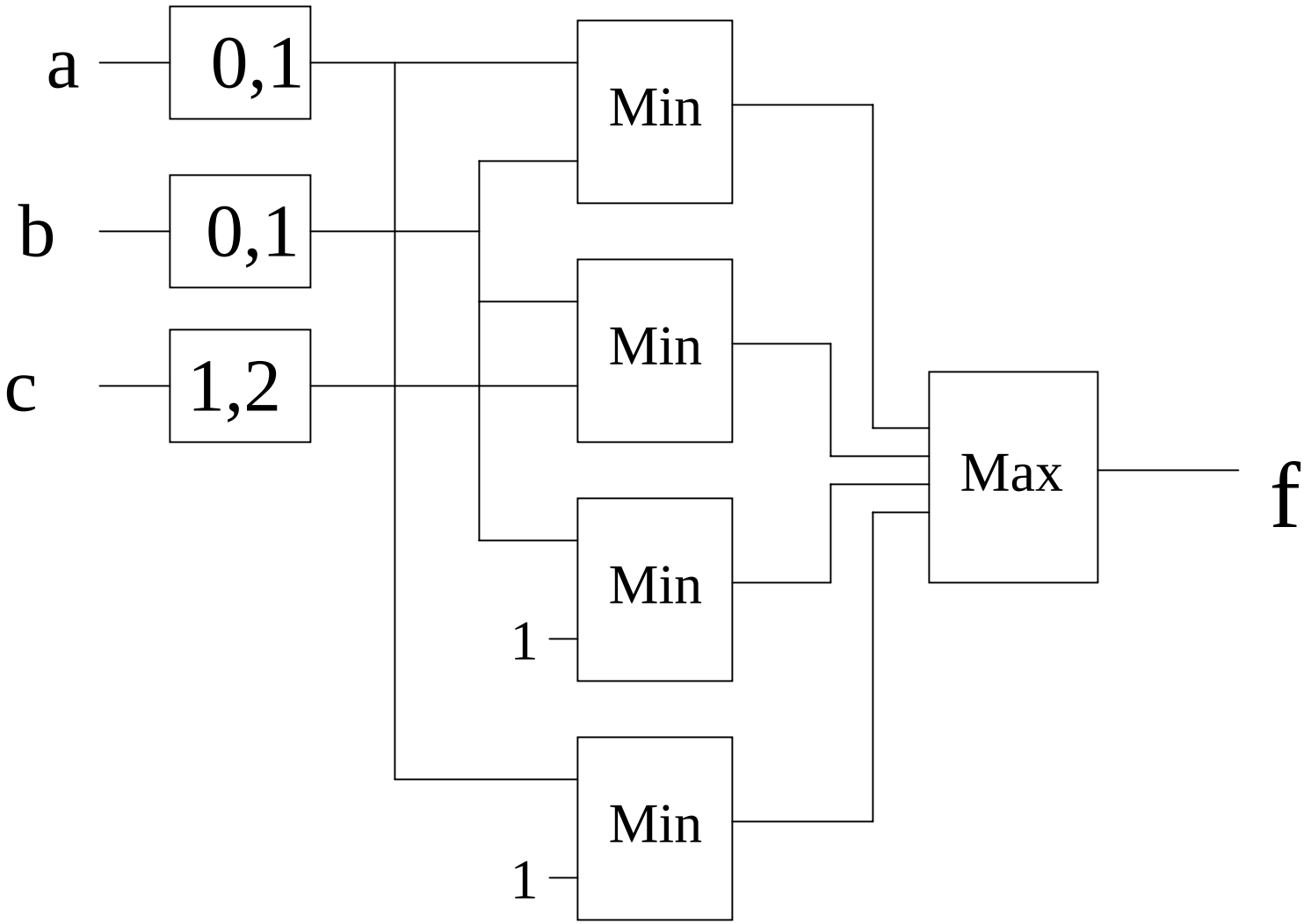
"Covering 2's in the map"




$1.a^{0,1} + 1.b^{0,1}$

"Covering 1's in the map"

**SOP** = $a^{0,1}\ b^{0,1} + b^{0,1}\ c^{1,2} + 1.a^{0,1} + 1.b^{0,1}$



# Example of Application of logic with MV inputs and binary outputs to minimize area of custom PLA with decoders.

- Pair of binary variables corresponds to quaternary variable X
- a'+b'=$X^{012}$
- a'+b=$X^{013}$

a + b' = $X^{023}$

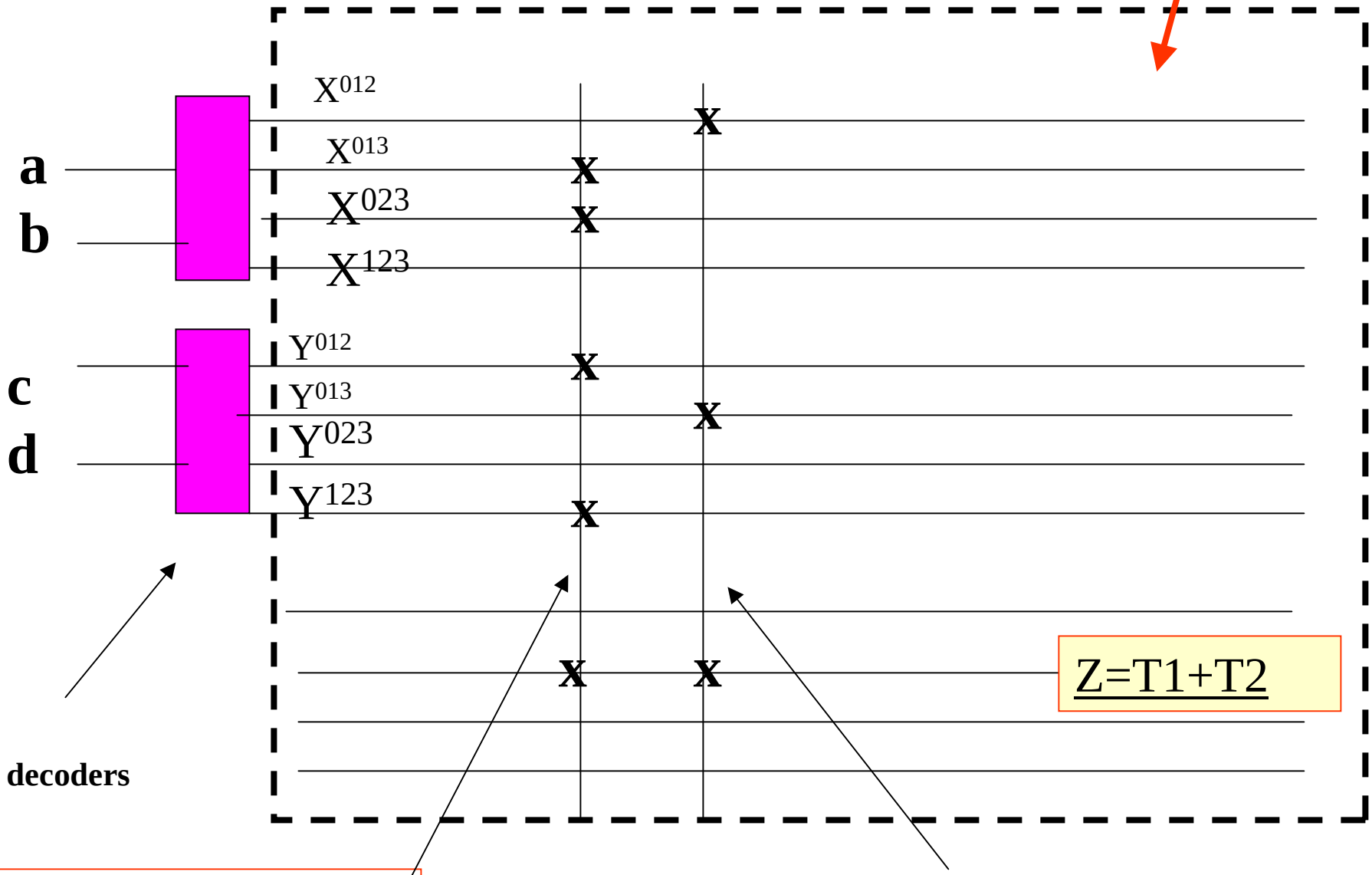
a + b = $X^{123}$

| a b | 0 | 1 |
|---|---|---|
| 0 | 0 | 1 |
| 1 | 2 | 3 |

$$(a + b') * (a'+b) = X^{023} * X^{013}$$

Thus equivalence can be realized by single column in PLA with input decoders instead of two columns in standard PLA

# PLA with decoders



# PLA with decoders

- Such PLAs can have much smaller area thans to more powerful functions realized in columns
- The area of decoders on inputs is negligible for large PLAs
- Multi-valued logic is used to minimize mv-input, binary-output functions with capital letter inputs X,Y,Z,etc.

# Why we need Multiple-Valued logic?

- In new technologies the most delay and power occurs in the **connections** between gates.
- When designing a function using Multiple-Valued Logic, we need **less gates**, which implies **less number of connections**, then **less delay**.

- Same is true in case of software (program) realization of logic

- Also, most the **natural variables** like color, is multi-valued, so it is better to use multi-valued logic to realize it instead of coding it into binary.

- In multi-valued logic, the binary AND gate is replaced by **MIN** gate, and OR by **MAX**
- But, **AND** can be also replaced by **arithmetic multiplication**, or **modulo multiplication**, or **Galois multiplication**
- OR can be also replaced by **modulo addition**, or by **Galois addition**, or by **Boolean Ring addition**, or by…..

- Finally, the number of values in **infinite**
- **This way we get Lukasiewicz logic, fuzzy logic, possibilistic logic, and so on…**
- ***Continuous logics***
- There are *very many ways* of creating gates in MVL
- They have different mathematical properties
- They have very different costs in various technologies
- The values and operators can describe time, moral values, energy, interestingness, utility, emotions…….

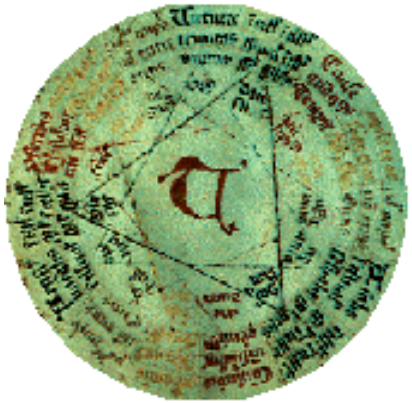
# Mathematical, logical, system science, or psychological/ methodological/ philosophical foundations

- ***Functional completeness*** theory studies the construction of logical functions from a set of primitives and enumeration of bases.
- The problems which are investigated include:
  - **classification** of functions
  - **enumeration of bases** of a closed subset of the set of all k-valued logical functions
  - study of **particular kinds of functions** (monotone, symmetric, predicate, etc.) in multi-valued logics.

# Are we sure that Lukasiewicz was the first human who had these ideas?

- Some Chinese philosophers claim that the **Buddhist logic,** invented Before Christ Era, was ***very similar*** to fuzzy logic
- Raymon Lullus (Ramon Llull) invented many concepts that were *hundreds years ahead of his time*

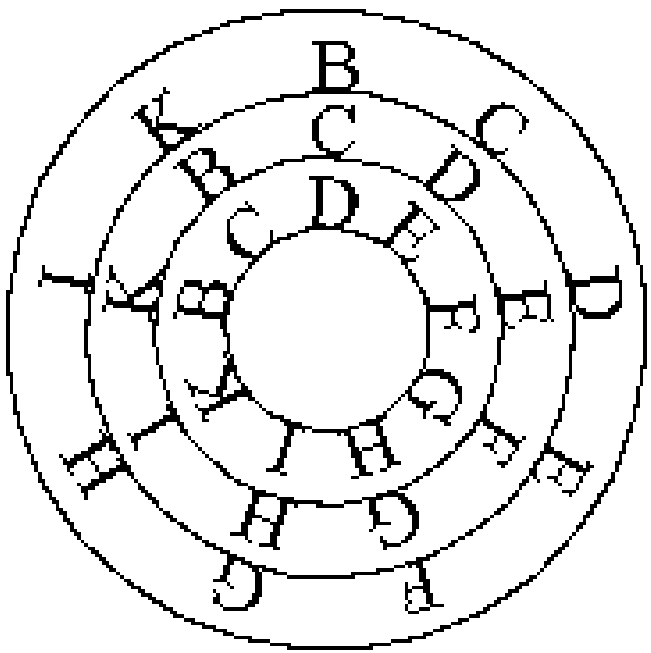
# **Raymon Lullus**, 1235-1316 (probably)



Monestir de **Miramar**

Fundat per Ramón Llull en 1276

*Trinity College founded by Raymond Lully in 1276.*
*The first Cultural Lantern of the Balearic Islands*
*(Spain) in the INTERNET*




- Creator of **"Cartesian Product"**
- Creator of **binary counting system**
- Creator of **multi-valued logic and counting system**
- Creator of the concept of **logic computer**

# Lotfi Zadeh (1921- )

- Father of Fuzzy Logic
- Professor of University of California in Berkeley
- First paper on fuzzy logic published in 1956

# Continuous Logic

- From two values to many values to infinite number of values
  - **Fuzzy logic** (Lotfi Zadeh),
  - **Lukasiewicz logic**,
  - **Probabilistic** logic,
  - **Possibilistic** logic,
  - **Arithmetic** logic,
  - **Complex** and **Quaternion** logic,
  - other **continuous** logics
- Find now applications in **software** and in hardware
- Are studied now outside the area of MV logic, but historically belong to it.

# Example:Arithmetic Logic

$A \vee B = A + B - A*B$

Logic sum

Arithmetic sum

***We express logic operators by arithmetic operators***

A=0.5 B=0.5 then result = 0.5+0.5-0.5*0.5=1-0.25=0.75

1-(1-A)(1-B)=1-(1-A-B+AB)=A+B-AB

This corresponds to probabilistic reasoning

# Example:Arithmetic Logic

$A \oplus B = A+B - 2*A*B$

Logic exor

Arithmetic sum

***We express logic operators by arithmetic operators***

*Various operators can be defined to model certain reasonings and because their hardware realization is simple*

A=0.5 B=0.5 then result = 0.5+0.5-2*0.5*0.5=1-0.5=0.5

$A \oplus B = A'B + AB' = (1-A)B+A(1-B)-(1-A)B(1-B)A = B-AB+A-AB-AB(1-A)(1-B) = A+B-2AB-AB(1-A-B+AB)$

In this definition, the same results for natural numbers 0 and 1, but slightly different for A=B=0.5

**Check it, define other operators of similar properties.**

# Functional Representations in Logic Synthesis

- New representations aim at more **compact representation** of discrete data that allows:
  - less memory space,
  - smaller processing time.
- Data can be functions, relations, sets of functions and sets of relations.
- Result of logic synthesis is a computer program for a robot
- Logic Synthesis = Automatic Program Synthesis
- Good synthesis = better program (smaller, faster, more reliable - noise, generalization)

- **Examples of representations** :

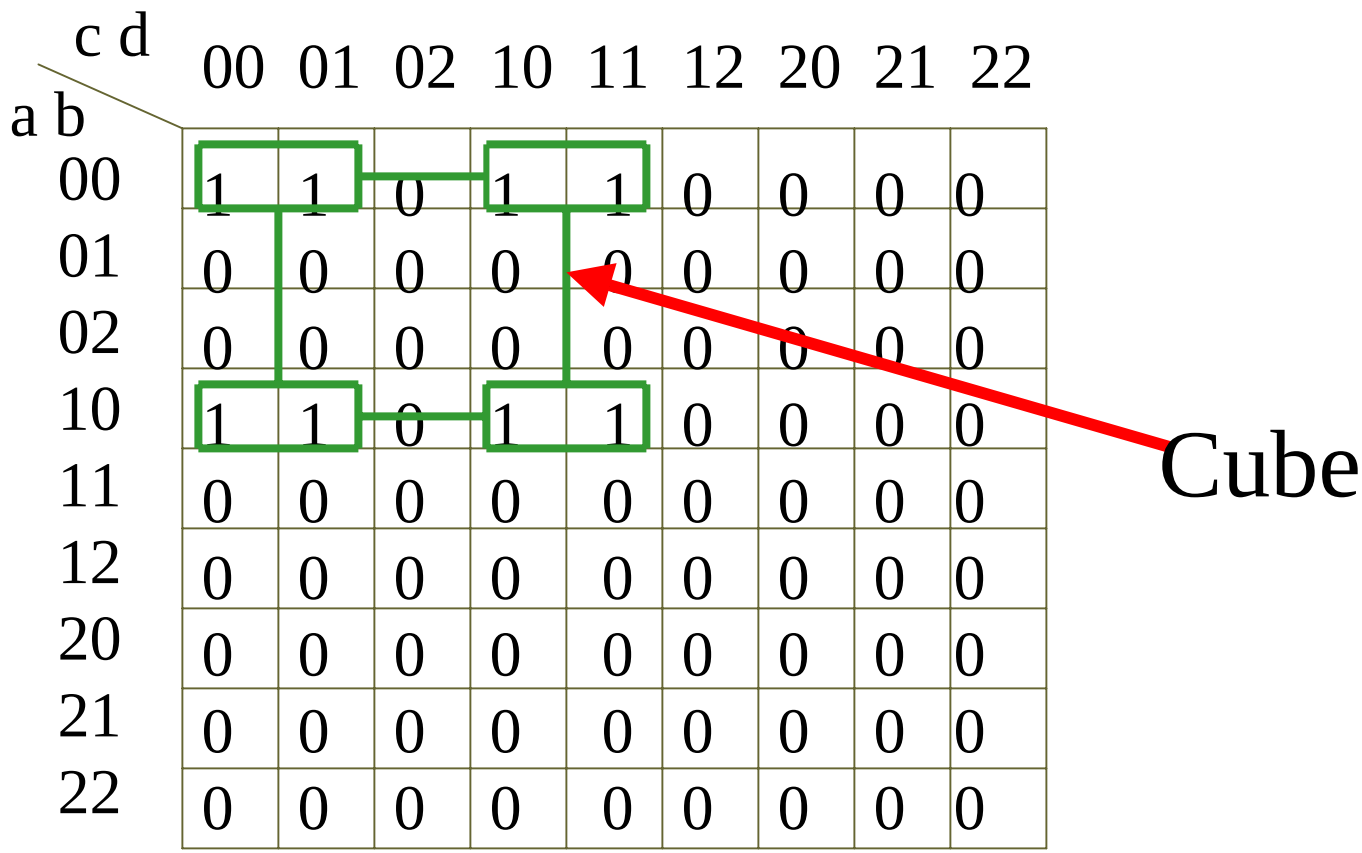
1. **Cube Representation** and the corresponding Cube operations (Cube Calculus).

2. **Decision Diagram** (DD) Representation and the corresponding DD operations.

3. **Labeled Rough Partitions** encoded with BDDs.

- **Cube Representations** :

1-a. Graphical Cube Representations of Multi-Valued Input Binary Output.



- 1-b. Expression (Flattened Form)

  Representation of Cubes of Multi-Valued Input Binary Output .

For the previous example :-

$$F = 1.a^0b^0c^0d^0 + 1.a^0b^0c^0d^1 + 1.a^0b^0c^1d^0 + 1.a^0b^0c^1d^1 + 1.a^1b^0c^0d^0 + 1.a^1b^0c^0d^1 + 1.a^1b^0c^1d^0 + 1.a^1b^0c^1d^1$$

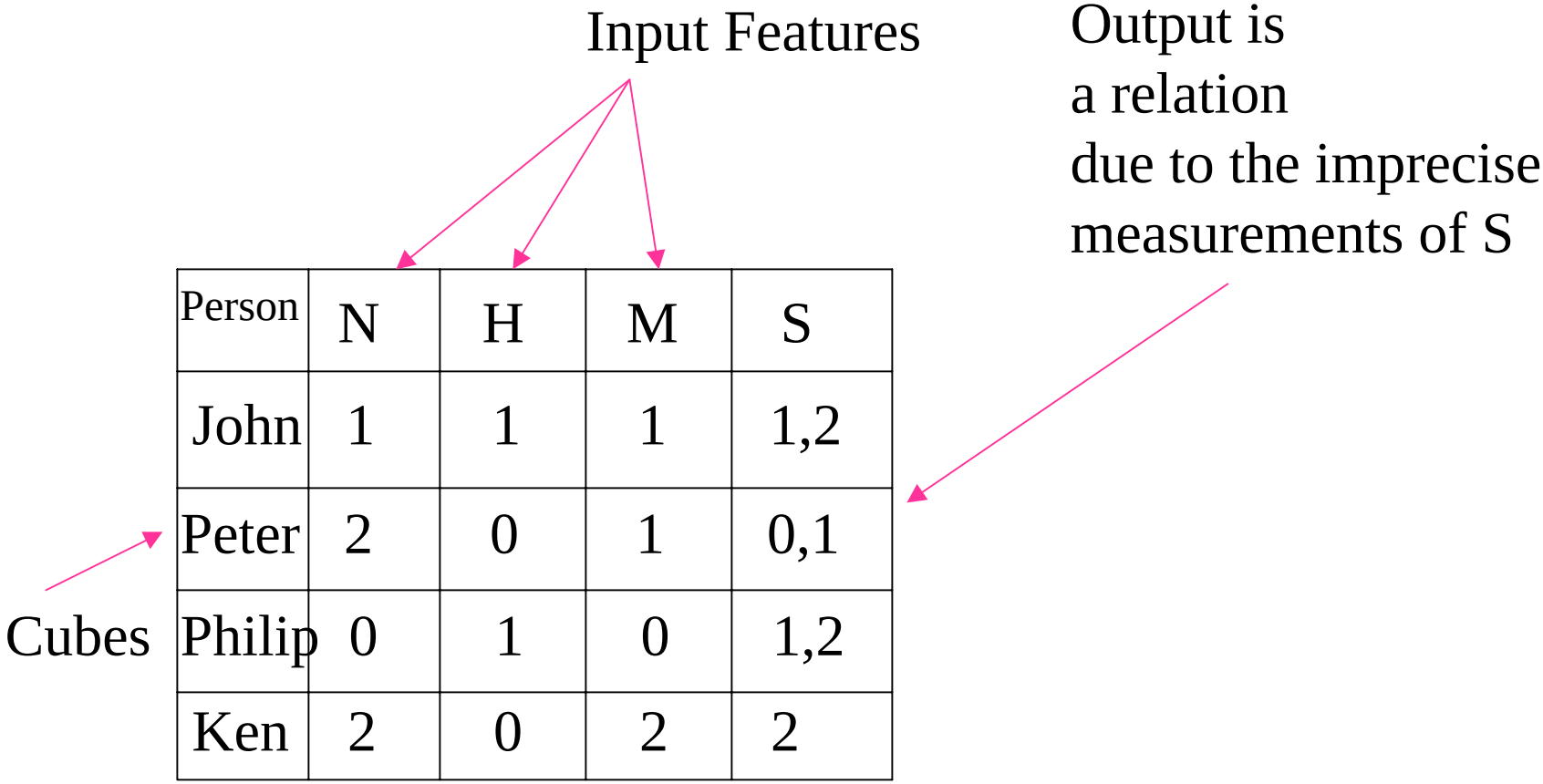
$$= 1 \,.\, a^{0,1}\, b^0 c^{0,1}\, d^{0,1}$$

# *Tabular Representation*

| Cube# | a | b | f | g |
|---|---|---|---|---|
| 0 | 0,2 | 1 | _ | 2 |
| 1 | 0,1 | 0 | 0,2 | 0 |
| 2 | 2 | 0 | 1,2 | 0 |
| 3 | 1 | 1 | 1,2 | 2 |

Functions and Relations are just **mappings**

- To recognize faces we obtain the following tabular representation :-




| Person | N | H | M | S |
|---|---|---|---|---|
| John | 1 | 1 | 1 | 1,2 |
| Peter | 2 | 0 | 1 | 0,1 |
| Philip | 0 | 1 | 0 | 1,2 |
| Ken | 2 | 0 | 2 | 2 |

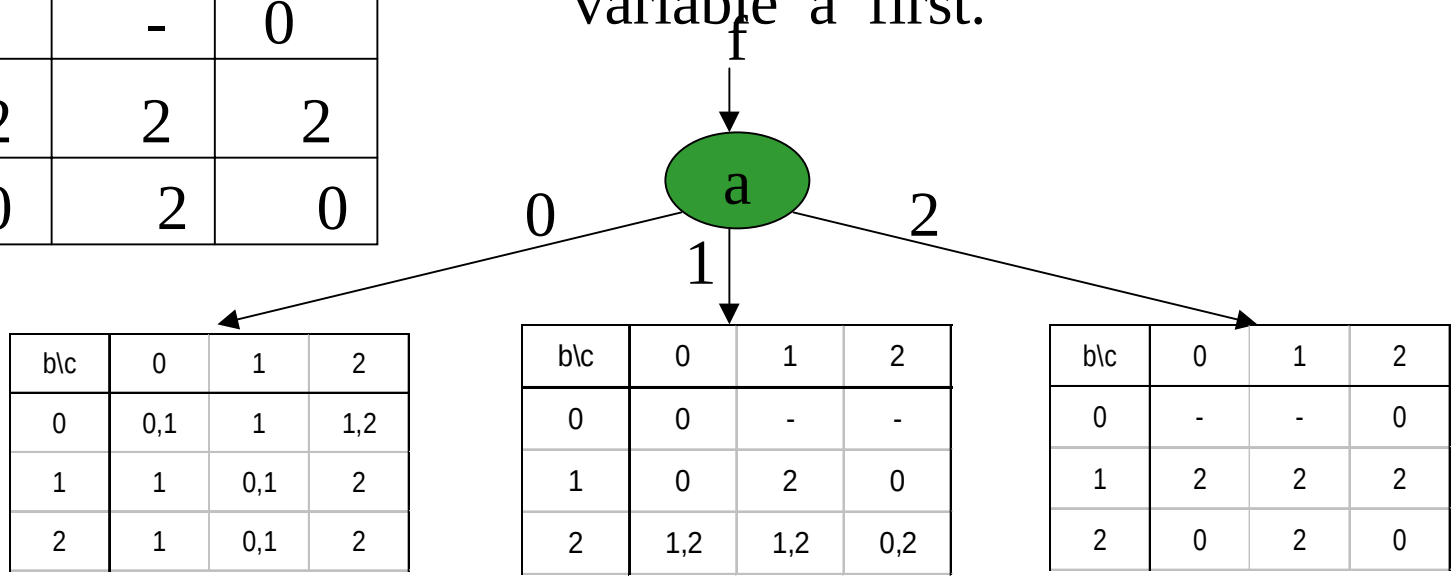
# Multi-valued Logic

| a b \ c | 0 | 1 | 2 |
|---|---|---|---|
| 00 | 0,1 | 1 | 1,2 |
| 01 | 1 | 0,1 | 2 |
| 02 | 1 | 0,1 | 2 |
| 10 | 0 | - | - |
| 11 | 0 | 2 | 0 |
| 12 | 1,2 | 1,2 | 0,2 |
| 20 | - | - | 0 |
| 21 | 2 | 2 | 2 |
| 22 | 0 | 2 | 0 |

To get it's Decision Diagrams, we follow these steps.

***Step 1: Expand the function with respect to variable "a","b" and "c".***

Expand the function with respect to variable 'a' first.




| b\c | 0 | 1 | 2 |
|---|---|---|---|
| 0 | 0,1 | 1 | 1,2 |
| 1 | 1 | 0,1 | 2 |
| 2 | 1 | 0,1 | 2 |

| b\c | 0 | 1 | 2 |
|---|---|---|---|
| 0 | 0 | - | - |
| 1 | 0 | 2 | 0 |
| 2 | 1,2 | 1,2 | 0,2 |

| b\c | 0 | 1 | 2 |
|---|---|---|---|
| 0 | - | - | 0 |
| 1 | 2 | 2 | 2 |
| 2 | 0 | 2 | 0 |

# Multi-valued Logic

When a=0, expand the function with respect to ‘b’ and ‘c’.

| b\c | 0 | 1 | 2 |
|---|---|---|---|
| 0 | 0,1 | 1 | 1,2 |
| 1 | 1 | 0,1 | 2 |
| 2 | 1 | 0,1 | 2 |

b= 0 1 2

| C | 0 | 1 | 2 |
|---|---|---|---|
| | 0,1 | 1 | 1,2 |

1

| C | 0 | 1 | 2 |
|---|---|---|---|
| | 1 | 0,1 | 2 |

0 1 2

1 | 0,1 | 2

| C | 0 | 1 | 2 |
|---|---|---|---|
| | 1 | 0,1 | 2 |

0 1 2

1 | 0,1 | 2

Because this group can be 1, the 0 can be ignored

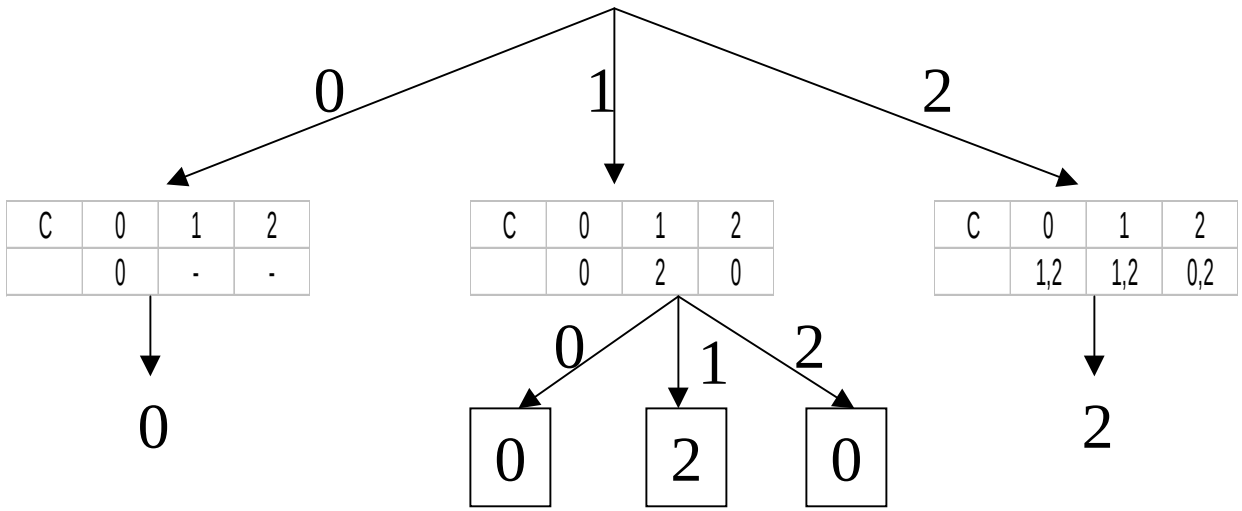
f equals ‘1’ here no matter if c is 0, 1 or 2, so it terminals at 1.

# Multi-valued Logic

a=1

| b\c | 0 | 1 | 2 |
|---|---|---|---|
| 0 | 0 | - | - |
| 1 | 0 | 2 | 0 |
| 2 | 1,2 | 1,2 | 0,2 |

b=

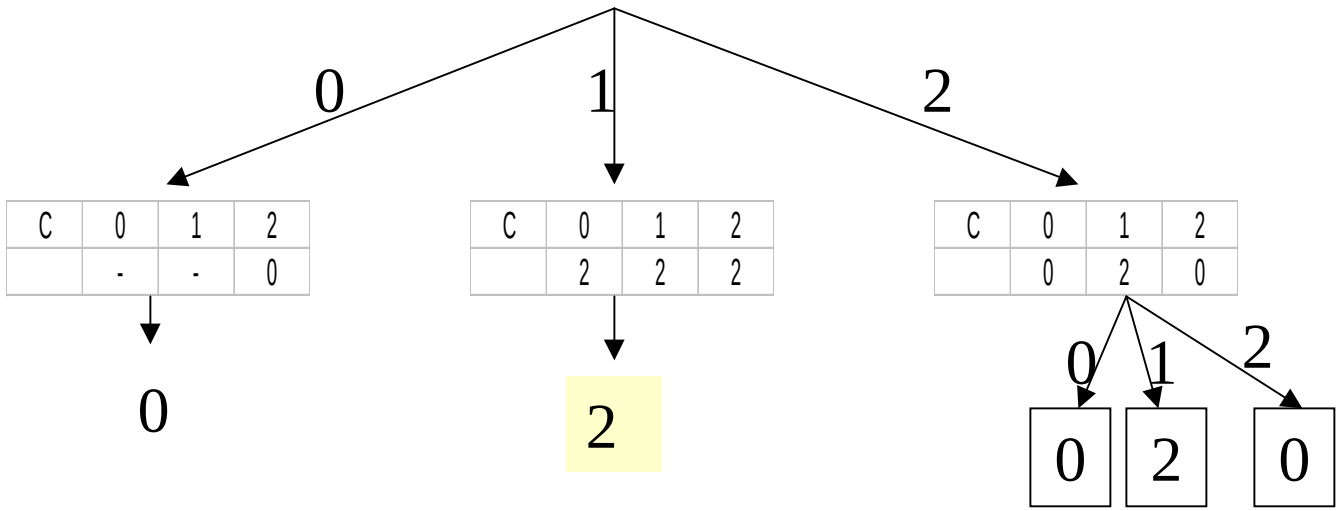


# Multi-valued Logic

a=2

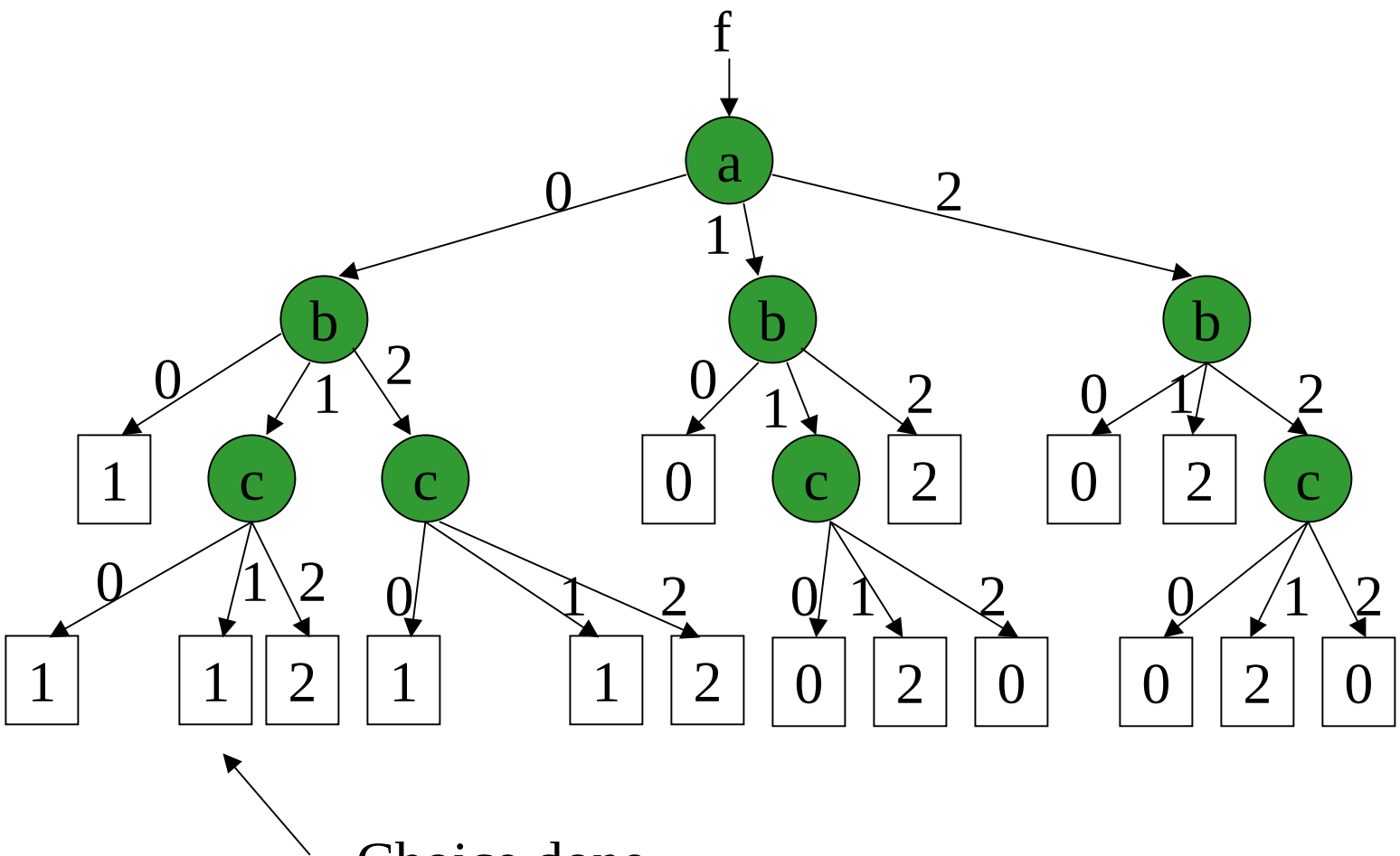
| b\c | 0 | 1 | 2 |
|---|---|---|---|
| 0 | - | - | 0 |
| 1 | 2 | 2 | 2 |
| 2 | 0 | 2 | 0 |

b=



# Multi-valued Logic

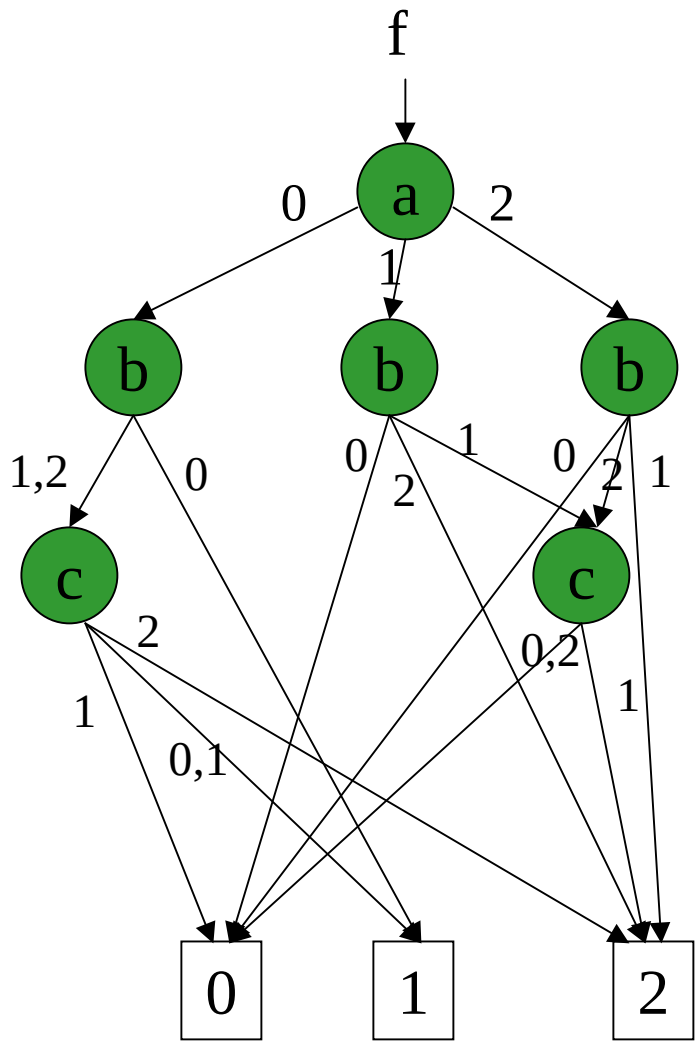
***Step 2: Draw the Decision Tree***




Choice done for simplification

# Multi-valued Logic

***Step 3: Combine the same terminals to get Decision Diagram***



# First Extension from Binary

## Binary Logic ---------- MV Logic

And Gate ---------- MIN gate

Or Gate ---------- MAX gate

Inverter ---------- Literal

Post, generalized Post or Universal

This is standard, many published results, both two-level and multi-level, **complete** system

# Second Extension from Binary

## Binary Logic ---------- MV Logic

And Gate <----------→ Galois Multiplication gate

EXOR Gate <----------→ Galois Addition gate

Inverter <----------→ Power of variable

This system was introduced by Pradhan and Hurst, few papers have been published, no software, most is two-level logic

# Another very recent extension

## Binary Logic ---------- MV Logic

And Gate ←----------→ MIN gate

Exor Gate ←----------→ MODSUM gate

Inverter ←----------→ Literals

Perhaps universal literals will increase the power, not investigated yet

This system was introduced by Muzio and Dueck and independently by Elena Dubrova in her Ph.D. Two papers have been published. Recent interest.