Visual Prolog 的搜索控制机制分析*)

雷英杰[1,2] 王宝树[1] 赵 晔[2] 王 涛[2]

(西安电子科技大学计算机学院 西安710071)[1] (空军工程大学导弹学院 陕西三原713800)[2]

摘 要 回溯机制是逻辑程序设计的重要设施。回溯本身是一种获得目标所有可能解的良好方法。然而回溯也有副作用,一是它可能导致 Visual Prolog 给出多余的答案,而 Visual Prolog 自己不能区分实质上相同的两个解,因此会降低效率;二是尽管一个特殊的目标已被满足,但是回溯机制可能还会强迫 Visual Prolog 继续寻找另外的解,因此会增加系统开销。在这些情况下,必须仔细控制目标搜索求解的回溯过程。本文在揭示 Visual Prolog 回溯机制所存在问题的基础上,通过实例,对 Visual Prolog 的静态截断机制、失败谓词 fail 与否定谓词 not 等控制谓词,以及动态截断机制等所构成的完整的目标搜索求解控制机制进行了详细分析,从而揭示出回溯机制和搜索求解控制机制的本质特性及应用机理。

关键词 Visual Prolog,逻辑程序设计,编程语言,人工智能,专家系统

Analysis of the Control Mechanism of Searching in Visual Prolog

LEI Ying-Jie[1,2] WANG Bao-Shu[1] ZHAO Ye[2] WANG Tao[2]

(School of Computer Science and Engineering, Xidian University, Xi'an 710071)[1]

(Missile Institute, Air Force Engineering University, Sanyuan Shanxi 713800)[2]

Abstract The backtracking mechanism is an important facility for programming in logic. The backtracking itself is a good approach for obtaining all the possible solutions. However, the backtracking mechanism has side-effects. First, it may lead that Visual Prolog finds superabundant solutions out, and Visual Prolog itself cannot distinguish both the solutions of homology in essence, hence the efficiency is reduced. Second, a special goal is satisfied, but it would still constrain Visual Prolog to find an alternate solution, and overheads of the system in running time and memory used are increased. In these cases, it is necessary to control carefully the backtracking courses of searching for goals and problem-solving. On the basis of exposing the existed problems of backtracking mechanism in Visual Prolog illustrated with an instance, the full control mechanism of goal-searching and problem-solving, consisting of a static mechanism of cut, both the control predicates fail and not, and a dynamic mechanism of cut, are analyzed and investigated in detail. Thus the essential characteristics and fundamentals of applications of the backtracking mechanism and that of the full control mechanism of goal-searching and problem-solving are exposed.

Keywords Visual prolog, Programming in logic, Programming language, AI, Expert systems

Prolog 是人工智能与专家系统领域最著名的逻辑程序设计语言。Visual Prolog 意指可视化逻辑程序设计[1]。Visual Prolog 具有模式匹配、递归、回溯、对象机制、事实数据库和谓词库等强大功能[2]。它还支持模块化与面向对象程序设计、系统级编程、文件操作、字符串处理、位级运算、算术与逻辑运算,以及与其它编程语言的接口[3]。

回溯机制是逻辑程序设计的重要设施。回溯是一种获得目标所有可能解的良好方法。然而,回溯也有副作用,对 Visual Prolog 的搜索过程需要进行仔细的控制[4]。本文在考察 Visual Prolog 回溯机制基本问题的基础上,通过实例,对搜索求解控制机制进行了详细分析,以期揭示回溯机制和搜索求解控制机制的本质特性和应用机理。

1 回溯机制的基本问题

回溯是指示 Visual Prolog 到何处去寻找问题答案的机制。这种机制使得 Visual Prolog 具有了通过所有已知事实和规则进行搜索求解的能力。所谓回溯(backtrack),是指 Visual Prolog 采用"回退再试"的方式寻找给定问题的所有可能解的一种方法。当 Visual Prolog 开始为求解一个问题(或目标)寻找答案时,往往要在多种可能情况中做出抉择。它首先在分支点(即回溯点)设置好标志,然后选择要追踪的第一个子目标。如果该子目标失败,Visual Prolog 将回退到上一个回溯点尝试另一个目标。看下面的例子:

```
DOMAINS
    child = symbol
    age = integer
PREDICATES
    player(child, age)
CLAUSES
    player(peter,9).
    player(paul,10).
    player(chris,9).
    player(susan,9).
```

这个程序含有球类俱乐部成员的名字和年龄的有关事实。我们可以用 Visual Prolog 来安排一个球类俱乐部9岁年龄组的乒乓球比赛。每组要赛两场,目标是找到所有可能配对的俱乐部中9岁的球员。

利用回溯机制,Visual Prolog 不仅能找到问题的第一个解,而且能找到所有可能的解。但对于本例,Visual Prolog 不能判断出 Person1 = peter 和 Person2 = peter 是相同的,所以我们不得不显式地给出待求解的子目标:Person1 <> Person2。这就是回溯的副作用,它将导致 Visual Prolog 给出

*)基金项目:国防科技预研基金(51406030104DZ0120),教育部高等学校骨干教师资助计划项目(GG-810-90039-1003)。雷英杰 博士,教授,博士生导师,研究方向:智能信息处理与智能系统、智能决策等。王宝树 教授,博士生导师,研究方向:智能信息处理与模式识别、智能控制等。王 涛 博士生,研究方向:智能信息处理与智能决策。赵 晔 博士生,研究方向:智能信息处理与智能决策。

多余的答案，而 Visual Prolog 不能辨识实质上相同的两个解。

Visual Prolog 的回溯机制还可能产生不必要的搜索，因此会降低效率。譬如，当寻找给定问题的唯一解时可能要搜索好几遍。另外，尽管一个特殊的目标已被满足，但是回溯机制可能还会强迫 Visual Prolog 继续寻找另外的解。在这些情况下，必须控制回溯过程。

2 搜索求解控制机制

Visual Prolog 提供一组工具用来控制程序搜索求解的过程：失败谓词 fail 和否定谓词 not 等控制谓词，静态截断机制和动态截断机制，从而构成完整的搜索求解控制机制。

1) 失败谓词 fail，永远失败，用来强制产生回溯以寻找备选答案。

2) 否定谓词 not，当相关子目标不能被证明为真时，谓词 not 运行成功。

3) 静态截断机制，用来静态地阻断回溯。

4) 动态截断机制，通过两个标准谓词 getbacktrack 和 cutbacktrack 来实现，用来动态地阻断回溯。

截断允许丢弃任何已经存在的备份选项。所谓“截断(cut)”，其实质是指“一旦到达这一点，就忽视该谓词的可选子句和子句中早期目标的可选解”。这正是所需要的，因为可选项被排除了，所以再不需要堆栈帧，递归调用可以继续向前推进。

截断是关于思维的修饰。无论何时看到不确定性代码，都可以设置一个截断。当执行一个截断时，Visual Prolog 就认为没有未试过的可选项，因而也无须创建堆栈帧。

3 失败谓词 fail——强制回溯

当调用失败时，Visual Prolog 开始回溯。在一些情形下，往往需要为了找寻替代答案而强迫回溯。Visual Prolog 提供的特殊谓词 fail，其作用就是强制失败，从而导使回溯。考察下面的例子：

```
DOMAINS
    name = symbol
PREDICATES
    father(name, name)
    everybody
CLAUSES
    father(leonard,katherine).
    father(carl,jason).
    father(carl,marilyn).
    everybody:-
        father(X,Y),
        write(X," is ",Y,"'s father\n"),
        fail. /* 使用谓词 fail 强制产生回溯 */
```

假设我们希望找到 father(X,Y)的所有解。但是一个内部对 father 的调用只会找到一个解，这显然不是所需要的。本例程序中的谓词 everybody 使用谓词 fail 强制产生回溯，从而就可找到所有可能的解。通过强制 Visual Prolog 回溯经由 everybody 的规则体，谓词 everybody 即可通过回溯对 father(X,Y)产生更多的解。谓词 fail 永远不能被满足(它总是失败)，所以 Visual Prolog 被强制回溯。当回溯发生时，Visual Prolog 回溯到最近一个能产生多个解的调用。这个调用目标有一个不确定性(non-deterministic)标志。谓词 write 不能重新被满足，亦即它不能提供新的解，所以 Visual Prolog 一直要回溯到该规则的第一个子目标 father(X,Y)。

注意，在规则体中，把一个子目标放在谓词 fail 之后是无效的。因为谓词 fail 总是失败，所以位于谓词 fail 之后的子目标是不可到达的。

4 静态截断 cut——静态地阻断回溯

Visual Prolog 的静态截断机制，常简称为截断(cut)，可用来静态地阻断回溯。截断在程序编码中写作一个惊叹号(“!”)。越过截断进行回溯是不可能的，这就是截断的作用。

截断可以被放置在程序中，也可放置在规则体中，其方法与放置一个子目标的方法相同。当一个程序越过截断时，对截断的调用立即成功，下一个子目标(如果有的话)将被调用。一旦截断已经通过，在当前执行子句中回溯到截断前的子目标是不可能的，回溯到当前进程中定义的其它谓词(包含截断)也是不可能的。

截断主要有两种用途：

1) 当预先知道某些可能性永远不能给出有意义的解时，查找备选的解既浪费时间又浪费存储空间。如果在这样的情况下使用截断，程序最终将使用较少内存并运行得更快，即用截断来减小系统的时空开销，提高运行效率。这种截断称为绿色截断(green cut)。

2) 当程序逻辑需要用截断来防止考察备选的子目标时，即用截断来改变程序的逻辑。这种截断称为红色截断(red cut)。

4.1 阻断到规则中前一个子目标的回溯

截断可用来阻断到规则中前一个子目标的回溯。例如子句

```
r1 :- a, b, !, c.
```

它告诉 Visual Prolog，我们对它给子目标 a 和 b 找到的第一个解感到满意。虽然 Visual Prolog 能通过回溯调用 c 找到更多的答案，但是不允许越过截断回溯到调用 a 和 b 来寻找其他的解。回溯到另外一个定义谓词 r1的子句也是不允许的。

4.2 阻断到下一个子句的回溯

截断可以用作这样一个方法，以告诉 Visual Prolog 已经为一个特定谓词选择了正确的子句。例如，考察下面的代码：

```
r(1):- !, a, b, c.
r(2):- !, d.
r(3):- !, c.
r(_):- write("This is a catchall clause.").
```

用截断使谓词 r 成为确定性的。这里，Visual Prolog 带着一个整数参数调用谓词 r。假定调用是 r(1)，Visual Prolog 搜索程序，寻找匹配；Visual Prolog 找到定义 r 的第一个子句。因为该调用有不止一个可能的解，所以 Visual Prolog 在这个子句后面设置一个回溯点。规则被激活后，Visual Prolog 开始处理规则体。首先遇到截断，因此回溯到另一个 r 子句的可能性被排除。

4.3 用截断改变不确定性

如果一个谓词中没有截断，它将是一个不确定性谓词，亦即可以通过回溯产生多个解。在许多 Prolog 的实现中，程序员必须对不确定性谓词给予特别关注，因为这意味着运行时对内存资源的需求。然而，Visual Prolog 可以对不确定性谓词进行内部检查以减轻程序员的负担。

截断可以插入到定义谓词的规则体中，从而使不确定性子句变为确定性子句。例如，把截断放在定义某个谓词的子句中，这个子句就变成确定性子句。因为有截断的存在，调用该谓词时只能返回一个解。

截断机制功能强大，但它也有副作用，即截断的使用常使 Prolog 程序变得很混乱。在这一点上，它很像其它编程语言中

的 GoTo 语句——用它可以做很多事情，但却使程序变得难于理解。

5 动态截断——动态地阻断回溯

传统的 Prolog 截断机制是静态的。静态截断机制存在两个问题：1) 当执行过程通过截断符号"!"时，截断即发生作用，但是它只影响那些已经把它放入其中的子句(在源文本中)。无法把一个截断的作用通过参数传递给另外一个谓词，如果条件满足，截断就只能被评估。2) 如果还没有截断谓词中下一个子句的回溯点，则先要截断子句中子目标更多的解是不可能的。

Visual Prolog 有一个动态截断机制，通过两个标准谓词 getbacktrack 和 cutbacktrack 来实现。这种机制使我们能够处理上述两个问题。谓词 getbacktrack 返回一个指针，指向当前回溯点堆栈的栈顶。在以后的某个时候，通过给出谓词 cutbacktrack 所找到的指针，就可以删除这个位置之上的所有回溯点，

如果出现回溯，尽管谓词可能返回许多答案，但是调用 cutbacktrack 来截断是可能的。随后的一个失败将回溯到上一个谓词。

动态截断更重要的应用是将回溯指针传递给另一个谓词并有条件地执行截断。指针是 unsigned 类型的，并且能够以 unsigned 类型的参数形式进行传递。

使用动态截断需要十分理智。利用动态截断非常容易破坏程序的结构，粗心的使用会产生许多难以解决的问题。

6 否定谓词 not

谓词 not 称为否定谓词，常用来对一个子目标的结果取反。考察下面的程序，它说明如何使用谓词 not 来识别一个好学生：积分点(GPA)不低于3.5且不是受处罚的试读期。

```
DOMAINS
    name = symbol
    gpa = real
PREDICATES
    honor_student(name)
    student(name, gpa)
    probation(name)
CLAUSES
    honor_student(Name):-
        student(Name, GPA),
        GPA>=3.5,
        not(probation(Name)). /* 使用谓词 not */
    student("Betty Blue", 3.5).
    student("David Smith", 2.0).
    student("John Johnson", 3.7).
    probation("Betty Blue").
    probation("David Smith").
GOAL
    honor_student(X).
```

使用谓词 not 时需要注意的一点是：当子目标不能被证明为真时，谓词 not 运行成功。这导致发生这样一种情形：未绑定变量在谓词 not 中被绑定。当带有一个自由变量的子目标被谓词 not 调用时，Visual Prolog 将返回错误信息：在 not 或 retrackall 中不允许有自由变量。产生这个信息的原因是，Visual Prolog 在一个子目标中绑定了自由变量，而这个子目标必须与其它一些子句合一且该子句必须成功。在含有谓词 not 的子目标中处理未绑定变量的正确方法是使用匿名变量。

结束语 回溯机制是逻辑程序设计的重要设施。回溯本身是一种获得目标所有可能解的良好方法。然而，回溯也有副作用，一是它可能导致 Visual Prolog 给出多余的答案，而 Visual Prolog 自己不能区分实质上相同的两个解，所以当寻找给定问题的唯一解时可能要搜索好几遍，因此会降低效率。二是尽管一个特殊的目标已被满足，但是回溯机制可能还会强迫 Visual Prolog 继续寻找另外的解。在这些情况下，必须控制回溯过程。Visual Prolog 的静态截断机制、失败谓词 fail 与否定谓词 not 等控制谓词，以及动态截断机制，构成了完整的目标搜索求解控制机制，可以实现对搜索过程的仔细控制。

本文在考察 Visual Prolog 回溯机制基本问题的基础上，通过实例，对搜索求解控制机制进行了详细分析，从而揭示回溯机制和搜索求解控制机制的本质特性和应用机理。

参 考 文 献

1 雷英杰，邢清华，孙金萍，张雷. Visual Prolog 智能集成开发环境评述[J]. 空军工程大学学报(自然科学版)，2002，3 (5)：39～43

2 雷英杰，张雷，邢清华，孙金萍. Visual Prolog 语言教程[M]. 西安：陕西科学技术出版社，2002

3 雷英杰，邢清华，孙金萍，张雷. Visual Prolog 编程、环境及接口[M]. 北京：国防工业出版社，2004

4 雷英杰，王涛，赵晔. Visual Prolog 的回溯机制分析[J]. 空军工程大学学报(自然科学版)，2004，5 (5)：80～84

(上接第51页)

1.3%。这表明语义网络的深度因数，在一定程度上影响匹配的效果。

综上，基于义素的信息项匹配方法不仅有较高的准确率，而且最大准确率与最小准确率之差较小，是一种较好的语义匹配方法。

结束语 基于义素的网页信息项的语义匹配方法，从义素这个基本语义单元角度，给出适用于变化网页信息的语义匹配方法。这个方法是目前较好的语义匹配方法中的一种，已经用于网上非法药品广告的检测系统。

参 考 文 献

1 Da L G, Facon J, Borges D L. Visual speech recognition: a solution from feature extraction to words classification. In: Proc. of Symposium on Computer Graphics and Image Processing, XVI Brazilian, 2003.399～405

2 Nakashima T. Classification of characteristic words of electronic newspaper based on the directed relation. In: 2001 IEEE Pacific Rim Conf. on Communications, Computers and signal Processing, 2001(2):591～594

3 Vladimir A O. Ontology based semantic similarity comparison of documents. In: 14th Intl. Workshop on Database and Expert Systems Applications (DEXA'03), 2003.735～738

4 程莉，卢正鼎，文坤梅. 基于语义的模糊匹配探索与应用. 华中科技大学学报(自然科学版)，2003，31(2)：23～25

5 Rodriguez M A, Egenhofer M J. Determining semantic similarity among entity classes from different ontologies, Knowledge and Data Engineering. IEEE Transa.on, 2003, 15(2): 442～456

6 张茂元，卢正鼎. 基于特征选取及模糊学习的网页分类方法研究. 小型微型计算机系统，已录用。