实验 3　因果图法与旅馆系统用例设计

1. 实验目标

(1) 能够依据需求分析原因和结果。
(2) 能够画出因果图,标注相应关系符号。
(3) 能够将因果图转化判定表。
(4) 能够使用因果图法进行用例设计。

2. 背景知识

(1) 需求：某旅馆住宿系统可为游客办理房间选定、房间支付及房间管理相关业务。其需求描述如下：当支付房间全款(即预期入住天数内所有房款)或支付房间房款不足(仅支付定金),选择“单人间”、“双人间”或“豪华间”,若该类型房间有空房,则相应类型的房间被开启;若该类型房间无空房,则“房间已满”提示灯亮。此时,支付房款不足的游客选择该类型的房间,则该类型房间不被开启且提示办理退款;若此期间,该房间类型有客人退房,则“房间已满”提示灯灭,该类型房间的某间房被开启的同时提醒游客房款不足。

(2) 界面原型：如图 3.1 所示。

图 3.1　旅馆住宿系统业务办理页面

首先,基于上述需求,请思考采用等价类划分法如何进行用例设计。读者采用等价类划分法进行上述需求的用例设计时,不难发现,设计出的测试用例存在如下特点。其一,数量甚少。其二,仅着重考虑了各项输入条件,并未考虑到输入条件的各种组合情况。例如选择不同的房款支付方式及房间类型,在“房间已满”和“房间空余”不同前提下,产生的结果会有所差异,此情况便未于等价类划分法中涉及和考虑。其三,未考虑到各输入情况之间的相互制约关系。例如“支付房款”与“支付定金”不能同时成立,最多仅能成立一个;选择“单人间”、“双人间”和“豪华间”不能同时成立,最多仅能成立一个,等等,上述列举的两种情况亦未于等价类划分法中涉及和考虑。

再如某保险公司的预约投保系统，界面原型如图 3.2 所示。读者针对此界面原型采用等价类划分法进行用例设计时，同样会忽略多种关系的存在。如“称谓”字段中“先生”与“女士”不能同时成立，最多仅能成立二者之一；“所在地”字段中，当“某市”出现时，“该市所属省份”必须也同步出现，决不应前者（市）出现而后者（市所属省份）不出现的情况发生；“联系电话”字段中，“固定电话”、“小灵通”及“手机号”至少要有一个被填写即可等。

图 3.2　预约投保系统页面

读者充分理解了上述两实例后，则不难理解仅采用等价类法并不能很好地处理“当系统中输入项之间以及输入项与输出之间存在多种关系”时的测试用例设计问题，而恰恰这正是因果图法引入的主要原因。

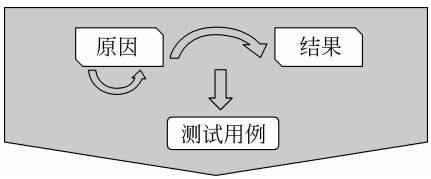


图 3.3　因果图法介绍

因果图法是从需求中找出因（输入条件）和果（输出或程序状态的改变），通过分析输入条件之间的关系（组合关系、约束关系等）及输入和输出之间的关系绘制出因果图，再转化成判定表，从而设计出测试用例的方法，如图 3.3 所示。不难理解，该方法主要适用于各种输入条件之间存在某种相互制约关系或输出结果依赖于各种输入条件的组合时的情况。

多次提及“因果图”，究竟“图”为何模样？如图 3.4 所示。

读者易于发现，图 3.4 中无法识别的符号甚多，例如 E、V 等。以下结合图 3.5 所示，就因果图中的常用符号含义进行作答。

因果图符号种类繁多，结合常用符号解释如下：

（1）CI：原因。

（2）EI：结果。

（3）恒等：原因结果同时出现。

（4）非～：原因出现，结果不出现；原因不出现，结果出现。

（5）或∨：原因 1 个出现，结果就出现；原因都不出现，结果就不出现。

（6）且∧：原因都出现，结果才出现。

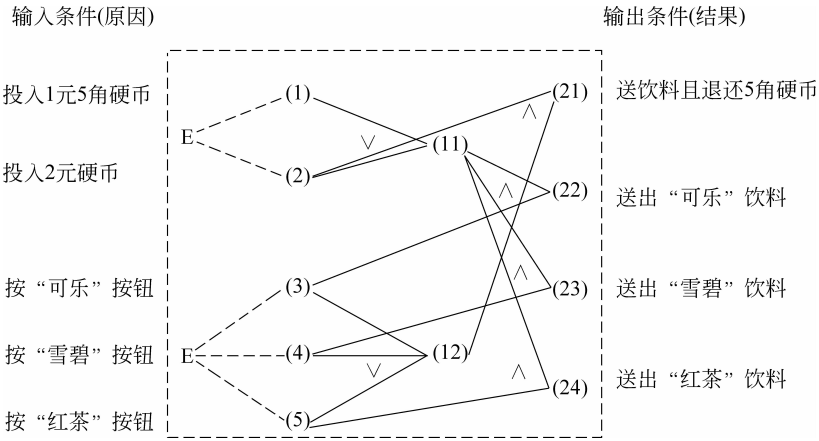


图 3.4　因果图初识

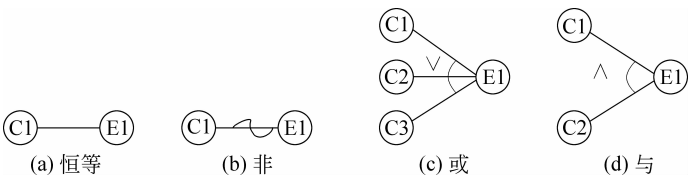


图 3.5　因果图符号

注意：

① 图 3.5 中所示的每个结点，表示状态。

② 图 3.5 中所示的 I 取“0”表示状态不出现，“1”表示状态出现，若有多状态，可取大于 1 的多个值表示。

为了表示原因与原因之间、结果与结果之间可能存在的约束条件，因果图中还附加一些表示约束条件的符号，以下结合图 3.6 所示，就因果图中的约束符号含义进行作答。

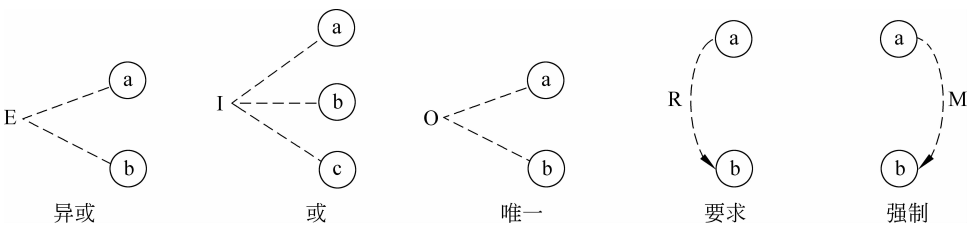


图 3.6　因果图符号_约束条件

约束符号亦包含多种类型，据“从输入考虑”和“从输出考虑”两方面进行归类如下。

(1) 从输入考虑。

① E(互斥/异或)：表示 ab 两原因不会同时成立，最多一个能成立。

② I(包含)：abc 三个原因中至少有一个必须成立。

③ O(唯一)：ab 当中必须有一个，且仅有一个成立。

④ R(要求)：当 a 出现时，b 必须也出现，不可能 a 出现 b 不出现。

(2) 从输出考虑。

M(强制或屏蔽)：a 是 1 时，b 必须是 0；a 是 0 时，b 的值不定。

基于上述因果图的常见重要符号的含义介绍，回过头来针对上文中的“某保险公司的预约投保系统”进行分析，结果如图 3.7 所示。

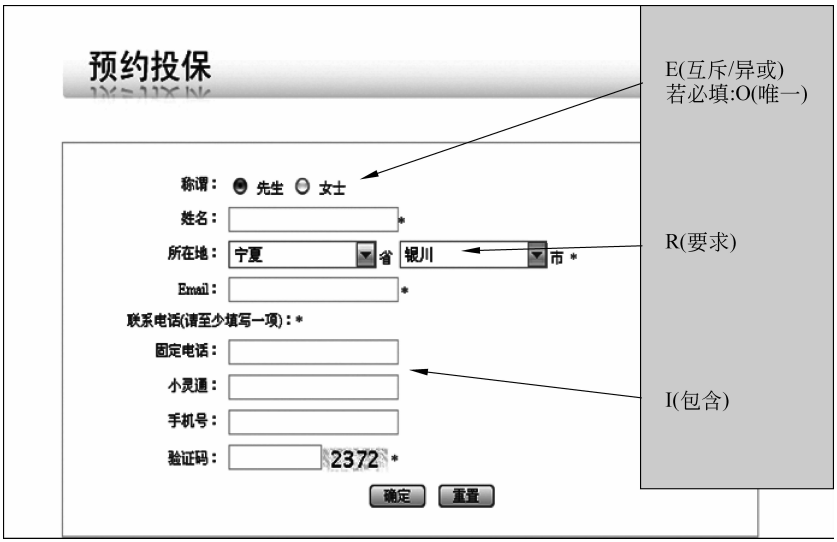


图 3.7　预约投保系统页面_关系分析

到目前为止，以上知识强调了因果图的重要符号的含义，读者或多或少对因果图的符号有了一定了解。究竟如何来使用因果图法呢？可参照如下步骤进行测试用例设计。

(1) 分析需求，提取原因和结果，并赋予标识符。

(2) 分析需求，提取因果关系，并表示成“因果图”。

(3) 标明因果图中约束条件。

(4) 因果图转换成判定表。

(5) 为判定表中每一列表示的情况设计测试用例。

注意：原因常常是输入条件或输入条件的等价类；结果常常是输出条件。

以下实验，以旅馆住宿系统为例，针对忽略房间状态和考虑房间状态两种不同的需求情况，结合上述因果图法的开展步骤进行具体方法应用的介绍。

3. 实验任务

任务 1：旅馆住宿系统测试用例设计(忽略房间状态)。

(1) 需求：某旅馆住宿系统可为游客办理房间选定、房间支付及房间管理相关业务，此系统默认房间资源始终保持充足的状态。其需求描述如下：当支付房间全款(即预期入住天数内所有房款)或支付房间房款不足(仅支付定金)，选择“单人间”、“双人间”或“豪华间”，则相应类型的房间被开启。若游客支付房款不足，则在开启房门的同时系统提示房款支付不足。

(2) 界面原型：如图 3.8 所示。

(3) 问题：采用因果图法进行测试用例设计。

图 3.8　旅馆住宿系统业务办理页面(忽略房间状态)

前提条件：经分析需求和界面原型，可发现该需求的输入项与输入项之间以及输入项与结果之间存在多种关系，处理此种情况采用因果图法更为合适。

第 1 步：分析需求说明，找出原因(即输入)和结果(即输出)。

原因：

1. 游客支付房间全款

2. 游客支付房款不足

3. 游客选择“单人间”

4. 游客选择“双人间”

5. 游客选择“豪华间”

结果：

21. 该类型房间被打开且提醒房款支付不足

22. 某“单人间”被打开

23. 某“双人间”被打开

24. 某“豪华间”被打开

第 2 步：画出因果图，并标注相应关系符号，如图 3.9 所示。在图 3.9 中所有原因结点显示于左侧，所有结果结点显示于右侧，并建立中间结点以表示处理的中间状态。

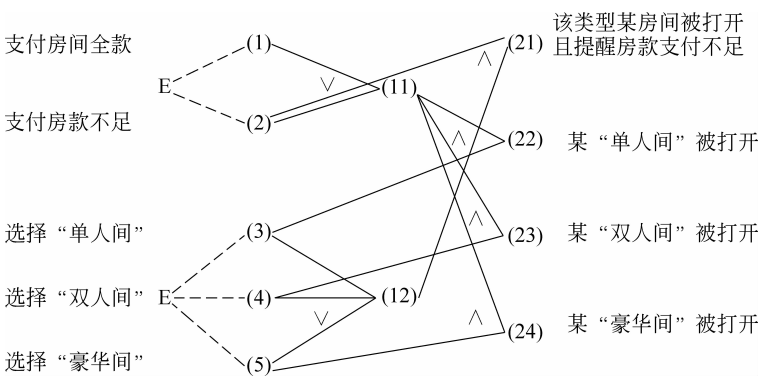


图 3.9　业务办理_因果图(忽略房间状态)

中间结点：

11. 已支付房款

12. 已选择房间类型

注意：中间结点的设立并非必须要完成的工作，但是它的设立可使绘制出的因果图显示起来更简单和美观，阅读起来也较为方便。

第 3 步：转换成判定表，如表 3.1 所示。

表 3.1　业务办理_判定表（忽略房间状态）

输入条件	游客支付房间全款	(1)	1	1	1	1	0	0	0	0	0	0	0
	游客支付房款不足	(2)	0	0	0	0	1	1	1	1	0	0	0
	游客选择“单人间”	(3)	1	0	0	0	1	0	0	0	1	0	0
	游客选择“双人间”	(4)	0	1	0	0	0	1	0	0	0	1	0
	游客选择“豪华间”	(5)	0	0	1	0	0	0	1	0	0	0	1
中间结果	已支付房款	(11)	1	1	1	1	1	1	1	1	0	0	0
	已选择房间类型	(12)	1	1	1	0	1	1	1	0	1	1	1
输出结果	该类型房间被打开且提醒房款支付不足	(21)	0	0	0	0	1	1	1	0	0	0	0
	某“单人间”被打开	(22)	1	0	0	0	1	0	0	0	0	0	0
	某“双人间”被打开	(23)	0	1	0	0	0	1	0	0	0	0	0
	某“豪华间”被打开	(24)	0	0	1	0	0	0	1	0	0	0	0
测试用例			Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y

第 4 步：在判定表中，可根据显示的 11 列作为确定测试用例的依据。设计测试用例如表 3.2 所示。

表 3.2　业务办理_测试用例设计（忽略房间状态）

编号	输　　入	预 期 结 果
1	游客支付房间全款，选择“单人间”	某“单人间”被打开
2	游客支付房间全款，选择“双人间”	某“双人间”被打开
3	游客支付房间全款，选择“豪华间”	某“豪华间”被打开
4	游客支付房间全款，未选择任何类型的房间	所有房间均不被打开
5	游客支付房款不足，选择“单人间”	某“单人间”被打开且系统提醒房款支付不足
6	游客支付房款不足，选择“双人间”	某“双人间”被打开且系统提醒房款支付不足
7	游客支付房款不足，选择“豪华间”	某“豪华间”被打开且系统提醒房款支付不足
8	游客支付房款不足，未选择任何类型的房间	所有房间均不被打开
9	游客不进行支付，选择“单人间”	所有房间均不被打开
10	游客不进行支付，选择“双人间”	所有房间均不被打开
11	游客不进行支付，选择“豪华间”	所有房间均不被打开

任务 2：旅馆住宿系统测试用例设计(考虑房间状态)。

(1) 需求：某旅馆住宿系统可为游客办理房间选定、房间支付及房间管理相关业务。其需求描述如下：当支付房间全款(即预期入住天数内所有房款)或支付房间房款不足(仅支付定金)，选择“单人间”、“双人间”或“豪华间”，若该类型房间有空房，则相应类型的房间被开启；若该类型房间无空房，则“房间已满”提示灯亮。此时，支付房款不足的游客选择该类型的房间，则该类型房间不被开启且提示办理退款；若此期间，该房间类型有客人退房，则“房间已满”提示灯灭，该类型房间的某间房被开启的同时提醒游客房款不足。

(2) 界面原型：如图 3.10 所示。

图 3.10　旅馆住宿系统业务办理页面(考虑房间状态)

(3) 问题：采用因果图法进行测试用例设计。

前提条件：经分析需求和界面原型，可发现该需求的输入项与输入项之间以及输入项与结果之间存在多种关系，处理此种情况采用因果图法更为合适。

第 1 步：分析需求说明，找出原因(即输入)和结果(即输出)。

原因：

1. 该类型房间有空房
2. 游客支付房款不足
3. 游客支付房间全款
4. 游客选择“单人间”
5. 游客选择“双人间”
6. 游客选择“豪华间”

结果：

21. 该类型房间“房间已满”灯亮
22. 提示办理退款
23. 提醒房款不足
24. 某“单人间”被打开
25. 某“双人间”被打开
26. 某“豪华间”被打开

第 2 步：画出因果图，并标注相应关系符号，如图 3.11 所示。在图 3.11 中所有原因结点显示于左侧，所有结果结点显示于右侧，并建立中间结点以表示处理的中间状态。

中间结点：

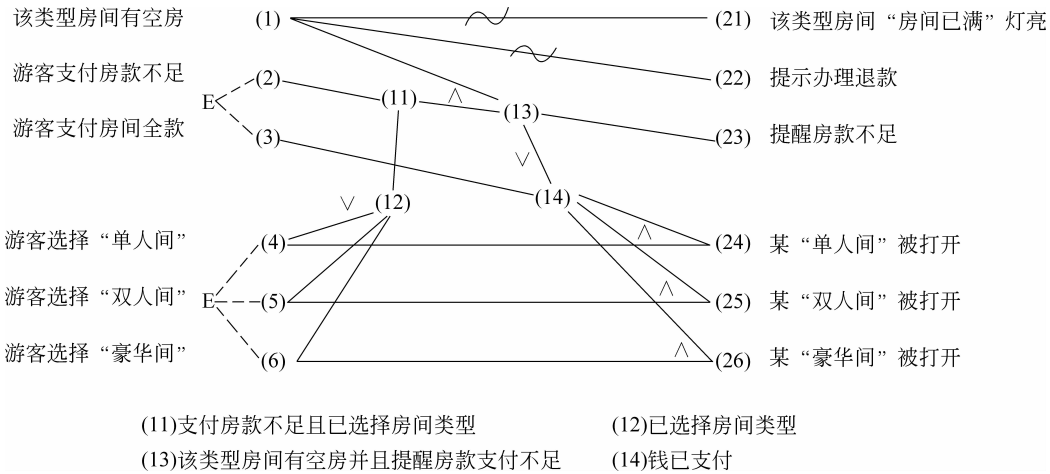
11. 支付房款不足且已选择房间类型

12. 已选择房间类型

13. 该类型房间有空房并且提醒房款支付不足

14. 钱已支付

注意：中间结点的设置并非必须要完成的工作，但是设立中间结点可使绘制出的因果图显示更简单和美观，阅读起来也较为方便。



(11)支付房款不足且已选择房间类型　　(12)已选择房间类型

(13)该类型房间有空房并且提醒房款支付不足　　(14)钱已支付

图 3.11　业务办理_因果图(考虑房间状态)

第 3 步：转换成判定表，如表 3.3 所示。

表 3.3　业务办理_判定表(考虑房间状态)

输入条件	(1)	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	(2)	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	(3)	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0
	(4)	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0
	(5)	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0
	(6)	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
输出结果	(21)										0	0	0				0		0	0	0				0		0	0	0				0
	(22)										0	0	0				0		0	0	0				0		0	0	0				0
	(23)										1	1	0				1		0	0	0				0		0	0	0				0
	(24)										1	0	0				0		1	0	0				0		0	0	0				0
	(25)										0	1	0				0		0	1	0				0		0	0	0				0
	(26)										0	0	0				1		0	0	0				1		0	0	0				0
测试用例											Y	Y	Y				Y		Y	Y	Y				Y		Y	Y	Y				Y

续表

输入条件	(1)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	(2)	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	(3)	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0
	(4)	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0
	(5)	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0
	(6)	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
输出结果	(21)										1	1	1				1		1	1	1				1		1	1	1				1
	(22)										1	1	0				1		1	1	1				1		0	0	0				0
	(23)										0	0	0				0		0	0	0				0		0	0	0				0
	(24)										0	0	0				0		0	0	0				0		0	0	0				0
	(25)										0	0	0				0		0	0	0				0		0	0	0				0
	(26)										0	0	0				0		0	0	0				0		0	0	0				0
测试用例											Y	Y	Y				Y		Y	Y	Y				Y		Y	Y	Y				Y

注意：

① 读者于第 3 步转化判定表时，通过分析，可先将违反约束条件的组合省略，再列出上表，则可大大减轻工作量。此实例组合项较多，避免讲解不充分，特将所有组合进行全部列举。

② 表 3.3 和续表 3.3 中未列出“中间结点”的取值情况，读者可自行列举。

第 4 步：在判定表中，阴影部分表示因违反约束条件而不可能出现的情况，故不对此进行用例设计。最后，剩余的 24 列将作为确定测试用例的依据。设计测试用例如表 3.4 所示。

表 3.4　业务办理_测试用例设计（考虑房间状态）

编号	输　　入	预 期 结 果
1	游客支付房款不足，选择“单人间”且有空房	某“单人间”被打开且系统提醒房款不足
2	游客支付房款不足，选择“双人间”且有空房	某“双人间”被打开且系统提醒房款不足
3	游客支付房款不足，未选择任何类型的房间	所有房间均不被打开且“房间已满”灯为灭的状态
4	游客支付房款不足，选择“豪华间”且有空房	某“豪华间”被打开且系统提醒房款不足
5	游客支付全款，选择“单人间”且有空房	某“单人间”被打开
6	游客支付全款，选择“双人间”且有空房	某“双人间”被打开
7	游客支付全款，未选择任何类型的房间	所有房间均不被打开且“房间已满”灯为灭的状态
8	游客支付全款，选择“豪华间”且有空房	某“豪华间”被打开

续表

编号	输　入	预期结果
9	游客不进行支付,选择“单人间”且有空房	所有房间均不被打开且“房间已满”灯为灭的状态
10	游客不进行支付,选择“双人间”且有空房	所有房间均不被打开且“房间已满”灯为灭的状态
11	游客不进行支付,未选择任何类型的房间且有空房	所有房间均不被打开且“房间已满”灯为灭的状态
12	游客不进行支付,选择“豪华间”且有空房	所有房间均不被打开且“房间已满”灯为灭的状态
13	游客支付房款不足,选择“单人间”且没有空房	“房间已满”灯为亮的状态且系统提示办理退款
14	游客支付房款不足,选择“双人间”且没有空房	“房间已满”灯为亮的状态且系统提示办理退款
15	游客支付房款不足,未选择任何类型的房间	“房间已满”灯为亮的状态
16	游客支付房款不足,选择“豪华间”且没有空房	“房间已满”灯为亮的状态且系统提示办理退款
17	游客支付全款,选择“单人间”且没有空房	“房间已满”灯为亮的状态且系统提示办理退款
18	游客支付全款,选择“双人间”且没有空房	“房间已满”灯为亮的状态且系统提示办理退款
19	游客支付全款,未选择任何类型的房间	“房间已满”灯为亮的状态
20	游客支付全款,选择“豪华间”且没有空房	“房间已满”灯为亮的状态且系统提示办理退款
21	游客不进行支付,选择“单人间”且没有空房	所有房间均不被打开且“房间已满”灯为亮的状态
22	游客不进行支付,选择“双人间”且没有空房	所有房间均不被打开且“房间已满”灯为亮的状态
23	游客不进行支付,未选择任何类型的房间且没有空房	所有房间均不被打开且“房间已满”灯为亮的状态
24	游客不进行支付,选择“豪华间”且没有空房	所有房间均不被打开且“房间已满”灯为亮的状态

注意：需求中描述“当房间没有空余时,‘房间已满’灯亮”。但是,读者会发现界面原型中并未见“灯”的显示。在此,值得一提的是,真实项目中界面原型可能会采用多种其他方式来实现需求说明中的要求,所采取的表现方式或许更加易于理解和使用。但是,建议开发方人员在界面原型确定后,及时同客户进行沟通并确认,便于后续工作在此基础上顺利开展。

思考：若此处采用等价类划分法设计用例,又该如何考虑呢?

4. 拓展练习

(1) 请采用因果图法针对如下需求进行测试用例设计。

需求：输入的第一个字符必须是＃或＊,第二个字符必须是一个数字,此情况下进行文件的修改;若第一个字符不是＃或＊,则给出信息 N,若第二个字符不是数字,则给出信

息 M。

(2) 请采用因果图法针对如下需求进行测试用例设计。

需求：有一个处理单价为 5 角钱的饮料的自动售货机软件测试用例的设计。其规格说明如下：若投入 5 角钱或 1 元钱的硬币，按下“橙汁”或“啤酒”按钮，则相应的饮料就送出来。若售货机没有零钱找，则一个显示“零钱找完”的红灯亮，这时在投入 1 元硬币并按下按钮后，饮料不送出来而且 1 元硬币也退出来；若有零钱找，则显示“零钱找完”的红灯灭，在送出饮料的同时退还 5 角硬币。

实验 4　决策表法与旅馆系统用例设计

1. 实验目标

(1) 理解决策表法内涵。

(2) 能够使用决策表法进行用例设计。

(3) 能够在真实项目中灵活采用决策表法。

2. 背景知识

读者已知晓,因果图法设计测试用例的步骤如下:

(1) 分析需求,提取原因和结果,并赋予标识符;

(2) 分析需求,提取因果关系,并表示成“因果图”;

(3) 标明因果图中约束条件;

(4) 因果图转换成判定表;

(5) 为判定表中每一列表示的情况设计测试用例。

显然,第(4)步“因果图转换成判定表”中已使用了判定表。判定表又称作决策表,为决策表法的核心,是分析和表达多逻辑条件下执行不同操作情况的有效工具。因此,决策表法是一种能够将复杂逻辑关系和多条件组合情况表达得较为明确的方法,适用于程序中输入输出较多或输入与输出之间相互制约条件较多的情况。综合所有黑盒测试方法来讲,基于决策表法的测试是最严格,最具有逻辑性的。

决策表法如此重要,何为决策表则显得尤为关键? 通过表 4.1 所示实例加以说明。

表 4.1　决策表初识

		1	2	3	4	5	6	7	8
问题	是否劳累?	Y	Y	Y	Y	N	N	N	N
	是否喜欢?	Y	Y	N	N	Y	Y	N	N
	是否难理解?	Y	N	Y	N	Y	N	Y	N
建议	重听一遍					√			
	继续进行						√		
	进行下一题							√	√
	休息	√	√	√	√				

不难理解,决策表能够将看似复杂的问题依据各种可能的情况进行全部罗列,简明且无遗漏。同理可悟出,在软件测试中,利用决策表法也应能够设计出完整的测试用例集合。

随后,将表 4.1 抽象为图 4.1 所示的决策表模型图。

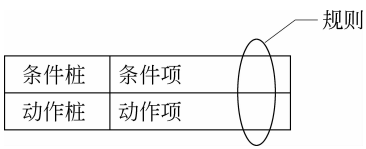


图 4.1　决策表模型图

可见，图 4.1 中包含了条件桩、动作桩、条件项和动作项 4 项元素，简要解释如下。

(1) 条件桩：为问题的所有条件的集合，包含了各种条件，其中各条件次序无严格限制。

(2) 条件项：为问题的所有条件的各种取值的集合，包含了左侧条件桩中各种条件的各种取值的组合，其中各条件次序无严格限制。

(3) 动作桩：为问题的所有可采取操作的集合，包含了各种可采取的操作，其中各操作次序无严格限制。

(4) 动作项：为针对条件项的各种组合的取值情况下，应该采取的对应操作。

值得提醒的是，图 4.1 中所示的任何一个条件组合的特定取值及其相应要执行的操作称为规则。

不难理解，决策表法实质为直接把测试输入中所有可能的情况进行组合，并汇总所对应的操作结果，这即为决策表法之优势所在。显然，利用决策表法能够设计出各种组合类型的完整测试用例集合。但不得不承认，决策表法并非十全十美，它不能表达重复执行的动作，如循环结构等。因此，读者应辩证地看待该方法。

至此，读者已对决策表法内涵有了相关见解，如何使用决策表法成为下一步要研究的重点。读者可参照如下步骤进行测试用例设计。

(1) 列出所有的条件桩和动作桩；

(2) 确定规则的个数；

(3) 填入条件项；

(4) 填入动作项；

(5) 简化决策表，合并类似规则或相同动作。

注意：

① 针对“确定规则个数”，值得提醒的是，若某决策表中有 n 个条件，且每个条件可取真、假两种值，则共有 2^n 条规则；若某决策表中有 n 个条件，且每个条件可取 1、2、3、…、m 种值，则共有 x^n 条规则。

② 针对决策表的简化过程，值得提醒以下两点：其一，若表中有两条或两条以上的规则具有相同的操作，且在条件项之间存在较为类似的关系，则可进行规则合并；其二，规则合并后得到的条件项用符号“-”表示，代表执行的动作与该条件的取值无关，即称为无关条件。

就目前而言，读者已从理论层面上认识了决策表法，以下实验，将以旅馆住宿系统及经典的 NextDate 函数为例，结合上述决策表法的开展步骤，从实践角度进一步揭示该方法的应用。

3. 实验任务

任务 1：旅馆住宿系统测试用例设计。

(1) 需求：为了进一步扩大业务和提升营业额，旅馆住宿系统支持房间提前预定支付及会员卡办理，且规定在旅游旺季客房紧张的情况下，“进行了房间预订且已支付定金”或“是本旅馆会员，即持有会员卡”的游客，应优先为其办理房间入住。

(2) 问题：采用决策表法进行测试用例设计。

前提条件：需求中输入与输出之间相互制约条件较多，故适合采用决策表法设计用例。

第 1 步：分析需求说明，列出所有的条件桩和动作桩。

① 条件桩：

- 是否进行房间预定。
- 是否已支付定金。
- 是否为旅馆会员。

② 动作桩：

- 优先办理房间入住。
- 作其他处理。

第 2 步：确定规则的个数。在此有 3 个条件，且每个条件有两种取值（是或否），故应有 $2^3=8$ 种规则。此步骤得出表 4.2 所示的决策表。

表 4.2　旅馆系统_决策表_确定规则个数

		1	2	3	4	5	6	7	8
条件	是否进行房间预定？								
	是否已支付定金？								
	是否为旅馆会员？								
动作	优先办理房间入住								
	作其他处理								

第 3 步：填入条件项。即左侧条件桩中各种条件的各种取值的组合，其中各条件次序无严格限制。此步骤得出表 4.3 所示的决策表。

表 4.3　旅馆系统_决策表_填入条件项

		1	2	3	4	5	6	7	8
条件	是否进行房间预定？	Y	Y	Y	Y	N	N	N	N
	是否已支付定金？	Y	Y	N	N	Y	Y	N	N
	是否为旅馆会员？	Y	N	Y	N	Y	N	Y	N
动作	优先办理房间入住								
	作其他处理								

第 4 步：填入动作项。此步骤得出表 4.4 所示的决策表。

表 4.4　旅馆系统_决策表_填入动作项

		1	2	3	4	5	6	7	8
条件	是否进行房间预定？	Y	Y	Y	Y	N	N	N	N
	是否已支付定金？	Y	Y	N	N	Y	Y	N	N
	是否为旅馆会员？	Y	N	Y	N	Y	N	Y	N
动作	优先办理房间入住	X	X	X		X		X	
	作其他处理				X		X		X

第 5 步：简化决策表，合并类似规则或相同动作。经分析可知，“1 与 2”、“5 与 7”、“6 与 8”规则可进行合并，此步骤得出表 4.5 所示的决策表。

表 4.5　旅馆系统_简化后决策表

		1	2	3	4	5	6	7	8
条件	是否进行房间预定？	Y	Y	Y	N	N			
	是否已支付定金？	Y	N	N	—	—			
	是否为旅馆会员？	—	Y	N	Y	N			
动作	优先办理房间入住	X	X		X				
	作其他处理			X		X			

至此，依据表 4.5 所示的所有规则可得出最终测试用例，如表 4.6 所示。

表 4.6　旅馆系统_测试用例设计

编号	输入条件	输入数据	预期结果
1	已进行房间预定且已支付定金	房间编号、类型、定金金额	优先办理入住
2	已进行房间预定、未支付定金，是旅馆会员	房间编号、类型、会员卡号	优先办理入住
3	已进行房间预定、未支付定金，不是旅馆会员	房间编号、类型	作其他处理
4	未进行房间预定，是旅馆会员	会员卡号	优先办理入住
5	未进行房间预定，不是旅馆会员		作其他处理

注意：

① 实际使用决策表时，常常优先进行化简步骤。

② 请回顾决策表法的步骤。

③ 请体会因果图法和决策表法的不同。

任务 2：NextDate 函数测试用例设计。

(1) 需求：NextDate 函数中包含了 3 个输入变量，分别为 Month(月份)、Day(日期)和 Year(年)；函数输出为输入后一天的日期。如输入为 2010 年 10 月 10 日，则输出为 2010 年 10 月 11 日。其中，输入变量 Month、Day 和 Year 都为整数，且取值范围满足：$1 \leqslant Month \leqslant 12$；$1 \leqslant Day \leqslant 31$；$1980 \leqslant Year \leqslant 2020$。

(2) 问题：采用决策表法进行测试用例设计。

前提条件：需求中存在输入输出较多或输入与输出之间相互制约条件较多的情况，故适合采用决策表法设计用例。

第 1 步：分析需求说明，列出所有的条件桩和动作桩。

① 条件桩：

- Month；
- Day；
- Year。

② 动作桩：

- Day 变量值加 1；
- Day 变量值复位为 1；
- Month 变量值加 1；
- Month 变量值复位为 1；
- Year 变量值加 1。

注意：为获得输入后一天的日期，NextDate 函数需执行的操作仅上述 5 种类型。

第 2 步：确定规则的个数。依据“若某决策表中有 n 个条件，且每个条件可取真、假两种值，则共有 2^n 条规则；若某决策表中有 n 个条件，且每个条件可取 1、2、3…、m 种值，则共有 x^n 条规则。”可知，在此有 3 个条件，且每个条件并非仅有两种取值（是或否），具体取值如下所示：

① Month 可有如下 4 种取值：

- M1＝{Month 有 30 天}；
- M2＝{Month 有 31 天，12 月除外}；
- M3＝{Month 为 12 月}；
- M4＝{Month 为 2 月}；

② Day 可有如下 5 种取值：

- D1＝{1≤Day≤27}；
- D2＝{Day＝28}；
- D3＝{Day＝29}；
- D4＝{Day＝30}；
- D5＝{Day＝31}；

③ Year 可有如下 2 种取值：

- Y1＝{Year 为是闰年}；
- Y2＝{Year 为非闰年}。

综上所述，应有 4×5×2＝40 种规则。

第 3 步：填入条件项。即左侧条件桩中各种条件的各种取值的组合，其中各条件次序无严格限制。值得提醒的是，实际使用决策表时，常常优先进行化简步骤。在此，填入条件项的步骤即可结合实际情况进行适当化简。

填入条件项过程中，以 Day 为填写基准（即以条件中取值情况最多的为基准），进行四组数据的填入（因为 Month 有 4 种取值），而 Year 仅对“Month＝M4，且 Day＝D2 或 D3”时的情况有影响。故此步骤得出表 4.7 所示的决策表。

表 4.7　NextDate 函数_决策表_填入条件项

		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
条件	Month	M1	M1	M1	M1	M1	M2	M2	M2	M2	M2	M3	M3	M3	M3	M3	M4	M4	M4	M4	M4	M4	M4
	Day	D1	D2	D3	D4	D5	D1	D2	D3	D4	D5	D1	D2	D3	D4	D5	D1	D2	D2	D3	D3	D4	D5
	Year	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	Y1	Y2	Y1	Y2	—	—

续表

		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
动作	无效																						
	Day 加 1																						
	Day 复位																						
	Month 加 1																						
	Month 复位																						
	Year 加 1																						

第 4 步：填入动作项。此步骤得出表 4.8 所示的决策表。

表 4.8　NextDate 函数_决策表_填入动作项

		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
条件	Month	M1	M1	M1	M1	M1	M2	M2	M2	M2	M2	M3	M3	M3	M3	M3	M4	M4	M4	M4	M4	M4	M4
	Day	D1	D2	D3	D4	D5	D1	D2	D3	D4	D5	D1	D2	D3	D4	D5	D1	D2	D2	D3	D3	D4	D5
	Year	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	Y1	Y2	Y1	Y2	—	—
动作	无效					√															√	√	√
	Day 加 1	√	√	√			√	√	√	√		√	√	√	√		√	√					
	Day 复位				√						√					√			√	√			
	Month 加 1				√						√								√	√			
	Month 复位															√							
	Year 加 1															√							

第 5 步：简化决策表，合并类似规则或相同动作。经分析可知，“1、2 与 3”、“6、7、8 与 9”、“11、12、13 与 14”及“21 与 22”规则可进行合并，此步骤得出表 4.9 所示的决策表。

表 4.9　NextDate 函数_简化后决策表

		1～3	4	5	6～9	10	11～14	15	16	17	18	19	20	21～22
条件	Month	M1	M1	M1	M2	M2	M3	M3	M4	M4	M4	M4	M4	M4
	Day	D1～D3	D4	D5	D1～D4	D5	D1～D4	D5	D1	D2	D2	D3	D3	D4～D5
	Year	—	—	—	—	—	—	—	—	Y1	Y2	Y1	Y2	—
动作	无效			√									√	√
	Day 加 1	√			√		√		√	√				
	Day 复位		√			√		√			√	√		
	Month 加 1		√			√					√	√		
	Month 复位							√						
	Year 加 1							√						

至此，依据表 4.9 所示的所有规则可得出最终测试用例，如表 4.10 所示。

表 4.10　NextDate 函数_测试用例设计

编号	输入条件	输入数据	预期结果
1	输入 3 个变量值	Year＝2010、Month＝9、Day＝10	2010-9-10
2	输入 3 个变量值	Year＝2010、Month＝9、Day＝30	2010-10-1
3	输入 3 个变量值	Year＝2010、Month＝9、Day＝31	无效
4	输入 3 个变量值	Year＝2010、Month＝10、Day＝10	2010-10-11
5	输入 3 个变量值	Year＝2010、Month＝10、Day＝31	2010-11-1
6	输入 3 个变量值	Year＝2010、Month＝12、Day＝10	2010-12-11
7	输入 3 个变量值	Year＝2010、Month＝12、Day＝31	2011-1-1
8	输入 3 个变量值	Year＝2010、Month＝2、Day＝10	2010-2-11
9	输入 3 个变量值	Year＝2012、Month＝2、Day＝28	2012-2-29
10	输入 3 个变量值	Year＝2010、Month＝2、Day＝28	2010-3-1
11	输入 3 个变量值	Year＝2012、Month＝2、Day＝29	2012-3-1
12	输入 3 个变量值	Year＝2010、Month＝2、Day＝29	无效
13	输入 3 个变量值	Year＝2010、Month＝2、Day＝30	无效

注意：

① 实际使用决策表时，常常优先进行化简步骤，请读者仔细体会该步骤。

② 请回顾决策表法的步骤。

③ 请读者尝试采用等价类划分法和边界值分析法进行用例设计，并体会与决策表法的不同。

4. 拓展练习

请采用决策表法针对如下需求进行测试用例设计。

需求：订购单的检查规则为：若金额超过 600 元，又未过期，则发出批准单和提货单；若金额超过 600 元，但过期了，则不发批准单；如果金额低于 600 元，则不论是否过期都发出批准单和提货单，在过期的情况下还需发出通知单。