

EECS3342 (Sec. E) Fall 2025  
Lab4  
Specifying & Verifying Tabular Expressions

Chen-Wei Wang

**Release: Tuesday, November 18**  
**Due Date: 11:59 PM, Tuesday, December 2**

**©This document is not for public distribution.** This document may only be used by EECS3342 students registered at York University. By downloading this document from eClass, registered York students agree to keep this document private for their personal use.

If needed, use a **private** Github account to store your work. Consider [this tutorial series](#) on setting up a **private** Github repository.

- There **is** a submission required for this lab. While working as a group is allowed, you are encouraged to work on your own to begin with, so as to exercise all aspects of the assigned tasks.
- **To receive full marks of the lab, you must submit a valid file with the specified name, verbatim and case-sensitive.** You are responsible for any mark reduction due to not adhering to the submission instructions as specified in Section 6.
- A model solution will be released after the due date: you are advised to attempt the lab, and ask questions (e.g., office hours, any of the scheduled lab sessions), before looking at the solution.
- **[Texts in blue]** are hyperlinks to the corresponding documents/recordings.

# Contents

<b>1 Assumptions</b>	<b>3</b>
<b>2 Background: Introducing Tabular Expressions</b>	<b>4</b>
2.1 Function Tables: Syntax and Semantics . . . . .	4
2.2 Function Tables: Example . . . . .	5
<b>3 Background: A Problem of Self-Learning Problem</b>	<b>6</b>
<b>4 Task 1: Formalizing the Self-Learning Problem using Tables</b>	<b>6</b>
<b>5 Task 2: Verifying the Self-Learning Problem in Rodin</b>	<b>7</b>
<b>6 Submission</b>	<b>8</b>
<b>7 Amendments</b>	<b>9</b>

## Learning Outcomes

By completing the assigned exercises of this lab, you are expected to be able to:

1. Understand how input-output requirements are formulated using function tables.
2. Formalize given informal input-output requirements as function tables.
3. Encode rows and columns of function tables in Rodin.
4. Analyze the healthiness conditions (i.e., completeness and disjointness) of function tables.
5. Prove properties about the input-output requirements specified in function tables.

## 1 Assumptions

- You have already set up a way for backing up your work (e.g., a Github **private** repository such as **EECS3342-workspace**).
- You have a working web browser and a stable internet connection for accessing the Rodin tool through the EECS remote lab facility: <https://remotelab.eecs.yorku.ca/>.
- You have already familiarized yourself with the basic workflow of constructing a formal model (consisting of context and machine): [https://www.eecs.yorku.ca/~jackie/teaching/tutorials/index.html#rodin\\_intro](https://www.eecs.yorku.ca/~jackie/teaching/tutorials/index.html#rodin_intro).
- Here is a document summarizing the mathematical language of the Event-B modelling notation:

<https://wiki.event-b.org/images/EventB-Summary.pdf>

Navigate this document and find those concepts reviewed in your Week 2 and Week 3 lectures:

<https://www.eecs.yorku.ca/~jackie/teaching/lectures/2025/F/EECS3342/slides/01b-Review-of-Math.pdf>

For each relevant concept, study its syntax (written in either the math form or ASCII characters) and descriptions (e.g., formal definition).

## 2 Background: Introducing Tabular Expressions

### 2.1 Function Tables: Syntax and Semantics

Tabular expressions are a proven and effective approach to describing conditionals and relations, and they are thus ideal for documenting many system requirements. They are arguably easier to comprehend and to maintain than conventional mathematical expressions. For our purpose of capturing input-output requirements, tabular expressions of the form shown in Fig. 1 are appropriate. These tabular expressions are called *horizontal condition tables* (HCTs). The input domain is partitioned into condition rows in the left column(s), while rows in the right column(s), inside double borders, denote the corresponding output results. Rows in the input columns may be divided to specify sub-conditions.

		Result												
		<b>F</b>												
		Condition												
$C_1$	<table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 15%;"><math>C_{1.1}</math></td><td style="width: 15%;"><math>RES_1</math></td></tr> <tr> <td><math>C_{1.2}</math></td><td><math>RES_2</math></td></tr> <tr> <td>...</td><td>...</td></tr> <tr> <td><math>C_{1.m}</math></td><td><math>RES_m</math></td></tr> <tr> <td>...</td><td>...</td></tr> <tr> <td><math>C_n</math></td><td><math>RES_n</math></td></tr> </table>	$C_{1.1}$	$RES_1$	$C_{1.2}$	$RES_2$	...	...	$C_{1.m}$	$RES_m$	...	...	$C_n$	$RES_n$	<b>IF</b> $C_1$ <b>IF</b> $C_{1.1}$ <b>THEN</b> $F = RES_1$ <b>ELSEIF</b> $C_{1.2}$ <b>THEN</b> $F = RES_2$ ... <b>ELSEIF</b> $C_{1.m}$ <b>THEN</b> $F = RES_m$ <b>ELSEIF</b> ... <b>ELSEIF</b> $C_n$ <b>THEN</b> $F = RES_n$
$C_{1.1}$	$RES_1$													
$C_{1.2}$	$RES_2$													
...	...													
$C_{1.m}$	$RES_m$													
...	...													
$C_n$	$RES_n$													

Figure 1: Semantics of Horizontal Condition Table (HCT)

We may interpret the tabular structure in Fig. 1 as a list of “if-then-else” statements, without the sequence implications of the “if-then-else” construct. This is shown in the right part of the figure. Each row defines the input circumstances under which the output  $F$  is bound to a particular result value. For example, the first row corresponds to the predicate ( $C_1 \wedge C_{1.1} \Rightarrow F = RES_1$ ), and so on.

In documenting input-output behaviours using HCTs as illustrated in Fig. 1, we need to reason about their *completeness* and *disjointness*.

- *Completeness* ensures that there is an output specified for every combination of inputs: the rows cover all input combinations.

$$\begin{aligned} C_1 \vee C_2 \vee \dots \vee C_n &\equiv \text{TRUE} \\ C_{1.1} \vee C_{1.2} \vee \dots \vee C_{1.m} &\equiv C_1 \end{aligned}$$

- *Disjointness* ensures that the rows do not overlap.

$$\forall i, j \bullet i \in \{1, 2, \dots, n\} \wedge j \in \{1, 2, \dots, n\} \Rightarrow (i \neq j \Rightarrow \neg(C_i \wedge C_j))$$

Similar constraints apply to the sub-conditions, if any. For example:

$$\forall i, j \bullet i \in \{1, 2, \dots, m\} \wedge j \in \{1, 2, \dots, m\} \Rightarrow (i \neq j \Rightarrow \neg(C_{1.i} \wedge C_{1.j}))$$

## 2.2 Function Tables: Example

As an example, we consider the requirement for an intended function *LIMITS\_ALARM*. An alarm monitors the quantity of some input variable  $X$  (e.g., temperature of a nuclear power plant), subject to a low limit  $L$  and a high limit  $H$ , with a hysteresis band of size  $\text{EPS}$ . As you can imagine, when the monitored quantity falls outside the limit, e.g., when the temperature has become too high, some measure (e.g., shut down the power plant) must be imposed immediately.

- We first consider the declaration of the alarm's inputs and outputs:

```
(* DECLARATION *)
+-----+
| LIMITS_ |
| ALARM    |
REAL--|H      QH|--BOOL
REAL--|X      Q|--BOOL
REAL--|L      QL|--BOOL
REAL--|EPS    |
+-----+
INPUT VARIABLES
H : REAL; (* High limit      *)
X : REAL; (* Variable value   *)
L : REAL; (* Lower limit      *)
EPS : REAL; (* Hysteresis      *)
OUTPUT VARIABLE
QH : BOOL; (* High flag      *)
Q : BOOL; (* Alarm output   *)
QL : BOOL; (* Low flag       *)
```

Figure 2: *LIMITS\_ALARM* declaration

- In Figure 3, we consider the alarm's (informal) expected input-output behaviour, as well as its (formal) tabular requirement (which constrains the relation between inputs  $X, H, L, \text{EPS}$  and outputs  $Q, QH, QL$ ). It is expected that the input  $X$  is sampled periodically, and the values of  $Q, QH$ , and  $QL$  should be updated accordingly. The input-output requirement is captured in the three function tables. We use *NC* to denote “No Change”, e.g., the value of variable  $QH$  is equal to its value from the last sampling point.

When variable value  $X$  exceeds the high limit  $H$ , the high flag  $QH$  becomes *TRUE*. Symmetrically, when  $X$  goes below the low limit  $L$ , the low flag  $QL$  becomes *TRUE*. Both flags  $QH$  and  $QL$  are set to *FALSE* when  $X$  is in the exclusive range of  $(L + \text{EPS}, H - \text{EPS})$ . There exists a hysteresis band for the high limit inside which the value of  $QH$  remains unchanged:  $[H - \text{EPS}, H]$ . Symmetrically, there exists a hysteresis band for the low limit:  $[L, L + \text{EPS}]$ . Finally, the alarm output  $Q$  is set to *TRUE* if and only if either of the flags is set to *TRUE*, and  $Q$  is set to *FALSE* otherwise.

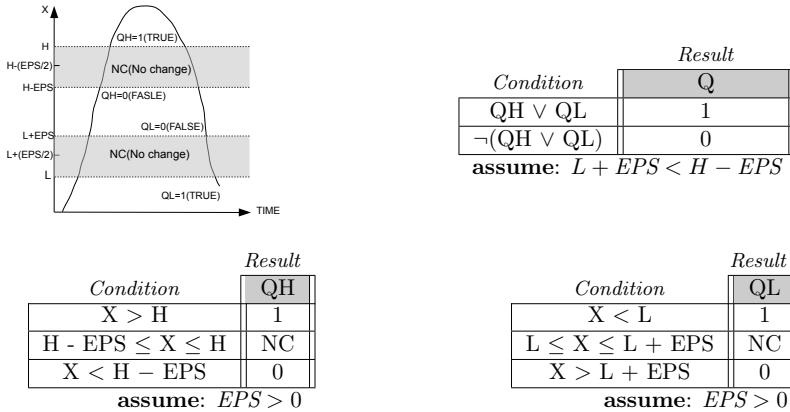


Figure 3: *LIMITS\_ALARM* requirement in tabular expression

- **Exercise.** Justify that the input-output requirements of *LIMITS\_ALARM*, as formulated in the three function tables, are both *complete* and *disjoint*.

### 3 Background: A Problem of Self-Learning Problem

Consider a working professional who wishes to start an initiative of continuous learning, to be accomplished outside their day-time work hours. This working professional wonders what study pattern might suit them the best.

The working professional starts by considering any 7-day period in the year in which Saturday might fall on the first day or on one of the other 6 days. There are two possible patterns.

- If the first day is a Saturday, then following *Pattern 1* would mean that the working professional will study all 7 days. Otherwise, if Saturday is one of the other days, then they will rest on Saturday and study on the other 6 days.
- If the last day is a Saturday, then following *Pattern 2* would mean that the working professional will study all 7 days. Otherwise, if Saturday is one of the other days, then they will rest on Saturday and study on the other 6 days.

*Problem.* What conclusions can one draw from the above two pattern rules? Specifically:

*Is there any 7-day period in which following either pattern would entail studying for all 7 days?*

How would we prove such a constraint for all possible 7-day chunks of time?

### 4 Task 1: Formalizing the Self-Learning Problem using Tables

Using the tabular notation, formulate the requirements described in Section 3.

- Assume a set  $W = \{1, 2, 3, 4, 5, 6, 7\}$  which represents a seven-day period, where 1 denotes the first day, 7 denotes the last day, and so on.
- There is only one input variable  $s$ , whose type is  $s \in W$ , which denotes where the Saturday falls onto  $W$ .
- There are three output variables:
  - $P_1 \in \mathbb{P}(W)$ : a set of study days according to *pattern1*
  - $P_2 \in \mathbb{P}(W)$ : a set of study days according to *pattern2*
  - $(D_1, D_2) \in \mathbb{N} \times \mathbb{N}$ : a pair of numbers, where  $D_1$  and  $D_2$  denote, respectively, the numbers of study days according to *pattern1* and *pattern2*

You are required to submit a one-page document `Lab5_table.pdf` which contains:

1. Your name (Last name, First name) and student number
2. A single function table
3. Brief written justifications on why the table is both *complete* and *disjoint*
4. A stated conclusion, drawn from the table, that answers the question raised in Section 3

**Hints:** Your function table should resemble that in Figure 1 (page 4), where  $C_1 \dots C_n$  are input conditions and should be instantiated to proper Boolean conditions. Output values should be specified in the three columns  $P_1$ ,  $P_2$ , and  $(D_1, D_2)$ .

## 5 Task 2: Verifying the Self-Learning Problem in Rodin

- Create a new Rodin project `SelfLearningPatterns`.
- Create a context `c0`.
- Create the following constants:

– <code>W</code>	[ seven-day period ]
– <code>P1</code>	[ set of study days according to <i>Pattern1</i> ]
– <code>P2</code>	[ set of study days according to <i>Pattern2</i> ]
– <code>s</code>	[ where Saturday falls onto a given seven-day period ]
– <code>P1.days</code>	[ number of study days according to <i>Pattern1</i> ]
– <code>P2.days</code>	[ number of study days according to <i>Pattern2</i> ]

- Write the following axioms (i.e., `not theorem`) with the specified labels:

- `axm1`: `W` is the interval ranging from 1 to 7.
- `axm2`: Saturday can fall onto anywhere in a given seven-day period.  
Your answer should involve both `s` and `W`.
- `axm3`: The number of days in a given seven-day period, when excluding the Saturday, should be six.  
Your answer should involve both `s` and `W`.
- `axm4`: `P1` can include some or all days in `W`.
- `axm5`: `P2` can include some or all days in `W`.
- `axm6`: `P1.days` is typed as a natural number.
- `axm7`: `P2.days` is typed as a natural number.
- `axm8`: `P1.days` should match the number of members in `P1`.
- `axm9`: `P2.days` should match the number of members in `P2`.
- `table.r1`: What are values of `P1` and `P2` when Saturday is the first day of a given seven-day period?
- `table.r2`: What are values of `P1` and `P2` when Saturday is neither the first nor the last day of a given seven-day period?
- `table.r3`: What are values of `P1` and `P2` when Saturday is the last day of a given seven-day period?

**Hint.** For each of `table.r1` to `table.r3`, use a logical implication.

- `property_completeness`: Do the three antecedents of `table.r1` to `table.r3` cover all possible conditions of the input variable `s`?
- `property_disjointness`: Are the three antecedents of `table.r1` to `table.r3` mutually disjoint?
- `conclusion.1a`: When Saturday is the first day of a given seven-day period, it is not the case that both study patterns require seven days of study.
- `conclusion.1b`: When Saturday is neither the first nor the last day of a given seven-day period, it is not the case that both study patterns require seven days of study.
- `conclusion.1c`: When Saturday is the last day of a given seven-day period, it is not the case that both study patterns require seven days of study.

**Hint.** For each of `conclusion.1a` to `conclusion.1c`, use an implication, and your answer should involve `s`, `P1.days`, and `P2.days`.

- `conclusion.2`: Despite where Saturday might fall into, it is not the case that both study patterns require seven consecutive days of study.

**Hint.** For `conclusion.2`, there is no need to use an implication, and your answer should involve `P1.days` and `P2.days` only.

## 6 Submission

You are required to submit both a PDF file **Lab5\_table.pdf** (Task 1) an archive file **SelfLearningPatterns.zip** exported from the Rodin project (Task 2).

1. Before you submit, you must make sure that there are no errors reported by the Rodin IDE.

Submitting models with errors (meaning that it cannot be run for grading) will result in possible partial, but low, marks.

2. You are required to submit a Rodin project archive file (.zip) consisting all subfolders.

In Rodin:

1. Right click on project **SelfLearningPatterns**.  
Then click **Export**
2. Under General, choose **Archive File**.

3. Check the top-level **SelfLearningPatterns**

Under **To archive file**: browse to, e.g., desktop, and save it as **SelfLearningPatterns.zip** (**case-sensitive**)  
Then **Finish**.

3. Click on the following link (for which you will be prompted to enter your EECS account login credentials):

<https://webapp.eecs.yorku.ca/submit/?acadyear=2025-26&term=F&course=3342&assignment=Lab4>

- You **must** login into the web submit page using your EECS login credentials (otherwise, your submitted folder on the EECS server may not be identified properly):

The screenshot shows a login form titled "Web Submit Login". It contains two main sections: one for "Passport York" login and one for "EECS account" login. The "EECS account" section is highlighted with a red box and includes fields for "EECS Username" and "EECS Password", along with a "Login" button.

Note. If you are prompted for your PPY login instead, then it might be due to an earlier login session. In this case, login first with your PPY account credentials, then log out. Then, clicking on the above submission link should lead you to the login page for EECS account credentials.

- Ensure that the correct academic year, term, course, and assignment are chosen. Then, **browse to the PDF file **Lab5\_table.pdf** archive file **SelfLearningPatterns.zip** and click on **Submit Files****.
- You may upload as many draft versions as you like before the deadline.

## **7 Amendments**

Clarifications or corrections will be added to this section.

-