**7.3**

a.

代码见附录

b.

对于任意的模型，恒有下列几个语句可以判断真假

c.

在实际中，可能会有一些模型比较复杂，比如说是多项、、、运算的复合运算，当函数复合程度越高、逻辑运算次数越多。即使真值存在，在有限时间内也难以判断真值。

d.

可以将原公式化成主析取范式或者主合取范式。这样通过判断每个子句的结果，然后进行析取或者合取得到公式的真值。例如，在部分模型中，A为真，B未知，，。但是存在着在部分模型中无法检测真值的情况，例如A为真，B未知时，、、均未知。

e.

改进的算法是将公式化为主析取范式或者主合取范式，通过对子句析取或者合取获得公式的真值。对于子句，通常可以快速判断真值，如，如果有任何一个变量为假，则该子句为假。这样的例子还有很多。由此可以看到，改进的算法可以加速判断一个语句是否由KB包含，即TT-ENTAILS?更有效。

**7.4**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **题号** | a | b | c | d |
| True/False | true | false | true | false |
| **题号** | e | f | g | h |
| True/False | true | true | true | true |
| **题号** | i | j | k | l |
| True/False | false | true | true | false |

**7.7**

a.

总模型个数为，当时，当且仅当，因此该语句对应的模型数为。

b.

当时，当且仅当，因此该语句对应的模型数为。

c.

考虑时，当且仅当，

此时，因此语句，该语句对应的模型数为0。

**7.20**

首先将s1~s6蕴含式化简成合取范式

DPLL算法过程：

|  |
| --- |
| 组合成子句{, } |
| 初始化  提取纯文字从消去，  从子句中消除相关子句 |
| **令** |
| 考察子句{,} |
| 令  考察子句{}  令  此时子句为假  **Return False** (此时说明A=B=true 不满足要求) |
| 令  考察子句{}  显然此时不存在使得子句为真的变量  **Return False** |
| 上述过程得到不能使得子句为真 |
| 令  考察子句{}  令  发现此时子句全为真  **Return True** |
| **Finsh** |

**附录**

**def** pl\_true(s, m):  
 **if** (s == **True**):  
 **return True  
 elif** (s == **False**):  
 **return False** op, args = s.op, s.args  
 **elif** (is\_symbol(op)):  
 **return** lookup(s, m)  
 **elif** (op == **'~'**):  
 **return not** pl\_true(args[0], m)  
 **elif** (op == **'or'**):  
 **return** pl\_true(args[0], m) | pl\_true(args[1], m)  
 **elif** (op==**'and'**):  
 **return** pl\_true(args[0], m) & pl\_true(args[1], m)  
 **elif** (op==**'->'**):  
 **return** (**not** pl\_true(args[0], m))|pl\_true(args[1], m)  
 **elif** (op=**'<->'**):  
 **return** ((**not** pl\_true(args[0], m)) | pl\_true(args[1], m))=((**not** pl\_true(args[1], m)) | pl\_true(args[0], m))  
 **else**:  
 print(**'关键词错误'**)