Q-M 算法的 Matlab 实现　实验报告

陈彦熹

March 6, 2016

1 实验目的

本实验根据 Q-M 算法的原理，实现一个能处理十变量及以下的逻辑函数的化简工具。理论上该工具也可化简十变量以上的逻辑函数，但是无法保证运行效率。

2 理论依据

本实验的理论依据为 Q-M 算法，具体实现包括以下两个步骤：

- 搜索本原蕴含项
- 寻找最小覆盖

对于第一步，Q-M 算法的基本思想与卡诺图一致，即找出相邻最小项，消除变量因子；对于第二步，本实验采用了 Petrick's Method[1]，以便于计算机实现。

3 实验结果

本实验以 Matlab 函数的形式实现逻辑化简工具。

函数 Q_M 接受三个参量：N 表示逻辑函数变量个数，一般不超过 10；m 是逻辑函数规范形式的最小项代号组成的向量，元素范围从 0 到 2^N-1;d 是逻辑函数规范形式的无关项代号组成的向量，元素范围同 m。函数实现对于输入参量的合法性检查包括 m 与 d 元素范围，当出现输入错误时给出警告信息并终止函数运行；另外函数预先对 m 与 d 进行去重复元素处理。函数没有检查 m 与 d 中是否存在相同元素，根据函数具体实现，对于这种情况，将该重复元素视为最小项而非无关项（除非 m 与 d 元素数目之和恰等于 2^N，此时函数会错误地返回 '1'）。

函数 Q_M 返回值 lgcExpr 为字符串，表示经过化简得到的最简二级与或形式逻辑表达式，字母 A 到 J 为函数变量；变量取非用 X' 形式表示。

[1] Wikipedia. Petrick's Method. https://en.wikipedia.org/wiki/Petrick's_method

4 具体实现

4.1 预处理及初始化

对输入参量进行检查与处理，并且定义部分后续所需变量；将最小项与无关项的代号通过函数 dec2bin 转化成 N 位二进制字串，存储在矩阵 binstr 中。

4.2 结合最小项与无关项

比较 binstr 各行字符串，若只有一位不同，则必有两串该位字符分别为 0 和 1，两串合并，该位字符置为 ‘-’，并加入到 nextbinstr 中，cmb_flg 中相应值置 1；一轮循环结束后，cmb_flg 为 0 的对应项不可再合并，则放入矩阵 final，用 nextbinstr 更新 binstr，更新 cmb_flg，进行新一轮循环。注意到这里并没有按照手工做法，先将各项按照包含 1 的个数进行分组，是因为这对于运行效率没有大的影响；反而是每次循环后应对 binstr 进行去重复处理，否则 binstr 中出现大量重复项，严重影响运行效率。

4.3 寻找最小覆盖

构造本原蕴含图 ptk，应用 Petrick’s Method 寻找最小覆盖。首先找出本质本原蕴含项，将该项在 final 中的对应项加入到 rslt 中，并且删除 ptk 中对应的行和列；假如 ptk 尚不为空，说明还需要寻找尽可能少的本原蕴含项来覆盖剩余的最小项。最终选择的项对应 final 中的项也加入到 rslt 中。

4.4 输出结果

将 rslt 中的各行字符串转化为与或形式逻辑函数式中的各项。比如对于四变量输入，rslt 中一行字符串 “1-00” 转化为逻辑函数中的 “AC’D’ ”。

5 缺陷与不足

本实验实现的 Q_M 函数，最大的缺陷在于寻找最小覆盖过程中，处理 ptk 非空情况时所使用的一个 for 循环（第 114 行）。该处理过程涉及到类似于数学中因式相乘做展开的过程：$(A+B)(A+C)=(A^2+AC+BA+BC)$。对于 Q_M 函数，各因式存储在 cbcell 中，并且长度不一，因此不得不引入向量 cbarr，模拟一个轮转循环取元素相乘的过程。假如 cbcell 中包含 3、4、5 元素的向量各 1 个，那么 while 循环需要执行 $3\times4\times5=60$ 次，得到 60 个展开因子。由于 Matlab 作为一种解释性语言，循环效率较低，因此这部分代码成为了整个函数性能的瓶颈，尤其当本原蕴含图较为稠密时，这部分循环会消耗大量时间。事实上，若 cbcell 中的元素等长，比如 cbcell 包含 p 个二元素向量，那么可以用一个整数 k 从 0 依次加 1 到 2^p-1，对每个 k 值，通过代码 dec2bin(k,p)-‘0’+1，即可得到一个用于选取各因式元素的向量 cbarr，这样做的效率远远高于本实验的实现方式。