# 基于拼凑替换的定理机器证明的研究与实现

于尚超,李 阳,王 鹏
(解放军理工大学 指挥自动化学院,江苏 南京 210007)

**摘 要**:机器定理证明可以避免人工证明容易出现的低级错误,是人工智能的重要方面,有广泛的应用前景;函数式程序设计的设计思想更加接近于数学,在定理证明方面有天然优势。人们证明逻辑推理的过程通过函数式程序实现,并将其证明的步骤显示出来,采用了逻辑推理机器证明。通过思考人脑在证明定理时的思考方式,给出了一个简单易懂的机器证明的方式。首先将证明的已知和结论形式化,将已知设为 start,结果为 end,已经证明的公理就是 road,那么证明的过程就是从 start 沿着 road 到达 end 的过程。实验表明,逻辑证明通过函数式程序实现,达到了预期目的。

**关键词**:机器证明;人工智能;自动推理
**中图分类号**:TP31 **文献标识码**:A **文章编号**:1673-629X(2012)06-0135-04

# Research and Realization of Theorem Proving Based on Combination and Replace

YU Shang-chao,LI Yang,WANG Peng
(Institute of Command Automation,PLA University of Science & Technology,Nanjing 210007,China)

**Abstract**:Mechanical theorem proving can prevent stupid mistakes of human being. It is one of the most important aspects of artificial intelligence. It can be foreseen that after a few years this technique will be used in many fields. Functional programming language is good at theorem proving,as a result of its design idea. It is similar to mathematics. It programs the way human being often do when they do the job of theorem proving. The program can also show the detailed steps. The way of human thinking can be simulated,to realize proving of theorem. Treat what people know as "start",what people want as "end" and the acknowledged truth as "road". And the procedure to prove is to find the "road" from "start" to "end". It puts forward a simple way to realize the proving of mathematics formula. The experiments show that achieve the goal of theorem through the use of functional language.

**Key words**:mechanical theorem proving;artificial intelligence;automated deduction

## 0 引 言

由于常规方式证明定理可能会出现人为的错误,机器证明[1]能够保证证明准确无误,所以在一些关键领域机器证明是非常必要的。目前 Isabelle[2] 在定理证明中的应用很广,其作为一个通用的定理证明辅助工具,有些人会直接使用它。但是有的时候,直接采用这些"高级的"成熟的工具会不理解其中的原理。对于许多人来说,如果没有了解一些基本的知识,就直接使用 Isabelle,就可能不知道其中的原理。

文中首先回顾了机器证明的历史,并给出机器证明的常用方法。盘点国内外的发展现状。就机器定理自动证明提出一些自己的看法:提出一个证明定理的基本的形式,然后根据这个形式,实现了数学归纳法例子的证明。旨在确保证明的过程更加明了,深入地了解定理证明的本质。

## 1 历史回顾与环境介绍

机器证明的思想很早就出现了。近代的机器证明思想是由莱布尼茨提出的[3],再加上希尔伯特的数理逻辑,给人们提供了一个好的证明途径。在计算机发明之前,由于工具的限制,机器证明方面并没有取得可喜成绩。所以数学家也开始怀疑机器证明能否真正实现。

1948 年,塔斯基[4] 对这个问题给出肯定的回答。计算机的出现,使得许多以前非常困难的问题迎刃而解。比如,四色猜想[5] 问题等。在数学的多个分支中,开始有计算机的参与,毕竟计算机本身就是为了"计算"而发明的。比如,几何定理机器证明等方面。

自从计算机出现之后,人们一度对定理证明出现了非常大得热情,但是一段时间之后,没有显著的进展,所以出现了一段时间的冷门。二十世纪七十年代,

**收稿日期**:2011-10-28;**修回日期**:2012-02-01
**作者简介**:于尚超(1986-),男,硕士,CCF 会员,研究方向为人工智能。

吴文俊院士提出的吴方法[6,7]，给机器证明注入了新活力。借助国家的项目支持，我国在此方面虽不是遥遥领先，也算是处于先进水平。几何机器证明的方法有好多，比如基于 Gröbner 基的方法[4]、例证法、三角化消元法等。在几何定理证明领域，国内已经做得很好了。

与此对应，逻辑公式的机器证明在国内的研究相对比较弱了。数理逻辑方面，胡世华[8]院士是国内把逻辑和计算机结合起来的倡导者。

但是这种用于辅助逻辑推导的软件在国内几乎没有。数学界比较著名的三个软件中，MathLab 擅长算法开发、数据分析等；Maple 属于比较通用类型的，擅长工程软件；Methematica 很好地结合了数值和符号计算引擎、图形系统、编程语言、文本系统等。

我们知道，程序式语言[9]的基本单位是命令，其程序是由命令组成的。也就是说里面有许多的变量，命令的执行可能会改变某些变量的值。当执行一个命令改变了一些本来不想改变的值时，副作用就产生了。由于这个特点，程序式语言在证明定理时可能不太容易。比如，加法交换律 E1+E2=E2+E1，如果在计算 E1 的过程中，改变了 E2 的某些变量，那么结果就可能与 E2+E1 的结果不一样。目前用于工程实践的开发语言大部分属于程序式语言。

之所以说函数式语言在处理表达式方面有许多天然的优势，是由于其基本的单位是表达式，而不是命令，其执行的单位就是表达式，内部的“变量”仅仅是一个对象的符号，一旦指定，就不能改变对应的值，除非再给它另外指定一个值。文中所采用的环境 O'Caml[10,11]就是一个函数式语言，在处理递归、模式匹配方面有很高的效率。另外，它可以把函数当做普通的变量来处理，就是函数的参数可以是普通变量也可以是函数，这样的程序语言就有很高的自由度。

## 2 证明的形式

逻辑表达式的证明，经常使用的是语义 Tableau 法[12,13]和归结法。归结原理又称消解原理[14,15]，从理论上解决了定理证明问题。归结原理提出的是一种证明子句集不可满足性，从而实现定理证明的一种理论及方法。Tableau 是按照某种规则在有向二叉树的每个节点上都标记有一个合式公式而构成，换句话说，通过引入相应的谓词公式，将二元关系的性质用逻辑公式表示出来，对于不同的逻辑系统，只是对公式构造集进行扩展。

这些方法都比较复杂，涉及的是基本的逻辑规则[16]。而对于程序实现上，有一类很简单的方法，基本规则就是用已知来替换未知，等号两边互换，也就是拼凑替换法。

拼凑替换法的基本原理是，前提和结论分别视为两个集合 $A$ 和 $B$。集合 $A$ 中的元素是已知条件，设默认的基本形式为 $e->f$ 的形式；$B$ 中的元素与 $A$ 类似。$B$ 中的左边的式子中的元素 $b1$ 在 $A$ 中搜索，如果找到 $e1->f1$（其中 $b1=e1$），则用 $f1$ 代替 $b1$；之后 $B$ 中右边的式子 $bb1$ 在 $A$ 中搜索，并进行相应代换，之后得到的是 $B$ 中为恒等式，则定理得证。如果不是，则可以运用其他的方式，比如消解法等。再进行拼凑替换。

### 2.1 证明原则

对于形如 $A_1,A_2,\cdots,A_n\rightarrow B$（其中 $A_i(i=1,2,\cdots,n)$，$B$ 均为 $f(x)=f(y)$ 型的式子，即由一些等式推出另外一些等式）。可以按照如下的方法进行证明。

（1）对 $B$ 的等号左边的式子从 $A_i$ 中寻找匹配，并进行代换；

（2）对 $B$ 的等号右边的式子从 $A_i$ 中寻找匹配，并进行代换；

（3）分别单独重复进行（1），（2），直到等号左右两边完全相等。

其中匹配的原则如下：

（1）$x$ 可以与 $x$ 匹配，但不可以与 $y$ 匹配，其中 $x\neq y$；

（2）$x$ 可以与 $\forall z f(z)$ 中的 $z$ 匹配。

代换的原则为：

将找到的对应匹配的字母取代为进行匹配的字母即得到式子。

### 2.2 正确性分析

（1）正向推导。

根据等词的传递性，即如果有 $A=B$ 及 $B=C$ 成立，则可以知道 $B=C$。

例如现在证 $A=B,B=C\mapsto A=C$，可以将右边的 $A$ 代换为 $B$（根据 $A=B$），因为由前提知道 $A$ 与 $B$ 是等价的，即 $A=B,B=C\mapsto B=C$。

（2）反向推导。

由于等词具有对称性，再结合（1）的说明，即如果有 $A=B$ 及 $A=C$ 成立，则可以知道 $B=C$。

例如现在证 $A=B,A=C\mapsto B=C$，可以将右边的 $B$ 代换为 $A$（根据 $A=B$），因为由前提知道 $A$ 与 $B$ 是等价的，即 $A=B,A=C\mapsto A=C$。

（3）反复利用上面的正向与反向推导，将右边的将要推出的部分进行代换，直到等式的左右两边完全相同为止。

## 3 例子分析

### 3.1 要证明如下的式子

$(\forall x.0+x=x)\cap(\forall x.\forall y.\mathrm{suc}(x)+y=\mathrm{suc}(x$

$+y)) \rightarrow \forall x.\ \forall y.\ (x+y=y+x)$(其中 suc 表示 $x$ 的后继,比如 2 为 1 的后继,10 为 9 的后继,等)

数学推导过程如下:

(1)首先利用蕴含右规则,将题中的式子化为:

$(\forall x.0+x=x) \cap (\forall x.\ \forall y.\ \mathrm{suc}(x)+y=\mathrm{suc}(x+y)) \mapsto \forall x.\ \forall y.\ (x+y=y+x)$

(suc $(x)$ 表示 $x$ 的后继)

(2)利用合取左规则,将左边的式子进行分解:

$(\forall x.0+x=x),(\forall x.\ \forall y.\ \mathrm{suc}(x)+y=\mathrm{suc}(x+y)) \mapsto \forall x.\ \forall y.\ (x+y=y+x)$

(3)利用数学归纳法中的策略,将式子分为两个子式:

$(\forall x.0+x=x),(\forall x.\ \forall y.\ \mathrm{suc}(x)+y=\mathrm{suc}(x+y)) \mapsto \forall y.\ (0+y=y+0)$

$\forall y.\ (x+y=y+x),(\forall x.0+x=x),(\forall x.\ \forall y.\ \mathrm{suc}(x)+y=\mathrm{suc}(x+y)) \mapsto \forall y.\ (\mathrm{suc}(x)+y=y+\mathrm{suc}(x))$

(4)再次对两个式子利用上述策略,得到四个式子:

①$(\forall x.0+x=x),(\forall x.\ \forall y.\ \mathrm{suc}(x)+y=\mathrm{suc}(x+y)) \mapsto (0+0=0+0)$

②$(0+y=y+0),(\forall x.0+x=x),(\forall x.\ \forall y.\ \mathrm{suc}(x)+y=\mathrm{suc}(x+y)) \mapsto (0+\mathrm{suc}(y)=\mathrm{suc}(y)+0)$

③$\forall y.\ (x+y=y+x),(\forall x.0+x=x),(\forall x.\ \forall y.\ \mathrm{suc}(x)+y=\mathrm{suc}(x+y)) \mapsto (\mathrm{suc}(x)+0=0+\mathrm{suc}(x))$

④$(\mathrm{suc}(x)+y=y+\mathrm{suc}(x)),\forall y.\ (x+y=y+x),(\forall x.0+x=x),(\forall x.\ \forall y.\ \mathrm{suc}(x)+y=\mathrm{suc}(x+y)) \mapsto (\mathrm{suc}(x)+\mathrm{suc}(y)=\mathrm{suc}(y)+\mathrm{suc}(x))$

现在对以上四个子目标分别证明。

①式:

由于等号的两边是完全一致的,故显然成立。

②式:

分别对其各个式子进行编号:

$\underbrace{(0+y=y+0)}_{(1)},\underbrace{(\forall x.0+x=x)}_{(2)},\underbrace{(\forall x.\ \forall y.\ \mathrm{suc}(x)+y=\mathrm{suc}(x+y))}_{(3)} \mapsto \underbrace{(0+\mathrm{suc}(y)=\mathrm{suc}(y)+0)}_{(4)}$

对(4)进行变换:

左边

$=0+\mathrm{suc}(y)$

$=\mathrm{suc}(y)$ ----〈全部,依据(2),正向推导〉

右边

$=\mathrm{suc}(y)+0$

$=\mathrm{suc}(y+0)$ -----〈全部,依据(3),正向推导〉

$=\mathrm{suc}(0+y)$ ---〈suc 内部,依据(1),反向推导〉

$=\mathrm{suc}(y)$ ------〈suc 内部,依据(2),正向推导〉

=左边

③式:

$\underbrace{\forall y.\ (x+y=y+x)}_{(1)},\underbrace{(\forall x.0+x=x)}_{(2)},\underbrace{(\forall x.\ \forall y.\ \mathrm{suc}(x)+y=\mathrm{suc}(x+y))}_{(3)} \mapsto \underbrace{\mathrm{suc}(x)+0=0+\mathrm{suc}(x)}_{(4)}$

对(4)式变换:

左边 $=\mathrm{suc}(x)+0$

$=\mathrm{suc}(x+0)$ -----〈全部,依据(3),正推导〉

$=\mathrm{suc}(0+x)$ --〈suc 内部,依据(1),正推导〉

$=\mathrm{suc}(x)$ -----〈suc 内部,依据(2),正推导〉

右边 $=0+\mathrm{suc}(x)$

$=\mathrm{suc}(x)$ -----〈全部,依据(2),正向推导〉

=左边

④式:

$\underbrace{(\mathrm{suc}(x)+y=y+\mathrm{suc}(x))}_{(1)},\underbrace{\forall y.\ (x+y=y+x)}_{(2)},\underbrace{(\forall x.0+x=x)}_{(3)},\underbrace{(\forall x.\ \forall y.\ \mathrm{suc}(x)+y=\mathrm{suc}(x+y))}_{(4)} \mapsto \underbrace{(\mathrm{suc}(x)+\mathrm{suc}(y)=\mathrm{suc}(y)+\mathrm{suc}(x))}_{(5)}$

对(4)式变换:

左边

$=\mathrm{suc}(x)+\mathrm{suc}(y)$

$=\mathrm{suc}(x+\mathrm{suc}(y))$ ---〈全部,依据(4),正向推导〉

$=\mathrm{suc}(\mathrm{suc}(y)+x)$ ----〈suc 内部,依据(2),正向推导〉

$=\mathrm{suc}(\mathrm{suc}(y+x))$ ---〈suc 内部,依据(4),正向推导〉

$=\mathrm{suc}(\mathrm{suc}(x+y))$ --------

----------〈suc 内部的 suc 内部,依据(2),反向推导〉

右边

$=\mathrm{suc}(y)+\mathrm{suc}(x)$

$=\mathrm{suc}(y+\mathrm{suc}(x))$ ------〈全部,依据(4),正向推导〉

$=\mathrm{suc}(\mathrm{suc}(x)+y))$ ----〈suc 内部,依据(1),反向推导〉

$=\mathrm{suc}(\mathrm{suc}(x+y))$ -----〈suc 内部,依据(4),正向推导〉

=左边

### 3.2 代　码

#### 3.2.1 伪代码

控制代换的伪代码如图 1 所示:

```
rep suc_n_out left_or_right replace_n direction =
    {
    if left_or_right = 左边 then
    formula_to_match =  结论的左边
       else if left_or_right = 右边 then
    formula_to_match = 结论的右边
       else .
    Error

        然后,将 formula_to_match 除去外层的 suc_n_out 个
suc,比如 suc_n_out 为 0,则一个也不去掉;如果为 1 则去掉 1 个
……产生 formula_to_match2

        如果 direction 为"从左向右",则从前提中的等号左边
找匹配;否则,从前提中的等号右边找匹配;
        找到第 replace_n 个匹配,然后将 formula_to_match2 代
换为匹配所在等式的右边。
    }
```

图 1 控制代换的伪代码

函数名为 rep，后面四个变量为其参数。其中第一个 suc_n_out 表示外面有几个 suc，其对应的是 3.1 中的理论推导的尖括号内的推导依据(以下简称"推导依据")的第一个参数;left_or_right 表示是对等式左边的进行推导还是对右边的进行推导;replace_n 表示的是按照第几个匹配进行代换,其是推导依据的第二个的变形,上面的推导依据直接是"依据 n",此处是自动进行匹配,将找到的匹配返回来;最后一个参数 direction 表示是从左向右推导还是从右向左推导,其控制着是从前面的前提的等式的左边还是右边进行匹配。

3.2.2 推导实例

$(0+y=y+0),(\forall x.0+x=x),(\forall x.\ \forall y.\ \mathrm{suc}(x)+y=\mathrm{suc}(x+y))\mapsto(0+\mathrm{suc}(y)=\mathrm{suc}(y)+0)$

rep 0 左边 0 从左向右
rep 0 右边 0 从左向右
rep 1 右边 1 从右向左
rep 1 右边 0 从左向右

$\forall y.(x+y=y+x),(\forall x.0+x=x),(\forall x.\ \forall y.\ \mathrm{suc}(x)+y=\mathrm{suc}(x+y))\mapsto(\mathrm{suc}(x)+0=0+\mathrm{suc}(x))$

rep 0 左边 0 从左向右
rep 1 左边 0 从左向右
rep 1 左边 0 从左向右
rep 0 右边 0 从左向右

$(\mathrm{suc}(x)+y=y+\mathrm{suc}(x)),\forall y.(x+y=y+x),(\forall x.0+x=x),(\forall x.\ \forall y.\ \mathrm{suc}(x)+y=\mathrm{suc}(x+y))\mapsto(\mathrm{suc}(x)+\mathrm{suc}(y)=\mathrm{suc}(y)+\mathrm{suc}(x))$

rep 0 左边 0 从左向右
rep 1 左边 0 从左向右
rep 1 左边 0 从左向右
rep 0 右边 0 从左向右
rep 1 右边 1 从右向左
rep 1 右边 0 从左向右
rep 2 右边 0 从左向右
得到结果如图 2 所示：

```
val st36:unit=()
(ALL x.0+x=x) & (ALL x.  ALL y.suc(x)+y=suc(x+y))-->
(ALL x.ALL y.x+y=y+x)
No subgoals left!
```

图 2 程序运行结果

## 4 结束语

文章首先盘点了关于自动证明的一些基本情况，然后提出了自己就自动证明的一些认识，并且以一个逻辑证明为例，进一步阐述了自己的观点。所提到的程序还没有实现完全的自动化，还需要一些策略，可以省去输入一些复杂参数的麻烦。并且程序比较复杂，本身要的实现工作量就很大，要实现更好的效果还需要更大的任务，且不好控制。

下一步的工作就是实现程序的自动运行，去除那些输入的参数，实现自动证明。

**参考文献：**


[1] 王世强，别荣芳，史 璟. 关于机器证明[J]. 前沿科学，2011,5(1):76-77.

[2] Paulson L C. Introduction to Isabelle[M]. Cambridge:University of Cambridge,2004.

[3] 傅海伦. 定理机器证明思想的产生与发展[J]. 科技导报，2001(6):14-30.

[4] 张景中，李永彬. 几何定理机器证明三十年[J]. 系统科学与数学，2009,29(9):1155-1168.

[5] 徐志才. 四色问题的探讨[J]. 北京邮电大学学报，2003(2):105-112.

[6] Wu W T. On the decision problem and the mechanization of theorem-proving in elementary geometry[J]. Scientia Sinica, 1978(21):159-172.

[7] 吴文俊. 初等几何判定问题与机械化证明[J]. 中国科学(A),1977(7):507-516.

[8] 胡世华. 信息时代的数学[J]. 数学进展，1988,17(1):12-20.

[9] Paulson L C. ML for the Working Programmer[M]. Cambridge:University of Cambridge,1996.

[10] Leroy X. The objective Caml system release 3.10-documentation and user's manual[M]. Ville de Paris:Institut National de Recherche en Informatique et en Automatique,2007:11-23.


(下转第 146 页)

Windows 系统。

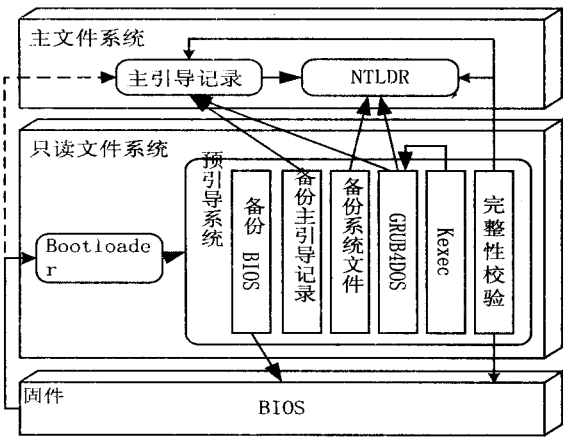


图 3 引导 Windows 的系统架构图

如果采用安装在 USB 存储设备上的预引导系统启动 Windows，USB 设备可能被识别成第一磁盘(hd0)，原第一硬盘将被识别成第二磁盘(hd1)，由于 Windows 限制引导非安装在第一磁盘的系统，所以在配置 GRUB4DOS 时，需要对磁盘进行映射，使用的 GRUB4DOS 命令序列如下：

```
map(hd0)(hd1)
map(hd1)(hd0)
rootnoverify(hd1,0)
makeactive
```

若将预引导系统存放在光盘中，不需要进行上述磁盘映射的操作。

GRUB4DOS 加载 Windows 有 2 种方式，一种是直接加载 ntldr 文件(注：ntldr 是 Windows NT 5.x 系列操作系统的引导文件，而 Windows NT 6.x 系列则使用文件 bootmgr)，另一种是先加载磁盘的 MBR，再由 MBR 来加载 ntldr 文件，在 GRUB4DOS 中分别使用命令 chainloader +1 与 chainloader /ntldr 实现上述 2 种加载方式。文中支持上述 2 种加载方式，以提高加载成功率。

## 2 结束语

文中设计的安全引导方式在虚拟机 VirtualBox 和实际的 PC 机上实现，其中 PC 机采用 USB-CD 启动，分析安全引导增加的系统启动时延。虚拟机的平均增加启动时延在 10 秒以内，在 PC 机上实验，平均时延为 18 秒(不含等待用户选择的时间)。实验表明文中的安全引导的方式对整个系统的启动时间的影响是可以接受的。

**参考文献：**


[1] Parno B. Bootstrapping trust in a "trusted" platform[C]// Proceedings of the 3rd conference on hot topics in security. Berkeley, CA, USA: USENIX Association, 2008.

[2] 陈书义，闻英友，赵 宏. 基于条件谓词逻辑的可信计算形式化分析[J]. 华南理工大学学报：自然科学版，2009，37(5)：106-110.

[3] TPM Main Specification Level 2 Version 1.2, Revision 116 [EB/OL]. (2011-03-01)[2011-11-14]. http://www.trustedcomputinggroup.org/developers/trusted_platform_module/specifications.

[4] 陈建勋，侯方勇，李 磊. 可信计算研究[J]. 计算机技术与发展，2010，20(9)：1-4.

[5] 张 颖，周长胜. EFI 下基于便携式 TPM 的可信计算平台研究[J]. 计算机技术与发展，2010，20(1)：167-171.

[6] Horman S. Kexec[EB/OL]. 2010[2011-11-14]. http://horms.net/projects/kexec/.

[7] Nellitheertha H. Reboot Linux faster using kexec[EB/OL]. 2004[2011-11-14]. http://www.ibm.com/developerworks/linux/library/l-kexec/index.html.

[8] Heasman J. Implementing and Detecting an ACPI BIOS Rootkit[C]//Blackhat Federal 2006. Washington, DC: [s.n.], 2006.

[9] Shingledecker R. Tiny Core Linux[EB/OL]. 2008[2011-11-14]. http://www.tinycorelinux.com.

[10] Advanced Computing Laboratory at Los Alamos National Laboratory (LANL). Coreboot[EB/OL]. (2011-06-24)[2011-11-14]. http://www.coreboot.org/.

[11] Eastlake D, Jones P. US Secure Hash Algorithm 1 (SHA1)[S]. Internet RFC 3174, 2001.

[12] GRUB4DOS and WINGRUB[EB/OL]. 2009[2011-11-14]. http://grub4dos.sourceforge.net/.


(上接第 138 页)


[11] de Rauglaudre D. Camlp5-Reference Manual[M]. Ville de Paris: Institut National de Recherche en Informatique et Automatique, 2008: 45-61.

[12] 刘 全，孙吉贵. 基于语义 tableau 的一阶逻辑自动定理证明[J]. 计算机工程与应用，2005(23)：22-24.

[13] 刘 全. 基于 tableau 的自动推理研究[D]. 长春：吉林大学，2004.

[14] 陆汝钤. 强有序输入消解原理[J]. 中国科学，1981(8)：1035-1042.

[15] 王元元. 计算机科学中的现代逻辑学[M]. 北京：科学出版社，2001：74-100.

[16] 张灵峰，夏战锋，彭志平. 基于 Tbox 和 Abox 的描述逻辑推理研究[J]. 计算机技术与发展，2010，20(11)：122-125.