第 26卷 第 1期 电子科技大学学报 Vol. 26 No. 1
1997年 2月 Journal of UEST of China Feb. 1997

二元判决图应用中函数组合方法的改进*

章小兵** 陈光
(电子科技大学 CAT研究室 成都 610054)

【摘要】 文中确定了 Bryant的基于图的函数组合方法[1]的时间复杂度为 $O(|G_1|^2 \cdot |G_2|)$,并提出了基于改进 ITE算符的函数组合方法。该方法省去了对结果二元判决图的约简步骤,保持了二元判决图的强正则性,提高了效率。

关 键 词 二元判决图; 函数组合; 时间复杂度; 优化
中图分类号 TP301

二元判决图是逻辑函数有效的表示和处理方法,被越来越多地应用到计算机辅助设计的领域中,譬如逻辑综合、逻辑校验、逻辑优化和测试自动生成等。

二元判决图是一种带标号的有向无环图(DAG). Bryant[1]对二元判决图增加了两个约束条件:固定编序和约简,使二元判决图成为逻辑函数的正则形式,称为约简编序二元判决图(ROBDD)。ROBDD有两类节点:终节点和非终节点。终节点代表逻辑常数 1或 0;非终节点由一个逻辑变量(标记为 v)和两个输出边(1边和 0边)组成。节点 F的两个输出边指向的节点分别标记为 F. high和 F. low,分别代表函数 F在其顶点变量取 1和 0时的值。BDD的节点也可标记为 (v, G, H),表示函数 $F = vG + \bar{v}H$

Bryant也对常用的逻辑函数操作提供了有效的基于图的算法[1],譬如运算(apply)、约简(reduce)和组合(compose)等,以用于二元判决图的构造和处理。这些操作是有效的,其时间复杂度是二元判决图大小($|G|$,即 ROBDD的节点数)的多项式。Fujita等[3]在评价和改进基于二元判决图的逻辑校验过程中,发现函数组合方法是提高系统性能的方法之一。他在电路中间网络的 BDD节点过多时,就将该网络分割开来并作为外部输出来处理。在所有的 BDD都建好后,再用函数组合的方法将整个电路的 BDD构造出来,这样可以减少计算过程中的节点数,也就加快了计算的速度。但是,基于图的算法还不是最优的。Brace[5]实现了 BDD中高效的 ITE算符,并使 BDD成为逻辑函数的强正则表达形式,也就是所有相同功能函数的 BDD都是同一 DAG,由唯一的指针表示。但 Brace没有实现函数组合算法,而且直接用 ITE来实现组合算法,效果也不好。

1 基于图形的函数组合方法的时间复杂度

Bryant在文献[1]中认为在最坏的情况下,组合操作的时间复杂度应该是 $O(|G_1|^2 \cdot |G_2|)$,但他不能确定是否正确,因为他没有发现比 $O(|G_1|^2 \cdot |G_2|)$更高的时间复杂度。我们通过下面的例子来说明组合操作的时间复杂度。

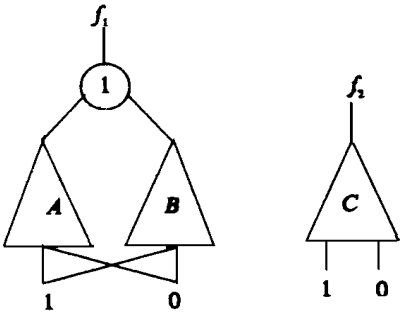
1996年 5月 15日收稿,1996年 6月 17日修改定稿
* 国家"八五"重点科研项目
** 男 26岁 博士生

图 1a表示函数 f_1 的 BDD,由一个顶点节点(序为 1)和两个互不交叉的子图 A 和 B 构成,子图 A 和 B 中节点数相等,且它们的序都不交叉,也就是说如果序 $\in I_A$ 则 $\notin I_B$;图 1b表示另一函数 f_2 的 BDD,其中节点的序与子图 A 和 B 的节点的序不交叉。我们来观察 compose($f_1, f_2, 1$),即是将函数 f_2 代入函数 f_1 中的 x_1。

基于图形的 compose算法的基础是公式

$$f_1|_{x_1=f_2} = f_2 \cdot f_1|_{x_i=1} + \bar{f}_2 \cdot f_1|_{x_i=0} \qquad (1)$$

详细算法见文献 [1]。根据这个算法,每一步迭代都要在节点集合 {A}, {B}和 {C}中选出序最小的节点,分别用它的 1边和 0边节点与另外两个集合中序最小的节点进行下一步迭代,一直到到达终节点为止。在这个算法中,可采用一个表来存贮已组合过的节点对的结果,这样图中节点的每种排列组合最多被调用一次。由于 A,B 和 C的节点序互不交叉,节点集合 {A}, {B} 和 {C}中各取一个节点的任何排列组合都会被调用一次,即一共迭代$|A| \cdot |B| \cdot |C|$次。而每次调用占用定长时间,所以这个运算的时间复杂度为 $O(|A| \cdot |B| \cdot |C|)$。在实际应用中,若 f_1 的节点较多,$|A|$ 和 $|B|$ 可以约等于 f_1的节点数的一半,所以其时间复杂度为 $O(|G_1|^2 \cdot |G_2|)$,其中 G_1 和 G_2 为表示 f_1 和 f_2 的 BDD。



(a) 函数 f_1　　(b) 函数 f_2

图 1　函数组合操作的例子

2 改进的函数组合方法

在 Bryant的算法中,每进行完一种 ROBDD之间的运算,就要对结果 ROBDD进行约简。约简的时间复杂度是 $O(|G| \log(|G|))$,这就在总体上降低了其算法的效率。Brace[5]和 MINATO等[4]提出了更高效的 BDD表示和运算方法,他们采用逻辑函数的强正则表达形式,将多个 ROBDD放入一个图中,使它们共享相同的子图,另外采用了 ITE算符、属性边等技术,在运算中动态地约简子图,并利用缓存技术提高 BDD运算的效率。所以采用这些技术,利用 ITE来实现函数组合算法,可以得到比基于图的方法更高的效率。

文献 [5]中定义 ITE算符为

$$\mathrm{ITE}(f,g,h) = fg + \bar{f}h \qquad (2)$$

根据式 (1)和式 (2),我们可以把函数组合表示为

$$f_1|_{x_1=f_2} = \mathrm{ITE}(f_2, f_1|_{x_1=1}, f_1|_{x_i=0}) \qquad (3)$$

根据式 (3)所知,此段正斜对换,可以直接利用 Brace的 BDD运算包中的 ITE算符[5]实现函数组合算法。在这种实现方法中,首先要计算 $f_1|_{x_i=1}$和 $f_1|_{x_i=0}$,这可以用将 f_1 的 BDD中所有序为 i 的节点分别替换为它们的 1边和 0边节点的方法来完成。但在 Brace的 BDD包中,要遍历序为 i 的所有节点而不遍历整个图是不可能的。因为在其 BDD包中,一个 BDD图由其根节点的指针表示,并且为了提高内存的使用效率和保持 ROBDD的强正则性,图中所有节点(具有不同的序)都保存在一个哈希表(称为唯一表)中,而没有按它们的序进行分级保存。所以,要对其中某个序的节点进行处理,只能从其根节开始,对整个图进行遍历。遍历图 G的时间复杂度为 $O(|G|)$,这也影响了函数组合的效率。

根据式 (3)可以得到下面的迭代公式

$$f_1|_{x_i=f_2} = \begin{cases} x_j (f_1|_{x_j=1})|_{x_j=1}|_{x_i=f_2} + \bar{x}_j (f_1|_{x_j=0})|_{x_1=f_2} & i \neq j \\ \mathrm{ITE}(f_2, f_1|_{x_1=1}, f_1|_{x_i=0}) & i = j \end{cases} \qquad (4)$$

式中 j 是 f_1 的顶点节点的序。

根据式(4),我们可以类似 Brace的 ITE算法[5]来设计函数组合算法,并将 $f_1|_{x_i=1}$ 和 $f_1|_{x_i=0}$ 的计算并入 compose迭代计算中去。改进的函数组合算法如图 2所示。在对唯一表和私有的计算表进行查找和插入只花费固定时间时的前提下,该算法在最坏情况下的时间复杂度也是 $O(|G_1|^2|G_2|)$。

```
1  compose(f, g, i) {   //将 g代入 f 的 x_i
2  if(是结束情况) return结果;
3  if(compose-table中有 (f, g, i)的结果) return结果;
4  v= min(f.index, g.index);
5  if(v= = i)
6     return ITE(g, f.high, f.low);
7  else {
8     T= compose(f_v, g_v, i);
9     E= compose(f_v̄, g_v̄, i);
10    if(T= = E) return T;
11    R= find-or-add-unique table(v, T, E);
12    insert-compose-table({f, g, i}, R);
13    return R;
14  }
15 }
```

图 2 基于改进的 ITE算符的函数组合算法

在图 2的算法中,为保持整个 BDD包的强正则性,compose算法生成的节点放在 BDD包的唯一表中(图中第 11行)。同 Brace的 BDD包一样,我们可以使用内存函数的概念,利用计算表来避免相同参数下的重复计算。具体的实现为:在计算出 $R=$ compose(f, g, i)时,将参数对 $\{f, g, i\}$和结果 R联系起来,存放到表 compose-table中(第 12行)。在每次迭代之前,查找 compose-table,如果表中有本次迭代的参数对,就直接返回对应的结果(第 3行)。值得注意的是,即使在计算另外的函数组合时,compose-table中的结果都是有效的,所以只需要对 compose-table初始化一次,而不用在每次计算新的函数组合时对其初始化。

为了保证 ROBDD节点的编序约束,每次迭代选择 f和 g中序小的一个进行香农展开(第 4行),再对它的两个香农因子进行进一步的迭代(第 8、9行)。在这个算法中计算香农因子是很简单的,由图 2的第 4行知 $v \leqslant f.$index,如果 $v= f.$index,则 $f_v= f.$high,$f_{\bar{v}}= f.$low;如果 $v < w$则 $f_v= f_{\bar{v}}= f$。对 g的香农展开也是这样。

根据 ROBDD的定义,从根节点到终节点的任意路径上节点是按序排列的,所以只要计算到 f中的序为 i的节点(第 5行),从该节点到终节点的任意路径上都不会再有序为 i的节点,就可以根据式(3)直接用 ITE来计算(第 6行),也就是式(4)中 $i= j$的情况。这样的好处是充分利用原来的 BDD包中的资源,譬如计算表中的结果。

图 2的算法是一个迭代过程,组合迭代结束情况有三种: compose$(1, g, i)= 1$, compose$(0, g, i) = 0$(第 2行)以及转入 ITE迭代(第 6行)。

改进的函数组合算法的一个例子见图 3,其中 $a= x_1+ x_2x_3$,$d= x_1+ x_3$。图中的有些节点,实际上是共享了 BDD包唯一表中的同一节点,所以使用一个标记,譬如各图中序为 3的所有节点,均标记为 c。由图 2的算法得结果为 compose$(a, d, 2)= d$。可以看出,计算出的结果已经是约简的,这就

省去了基于图的组合算法对结果 BDD的约简步骤,并保持了强正则性.

$$R = \text{compose}(a, d, 2) =$$
$$(x, \text{compose}(a_{x_1}, d_{x_1}, 2), \text{compose}, (a_{\bar{x}_1} - d_{\bar{x}_1} - 2)) =$$
$$(x_1, \text{compose}(1, 1, 2), \text{compose}(b, c, 2)))$$
$$= (x_1, 1, \text{ite}(c, c, 0)) = (x_1, 1, c) = d$$

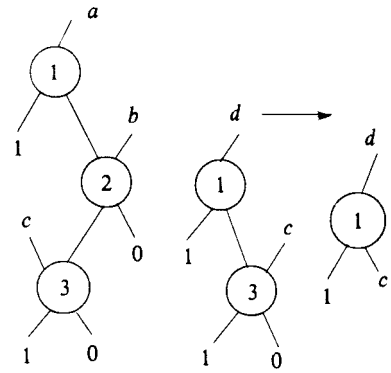


图 3　改进的 compose算法的例子

3 结　论

本文对 Bryant[1]的基于图其函数组合算法进行分析,确定在最坏情况下其时间复杂度为 $O(|G_1|^2|G_2|)$,并加以改进,提出了一种基于 ITE算符的更高效的算法. 新算法的优点是: 省去了图算法的约简步骤;保证了 BDD在应用中的强正则性: 利用 Brace[5]对 BDD包的改进技术,得到了比图形算法速度更快、内存更有效的算法.

参 考 文 献

1　Bryant Randal E. Graph-based algorithm for boolean function manipulation. IEEE Trans on Computer, 1986, 35(8): 677~691

2　Malik Sharad, Wang Albert R, Brayton Robert K et al. Logic verification using binary decision diagrams in a logic synthesis environment. Proc Int'l Conf on CAD, California, 1988: 6~9

3　Fujita M, Fujisawa H, Kawato N. Evaluation and improvements of boolean comparison method based on binary decision diagrams. Proc Int'l Conf on CAD, California, 1988: 2~5

4　Minato Shin-ichi, Ishiura Nagisa, Yajima Shuzo. Shared binary decision diagram with attributed edges for efficient boolean function manipulation. 27th ACM/IEEE DAC, Florida, 1990: 52~57

5　Brace Karl S, Rudell R L, Bryant R E. Efficient implementaion of a BDD package. 27th ACM/IEEE DAC, Florida, 1990: 40~45

6　Fujita Masahiro, Fujisawa Hisanori, Matsunaga Yusuke. Variable ordering algorithms for ordered binary decision diagrams and their evaluation. IEEE Trans on CAD of Integrated Circuit and System, 1993, 12(1): 6~12

Improvement on Composition Methods in Binary Decision Diagrams Based Applications

Zhang Xiaobing　　Chen Guangju

(CAT Research Lab., UEST of China　Chengdu　610054)

Abstract　Function composition is a basic logical operation in most CAD applications, and is adopted as an improvement to a binary decisions diagram (BDD) based application. This paper determines the time complexity of the graph-based composition algorithm proposed by Bryant to $O(|G_1|^2|G_2|)$, then improves the composition algorithm based on the efficient ITE operator. The new algorithm omits the reducing step for result BDD, keeps the BDD representation as a strong canonical form and therefore improves the composition effciency.

Key words　binary decision diagram;　compose;　time complexity;　optimization

编辑　黄　辛