计算机科学1992 Vol.19 No.5

微机上实现的逻辑推理语言—Tuili 1.1

高全泉　陆汝钤　（中国科学院数学研究所，北京100080）

摘　要

Tuili is a new AI Language which has many kind of logic inference functions and high flexible control mechanism, and it can perform logic inference with different directions and different search modes. It is more powerful, more convenient and more effictive and suitable to building expert systems and various knowledge-based system quickly than Prolog. In this paper, we will introduce a version of Tuili which has been implemented on micro computer.

一、概　述

Tuili是一个基于谓词逻辑的人工智能语言，具有自然的说明性知识表示方式和能够运用这种说明性知识进行多种推理的显著特点。Tuili的设计不但成功地解决了将产生式语言、逻辑程序设计和过程型语言有机结合的问题，而且在知识程序的模块化结构、元级推理以及计算功能等方面有重要创新，因而比目前其它的一些人工智能语言，如OPS5、LISP、Prolog等，更适合建造专家系统和基于知识的系统。Tuili以其设计新颖、推理功能丰富、高度灵活的控制结构以及表达形式的优雅，受到了国内外同行的普遍重视和高度评价。Tuili一词具有双重含义，既是中文"推理"的拼音，也是英文短语 "Tool of Universal Interactive Logic Inference" 之简写。Tuili语言文本由本文第二位作者陆汝钤设计。文本的设计始于1983年，完成于1986年。

Tuili是一个功能很强的逻辑推理语言。Tuili中的许多语言技术是多年研究的结果，是国内外其它逻辑推理语言中所没有的。它能否在计算机上有效地实现是人们普遍关心的问题。无疑，语言的独创性和新颖性势必增加实现的难度，向实现技术提出新的问题。当Tuili在国际上首次提出时，有关专家在赞扬和高度评价的同时，也对Tuili的实现可能性提出疑问。从1986年起，本文第一作者高全泉开始设计和实现Tuili系统，到1990年，在UV-68020微机上用C语言实现了Tuili语言的基本集——Tuili1.1。之后，由刘晓华移植到PC系列微机。Tuili1.1已在一些部门实际应用，取得了很好的效果。以后，本文提到的Tuili，均指这一已经实现的版本。

二、Tuili程序结构

Tuili以产生式系统为主要的知识表示方法，它的知识基分为四部分，即：图象基、数据基、过程基和规则基。

Tuili的四个组成部分具有不同的作用。简单说来，图象基是程序中基本对象的说明部分，数据基是程序的数据空间，过程基是用Tuili以外的过程型语言定义的程序体，规则基是程序的主体，可由一般规则和元规则组成。数据基和过程基在某些特定的情况下可以缺省。图象基和规则基是必需的成份。

1. 图象基

图象基（pattern-base）也叫模式库。Tuili以谓词为知识表示的基本单位，把每个

谓词看作一个图象（或模式），以图象匹配（matching）作为推理的基本方式。图象分为谓词图象和参数图象。

参数图象由图象元构成。图象元包括变量、常量和形式函数三种。其中，形式函数相当于Prolog的结构项，只供形式上的匹配，并不具有计算（函数调用）的意义。图象元用于定义谓词图象。每个图象元都有类型，因而必须在定义谓词图象之前，首先定义图象元的类型。图象元的类型是八种基本类型之一。这八种基本类型是：

- 整型（int）
- 实型(real)
- 布尔型(bool)
- 串型(string)
- 串元型(elem，字符串和变量组成的表达式)
- 表型（list）
- 记录型(record)
- 任意型(any)

在图象基中给出的是形式图象元，用于描述形参图象。其它图象元包括Tuili固有的常量、变量名以及Tuili 固有的函数调用、内部谓词、内部过程以及过程基中说明的函数过程调用。

在图象基中定义的谓词图象由谓词名后跟一组（排好序的）形参图象组成。它规定了将来代入实参图象（在数据基中）和半实参图象（在规则基中）对必须遵循的关于参数个数、次序和类型的限制。

在说明一个常量时，可以带定义值。凡是带“=〈直接量〉”说明的常量名，当它在数据基或规则基中出现时，一律在程序运行前自动换成相应的直接量。

在用户自定义谓词前加上问号“？”，即可在规则中作为咨询谓词使用。咨询谓词的真假值（包括其参数的约束值）在执行阶段通过询问用户获得。咨询谓词的使用是简化交互式程序设计的有效途径。希望用户回答的问题和对这一问题的解释作为谓词说明的一部分在图象基里说明，咨询谓词加上惊叹号“！”表示把通过咨询获得的数据存入系统数据基systore备用，当程序再次执行到这咨询谓词时，不再向用户提问。

2．数据基

为了有别于通常意义下的数据库，Tuili中的database叫做数据基，其内容是已知为真的谓词。数据基定义推理的数据搜索空间，有两种用法。一是用于输入，好比通常的函数或过程调用的输入参数，作为推理的已知前提。一是用于输出，用于存放经过推理获得或证明了的结果。在一个推理任务（或子任务）中，可以指定一个或多个数据基作为输入数据基，只能指定一个数据基作为输出数据基。一个数据基可以既作输入又作输出之用。

在数据基中出现的谓词，其图象必须是在图象基中定义过的。每个数据基就是一个世界，同一个谓词在不同的世界中可以有不同的真假值，若谓词p在一个数据基中出现，则任何一个谓词p′，凡能匹配到 p者，匹配以后的谓词实例在此世界内均取真值。

在数据基部分出现的数据基是用户自行定义的数据基。系统自身有四个内部数据基，其名字和作用如下：

- **sysin** 系统输入数据基。这是一个逻辑数据基，其内容为系统提供的内部谓词。
- **sysout** 系统输出数据基。当用户在程序中未指明输出数据基时，自动作为输出数据基之用。
- **systore** 系统样品数据基。用于存放咨询结果。
- **sysgoal** 系统目标数据基。用于存放向后推理成功时目标谓词的实例。其作用是使得向后推理也产生新的数据。

3．过程基

过程基是用Tuili 以外的过程性语言编写的程序体，允许使用的过程性语言是C语言。过程分为函数过程和普通过程，前者有值，后者无值，取布尔值的函数过程称为布尔过程。

对于在过程基中定义过的过程，相应的过程调用的使用规则如下：

• 函数过程调用可以出现在谓词或另一个过程调用的参数中。

• 布尔过程调用可以作为左部元出现在规则的左部。

• 普通过程调用可以作为右部元出现在推理规则的右部。

• 任何外部过程调用都可出现在任一过程定义的过程体中。

4. 规则基

规则基由规则组成，每个规则代表推理的一个步骤。每个规则基表示在一个推理过程（或子过程）中允许使用的规则的全体。每个规则基都有一个级别，级别数愈小，级别愈高，高级规则基可以通过系统提供的控制过程控制低级规则基的运行，起着元规则的作用。零级规则基的级别是最高的，整个系统由此启动。

若上级规则基中一条规则的右部有系统固有过程调用run（<下级规则基名>），则上级规则基运行至此时即转入下级规则基的运行。前者称为直接上级，后者称为直接下级。该下级规则基的运行策略，由和上述run调用位于同一规则右部的其它系统固有过程调用指明，指明不完全时，由缺省规则决定。

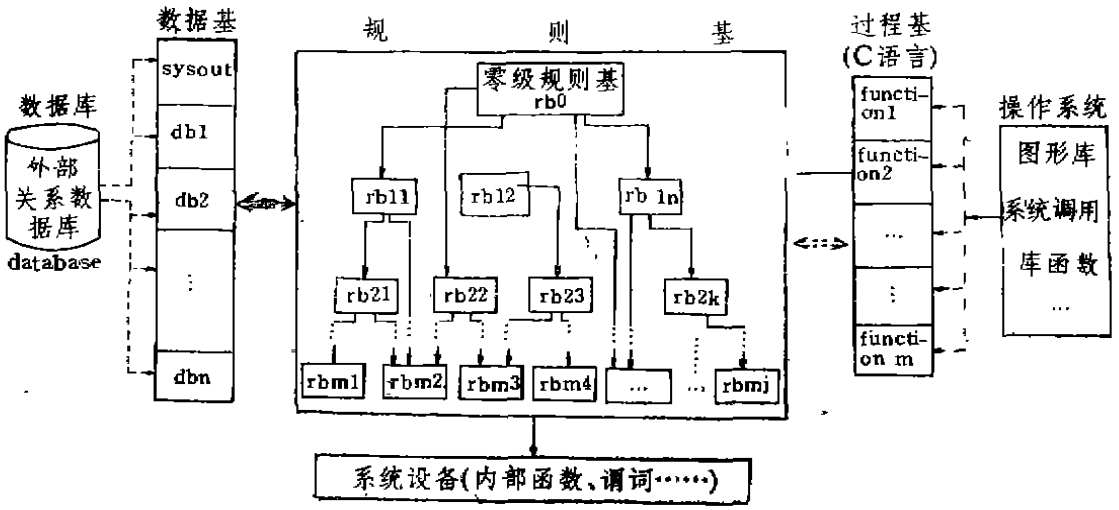
运行一个规则基就是反复做如下事情：用输入数据基或推理目标（可以为一单独的目标序列，也可指定一个数据基的内容作为推理目标）和规则里的谓词条件向前匹配（从左到右执行规则）或向后匹配（与规则右部谓词匹配），包括执行规则中含有的过程调用，同时按控制策略规定的方式搜索，直到下列五种情形之一出现。

1）完成了（或部分完成了）上级规则基指定的任务，回到上级规则基中标号为done（或success）的规则继续运行。

2）上级规则基指定的目标尚未达到，但已不能再通过匹配产生新的结果，或者已经到了由限制函数 limitf 规定的限制条件的边界，此时，即分别回到上级规则基中标号为 fail 和 limit 的地方继续运行。

3）推理过程中遇到了规则中的中断名，或具备了出现系统中断的条件，即中止本规则基的运行，回到上级规则基中以相应中断名为标号的规则处继续运行。若找不到，继续向上找，最终找不到，则返回顶级命令控制。

4）执行运行下一级规则基的命令后转



*虚线表示信息流 *实线表示规则基的支配关系

图1 Tuili 的模块结构图.（上层的规则基rbi 可以启动下层的规则基rbj，对任一i, j, k, dbi, functionj 可为 rbk 引用）

入更下一级规则基的运行。

5）遇到系统过程调用 pause 即暂停运行，等待操作员命令；遇到系统过程调用 stop，即结束运行，回到操作系统命令级。

规则基的控制关系以及规则基与数据基等成份的引用关系如图1所示。

规则基中的规则是产生式规则，由左部和右部两部分组成。左部是条件部分（或曰子目标），各个条件或者子目标间的关系是“与”。它们可以是半实谓词、咨询谓词、布尔过程调用、中断名、系统固有谓词以及一些系统固有过程。右部是动作和结论（或曰目标）部分。目标是半实谓词，动作是普通过程调用或中断名。一规则右部可有多个动作或结论，它们在向前推理时左部诸条件都满足之时，被依次执行—或执行过程调用或产生新的谓词实例，或执行中断（下级规则基返回上级规则基）。在向后推理时，要求右部都是以半实谓词形式出现的目标，此时，右部中的诸半实谓词的关系是“或”，即如果这些半实谓词中有一个能与当前的目标调用匹配成功，则认为右部匹配成功，并执行变量约束，然后执行左部诸子目标。

三、Tuili程序举例

以下，我们给出一个 Tuili 程序实例，以便对 Tuili 程序结构有基本的了解。

例：乘车问题。从甲地出发要到达乙地，选择一条可到达目的地的乘车路线并计算出所需票价。

```
tuili (traffic), /* Tuili 程序首部*/
  pattern-base, /* 图象基*/
    var  k, i, j, num, num1:int; /*  变量说
                                    明*/
      stop, stop1, stop2:elem;
      x, y: list;
      cash1, cash2:int;
    pred from (stop): "你现在在哪里？ "; /*
                                    谓词说明*/
       to (stop): "你要到哪里？ ";
       node (stop, num, i);
       take (stop, num, i);
       cross (stop);
       reach (stop1, stop2, x, num);
       cash (num, i, j);
eof-pb; /*图象基到此结束*/
database (bus); /*数据基*/
  node ("白石桥", 320, 7); /*车站白石桥是
                          320路的第7站*/
  node ("中关村", 332, 8),
  node ("动物园", 103, 16);
  node (白石桥", 332, 2);
  node ("动物园", 332, 1);
  node ("木樨地", 0, 8); /*0路表示地铁*/
        ……
  node ("北京站", 103, 1);
  cross ("动物园"); /*换乘站*/
  cross ("木樨地");
  cross ("人民大学");
  cross ("白石桥");
  cash (0, 0, 50);       /*票价，地铁一律
                          5角*/
  cash (103,1,2);      /*103路，每站2分*/
  cash (104, 1, 2);
  cash (111, 1, 2);
  cash (332, 1, 4);
  cash (1, 0, 20);    /*大1路，一律2角*/
  cash (320, 1, 4);
  ……
eof bus,          /*数据基bus到此结束*/
procbase (compute);  /*过程基*/
  sum1 (j, k, l)      /*C语言写的票价计算
                          函数*/
  int j, k, li
  {
  if (k==0) return(1);
  else  return (j * l);
  }
eof  compute          /*过程基结束*/
rulebase (control, 0);    /*0级规则基*/
  start: ? from (stop1)↑∧? to (stop2)↑→in
       (bus) ∧out (bus)∧goals(reach (stop
      1,stop2,x,i))∧ford∧
      breadth∧run(search); /*执行宽度优先
                          向前搜索*/
  done: →print3∧stop;
eof control;
```

```
rulebase(search, 1);
  nowork: ? from(stop) ∧ ? to(stop) →reach
          (stop, stop, (), 0);
  geton: ? from(stop) ∧ node(stop, num, i)
       →take(stop, num, i);
  arrive: take(stop, num,i) ∧ ? to(stop1) ∧
    node(stop1, num, j) ∧ cash(num,k, l)→
  reach (stop, stop1, (num, stop, stop1),
  sum1(abs(i-j), k, l));
  getoff: take(stop, num, i) ∧ node(stop1,
  num, j) ∧ cross(stop1) ∧ nq(stop, stop1)
  ∧ cash(num, k, l)→take (stop1, num,
  j) ∧ reach(stop, stop1, (num,stop,
  stop1), sum1(abs(i-j), k,l));
  change: take(stop, num,i) ∧ node(stop,
  num1, j) ∧ nq(num, num1)→ take (stop,
    num1, j);
  connect, reach(stop, stop1, x, cash1) ∧
  reach(stop1, stop2, y, cash2) ∧
    nq(stop, stop2)→reach (stop,stop2,(x,
  y), cash1+cash2);
eof search;
eof tuili;      /*Tuili: 程序到此结束*/
```

这一程序在计算机上的运行记录如下：

```
…Now, your program is running, please wait
for result…
  你现在在哪里?
   (please input value of variable: stop1)
  中关村↵     /*程序员对咨询谓词的回答*/
  你要到哪里?
   (please input value of variable: stop2)
  北京站↵   /*程序员对咨询谓词的回答*/
  The results of forward reasoning are in the
  database: bus;
  The contents are as follows:
                ……
    reach (中关村, 北京站,
      ((332, (中关村, (动物园, nil))),
      ((103, (动物园, (北京站, nil))),
      nil)), 58);
```

注：从中关村到北京站的乘车路线：乘332，从中关村到动物园，换乘103，从动物园到北京站，票价是5角8分（只是举例，并非实价）。

四、匹　配

匹配是两个谓词图象（简称谓词）之间的一种动态对应和相互约束关系。在匹配时，首先判断两个同名谓词之间是否存在最广通代。存在则匹配成功，否则匹配失败。在存在最广通代的情况下，把该最广通代中的变量置换施行于这两个谓词中的所有有关变量。在匹配过程中，若有一变量名x被约束为表达式e，则x所在的谓词、过程调用及规则中的所有同名变量皆应同时约束为e。

两个同名谓词匹配当且仅当它们的参数顺序两两匹配。参数是基本匹配对象，每个参数都可以是一个表达式。表达式有以下几种：

• **算术表达式**　与通常过程性语言一致。内部函数和外部过程(在过程基里定义)调用可作为运算分量。

• **形式表达式**　即形式函数，类似于Prolog的结构项。

• **记录表达式**　一维向量，其形如〔5, x, 6, y〕。

• **串元表达式**　由字符串、串连接符"/"和常量、变量组成的表达式。如，"中国科学院"/x/"研究所"即是一串元表达式。

• **表表达式**　包含两类。一类与Prolog中的表一致。另一类是表的操作函数，它们完成取表头、取表尾和表的合成，如LISP中的car、cdr和cons那样。

• **原子**　原子表达式是上述各类表达式的特例（不含运算符）。包括：整数、实数、常量、变量、字符串。

在将作为参数的一对表达式匹配时，使用以下规则。

• 变量可与同类型的任意表达式e匹配。

• 两个常量值相等可以匹配，常量和等值常量名匹配。

• 对所有的形式函数f，当两组表达式e_{1j}和e_{2j}（$1\leqslant j\leqslant n$）顺序两两匹配（即存在

置换φ，φ(e_{1j})=φ(e_{2j})，1≤j≤n)时，f(e_{11}…，e_{1n}) 与f(e_{21}，…，e_{2n}) 也匹配。

• 对表表达式L_1和L_2，若car(L_1) 和car(L_2) 匹配，且cdr(L_1) 和cdr(L_2) 是相匹配的表表达式，则L_1和L_2匹配。car、cdr分别表示表头、表尾。

• 两组表达式e_{1j}和e_{2j} (1≤j≤n) 顺序两两匹配时，记录表达式r_1=〔e_{11}，…，e_{1n}〕和r_2=〔e_{21}，…，e_{2n}〕也匹配。

• 两个串表达式的匹配分为将两个含运算符"/"的串元表达式匹配和将一字符串与一串元表达式进行匹配，其匹配规则较为复杂，我们只给出基本思想。对前者，要求两个串元表达式中的对应的运算分量两两匹配。对后者，需要根据串元表达式中的子串将字符串分解成一组子串，再对这两组分量进行匹配。

五、控制策略和系统设备

Tuili使用语言一级的控制策略，允许用户通过使用系统提供的控制过程，达到对推理方向、搜索方式等方面的控制。

1. 推理策略

推理策略分为向前推理和向后推理两种。向前推理形成一片推理树，即一组推理树。向前推理又分为带目标谓词和不带目标谓词。带目标谓词是指给出了推理的目标，一旦这目标达到，便中止推理。反之，不带目标未指出推理的目标，是一种盲推理，产生所有可能的推理树，中止条件是不能形成新的推理树（动力耗尽）或者时、空超限。向前推理是以（各）输入数据基（包括system-ore）中的实在谓词作树叶，和规则基中的规则实行向前匹配，产生新的节点，新节点同时加入到当时的输出数据基。如果这个输出数据基同时又是输入数据基，则由这个新节点（以及重复使用新老树叶）又可通过向前匹配构造更大的子树，产生更高层的子树根，这样得到的树即是所谓的推理树。

向后推理是面向目标的推理，必须给出推理的目标谓词。向后推理通过在规则基中搜索适合的规则，将目标分解为一组子目标，子目标再分解成下层的子目标，……，最下层的子目标用数据基中的实在谓词或事实（条件部分为空的规则）得到证明，如此便形成一棵与或树，这与 Prolog 形成的推理树基本一致。

2. 搜索策略

对于向前推理，可以选择的搜索方式有以下几种：

- 深度优先
- 最佳优先
- 广度优先
- 线性搜索
- 简单规则优先
- 简单搜索
- 循环优先

其中的一些搜索方式是 Tuili 特有的。如，最佳优先是根据用户自定义的启发式函数进行搜索；再如，线性搜索是指对规则基中的规则只执行一遍，等等。

对向后推理，目前只有深度优先一种搜索方式。Prolog 中的搜索即是由系统限定的深度优先搜索。

3. 求解策略

求解策略有：求严格解、求第一个解和求所有解（对前推理，求所有解采取了一些限制，以避免无穷的推理）。

4. 限制策略

为了防止无穷的推理过程，以及过长的推理超过了计算机所能应付的时间和空间复杂性等问题，可由上级规则基指定推理限制策略，以便在一定条件下中断推理过程。限制策略可以对推理的深度、宽度、时间、空间进行限制。

5. 缺省策略

用户的Tuili 程序中，当控制推理的元规则中关于推理的信息不完全时，以及程序中未安排明显的出口等情形，使用系统的缺省策略。比如，如果运行一个规则基时未指明推理方向，则系统认为是向前推理。若未指明搜索方式，则向前推理时按广度优先，

向后推理时按深度优先、等等。

6. 系统设备

系统设备是系统的可直接使用的成份，涉及多方面的功能，主要有：内部标号、内部函数、内部过程（包括控制过程）、内部谓词、系统输入/出数据基等，总共一百多个。其中，既有可在程序中出现也可在系统命令级菜单状态下使用的两用过程，也有能以不确定个参数使用的内部过程和内部函数。

此外，系统还具有许多潜在的功能，如事件驱动的精灵（demo）机制等。它们与系统设备一起，能极大地方便用户的程序设计。

Tuili1.1系统是一个已经投入实用的系统。新近推出的Tuili1.2是一个具有友好的用户界面的程序开发环境。用户在此环境下，能够完成Tuili程序的编辑、编译和运行。Tuili语言采用编译方法实现，系统具有良好的可移植性（系统本身及生成的目标语言都是C语言），较高的运行效率，并与其它软件有着有效的接口。一方面，可在Tuili程序中用C语言编程，另一方面，Tuili系统在实现上保证了向下兼容Prolog的推理功能，Prolog程序只需经过形式上的变换，就可转换为Tuili规则。

参 考 文 献

〔1〕陆汝钤，关于Tuili的设计，计算机软件开发方法、工具和环境，西北大学出版社，西安，1985

〔2〕Lu Ruqian, Tuili—A language for designing expert systems, Advances in Chinese Computer Science, I, World Scientific Publishing House, Singapore, 1987

〔3〕陆汝钤，Tuili语言文本，中科院数学所研究报告，1986

〔4〕陆汝钤，高全泉，UV—68000微机上实现的Tuili1.1文本及使用手册，中科院数学所研究报告，1990、12

〔5〕高全泉，陆汝钤，刘晓华，通用逻辑推理语言Tuili1.2使用说明，中科院数学所研究报告，1991.7

〔6〕高全泉，Tuili（推理）语言的编译方法与实现技术，软件学报，1991年第2期

〔7〕高全泉，Tuili实现系统中的多推理机结构，计算机学报，1991年第10期

计算机应用数据库介绍

中国科技情报所重庆分所经过三年多的开发，已在微机上建成“计算机应用数据库”。该库系文献型数据库，目前存有二次文献两万余篇，从1991年起已向国内各单位提供机读产品软盘，专业内容为计算机在工程技术、事务管理、数据处理、情报科学和文献加工等领域中的应用。该库每年新增文献一万篇，文献来源于国内外期刊、会议录、汇编、专著、报告和学位论文等4000余种。文献语种为中、英、俄、德、法、日等，每篇文献附有中文摘要。凡拥有微机的单位均可使用该库的机读产品软盘，建立文献检索数据库。需要详细资料和软盘者可函寄重庆2104信箱三室沈群，邮编：630013