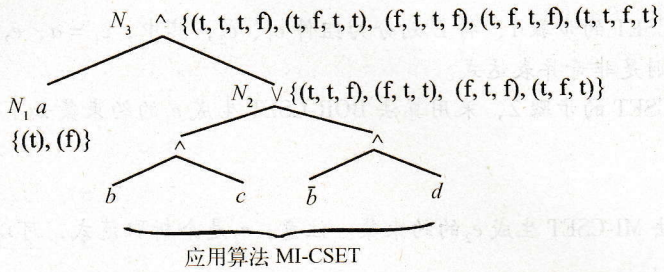

图 2-21 采用算法 BOR-MI-CSET 导出的谓词 $a(bc+\bar{b}d)$ 的约束集

注意，例 2.32 中采用算法 BOR-MI-CSET 导出的约束集 S_E 包含 5 个约束，而例 2.31 中采用算法 MI-CSET 生成的约束集则含有 6 个约束。在通常情况下，采用 BOR-MI-CSET 生成的约束集比用 MI-CSET 生成的小，且有更强的故障检测能力。练习 2.40、练习 2.41 有助于加深读者对 BOR-MI-CSET 及其生成测试集的故障检测效力的理解。

2.7.6 因果图与谓词测试

第 2.6 节中描述的因果图是一种需求建模和测试设计技术。因果关系是从软件需求规范当中提取出来的。在一个因果关系中，作为原因的那部分可以表示成谓词；作为结果的那部分用于构造测试预言，判断当相应原因成立时结果是否会出现。

为了测试代表因果图中原因的条件是否被正确实现，测试人员要么用第 2.6.3 节描述的判定表技术，要么用本节描述的四个算法之一设计测试集。

已有研究证明：用算法 BOR-CSET 生成的测试集比用算法 CEGDT 生成的测试集小得多；用算法 BOR-CSET 生成的测试集的故障检测效力比用算法 CEGDT 生成的测试集略差一点。

有两个理由可以认为，将因果图技术与本节介绍的谓词测试技术结合起来肯定会有更强的故障检测能力。首先，因果图是模拟软件需求的有效手段；其次，一旦因果图建立起来之后，可用本节介绍的四个基于谓词的测试设计算法中任意一个生成测试集。练习 2.42 有助于读者理解因果图与谓词测试组合起来是如何工作的。

2.7.7 故障传播

现在解释故障传播这个概念，它是本节描述的 4 个谓词测试设计算法的基础。设 p_r 是个谓词（不论是简单谓词还是复合谓词），p_c 是 p_r 的一个组件。抽象语法树上的每一个结点都是谓词的组件。例如，在图 2-17 中，谓词 p_r：$(a+b<c)\wedge\neg p\vee(r>s)$ 包含下面 6 个组件：

$$a+b<c,\ \wedge,\ \neg,\ p,\ \vee,\ r>s$$

设 p_f 是 p_r 的错误实现。针对某些测试用例 t，如果 $p_f(t)\neq p_r(t)$，说明 p_f 某个组件中的错误已经传播到了抽象语法树 $AST(p_f)$ 的根结点，该错误已影响到 p_f 的输出。同时，测试用例 t 也被称作 p_f 的错误/故障检测用例。

在谓词测试中，我们关心的是设计出至少一个测试用例 t，确保使 p_f 中的错误能够传播到抽象语法树 $AST(p_f)$ 的根结点，即 $p_f(t)\neq p_r(t)$。BOR、BRO、BRE 充分的测试集能确保使前面章节提到的某些故障传播到谓词抽象语法树的根结点。下面的例子说明故障传播这个概念。

例 2.33 设 p_r：$(a+b<c)\wedge\neg p\vee(r>s)$；$p_f$：$(a+b<c)\vee\neg p\vee(r>s)$ 是两个谓词，p_f 是 p_r 的错误实现。由于错误地使用了运算符 $\vee$，p_f 有一个布尔运算符故障。表 2-7 列出的 4 个