下面的测试集 T 满足约束集合 S 以及 BOR 测试准则：

$T = \{$ t_1: < a = 1, b = 2, c = 1, d = 0 >; 满足 (**t**, **t**),
t_2: < a = 1, b = 2, c = 1, d = 2 >; 满足 (**t**, **f**),
t_3: < a = 1, b = 0, c = 1, d = 0 >; 满足 (**f**, **t**).
$\}$

由于 T 满足 BOR 测试准则，从而确保能够检测出 p_r 中存在的所有单/多布尔运算符故障。通过针对测试集 T，计算 p_r 及其经注入布尔运算符故障后得到的变体的真值，就能验证这一点。

表 2-6 列出了谓词 p_r 以及经注入单/多布尔运算符故障后得到的 7 个故障谓词。对于每个谓词，都使用 T 中的 3 个测试用例分别进行计算。注意，对于故障谓词，至少存在一个测试用例，使其取值与 p_r 取值不同。

表 2-6　例 2.25 中 BOR 充分测试集对单/多布尔运算符故障的检测能力
（故障谓词与谓词 p_r 的计算结果的差异用斜体标识）

		谓　词	t_1	t_2	t_3
		$a<b\wedge c>d$	true	false	false
单布尔运算符故障	1	$a<b\vee c>d$	true	*true*	*true*
	2	$a<b\wedge \neg c>d$	*false*	*true*	false
	3	$\neg a<b\wedge c>d$	*false*	false	*true*
多布尔运算符故障	4	$a<b\vee \neg c>d$	true	*true*	false
	5	$\neg a<b\vee c>d$	true	false	*true*
	6	$\neg a<b\wedge \neg c>d$	*false*	false	false
	7	$\neg a<b\vee \neg c>d$	true	*true*	*true*

很容易验证，如果从表 2-6 中删去任意一个测试用例，则至少存在一个故障谓词对于剩余的两个测试用例，其真值与谓词 p_r 的真值相同。例如，删去 t_2，则故障谓词 4 与谓词 p_r 对于测试用例 t_1、t_3 所得结果一致。因此，可以肯定，T 是谓词 p_r 的最小且满足 BOR 充分性的测试集。

练习 2.28 与上面的例子类似，要求验证所给的两个测试集分别是 BRO 充分的和 BRE 充分的。在下一节，将讨论生成 BOR、BRO、BRE 充分测试用例的算法。

2.7.5 生成 BOR、BRO 和 BRE 充分性测试用例

现在来描述用于谓词测试的测试约束生成的算法。实际使用的测试用例通常都是根据测试约束产生的。回想一个有效的（feasible）测试约束可被一个或多个测试用例满足。因此，在这里着重讨论生成测试约束的算法，而不是具体的测试用例。满足测试约束的测试用例可以用手工或自动化的方式生成。我们从介绍针对具体谓词的 BOR 约束集合的生成方法开始。

首先，回顾关于集合笛卡儿积的两个定义。

有限集合 A 和 B 的笛卡儿积记为 $A\times B$，定义如下：

$$A\times B=\{(a,b)\mid a\in A, b\in B\}$$

为了能生成最小的测试约束集，还需要另一种集合积的计算方法。集合的 *onto* 积（其运算符记为⊗）定义如下：对于有限集合 A 和 B，$A\otimes B$ 为二元偶（u，v）构成的最小集合，其中 $u\in A$、$v\in B$，且 A 中的各个元素至少出现一次，B 中的各个元素也至少出现一次。根据该定义，当集合 A、B 包含两个或两个以上元素时，$A\otimes B$ 的计算结果不唯一。