PROCESS SPECIFICATION

Contents

1. MOTIVATION AND LEARNING GOALS

2. LEARNING UNIT 1
   Structured English specification

3. LEARNING UNIT 2
   Decision table based specifications

4. LEARNING UNIT 3
   Detecting
   - Incompleteness
   - Ambiguity
   - Contradictions
   - Redundancy
   in decision table specification

5. LEARNING UNIT 4
   Eliminating redundancy in specifications

6. LEARNING UNIT 5
   Decision trees for specifications

7. REFERENCES
PROCESS SPECIFICATION

MOTIVATION

Before designing a system an analyst must clearly understand the logic to be followed by each process block in a DFD. An analyst’s understanding must be crosschecked with the user of the information system. A notation is thus needed to specify process block in detail, which can be understood by a user. Notation used must be appropriate for the type of the application to be modelled. Different notations are needed to represent repetition structures, complex decision situation and situations where sequencing of testing of conditions is important. For complex logical procedures a notation is needed which can also be used to detect logical errors in the specifications. This is called Decision Table. A tabular structure for representing logic can be used as a communication tool and can be automatically converted to a program.

LEARNING GOALS

At the end of this module you will know

1. How to use structured English to precisely specify processes
2. The terminology used in structured English
3. Terminology of decision tables and how it is used to specify complex logic
4. How to detect errors in decision table specifications
5. Terminology and use of decision trees
6. Comparison of structured English, decision tables and decision trees
Structured English specification

PROCESS SPECIFICATION

Once a DFD is obtained the next step is to precisely specify the process. Structured English, Decision tables and Decision Trees are used to describe processes. Decision tables are used when the process is logically complex involving large number of conditions and alternate solutions. Decision trees are used when conditions to be tested must follow a strict time sequence.

STRUCTURED ENGLISH

Structured English is similar to a programming language such as Pascal. It does not have strict syntax rules as in programming languages as the intention is only to give precise description of a process. The structured English description should be understandable to the user.

Example:

if customer pays advance
    then Give 5% Discount
else
    if purchase amount >=10,000
        then
            if the customer is a regular customer
                then Give 5% Discount
                else No Discount
            end if
        end if
    else No Discount
end if

DECISION TABLE-EXAMPLE

Same structured English procedure given as decision table

          if          else
          ------      ------
          customer   customer
          pays        is  a
          advance      regular
  then                      then
  Give 5% Discount            Give 5% Discount
          else
          No Discount
end if
<table>
<thead>
<tr>
<th>CONDITIONS</th>
<th>RULE1</th>
<th>RULE2</th>
<th>RULE3</th>
<th>RULE4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Advance payment made</td>
<td>Y</td>
<td>N</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>Purchase amt &gt;=10,000</td>
<td>-</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
</tr>
<tr>
<td>Regular Customer?</td>
<td>-</td>
<td>Y</td>
<td>N</td>
<td>-</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ACTIONS</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Give 5% Discount</td>
<td>X</td>
</tr>
<tr>
<td>Give No Discount</td>
<td>-</td>
</tr>
</tbody>
</table>

**DECISION TABLE-EXPLANATION**

- Conditions are questions to be asked
- ‘Y’ is yes, ’N’ is no & ‘-’ is irrelevant
- A ‘X’ against the action says the action must be taken
- A ‘-’ against the action says the action need not be taken

Rule 2 in decision table DISCOUNT states:
if no advance payment and purchase amount >= 10000
and regular customer then give 5% discount

In Structured English, imperative sentences, actions to be performed should be precise and quantified
Good Example: Give discount of 20%
Bad Example: Give substantial discount

The operators and keywords in Structured English are as follows:

Operators - Arithmetic: +, -, /, *
   Relational: >, >=, <, <=, =, !=
   Logical: and, or, not
Keywords: if, then, else, repeat, until, while, do, case,
The structured English procedure given above is expressed as a Decision tree below.

**Structured English-Decision Structures**

```
If condition
    then
    { Group of statements }
else
    { Group of statements }
```

C1: Advance payment made
C2: Purchase amount >= 10,000
C3: Regular Customer

Y = Yes
N = No
end if

Example: if (balance in account >= min.balance)
    then honor request
    else reject request
end if

STRUCTURED ENGLISH-CASE STATEMENT

Case (variable)
    Variable = P: { statements for alternative P}
    Variable = Q: { statements for alternative Q}
    Variable = R: { statements for alternative R}
    None of the above: { statements for default case}
end case

Example: Case (product code)
    product code = 1: discount = 5%
    product code = 2: discount = 7%
    None of the above: discount = 0
end case

STRUCTURED ENGLISH-REPETITION STRUCTURE

for index = initial to final do
    { statements in loop }
end for

Example: Total = 0
for subject = 1 to subject = 5 do
total marks = total marks + marks(subject)
write roll no, total marks
end for

**STRUCTURED ENGLISH-WHILE LOOP**

while condition do
    { statements in loop }
end while

Example: while there are student records left do
    read student record
    compute total marks
    find class
    write total marks, class, roll no
end while

**EXAMPLE**

Update inventory file
for each item accepted record do
    { search inventory file using item code
        if successful
            then { update retrieved inventory record;
                    write updated record in inventory file using accepted record}
        else { create new record in inventory file;
                    enter accepted record in inventory file}
        end if
    end for

**LEARNING UNIT 2**

Decision table based specifications

**ADVANTAGES OF DECISION TABLE**
Easy to understand by non-computer literate users and managers. Good documentation of rules used in data processing. Simple representation of complex decision rules. Tabular representation allows systematic validation of specification detection of redundancy, incompleteness & inconsistency of rules. There exist algorithms to automatically convert decision tables to equivalent computer programs.

**METHOD OF OBTAINING DECISION TABLE FROM WORD STATEMENT OF RULES**

**EXAMPLE**
A bank uses the following rules to classify new accounts
If depositor's age is 21 or above and if the deposit is Rs 100 or more, classify the account type as A
If the depositor is under 21 and the deposit is Rs 100 or more, classify it as type B
If the depositor is 21 or over and deposit is below Rs 100 classify it as C
If the depositor is under 21 and deposit is below Rs 100 do-not open account
Identify Conditions:  Age >= 21  Cl
Deposits >= Rs 100: C2
Identify Actions : Classify account as A, B or C
Do not open account

**DECISION TABLE FROM WORD STATEMENT**
**Condition Stub**

<table>
<thead>
<tr>
<th>CONDITIONS</th>
<th>Rule 1</th>
<th>Rule 2</th>
<th>Rule 3</th>
<th>Rule 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>C1: Age &gt;= 21</td>
<td>Y</td>
<td>N</td>
<td>Y</td>
<td>N</td>
</tr>
<tr>
<td>C2: Deposit &gt;=100</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
<td>N</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ACTIONS</th>
</tr>
</thead>
<tbody>
<tr>
<td>A1: Classify as A</td>
</tr>
<tr>
<td>A2: Classify as B</td>
</tr>
<tr>
<td>A3: Classify as C</td>
</tr>
<tr>
<td>A4: Do not open Account</td>
</tr>
</tbody>
</table>

**Action Stub**

**DECISION TABLE NOTATION EXPLAINED**

<table>
<thead>
<tr>
<th>CONDITION STUB</th>
<th>CONDITION ENTRIES</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACTION STUB</td>
<td>ACTION ENTRIES</td>
</tr>
</tbody>
</table>
INTERPRETING DECISION TABLE-ELSE RULE

C1: Is applicant sponsored? | Y | Y
C2: Does he have min Qualification? | Y | Y | ELSE
C3: Is fee paid? | Y | N

A1: Admit letter | X | - | -
A2: Provisional Admit letter | - | X | -
Interpretation
R1: If applicant sponsored and he has minimum qualifications and his fee is paid – Send Admit letter
R2: If applicant sponsored and has minimum qualifications and his fee not paid send provisional admit letter
ELSE: In all other cases send regret letter. The else rule makes a decision table complete

DECISION TABLE FOR SHIPPING RULES

<table>
<thead>
<tr>
<th>C1: Qty ordered &lt;= Quantity in stock?</th>
<th>R1</th>
<th>R2</th>
<th>R3</th>
<th>R4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>C2: (Qty in stock-Qty ordered)&lt;=reorder level</td>
<td>N</td>
<td>Y</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>C3: Is the partial shipment ok?</td>
<td></td>
<td></td>
<td>Y</td>
<td>N</td>
</tr>
</tbody>
</table>

| A1: Qty shipped=Qty ordered         | X  | X  | -  | -  |
| A2: Qty shipped=Qty in stock        |    |    | X  | -  |
EXTENDED ENTRY DECISION TABLE

- Condition Entries not necessarily Y or N
- Action entries not necessarily X or - Extended Entry Decision Tables (EEDT) more concise
- EEDT can always be expanded to LEDT
## MIXED ENTRY DECISION TABLE

Can mix up Yes, No answers with codes

<table>
<thead>
<tr>
<th>Example</th>
<th>R1</th>
<th>R2</th>
<th>R3</th>
<th>R4</th>
<th>R5</th>
<th>R6</th>
</tr>
</thead>
<tbody>
<tr>
<td>C1: Product code</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>C2: Customer code</td>
<td>A</td>
<td>B</td>
<td>A</td>
<td>B</td>
<td>C</td>
<td>-</td>
</tr>
<tr>
<td>C3: Order amount</td>
<td>&lt;= 500</td>
<td>&lt;= 500</td>
<td>&gt; 500</td>
<td>&gt; 500</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Discount =</th>
<th>5%</th>
<th>7.5%</th>
<th>7.5%</th>
<th>10%</th>
<th>6%</th>
<th>5%</th>
</tr>
</thead>
</table>

**Example**

<table>
<thead>
<tr>
<th>C1: Product code = 1?</th>
<th>R1</th>
<th>R2</th>
<th>R3</th>
<th>R4</th>
<th>R5</th>
<th>R6</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
</tr>
<tr>
<td>C2: Customer code</td>
<td></td>
<td>A</td>
<td>B</td>
<td>A</td>
<td>B</td>
<td>C</td>
</tr>
<tr>
<td>C3: Order amount &lt; 500?</td>
<td></td>
<td>Y</td>
<td>Y</td>
<td>N</td>
<td>N</td>
<td>-</td>
</tr>
</tbody>
</table>

**Discount =**

<table>
<thead>
<tr>
<th>5%</th>
<th>7.5%</th>
<th>7.5%</th>
<th>10%</th>
<th>6%</th>
<th>5%</th>
</tr>
</thead>
</table>
Choice of LEDT, EEDT, MEDT depends on ease of communication with user. Softwares are available to translate DTs to programs. DT’s are easy to check.

### LINKED DECISION TABLE

#### Decision table 1

| Salary point=6 | Y | e |
| Conduct OK? | Y | l |
| Diligence OK? | Y | s |
| Efficiency OK? | Y | e |

- Go to table 2
- No promotion

#### Decision table 2

| Salary point>2 | N | N | N | Y |
| 1 yr as class 1 officer | Y | N | - | - |
| Departmental test Passed? | Y | - | N | - |

- Advance to next salary point
- No promotion
- Go to Table3

#### Decision table 3

| Complete departmental Course | Y else |
| 1 yr since last increment | Y |
1. Observe that one can branch between tables
2. Whenever complex rules are given it is a good idea to break them up into manageable parts

LOGICAL CORRECTNESS OF DECISION TABLE

Consider decision table DTI:

<table>
<thead>
<tr>
<th></th>
<th>R1</th>
<th>R2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cl: x&gt;60</td>
<td>Y</td>
<td>-</td>
</tr>
<tr>
<td>C2: x&lt;40</td>
<td>-</td>
<td>Y</td>
</tr>
<tr>
<td>A1</td>
<td>X</td>
<td>-</td>
</tr>
<tr>
<td>A2 :</td>
<td>-</td>
<td>X</td>
</tr>
</tbody>
</table>

We can expand decision table by replacing each – by Y & N

DT2:

<table>
<thead>
<tr>
<th></th>
<th>R11</th>
<th>R12</th>
<th>R21</th>
<th>R22</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cl: x&gt;60</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
<td>Y</td>
</tr>
<tr>
<td>C2: x&lt;40</td>
<td>Y</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>A1</td>
<td>X</td>
<td>X</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>A2 :</td>
<td>-</td>
<td>-</td>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>

A rule which has no – is an Elementary rule

DT2 is an Elementary Rule Decision Table (ERDT)
From this table we see that the rule YY has two contradictory actions. Thus we need to examine the table further and make sure it is not a serious mistake. Also the rule C1=C2=N is missing which needs further examination.

LEARNING UNIT 3

Detecting- Incompleteness, Ambiguity, Contradictions & Redundancy in decision table specification

LOGICAL CORRECTNESS OF DECISION TABLE (CONT'D)

A decision table with 1 condition should have 2 elementary rules, each elementary rule must be distinct, each elementary rule must have distinct action, if a decision table with k conditions does not have $2^k$ rules specified it is said to be incomplete.
For example: DT2 does not have the elementary rule C1:N, C2:N. It is thus incomplete.
If the decision table has the same elementary rule occurring more than once it is said to have multiplicity of specifications
For Example: In DT2 The rule C1:Y, C2:Y occurs twice. Thus it has multiplicity of specification.

If action specified for multiple identical rules are different then it is called ambiguous specifications
DT2 has an ambiguity. Rules R11 and R22 are identical but have different actions. Ambiguity may be apparent or real. It is said to be apparent if the rule leading to the ambiguity is logically impossible
For example, (x>60)=Y and (x<40)=Y cannot occur simultaneously.
Thus in DT2 rules R11 and R22 are apparently ambiguous rules
Apparently ambiguous rules is not an error.
If an apparently ambiguous specification is real then it is a contradiction.
For example: If C1: \((X > 60) = Y\) and C2: \((X > 40) = Y\) then 
\(X = 70\) will satisfy both inequalities.
As two actions are specified for \((C1 = Y, C2 = Y)\) and they are different the rule is really ambiguous and is called Contradictory Specification.

If all \(2^k\) elementary rules are not present in a \(k\) condition decision table is said to be incomplete.

DT2 is incomplete as rule C1:N, C2:N is missing.
Rule C1=N, C2:=N is logically possible as C1=N is X<=60 and C2=N is X >= 40. A value of X = 50 will make C1=N, C2=N
Thus DT2 has a real incomplete specification.
A decision table which has no real ambiguities or real incompleteness is said to be logically correct. Decision table with logical errors should be corrected.

**USE OF KARNAUGH MAPS**

KARNAUGH map abbreviated K-map is a 2 dimensional diagram with one square per elementary rule.

The k-map of DT2 is

<table>
<thead>
<tr>
<th>C1</th>
<th>N</th>
<th>Y</th>
</tr>
</thead>
<tbody>
<tr>
<td>C2</td>
<td></td>
<td>A1</td>
</tr>
<tr>
<td>N</td>
<td></td>
<td>A2</td>
</tr>
<tr>
<td>Y</td>
<td></td>
<td>A1, A2</td>
</tr>
</tbody>
</table>

If more than one action is in one square it is an ambiguous rule.
If a square is empty it signifies incomplete specification.

**USE OF KARNAUGH MAPS**

Structured English procedure:
If carbon content<0.7
  then if Rockwell hardness>50
    then if tensile strength>30000
      then steel is grade 10
    else steel is grade 9
  end if
  else steel is grade 8
end if
else steel is grade 7
end if

DT3:

**Decision table-Grading steel**

<table>
<thead>
<tr>
<th>C1: Carbon content &lt;0.7</th>
<th>C2: Rockwell hardness&gt;50</th>
<th>C3 tensile strength&gt;30000</th>
<th>Grade</th>
</tr>
</thead>
<tbody>
<tr>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>10</td>
</tr>
<tr>
<td>Y</td>
<td>Y</td>
<td>N</td>
<td>9</td>
</tr>
<tr>
<td>Y</td>
<td>N</td>
<td>N</td>
<td>8</td>
</tr>
<tr>
<td>Y</td>
<td>N</td>
<td>N</td>
<td>?</td>
</tr>
<tr>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>?</td>
</tr>
<tr>
<td>Y</td>
<td>N</td>
<td>Y</td>
<td>?</td>
</tr>
<tr>
<td>Y</td>
<td>Y</td>
<td>N</td>
<td>?</td>
</tr>
<tr>
<td>Y</td>
<td>N</td>
<td>N</td>
<td>?</td>
</tr>
</tbody>
</table>

**KARNAUGH MAPS – GRADING STEEL**

<table>
<thead>
<tr>
<th>C1</th>
<th>C2</th>
<th>C3</th>
</tr>
</thead>
<tbody>
<tr>
<td>N</td>
<td>NN</td>
<td>?</td>
</tr>
<tr>
<td>Y</td>
<td>?</td>
<td>?</td>
</tr>
</tbody>
</table>

Observe that the fact that the specification is incomplete is obvious in the Decision table whereas the structured English specification seems complete which is not.
### DT4: DECISION TABLE-ARREARS MANAGEMENT

<table>
<thead>
<tr>
<th>C1: Payment in current month &gt; min. specified payment</th>
<th>R1</th>
<th>R2</th>
<th>R3</th>
<th>R4</th>
<th>R5</th>
<th>R6</th>
</tr>
</thead>
<tbody>
<tr>
<td>Y</td>
<td>N</td>
<td>N</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>C2: Payment in current month &gt; 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-</td>
<td>Y</td>
<td>Y</td>
<td>-</td>
<td>N</td>
<td>N</td>
<td></td>
</tr>
<tr>
<td>C3: Any payment in last 3 months</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-</td>
<td>-</td>
<td>-</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
<td></td>
</tr>
<tr>
<td>C4: Actual arrears &gt; 3(min. Specified payment per month)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-</td>
<td>Y</td>
<td>N</td>
<td>Y</td>
<td>N</td>
<td>Y</td>
<td></td>
</tr>
</tbody>
</table>

| A1: Send letter A                                      | X  |    |    |    |    |    |
| A2: Send letter B                                      |    | X  |    |    |    |    |
| A3: Send letter C                                      |    | -  | X  |    |    |    |
| A4: Send letter D                                      |    | -  | -  | X  |    | X  |
| A5: Send letter E                                      |    | -  | -  | -  | X  | -  |
### KARNAUGH MAP

<table>
<thead>
<tr>
<th></th>
<th>C1C2</th>
<th>C3C4</th>
<th>NN</th>
<th>NY</th>
<th>YY</th>
<th>YN</th>
</tr>
</thead>
<tbody>
<tr>
<td>NN</td>
<td>?</td>
<td>A3</td>
<td>A1</td>
<td>A1</td>
<td>A1*</td>
<td></td>
</tr>
<tr>
<td>NY</td>
<td>A4</td>
<td>A2A4+</td>
<td>A1A4</td>
<td>A1A4*</td>
<td></td>
<td></td>
</tr>
<tr>
<td>YY</td>
<td>A4</td>
<td>A2</td>
<td>A1</td>
<td>A1</td>
<td>A1A4*</td>
<td></td>
</tr>
<tr>
<td>YN</td>
<td>A5</td>
<td>A4</td>
<td>A1</td>
<td>A1</td>
<td>A1A5*</td>
<td></td>
</tr>
</tbody>
</table>

C1: x>m C2: x>0 C3: y>0 C4: z>3m m>0
C3, C4 independent of C1, C2
C1, C2 dependent

C1: Y C2: Y x>m, x>0 possible
C1: N C2: Y x<=m, x>0 not logically possible
C1: N C2: N x<=m, x<=0 possible
Thus C1, C2, C3, C4: NNNN incomplete specification

BOXES MARKED * NOT LOGICALLY POSSIBLE

Rules C1 C2 C3 C4: NYNY and YYNY logical errors
Errors to be corrected after consulting users who formulated the rules
**CORRECT DECISION TABLE**

If users say that for rules C1C2C3C4:NYNY AND YYNY (marked with + in k-map) the action is A4 and for C1C2C3C4:NNNN also it is A4, the corrected map is

<table>
<thead>
<tr>
<th>C1C2</th>
<th>NN</th>
<th>NY</th>
<th>YY</th>
<th>YN</th>
</tr>
</thead>
<tbody>
<tr>
<td>NN</td>
<td>A4</td>
<td>A3</td>
<td>A1</td>
<td></td>
</tr>
<tr>
<td>NY</td>
<td>A4</td>
<td>A4</td>
<td>A4</td>
<td></td>
</tr>
<tr>
<td>YY</td>
<td>A4</td>
<td>A2</td>
<td>A1</td>
<td></td>
</tr>
<tr>
<td>YN</td>
<td>A5</td>
<td>A3</td>
<td>A1</td>
<td></td>
</tr>
</tbody>
</table>

**CORRECTED DECISION TABLE  DT4**

<table>
<thead>
<tr>
<th>C1</th>
<th>Y</th>
<th>Y</th>
<th>Y</th>
<th>N</th>
<th>N</th>
<th>N</th>
<th>N</th>
<th>N</th>
<th>Y</th>
<th>N</th>
<th>N</th>
<th>N</th>
<th>N</th>
<th>N</th>
</tr>
</thead>
<tbody>
<tr>
<td>C2</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>C3</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
<td>Y</td>
<td>N</td>
<td>N</td>
<td>Y</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>C4</td>
<td>N</td>
<td>Y</td>
<td>N</td>
<td>Y</td>
<td>N</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
</tr>
</tbody>
</table>

**IMPOSSIBLE RULES**
Question: Can the number of rules be reduced?
Answer: Yes, by combining rules with the same action

Action A1 can be represented by the Boolean expression:
\[ C_1C_2\overline{C}_3\overline{C}_4 + C_1C_2C_3\overline{C}_4 + C_1\overline{C}_2C_3C_4 = C_1C_2\overline{C}_3\overline{C}_4 + C_1C_2C_3 (\overline{C}_4 + \overline{C}_4) \]
\[ = C_1C_2C_3C_4 + C_1C_2C_3 = C_1C_2\overline{C}_4 + C_1C_2C_3 \]

**LEARNING UNIT 4**

**Eliminating redundancy in specifications**

**REDUNDANCY ELIMINATION**

Redundancy can be eliminated by systematically applying four identities of Boolean Algebra

These identities are
\[ A + \overline{A} = 1 \]
KARNAUGH MAP REDUCTION

C3 C4
NN NY YY YN
A1 A1 A1 A1
NN NY YY YN
A1 A1 A1 A1
NN NY YY YN
A1 A1 A1 A1
NN NY YY YN
A1 A1 A1 A1
NN NY YY YN
A1 A1 A1 A1

C1C2
C3 C4
NN NY YY YN
A2 A2 A2 A2
NN NY YY YN
A2 A2 A2 A2
NN NY YY YN
A2 A2 A2 A2
NN NY YY YN
A2 A2 A2 A2

C3 C4
NN NY YY YN
A3 A3 A3 A3
NN NY YY YN
A3 A3 A3 A3
NN NY YY YN
A3 A3 A3 A3
NN NY YY YN
A3 A3 A3 A3

A1 A2 A3
A3 = C1C2C3C4 + C1C2C3C4 + C1C2C3C4 + C1C2C3C4
    = C2C3C4(C1+C1) + C2C3C4(C1+C1)
    = C2C4(C3+C3) = C2C4

**REDUCING DECISION TABLES-USE OF K-MAP**

<table>
<thead>
<tr>
<th></th>
<th>NN</th>
<th>NY</th>
<th>YY</th>
<th>YN</th>
</tr>
</thead>
<tbody>
<tr>
<td>C1C2</td>
<td>A4</td>
<td>A3</td>
<td>A1</td>
<td></td>
</tr>
<tr>
<td>C3C4</td>
<td></td>
<td>A4</td>
<td>A4</td>
<td>A4</td>
</tr>
<tr>
<td>NN</td>
<td>A4</td>
<td>A4</td>
<td>A2</td>
<td>A1</td>
</tr>
<tr>
<td>NY</td>
<td></td>
<td>A5</td>
<td>A3</td>
<td>A1</td>
</tr>
<tr>
<td>YY</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>YN</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Boxes marked X correspond to impossible rules. They can be employed if they are useful in reducing rules.

Using k-map reduction rules we get

A1 : C1C4 + C1C3
A2 : C1C2C3C4
A3 : C1C2C4
A4 : C3C4 + C2C3 + C2C4
A5 : C2C3C4
REDUCING DECISION TABLES

| C1: Payment in current month > min specified payment | Y | Y | N | N | - | - | - | - |
| C2: Payment in current month > 0 | - | - | Y | Y | - | N | N | N |
| C3: Any payment in last 3 months | - | Y | Y | - | N | N | - | Y |
| C4: Actual arrears > 3(minimum specified payment per month) | N | - | Y | N | Y | - | Y | N |

**EXAMPLE-REDUCTION OF RULES IN WORD STATEMENT**

Rules to Insure Driver if following rules are satisfied:
1. Drivers annual income > 20000 & is married male
2. Drivers annual income > 20000 & is married and over 30
3. Drivers annual income <= 20000 & she is married female
4. Driver is male over 30
5. Driver is married and age is not relevant
   Else do not insure

Conditions:
- C1 : Annual income > 20000
- C2 : Male
- C3 : Married
- C4: Age > 30

Action: Insure or do not insure

**DECISION TABLE FOR INSURANCE RULES**
Decision trees for specifications

Reduced rules: Insure if married or male over 30
Observe 5 rules simplified to 2 and 1 condition removed

Decision Trees is used when sequence of testing condition is important. It is more procedural compared to Decision tables.

EXAMPLE – DECISION TREE TO BOOK TRAIN TICKET

C1: Is II AC ticket available on 4/8/04
C2: Is II AC ticket available on 5/8/04
C3: Is sleeper available on 4/8/04
C4: Is sleeper available on 5/8/04

Observe in the tree sequencing of conditions which is important in this example

CONDITIONS

- Decision trees are drawn left to right
- Circles used for conditions
- Conditions labelled and annotation below tree
- Conditions need not be binary

For example:

```
C1
  Y
  \--- Book II AC
       \--- Y
           \--- Book II AC
                 \--- Y
                      \--- Book sleeper
                                        \--- Y
                                             \--- Book sleeper
                                                  \--- N
                                                       \--- Return
```

C1: Is II AC ticket available on 4/8/04
C2: Is II AC ticket available on 5/8/04
C3: Is sleeper available on 4/8/04
C4: Is sleeper available on 5/8/04

GRADE A: \( \geq 60 \)
GRADE B: \( \geq 50 \)
GRADE C: \( \geq 40 \)
GRADE F: else
Sometimes Decision trees are more appropriate to explain to a user how decisions are taken.

### COMPARISON OF STRUCTURED ENGLISH, DECISION TABLES AND DECISION TREES

<table>
<thead>
<tr>
<th>CRITERION FOR COMPARISON</th>
<th>STRUCTURED ENGLISH</th>
<th>DECISION TABLES</th>
<th>DECISION TREES</th>
</tr>
</thead>
<tbody>
<tr>
<td>ISOLATING CONDITIONS &amp; ACTIONS</td>
<td>NOT GOOD</td>
<td>BEST</td>
<td>GOOD</td>
</tr>
<tr>
<td>SEQUENCING CONDITIONS BY PRIORITY</td>
<td>GOOD</td>
<td>NOT GOOD</td>
<td>BEST</td>
</tr>
<tr>
<td>CHECKING FOR COMPLETENESS, CONTRADICTION</td>
<td>NOT GOOD</td>
<td>BEST</td>
<td>NOT GOOD</td>
</tr>
</tbody>
</table>
WHEN TO USE STRUCTURED ENGLISH, DECISION TABLES AND DECISION TREES

Use Structured English if there are many loops and actions are complex

Use Decision tables when there are a large number of conditions to check and logic is complex

Use Decision trees when sequencing of conditions is important and if there are not many conditions to be tested

REFERENCES

