所有不超过二/三元的逻辑联结词构成最小完全集求法

问题难点：三元联结词的组合情况太多，无法计算

解决思路：缩小搜索空间

Post's Functional Completeness Theorem

**适用范围**：2值(True/False)真值函数

**作用**：判断一个真值函数/逻辑联结词集合是否是完全集

**定理**：提出了5条规则，每个规则对应着一个逻辑逻辑联结词的集合。对于一个逻辑联结词的集合S，如果S不是上述5个集合中任一集合的子集，那么S就是一个完全集。

**换种表达方式：**上述5条规则，如果对于每条规则，联结词集合S中都至少有一个元素不满足，则S是一个完全集。

缩小搜索空间

**定义5个集合**：

分别为不满足规则的逻辑联结词集合，相互之间可以有交集。

**完全集的定义**：

从集合中分别抽出至少一个元素构成的集合

**定义逻辑联结词集合**：

从集合中分别抽取一个元素组成的集合。

**搜索空间**：所有的集合

暴力搜索法

遍历所有的集合(可重复)，共

每次遍历，执行以下操作：

1. 对集合进行去重
2. 根据去重后集合中元素个数存入1,2,3,4,~~5~~相应的文件中

构造最小完全集集合，并依次遍历文件1,2,3,4：

1. 对文件中元素进行去重
2. 将子集不在集合中的元素加入到集合中

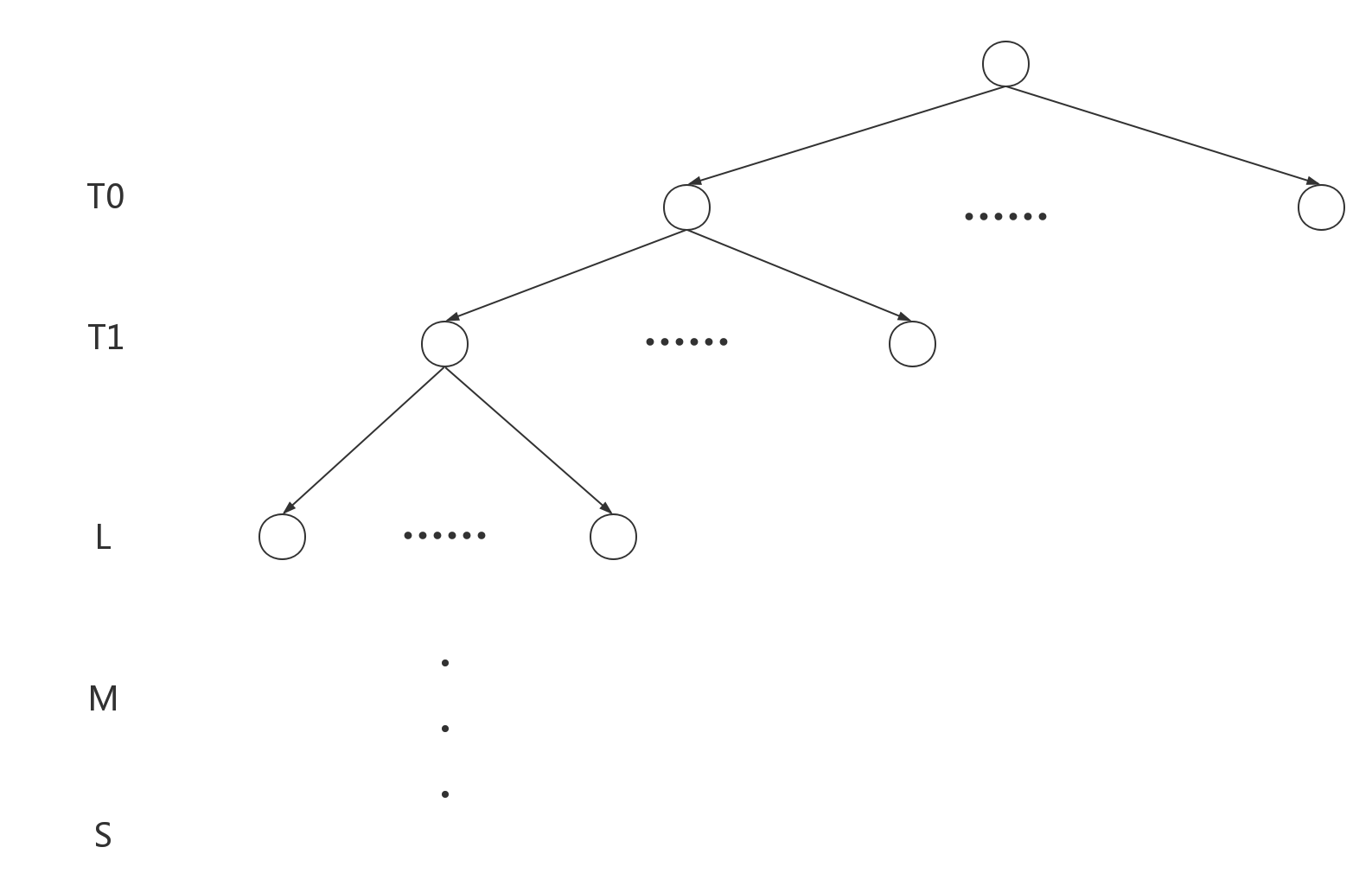
最小完全集不可能有5个不同元素：

反证法：如果最小完全集有5个不同元素，则上述5个集合中，每个集合都至少有一个元素只属于该集合，也就是只满足一个规则。这种情况是不存在的。

总时间：大概1天

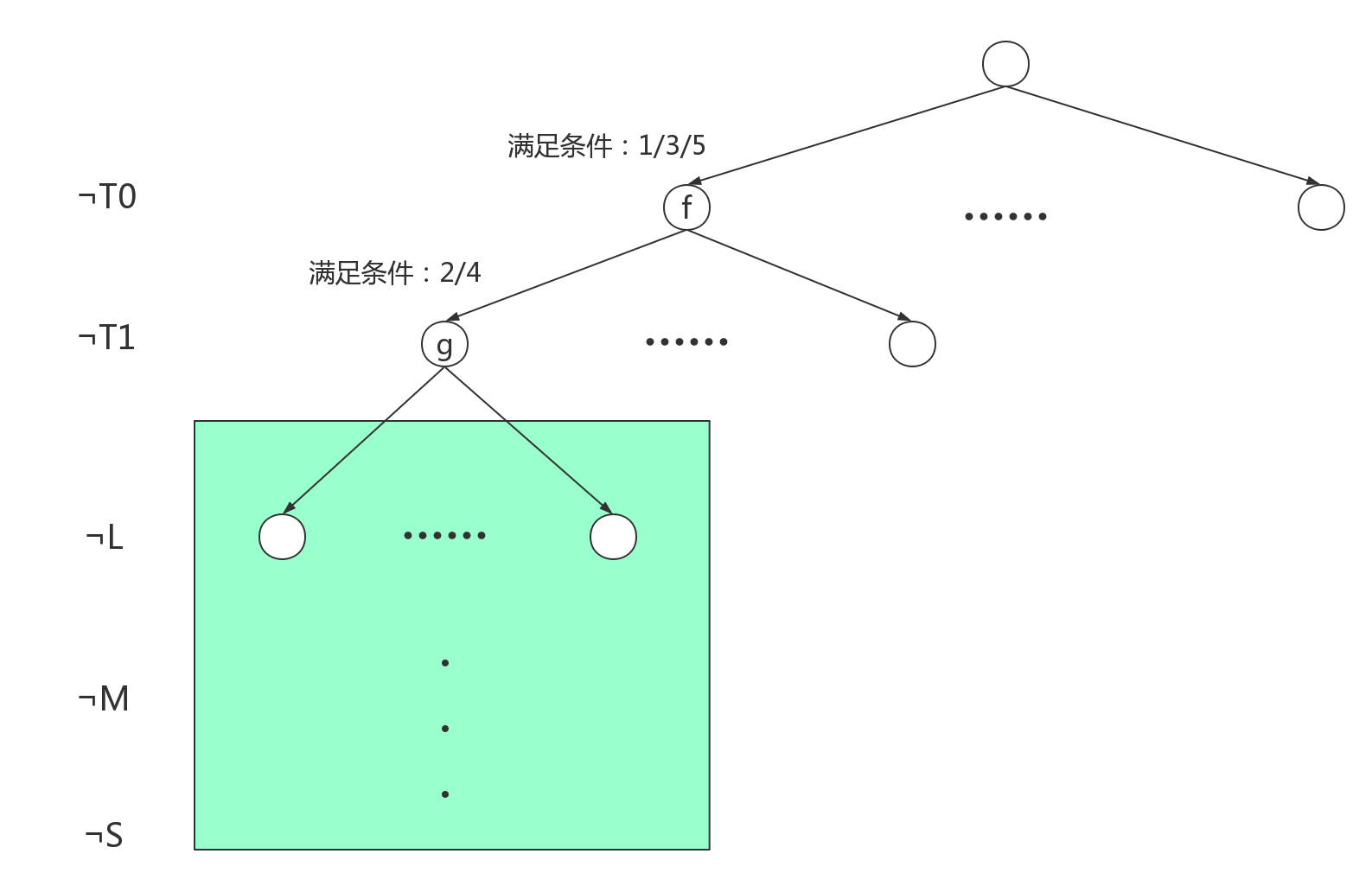
剪枝(+贪婪)搜索法

暴力搜索：



先纵向遍历，再横向遍历

剪枝：



当遍历完f和g之后，此时的集合就已经是一个完全集了，继续纵向遍历的结果一定不可能是一个最小完全集，所以可以停止纵向遍历，开始横向遍历。

贪婪：在遍历前，对集合都进行排序，满足条件多的元素放在前面。

遍历完再去除结果中非最小完全集

计算时间：大概1小时

二元结果：56

三元结果：6800+

不超过二元的联结词组成的最小完全集

{'f0': ['2', '1110']}

{'f0': ['2', '1100'], 'f1': ['2', '1101']}

{'f0': ['2', '1011'], 'f1': ['2', '1100']}

{'f0': ['2', '1010'], 'f1': ['2', '1101']}

{'f0': ['2', '1010'], 'f1': ['2', '1011']}

{'f0': ['2', '1001'], 'f1': ['2', '1100']}

{'f0': ['2', '1001'], 'f1': ['2', '1010']}

{'f0': ['2', '1000']}

{'f0': ['2', '0111'], 'f1': ['2', '1100']}

{'f0': ['2', '0111'], 'f1': ['2', '1010']}

{'f0': ['2', '0110'], 'f1': ['2', '1101']}

{'f0': ['2', '0110'], 'f1': ['2', '1100']}

{'f0': ['2', '0110'], 'f1': ['2', '1011']}

{'f0': ['2', '0110'], 'f1': ['2', '1010']}

{'f0': ['2', '0110'], 'f1': ['2', '0111'], 'f2': ['2', '1001']}

{'f0': ['2', '0101'], 'f1': ['2', '0110'], 'f2': ['2', '1001']}

{'f0': ['2', '0100'], 'f1': ['2', '1101']}

{'f0': ['2', '0100'], 'f1': ['2', '1100']}

{'f0': ['2', '0100'], 'f1': ['2', '1011']}

{'f0': ['2', '0100'], 'f1': ['2', '1010']}

{'f0': ['2', '0100'], 'f1': ['2', '1001']}

{'f0': ['2', '0011'], 'f1': ['2', '0110'], 'f2': ['2', '1001']}

{'f0': ['2', '0010'], 'f1': ['2', '1101']}

{'f0': ['2', '0010'], 'f1': ['2', '1100']}

{'f0': ['2', '0010'], 'f1': ['2', '1011']}

{'f0': ['2', '0010'], 'f1': ['2', '1010']}

{'f0': ['2', '0010'], 'f1': ['2', '1001']}

{'f0': ['2', '0001'], 'f1': ['2', '1100']}

{'f0': ['2', '0001'], 'f1': ['2', '1010']}

{'f0': ['2', '0001'], 'f1': ['2', '0110'], 'f2': ['2', '1001']}

{'f0': ['0', '1'], 'f1': ['2', '1100']}

{'f0': ['0', '1'], 'f1': ['2', '1010']}

{'f0': ['0', '1'], 'f1': ['2', '0110']}

{'f0': ['0', '1'], 'f1': ['2', '0100']}

{'f0': ['0', '1'], 'f1': ['2', '0010']}

{'f0': ['1', '10'], 'f1': ['2', '1101']}

{'f0': ['1', '10'], 'f1': ['2', '1011']}

{'f0': ['1', '10'], 'f1': ['2', '0111']}

{'f0': ['1', '10'], 'f1': ['2', '0101'], 'f2': ['2', '1001']}

{'f0': ['1', '10'], 'f1': ['2', '0101'], 'f2': ['2', '0110']}

{'f0': ['1', '10'], 'f1': ['2', '0100']}

{'f0': ['1', '10'], 'f1': ['2', '0011'], 'f2': ['2', '1001']}

{'f0': ['1', '10'], 'f1': ['2', '0011'], 'f2': ['2', '0110']}

{'f0': ['1', '10'], 'f1': ['2', '0010']}

{'f0': ['1', '10'], 'f1': ['2', '0001']}

{'f0': ['1', '10'], 'f1': ['0', '1'], 'f2': ['2', '1001']}

{'f0': ['1', '10'], 'f1': ['0', '1'], 'f2': ['2', '0101']}

{'f0': ['1', '10'], 'f1': ['0', '1'], 'f2': ['2', '0011']}

{'f0': ['0', '0'], 'f1': ['2', '1101']}

{'f0': ['0', '0'], 'f1': ['2', '1100']}

{'f0': ['0', '0'], 'f1': ['2', '1011']}

{'f0': ['0', '0'], 'f1': ['2', '1010']}

{'f0': ['0', '0'], 'f1': ['2', '1001']}

{'f0': ['0', '0'], 'f1': ['1', '10'], 'f2': ['2', '0110']}

{'f0': ['0', '0'], 'f1': ['1', '10'], 'f2': ['2', '0101']}

{'f0': ['0', '0'], 'f1': ['1', '10'], 'f2': ['2', '0011']}