Rough Set Theory (RST)

RST

- The notion of rough sets was introduced by Z Pawlak in his seminal paper of 1982 (Pawlak 1982).
- It is a formal theory derived from fundamental research on logical properties of information systems.
- Rough set theory has been a methodology of database mining or knowledge discovery in relational databases.
- In its abstract form, it is a new area of uncertainty mathematics closely related to fuzzy theory.

RST

- Rough sets and fuzzy sets are complementary generalizations of classical sets.
- The approximation spaces of rough set theory are sets with multiple memberships, while fuzzy sets are concerned with partial memberships.
- The rapid development of these two approaches provide a basis for "soft computing," initiated by Lotfi A. Zadeh.
- Soft Computing includes along with rough sets, at least fuzzy logic, neural networks, probabilistic reasoning, belief networks, machine learning, evolutionary computing, and chaos theory.

RST

- In Computer Science, a **rough set**, first described by a Polish computer scientist Zdzisław I. Pawlak, is a formal approximation of a crisp set (i.e., conventional set) in terms of a pair of sets which give the *lower* and the *upper* approximation of the original set. In the standard version of rough set theory (Pawlak 1991), the lower- and upper-approximation sets are crisp sets, but in other variations, the approximating sets may be fuzzy sets.

RST

- Basic problems in data analysis solved by Rough Set:
  - Characterization of set of objects in terms of attribute values.
  - Finding dependency between the attributes.
  - Reduction of superflous attributes.
  - Finding the most significant attributes.
  - Decision rule generation.

INFORMATION SYSTEM FRAMEWORK

- In Rough Set data model information is stored in a table.
- Each row (tuples) represents a fact or an object.
- Often the facts are not consistent to each other.
- In Rough Set terminology a data table is called an *Information System*.

An Information Table

	Attributes			
Case	Temperature	Headache	Nausea	Cough
1	high	yes	no	yes
2	very_high	yes	yes	no
3	high	no	no	no
4	high	yes	yes	yes
5	normal	yes	no	no
6	normal	no	yes	yes

Information system

- Data tables storing real-world objects

- Mathematically represented as $\mathbb{I} = (\mathbb{U}, \mathbb{A})$ where U is a non-empty set of finite objects, A is a non-empty finite set of attributes such that $\forall a \in \mathbb{A},\ a : \mathbb{U} \rightarrow V_a$ i.e. V_a is the set of values attribute *a* may take

An Information Table

Case	Attributes			
	Temperature	Headache	Nausea	Cough
1	high	yes	no	yes
2	very_high	yes	yes	no
3	high	no	no	no
4	high	yes	yes	yes
5	normal	yes	no	no
6	normal	no	yes	yes

Indiscernibility

- Tables may contain many objects having the same features
- A way of reducing table size is to store only one representative object for every set of objects with same features.
- These objects are called indiscernible objects or tuples

Indiscernibility

With any $P \subseteq \mathbb{A}$ there is an associated equivalence relation IND(*P*):

$$\mathrm{IND}(P) = \left\{(x,y) \in \mathbb{U}^2 \mid \forall a \in P, a(x) = a(y)\right\}$$

Sample Information System

Object	P_1	P_2	P_3	P_4	P_5
O_1	1	2	0	1	1
O_2	1	2	0	1	1
O_3	2	0	0	1	0
O_4	0	0	1	2	1
O_5	2	1	0	2	1
O_6	0	0	1	2	2
O_7	2	0	0	1	0
O_8	0	1	2	2	1
O_9	2	1	0	2	2
O_{10}	2	0	0	1	0

example

Sample Information System

Object	P_1	P_2	P_3	P_4	P_5
O_1	1	2	0	1	1
O_2	1	2	0	1	1
O_3	2	0	0	1	0
O_4	0	0	1	2	1
O_5	2	1	0	2	1
O_6	0	0	1	2	2
O_7	2	0	0	1	0
O_8	0	1	2	2	1
O_9	2	1	0	2	2
O_{10}	2	0	0	1	0

$P = \{P_1, P_2, P_3, P_4, P_5\}$

$$[x]_P = \begin{cases} \{O_1, O_2\} \\ \{O_3, O_7, O_{10}\} \\ \{O_4\} \\ \{O_5\} \\ \{O_6\} \\ \{O_8\} \\ \{O_9\} \end{cases}$$

$P = \{P_1\}$

$$[x]_P = \begin{cases} \{O_1, O_2\} \\ \{O_3, O_5, O_7, O_9, O_{10}\} \\ \{O_4, O_6, O_8\} \end{cases}$$

RST

- It is a formal approximation of a crisp set defined by its two approximations – upper approximation and lower approximation
- Upper approximation ($\overline{P}X$) is the set of objects which possibly belong to the target set
- Lower approximation ($\underline{P}X$) is the set of objects that positively belong to the target set
- The tuple $< \underline{P}X , \overline{P}X >$ represents the rough set

RST(contd.)

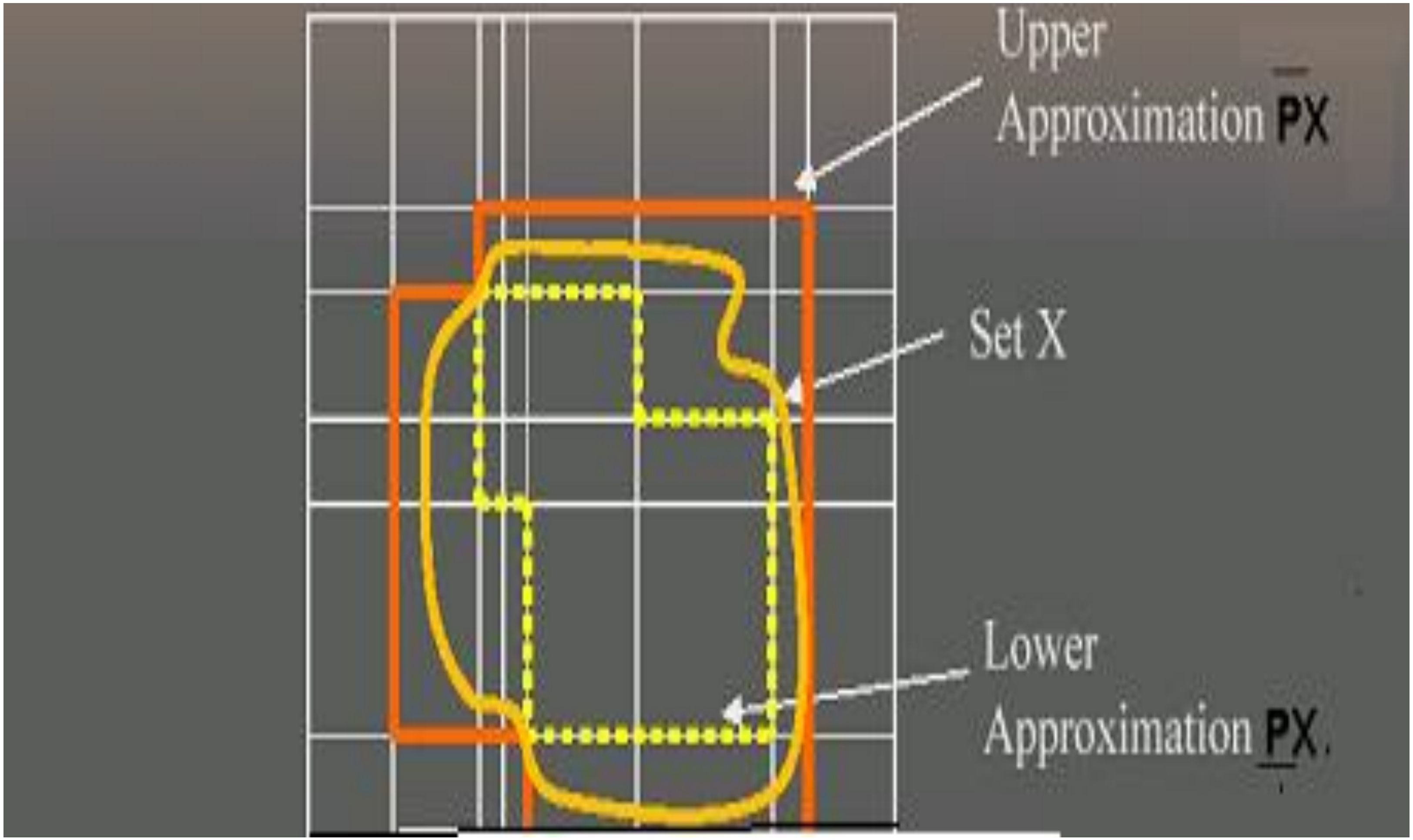
- Mathematically,
  lower approximation is denoted as
  $\underline{P}X = \{x \mid [x]_P \subseteq X\}$

- Mathematically,
  upper approximation is denoted as
  $\overline{P}X = \{x \mid [x]_P \cap X \neq \Phi\}$

- Partition of $\mathbb{U}$ generated by *IND*(*P*) is denoted by $\mathbb{U}/IND(P)$ or $\mathbb{U}/P$

RST(contd.)

- $\cup \underline{P}X$ represents the positive region which contains the objects definitely belonging to the target set X
- $\mathbb{U} - \cup \overline{P}X$ represents the negative region which contains the objects that can be definitely ruled out as a member of the target set X
- $\cup \overline{P}X - \cup \underline{P}X$ represents the boundary region which contains the objects that may or may not belong to the target set X



Upper
Approximation $\overline{PX}$
Set X
Lower
Approximation $\underline{PX}$.

example

Sample Information System

Object	P_1	P_2	P_3	P_4	P_5
O_1	1	2	0	1	1
O_2	1	2	0	1	1
O_3	2	0	0	1	0
O_4	0	0	1	2	1
O_5	2	1	0	2	1
O_6	0	0	1	2	2
O_7	2	0	0	1	0
O_8	0	1	2	2	1
O_9	2	1	0	2	2
O_{10}	2	0	0	1	0

$X = (\{O_1,O_2,O_3,O_5,O_8\}, \{O_4,O_6,O_7,O_9,O_{10}\})$

$P = \{P_1,P_2,P_3,P_4,P_5\}$

$IND(P) = (\{O_1,O_2\}, \{O_3,O_7,O_{10}\}, \{O_4\}, \{O_5\}, \{O_6\}, \{O_8\}, \{O_9\})$

$$\left\{\begin{array}{l} \{O_1, O_2\} \\ \{O_3, O_7, O_{10}\} \\ \{O_4\} \\ \{O_5\} \\ \{O_6\} \\ \{O_8\} \\ \{O_9\} \end{array}\right.$$

$\underline{P}X = \{x \mid [x]_P \subseteq X\} = \{O_1,O_2,O_5,O_8\}, \{O_4,O_6,O_9\}$

$\overline{P}X = \{x \mid [x]_P \cap X \neq \Phi\} = \{O_1,O_2,O_3,O_7,O_{10}, O_5,O_8\}, \{O_3,O_4,O_6,O_7,O_9,O_{10}\}$

Positive region $= \cup\, \underline{P}X = \{O_1,O_2,O_5,O_8,O_4,O_6,O_9\}$

Negative region $= \mathbb{U} - \cup\, \overline{P}X = \{\Phi\}$

Boundary region $= \cup\, \overline{P}X - \cup\, \underline{P}X = \{O_3,O_7,O_{10}\}$

RST(contd.)

- Accuracy of the rough set of set X, which measures how closely the rough set approximates target set X, is given as

$$0 \le \alpha_P(X) = \frac{|\underline{P}X|}{|\overline{P}X|} \le 1$$

- $\alpha_P(X) = 1$, the upper and lower approximations are equal and X becomes a crisp set.
- Whenever the lower approximation is empty, the accuracy is zero (regardless of the size of the upper approximation).

Attribute dependency

- One of the most important aspects of database analysis or data acquisition is the discovery of attribute dependencies
- It describes which variables are strongly related to which other variables.
- Generally, it is these strong relationships that will warrant further investigation, and that will ultimately be of use in predictive modeling.

Attribute dependency

- In rough set theory, the notion of dependency is defined very simply.
- Let us take two (disjoint) sets of attributes, set P and set Q, and inquire what degree of dependency obtains between them.
- Each attribute set induces an (indiscernibility) equivalence class structure, the equivalence classes induced by P given by $[x]_P$, and the equivalence classes induced by Q given by $[x]_Q$.

Attribute dependency

- Let , where Q_i is a given equivalence class from the equivalence-class structure induced by attribute set Q.
- The *dependency* of attribute set Q on attribute set P, $\gamma_P(Q)$, is given by

$$\gamma_P(Q) = \frac{\sum_{i=1}^{N} |\underline{P}Q_i|}{|\mathbb{U}|} \leq 1$$

- This approximation (for set X) is the number of objects on attribute set P can be positively identified as belonging to target set Q_i

Reduct

- A reduct is a sufficient set of features which by itself can fully characterize the knowledge in the database

- Produce same equivalence class structure as that expressed by the full attribute set
  $[x]_{RED} = [x]_P$

- It is minimal
  $[x]_{(RED-A)} \neq [x]_P \ \forall A \in RED$

- It is not unique

Core

- Core is the set of attributes which is common to all reducts
  $CORE(P) = \cap RED(P)$

- Consists of attributes which cannot be removed without causing collapse of the equivalence class structure

- It may be empty

- May be thought of as the set of necessary attributes

Decision System

- The information system is given as decision table.

a'	b'	c'	d'	D
M	L	3	M	1
M	L	1	H	1
L	L	1	M	1
L	R	3	M	2
M	R	2	M	2
L	R	3	L	3
H	R	3	L	3
H	N	3	L	3

- A decision table contains attributes and entries.
- The attributes are of two types 1.decision attribute
  2.Condition attribute

Discernibility Matrix

- Discernibility matrix is defined as:

$$M_{ij} = \{a \in A : a(x_i) \neq a(x_j)\} \quad \text{for } i, j = 1, 2, \ldots.., n.$$

a'	b'	c'	d'	D
M	L	3	M	1
M	L	1	H	1
L	L	1	M	1
L	R	3	M	2
M	R	2	M	2
L	R	3	L	3
H	R	3	L	3
H	N	3	L	3

a'	b'	c'	d'	D
M	L	3	M	1
M	L	1	H	1
L	L	1	M	1
L	R	3	M	2
M	R	2	M	2
L	R	3	L	3
H	R	3	L	3
H	N	3	L	3

Discernibility matrix

- the discernibility matrix is :

			a'b'	b'c'	a'b'd'	a'b'd'	a'b'd'
			b'c'd'	b'c'd'	a'b'c'd'	a'b'c'd'	a'b'c'd‘
			b'c'	a'b'c'	b'c'd'	a'b'c'd'	a'b'c'd'
					d'	a'd'	a'b'd’
					a'c'd‘	a'c'd‘	a'b'c'd'

- The entry (4,3) contains minimum number of attributes.
- The attribute in this entry is d’, which is called core attribute.

Reduct(NP-Complete)

- Reduct gives the minimal subset of attributes, for which essential properties of the information system are preserved.

- Core attributes are discarded from the attribute set, which gives {a',b',c‘}.

Quick Reduct Calculation Algorithm

- **Algorithm.** *QuickReduct*
- **Input:** $\mathbb{C}$, the set of all conditional features
- **Input:** $\mathbb{D}$, the set of all decisional features
- **Output:** $\mathbb{R}$, a feature subset
- $\mathbb{1}$. T := { }, $\mathbb{R}$: = { }
- 2. **repeat**
- 3. T : = $\mathbb{R}$
- 4. $\forall x \in (\mathbb{C} - \mathbb{R})$
- 5. if $\gamma_{\mathbb{R} \cup \{X\}}(\mathbb{D}) > \gamma_{\mathbb{T}}(\mathbb{D})$
- 6. T : = $\mathbb{R}$ U {x}
- 7. $\mathbb{R}$: = $\mathbb{T}$
- 8. **until** $\gamma_{\mathbb{R}}(\mathbb{D}) = \gamma_{\mathbb{C}}(\mathbb{D})$
- 9. **return** $\mathbb{R}$

Reduct

Core = {d’}

Non-Core = {a’, b’, c’}

Frequency of non-core attributes in discernibility matrix:

a’ : 15 b’ : 17 c’: 14

According to Order of importance:

Non-Core {b’, a’, c’}

Reduct

a'	b'	c'	d'	D
M	L	3	M	1
M	L	1	H	1
L	L	1	M	1
L	R	3	M	2
M	R	2	M	2
L	R	3	L	3
H	R	3	L	3
H	N	3	L	3

- Applying QuickReduct algorit

RED = Core = d’

[x]D = {1, 2, 3}, {4, 5}, {6, 7, 8}

To compute $\gamma_T(D)$

T = {d’}.   So [X]T = {1,3,4,5}, {2},{6,7,8}

When X = {1,2,3}, POS(X) = {2}

When X = {4,5}, POS(X) = NIL

When X = {6,7,8}, POS(X) = {6,7,8}

So, $\gamma_T(D) = |\{2,6,7,8\}|/|U| = 4/8 = 0.5$

Reduct

a'	b'	c'	d'	D
M	L	3	M	1
M	L	1	H	1
L	L	1	M	1
L	R	3	M	2
M	R	2	M	2
L	R	3	L	3
H	R	3	L	3
H	N	3	L	3

[x]D = {1, 2, 3}, {4, 5}, {6, 7, 8}

To compute $\gamma_{RU\{X\}}(D)$

So [X]d' U b' = {1,3},{2},{4,5},{6,7},{8}

When X = {1,2,3}, POS(X) = {1,2,3}

When X = {4,5}, POS(X) = {4,5}

When X = {6,7,8}, POS(X) = {6,7,8}

So, $\gamma_{RU\{X\}}(D) = |\{1,2,3,4,5,6,7,8\}|/|U| = 8/8 = 1.0$

$\gamma_{RU\{X\}}(D) > \gamma_{T}(D)$ and $\gamma_{RU\{X\}}(D) = 1.0$

=> {b', d'} is a reduct

Reduct

a'	b'	c'	d'	D
M	L	3	M	1
M	L	1	H	1
L	L	1	M	1
L	R	3	M	2
M	R	2	M	2
L	R	3	L	3
H	R	3	L	3
H	N	3	L	3

After the termination of QuickReduct algorithm:

Non-Core = {a’, c’}

Repeat the algorithm with Core = {d’}

[X]d’ U a’ = {1,5},{2},{3,4},{6}, {7,8}

When X = {1,2,3}, POS(X) = {2}

When X = {4,5}, POS(X) = NIL

When X = {6,7,8}, POS(X) = {6,7,8}

So, $\gamma_{RU\{X\}}(D)$ = |{2,6,7,8}|/|U| = 4/8 = 0.5

$\gamma_{RU\{X\}}(D) >= \gamma_{T}(D)$ so R = {d’, a’}

Reduct

Now, Non-Core = {c'}

Next Iteration:

[X]d' U a' U c' = {1},{2},{3},{4},{5},{6}, {7,8}

When X = {1,2,3}, POS(X) = {1,2,3}

When X = {4,5}, POS(X) = {4,5}

When X = {6,7,8}, POS(X) = {6,7,8}

So, $\gamma_{RU\{X\}}(D)$ = |{1,2,3,4,5,6,7,8}|/|U| = 8/8 = 1.0

$\gamma_{RU\{X\}}(D) >= \gamma_{T}(D)$ and $\gamma_{RU\{X\}}(D) = 1.0$

$\Rightarrow${d', a', c'} is a reduct

Also, Non-Core set is empty => terminate the process

Reduct

a'	b'	c'	d'	D
M	L	3	M	1
M	L	1	H	1
L	L	1	M	1
L	R	3	M	2
M	R	2	M	2
L	R	3	L	3
H	R	3	L	3
H	N	3	L	3

So, two possible reducts are:

{b’, d’} and

{a’,c’,d’}

CLASSIFICATION

Rule Extraction

- The category representations discussed above are all *extensional* in nature; that is, a category or complex class is simply the sum of all its members.
- To represent a category is, then, just to be able to list or identify all the objects belonging to that category.
- However, extensional category representations have very limited practical use, because they provide no insight for deciding whether novel objects are members of the category.

Rule Extraction

- What is generally desired is an *intentional* description of the category, a representation of the category based on a set of *rules* that describe the scope of the category.
- The choice of such rules is not unique. There is a few rule-extraction methods. We will start from a rule-extraction procedure based on Ziarko & Shan (1995).

Decision matrices

- Let us say that we wish to find the minimal set of consistent rules called logical implications that characterize our sample system.
- For a set of *condition* attributes $\mathcal{P} = \{P_1, P_2, P_3, \ldots, P_n\}$ and a decision attribute $Q, Q \notin \mathcal{P}$, these rules should have the form, $P_i^a P_j^b \cdots P_k^c \rightarrow Q^d$ or spelled out, $(P_i = a) \wedge (P_j = b) \wedge \ldots \wedge (P_k = c) \rightarrow (Q = d)$

- where $\{a, b, c, \ldots\}$ are legitimate values from the domains of their respective attributes.

Sample Information System

Object	P_1	P_2	P_3	P_4	P_5
O_1	1	2	0	1	1
O_2	1	2	0	1	1
O_3	2	0	0	1	0
O_4	0	0	1	2	1
O_5	2	1	0	2	1
O_6	0	0	1	2	2
O_7	2	0	0	1	0
O_8	0	1	2	2	1
O_9	2	1	0	2	2
O_{10}	2	0	0	1	0

Decision matrices

- This is a form typical of association rules, and the number of items in U which match the condition/antecedent is called the *support* for the rule.
- The method for extracting such rules given in Ziarko & Shan (1995) is to form a *decision matrix* corresponding to each individual value *d* of decision attribute *Q*.

Decision matrices

- Informally, the decision matrix for value *d* of decision attribute *Q* lists all attribute–value pairs that *differ* between objects having $Q = d$ and $Q \neq d$.
- Example - Consider the table above, and let P_4 be the decision variable and $\{P_1, P_2, P_3\}$ be the condition variables.
- We note that the decision variable P_4 takes on two different values, namely {1,2}. We treat each case separately.

Decision matrices

- First, we look at the case $P_4 = 1$, and we divide up into objects that have $P_4 = 1$ and those that have $P_4 \neq 1$
- The objects having $P_4 = 1$ are $\{O_1,O_2,O_3,O_7,O_{10}\}$ while the objects which have $P_4 \neq 1$ are $\{O_4,O_5,O_6,O_8,O_9\}$.
- The decision matrix for $P_4 = 1$ lists all the differences between the objects having $P_4 = 1$ and those having $P_4 \neq 1$
- Thus, the decision matrix lists all the differences between $\{O_1,O_2,O_3,O_7,O_{10}\}$ and $\{O_4,O_5,O_6,O_8,O_9\}$.

Decision matrix

Sample Information System

Object	P_1	P_2	P_3	P_4	P_5
O_1	1	2	0	1	1
O_2	1	2	0	1	1
O_3	2	0	0	1	0
O_4	0	0	1	2	1
O_5	2	1	0	2	1
O_6	0	0	1	2	2
O_7	2	0	0	1	0
O_8	0	1	2	2	1
O_9	2	1	0	2	2
O_{10}	2	0	0	1	0

Decision matrix for $P_4 = 1$

Object	O_4	O_5	O_6	O_8	O_9
O_1	P_1^1, P_2^2, P_3^0	P_1^1, P_2^2	P_1^1, P_2^2, P_3^0	P_1^1, P_2^2, P_3^0	P_1^1, P_2^2
O_2	P_1^1, P_2^2, P_3^0	P_1^1, P_2^2	P_1^1, P_2^2, P_3^0	P_1^1, P_2^2, P_3^0	P_1^1, P_2^2
O_3	P_1^2, P_3^0	P_2^0	P_1^2, P_3^0	P_1^2, P_2^0, P_3^0	P_2^0
O_7	P_1^2, P_3^0	P_2^0	P_1^2, P_3^0	P_1^2, P_2^0, P_3^0	P_2^0
O_{10}	P_1^2, P_3^0	P_2^0	P_1^2, P_3^0	P_1^2, P_2^0, P_3^0	P_2^0

Rule Extraction

- Thus, for the above table we have the following five Boolean expressions:

$$\begin{cases}
(P_1^1 \vee P_2^2 \vee P_3^0) \wedge (P_1^1 \vee P_2^2) \wedge (P_1^1 \vee P_2^2 \vee P_3^0) \wedge (P_1^1 \vee P_2^2 \vee P_3^0) \wedge (P_1^1 \vee P_2^2) \\
(P_1^1 \vee P_2^2 \vee P_3^0) \wedge (P_1^1 \vee P_2^2) \wedge (P_1^1 \vee P_2^2 \vee P_3^0) \wedge (P_1^1 \vee P_2^2 \vee P_3^0) \wedge (P_1^1 \vee P_2^2) \\
(P_1^2 \vee P_3^0) \wedge (P_2^0) \wedge (P_1^2 \vee P_3^0) \wedge (P_1^2 \vee P_2^0 \vee P_3^0) \wedge (P_2^0) \\
(P_1^2 \vee P_3^0) \wedge (P_2^0) \wedge (P_1^2 \vee P_3^0) \wedge (P_1^2 \vee P_2^0 \vee P_3^0) \wedge (P_2^0) \\
(P_1^2 \vee P_3^0) \wedge (P_2^0) \wedge (P_1^2 \vee P_3^0) \wedge (P_1^2 \vee P_2^0 \vee P_3^0) \wedge (P_2^0)
\end{cases}$$

statement, corresponding to object O_{10}, states that all the following must be satisfied:

1. Either P_1 must have value 2, or P_3 must have value 0, or both.
2. P_2 must have value 0.
3. Either P_1 must have value 2, or P_3 must have value 0, or both.
4. Either P_1 must have value 2, or P_2 must have value 0, or P_3 must have value 0, or any combination thereof.
5. P_2 must have value 0.

Rule Extraction

- It is clear that there is a large amount of redundancy here, and the next step is to simplify using traditional Boolean algebra. The statement

$$(P_1^1 \vee P_2^2 \vee P_3^0) \wedge (P_1^1 \vee P_2^2) \wedge (P_1^1 \vee P_2^2 \vee P_3^0) \wedge (P_1^1 \vee P_2^2 \vee P_3^0) \wedge (P_1^1 \vee P_2^2)$$

- Corresponding to objects $\{O_1, O_2\}$ simplifies to

$P_1^1 \vee P_2^2$,which yields the implication

$$(P_1 = 1) \vee (P_2 = 2) \rightarrow (P_4 = 1)$$

Rule Extraction

- Similarly, the statement

$$\left(P_1^2 \vee P_3^0\right) \wedge \left(P_2^0\right) \wedge \left(P_1^2 \vee P_3^0\right) \wedge \left(P_1^2 \vee P_2^0 \vee P_3^0\right) \wedge \left(P_2^0\right)$$

corresponding to objects $\{O_3, O_7, O_{10}\}$ simplifies to $P_1^2 P_2^0 \vee P_3^0 P_2^0$

- This gives the implication

$$\left(P_1 = 2 \wedge P_2 = 0\right) \vee \left(P_3 = 0 \wedge P_2 = 0\right) \rightarrow \left(P_4 = 1\right)$$

Possible rule s

Sample Information System

Object	P_1	P_2	P_3	P_4	P_5
O_1	1	2	0	1	1
O_2	1	2	0	1	1
O_3	2	0	0	1	0
O_4	0	0	1	2	1
O_5	2	1	0	2	1
O_6	0	0	1	2	2
O_7	2	0	0	1	0
O_8	0	1	2	2	1
O_9	2	1	0	2	2
O_{10}	2	0	0	1	0

$$\begin{cases} (P_1 = 1) \rightarrow (P_4 = 1) \\ (P_2 = 2) \rightarrow (P_4 = 1) \\ (P_1 = 2) \wedge (P_2 = 0) \rightarrow (P_4 = 1) \\ (P_3 = 0) \wedge (P_2 = 0) \rightarrow (P_4 = 1) \end{cases}$$