Challenge April-2024

Using Lookup Tables in Decision Models

A solution with DT5GL by Jack Jansonius – 3 July 2024

Introduction

Broadly speaking, this challenge is solved along the same lines as my solutions for the March Challenge:

1. Initially, I could not get a grip on the intent of this challenge. The specification of what to do was drafted very unclearly, for example, with phrases such as, "has a corresponding diagnosis," "If no corresponding pair is found, report all procedures without compatible diagnoses," and "If a pair [...] is found, report all incompatible pairs." For that reason, I entered the text of the challenge into ChatGPT with the request to clarify it. This already succeeded with some follow-up questions, upon which I asked for the solution to the challenge with a Python program in the same session. Again, ChatGPT came up with a complete, immediately working program and that, moreover, in 3 variants.[1]

2. The next challenge was to convert the algorithms in Python to DT5GL decision tables. It always remains exciting whether the expressiveness of a knowledge modeling language is sufficient to handle any desired specification.

Regarding the specification "If no corresponding pair is found, report all procedures without compatible diagnoses," all [diagnosis, procedure] combinations for a procedure must be checked until a combination is found (and the procedure is not reported) or no combination is found (and the procedure is reported).

I was pleasantly surprised that this turned out to be no problem at all. Reading in the Claim file in JSON format and reading in the 2 CSV files is entirely based on the provided Python code from ChatGPT.

Remarkably, of all procedures with type X or Y, only 1 appears in the default list "CompatibleCodes.csv", namely 11000. And this one has no corresponding pair, so all procedures of type X or Y are actually reported. For this reason, I have added to "CompatibleCodes.csv" 2 pairs, namely [I10,28003] and [A18.83,11011] which can now therefore be found, so that these procedure codes are no longer reported.

3. Now the 2 CSV files can be read and searched using Python, but it is still a lot easier and faster - especially in terms of performance [2] - to import and access these files within a relational database, for example PostgreSQL or SQLite (or any other database). This required only a slight modification of the solution in point 2.

---

[1] I have not included these solutions in this document but they can be found in the session with ChatGPT: https://chatgpt.com/share/1adacab7-5234-4d18-8632-4543f3d2ed6c

[2] After putting the necessary indexes on the files; again, ChatGPT provided the required instructions, both for PostgreSQL and SQLite.

Implementation of the decision model in DT5GL/Python:

```
Initial_instructions:
>>: fn = extract_claim('Claim.json')                # read claim (DTFunctions.py)
>>: fn = extract_compatible_codes('CompatibleCodesPlus.csv')    # read csv-file1
>>: fn = extract_incompatible_codes('IncompatibleCodes.csv')    # read csv-file2
>>: collected_procedures = ""           # procedures without compatible diagnoses
>>: collected_incompatible_pairs = ""    # incompatible Procedure-Diagnosis pairs
End_Instructions


Table 0:
If:                                          | 0| 1|
next procedure_nr in [0 - lastProcedure]     | Y| N|
Then:
Action is all_procedures_checked             |  | X|
Action is check_diagnoses                    | X|  |
# .......
# repeat until: all_procedures_checked

Attribute: lastProcedure     Type: Integer
Equals: len_procedures() - 1


rTable 1:
If:                                                    | 0| 1| 2| 3|
next diagnosis_nr in [0 - lastDiagnosis]               | Y| Y| N| N|
True = procedure_type in ["X", "Y"]                    | Y| -| Y| N|[3]
True = CompatibleCodes(procedure_code, diagnosis_code) | Y| -| -| -|
procedure_type = "Z"                                   | -| Y| -| -|
True = InCompatibleCodes(procedure_code, diagnosis_code) | -| Y| -| -|
Then:
Act_Diagnosis is all_diagnoses_handled_type_XY         |  |  | X|  |
Act_Diagnosis is all_diagnoses_handled_type_other      |  |  |  | X|
Act_Diagnosis is no_report_and_break                   | X|  |  |  |
Act_Diagnosis is report_incompatible_pair              |  | X|  |  |
# .......
# repeat until: all_diagnoses_handled_type_XY,
#               all_diagnoses_handled_type_other,
#               no_report_and_break

Attribute: fn                Type: Integer
Attribute: True              Type: Integer
Equals: 1

Attribute: lastDiagnosis     Type: Integer
Equals: len_diagnoses() - 1

Attribute: procedure_type
Equals: get_procedure_type_from_claim(procedure_nr)

Attribute: procedure_code    Type: Text
Equals: get_procedure_code_from_claim(procedure_nr)

Attribute: diagnosis_code    Type: Text
Equals: get_diagnosis_code_from_claim(diagnosis_nr)

Attribute: collected_procedures_print   Type: Text
Equals: collected_procedures[2:] if collected_procedures != "" else "None"
# [2:] => skip first ", "

Attribute: collected_incompatible_pairs_print  Type: Text
Equals: collected_incompatible_pairs[2:] if collected_incompatible_pairs != "" else
"None"
```

[3] In future versions, the addition “True = ” will no longer be necessary.

```
GOALATTRIBUTE: Action
Repeat_until: all_procedures_checked

Case: all_procedures_checked
Print: "-----------------------------------------------"
Print: "Procedures without compatible diagnoses: %s"  collected_procedures_print
Print: "Incompatible pairs: %s"
collected_incompatible_pairs_print
Print: "-----------------------------------------------"

Case: check_diagnoses
Print: "#REM# - "


GOALATTRIBUTE: Act_Diagnosis
Repeat_until: all_diagnoses_handled_type_XY, all_diagnoses_handled_type_other,
no_report_and_break

Case: all_diagnoses_handled_type_XY
>>: collected_procedures = collected_procedures + ", " + procedure_code

Case: all_diagnoses_handled_type_other
Print: "#REM# - "

Case: no_report_and_break
Print: "#REM# - "

Case: report_incompatible_pair
>>: collected_incompatible_pairs = collected_incompatible_pairs + ", " +
procedure_code + "-" + diagnosis_code
```

Required Python code in DTFunctions.py:

```
import json
import csv

def load_json2(file_path):
    with open(file_path, 'r') as file:
        return json.load(file)


def load_csv(file_path):
    with open(file_path, 'r') as file:
        reader = csv.reader(file)
        return list(reader)

def extract_claim(file_name):
    global procedures, diagnoses
    claim = load_json2(file_name)
    procedures = claim['claim']['procedures']
    diagnoses = claim['claim']['diagnoses']
    return 1

def extract_compatible_codes(file_name):
    global compatible_codes
    compatible_codes = load_csv(file_name)
    # Convert compatible codes to list of lists
    compatible_codes = [item for item in compatible_codes]
    return 1

def extract_incompatible_codes(file_name):
    global incompatible_ranges
    incompatible_ranges = load_csv(file_name)
    # Convert incompatible ranges to list of lists
    incompatible_ranges = [[item[0], item[1], item[2], item[3]]
                           for item in incompatible_ranges]
    return 1

def len_procedures():
    return len(procedures)

def len_diagnoses():
    return len(diagnoses)

def get_procedure_type_from_claim(procedure_nr):
    return procedures[procedure_nr]["type"]

def get_procedure_code_from_claim(procedure_nr):
    return procedures[procedure_nr]["code"]

def get_diagnosis_code_from_claim(diagnosis_nr):
    return diagnoses[diagnosis_nr]["code"]

def CompatibleCodes(procedure_code, diagnose_code):
    return [diagnose_code, procedure_code] in compatible_codes

def InCompatibleCodes(procedure_code, diagnose_code):
    for range_item in incompatible_ranges:
        if  (range_item[0] <= procedure_code <= range_item[1])
        and (range_item[2] <= diagnose_code  <= range_item[3]):
            return True
    return False
```

Output DT5GL/Python:

The following lines were added to the CompatibleCodes.csv file at a random location:

L23.00,28003
L23.01,28003
L23.02,28003
L23.03,28003
L23.04,28003
L23.05,28003
I10,28003
L23.06,28003
L23.07,28003

and

A18.83,11011

(and then saved as CompatibleCodesPlus.csv), so procedures 11011 and 28003 are not listed under Procedures without compatible diagnoses, since the pairs I10,28003 and A18.83,11011 can be found in the file.

```
PS C:\Users\Administrator\dt5gl> .\dt.exe[4] -source:Claim_JSON_v1.txt -nti
-----------------------------------------------
Procedures without compatible diagnoses: 12001, 28005, 11000, 27675, 49657, 62319[5]
Incompatible pairs: 28415-I10, 28415-J38.6, 28415-J38.3, 28415-H52.01, 28415-
H47.632, 28415-N64.81, 30420-I10, 30420-L76.01, 30420-N64.81
-----------------------------------------------
Time elapsed: 0:00:02.180757
```

[4] DT.exe is Python code, compiled to C, and runs directly under Windows (without pre-installation of Python). Download and all necessary files available at: https://github.com/JackJansonius/DT5GL

[5] Exercise: modify the code of the script "Claim_JSON_v1.txt" so that also appears here:
Procedures with compatible diagnoses: 28003, 11011
Procedures with unknown types: 49180 - N

Implementation of the decision model in DT5GL/PostgreSQL/SQLite:

```
PostgreSQL_database: "csv_files"
# SQLite_database: "Database/CSV files.db"

Initial_instructions:
>>: fn = extract_claim('Claim.json')      # read claim (DTFunctions.py)
>>: collected_procedures = ""             # procedures without compatible diagnoses
>>: collected_incompatible_pairs = ""     # incompatible Procedure-Diagnosis pairs
End_Instructions


Table 0:
If:                                          | 0| 1|
next procedure_nr in [0 - lastProcedure]     | Y| N|
Then:
Action is all_procedures_checked             |  | X|
Action is check_diagnoses                    | X|  |
# .......
# repeat until: all_procedures_checked

Attribute: lastProcedure     Type: Integer
Equals: len_procedures() - 1

rTable 1:
If:                                                    | 0| 1| 2| 3|
next diagnosis_nr in [0 - lastDiagnosis]               | Y| Y| N| N|
True = procedure_type in ["X", "Y"]                    | Y| -| Y| N|
'[Diagnosis, Procedure]-pair in Compatible Codes'      | Y| -| -| -|
procedure_type = "Z"                                   | -| Y| -| -|
'[Procedure, Diagnosis]-pair in inCompatible Codes'    | -| Y| -| -|
Then:
Act_Diagnosis is all_diagnoses_handled_type_XY         |  |  | X|  |
Act_Diagnosis is all_diagnoses_handled_type_other      |  |  |  | X|
Act_Diagnosis is no_report_and_break                   | X|  |  |  |
Act_Diagnosis is report_incompatible_pair              |  | X|  |  |
# .......
# repeat until: all_diagnoses_handled_type_XY,
#               all_diagnoses_handled_type_other,
#               no_report_and_break

Attribute: fn                Type: Integer
Attribute: True              Type: Integer
Equals: 1

Attribute: lastDiagnosis     Type: Integer
Equals: len_diagnoses() - 1

Attribute: procedure_type
Equals: get_procedure_type_from_claim(procedure_nr)

Attribute: procedure_code    Type: Text
Equals: get_procedure_code_from_claim(procedure_nr)

Attribute: diagnosis_code    Type: Text
Equals: get_diagnosis_code_from_claim(diagnosis_nr)

Proposition: '[Diagnosis, Procedure]-pair in Compatible Codes'
Obtain_instance_from_database_view: CompatibleCodes

Proposition: '[Procedure, Diagnosis]-pair in inCompatible Codes'
Obtain_instance_from_database_view: inCompatibleCodes

Attribute: collected_procedures_print   Type: Text
Equals: collected_procedures[2:] if collected_procedures != "" else "None"
# [2:] => skip first ", "
```

```
Attribute: collected_incompatible_pairs_print  Type: Text
Equals: collected_incompatible_pairs[2:] if collected_incompatible_pairs != "" else
"None"


Database_view: CompatibleCodes
With_attributes: code1, code2
Query:
SELECT *
  FROM CompatibleCodesPlus
 WHERE procedure_code = '%s' AND
       diagnosis_code = '%s'
 LIMIT 1
With_arguments: procedure_code, diagnosis_code


Database_view: inCompatibleCodes
With_attributes: code1, code2, code3, code4
Query:
SELECT *
  FROM inCompatibleCodes
 WHERE procedure_code_min <= '%s'
   AND procedure_code_max >= '%s'
   AND diagnosis_code_min <= '%s'
   AND diagnosis_code_max >= '%s'
 LIMIT 1
With_arguments: procedure_code, procedure_code, diagnosis_code, diagnosis_code



GOALATTRIBUTE: Action
Repeat_until: all_procedures_checked

Case: all_procedures_checked
Print: "-----------------------------------------------"
Print: "Procedures without compatible diagnoses: %s"   collected_procedures_print
Print: "Incompatible pairs: %s"
collected_incompatible_pairs_print
Print: "-----------------------------------------------"

Case: check_diagnoses
Print: "#REM# - "


GOALATTRIBUTE: Act_Diagnosis
Repeat_until: all_diagnoses_handled_type_XY, all_diagnoses_handled_type_other,
no_report_and_break

Case: all_diagnoses_handled_type_XY
>>: collected_procedures = collected_procedures + ", " + procedure_code

Case: all_diagnoses_handled_type_other
Print: "#REM# - "

Case: no_report_and_break
Print: "#REM# - "

Case: report_incompatible_pair
>>: collected_incompatible_pairs = collected_incompatible_pairs + ", " +
procedure_code + "-" + diagnosis_code
```

Output DT5GL/SQLite/PostgreSQL

```
SQLite_database: "Database/CSV files.db"

PS C:\Users\Administrator\dt5gl> .\dt.exe -source:Claim_JSON_v2.txt
-----------------------------------------------
Procedures without compatible diagnoses: 12001, 28005, 11000, 27675, 49657, 62319
Incompatible pairs: 28415-I10, 28415-J38.6, 28415-J38.3, 28415-H52.01, 28415-
H47.632, 28415-N64.81, 30420-I10, 30420-L76.01, 30420-N64.81
-----------------------------------------------
Time elapsed: 0:00:00.143317



PostgreSQL_database: "csv_files"

PS C:\Users\Administrator\dt5gl> .\dt.exe -source:Claim_JSON_v2.txt
-----------------------------------------------
Procedures without compatible diagnoses: 12001, 28005, 11000, 27675, 49657, 62319
Incompatible pairs: 28415-I10, 28415-J38.6, 28415-J38.3, 28415-H52.01, 28415-
H47.632, 28415-N64.81, 30420-I10, 30420-L76.01, 30420-N64.81
-----------------------------------------------
Time elapsed: 0:00:00.336505
```

Decision tables: variations on a theme.

The previous solutions are based on 2 decision tables, but they can also be merged into 1 table or split into multiple tables (with equal functionality, of course). Note that for all variations, the total number of columns in the tables is always 6.

1. merging into 1 table:

```
rTable 1:
If:                                                          | 0| 1| 2| 3| 4| 5|
next procedure_nr in [0 - lastProcedure]                     | Y| N| -| -| -| -|
next diagnosis_nr in [0 - lastDiagnosis]                     | -| -| Y| Y| N| N|
True = procedure_type in ["X", "Y"]                          | -| -| Y| -| Y| N|
True = CompatibleCodes(procedure_code, diagnosis_code)       | -| -| Y| -| -| -|
procedure_type = "Z"                                         | -| -| -| Y| -| -|
True = InCompatibleCodes(procedure_code, diagnosis_code)     | -| -| -| Y| -| -|
Then:
Action is all_procedures_checked                             |  | X|  |  |  |  |
Action is check_diagnoses                                    | X|  |  |  |  |  |
Act_Diagnosis is all_diagnoses_handled_type_XY               |  |  |  |  | X|  |
Act_Diagnosis is all_diagnoses_handled_type_other            |  |  |  |  |  | X|
Act_Diagnosis is no_report_and_break                         |  |  | X|  |  |  |
Act_Diagnosis is report_incompatible_pair                    |  |  |  | X|  |  |
# .......
# Action: repeat until: all_procedures_checked
#         Act_Diagnosis: repeat until: all_diagnoses_handled_type_XY,
#                                      all_diagnoses_handled_type_other,
#                                      no_report_and_break


Attribute: lastProcedure     Type: Integer
Equals: len_procedures() - 1

Attribute: lastDiagnosis     Type: Integer
Equals: len_diagnoses() - 1
```

2. split table 2 into 2 tables

```
rTable 0:
If:                                          | 0| 1|
next procedure_nr in [0 - lastProcedure]     | Y| N|
Then:
Action is all_procedures_checked             |  | X|
Action is check_diagnoses                    | X|  |
# .......
# repeat until: all_procedures_checked

Attribute: lastProcedure     Type: Integer
Equals: len_procedures() - 1

rTable 1:
If:                                                      | 0| 1|
next diagnosis_nr in [0 - lastDiagnosis]                 | Y| N|
True = procedure_type in ["X", "Y"]                      | Y| Y|
True = CompatibleCodes(procedure_code, diagnosis_code)   | Y| -|
Then:
Act_Diagnosis is all_diagnoses_handled_type_XY           |  | X|
Act_Diagnosis is no_report_and_break                     | X|  |
# .......
# repeat until: all_diagnoses_handled_type_XY, no_report_and_break


rTable 2:
If:                                                      | 0| 1|
next diagnosis_nr                                        | Y| N|[6]
True = procedure_type in ["X", "Y"]                      | -| N|
procedure_type = "Z"                                     | Y| -|
True = InCompatibleCodes(procedure_code, diagnosis_code) | Y| -|
Then:
Act_Diagnosis is all_diagnoses_handled_type_other        |  | X|
Act_Diagnosis is report_incompatible_pair                | X|  |
# .......
# repeat until: all_diagnoses_handled_type_other

Attribute: lastDiagnosis     Type: Integer
Equals: len_diagnoses() - 1
```

[6] When setting up this table, I intuitively used again the condition: next diagnosis_nr in [0 - lastDiagnosis], but that produces an error message: Error: attribute after 'next'-instruction in condition 1 in table 2 already exists. Optionally specify 'next attribute' without follow-up instructions.
In short, if an attribute is already provided with a range, this range must be omitted in subsequent tables.

3. Split 3 tables into 6 (but not recommended)

```
rTable 0a:
If:                                          | 0|
next procedure_nr in [0 - lastProcedure]     | N|
Then:
Action is all_procedures_checked             | X|
# .......
# repeat until: all_procedures_checked

rTable 0b:
If:                                          | 0|
next procedure_nr                            | Y|
Then:
Action is check_diagnoses                    | X|
# .......

Attribute: lastProcedure     Type: Integer
Equals: len_procedures() - 1


rTable 1a:
If:                                                      | 0|
next diagnosis_nr in [0 - lastDiagnosis]                 | N|
True = procedure_type in ["X", "Y"]                      | Y|
Then:
Act_Diagnosis is all_diagnoses_handled_type_XY           | X|
# .......
# repeat until: all_diagnoses_handled_type_XY

rTable 1b:
If:                                                      | 0|
next diagnosis_nr                                        | Y|
True = procedure_type in ["X", "Y"]                      | Y|
True = CompatibleCodes(procedure_code, diagnosis_code)   | Y|
Then:
Act_Diagnosis is no_report_and_break                     | X|
# .......
# repeat until: no_report_and_break


Attribute: lastDiagnosis     Type: Integer
Equals: len_diagnoses() - 1


rTable 2a:
If:                                                      | 0|
next diagnosis_nr                                        | N|
True = procedure_type in ["X", "Y"]                      | N|
Then:
Act_Diagnosis is all_diagnoses_handled_type_other        | X|
# .......
# repeat until: all_diagnoses_handled_type_other


rTable 2b:
If:                                                      | 0|
next diagnosis_nr                                        | Y|
procedure_type = "Z"                                     | Y|
True = InCompatibleCodes(procedure_code, diagnosis_code) | Y|
Then:
Act_Diagnosis is report_incompatible_pair                | X|
# .......
```