计算机科学1992Vol.19No.5

机器定理证明中的一般问题*)

贲可荣　陈火旺

（国防科技大学计算机系，长沙410073）

摘　要

本文从人工智能角度阐述了机器定理证明的重要意义，综述了以逻辑方法为重点的定理证明器的一般问题，如搜索策略、化简、语义、抽象、发现相关公理以及它涉及的一般软件工程问题。

一、概　论

形式逻辑是机器定理证明的重要基础，它适合于表达说明性（而非过程性）知识，这种说明性只陈述什么为真，而不具体说明知识如何使用。这就允许以新的创造性方式使用知识，但同时也产生了副作用：在特殊情况下，难以知道如何使用知识。然而过程性知识具体阐述了在特殊情况下如何去做，所以两种知识各有用途。下面我们将考虑用于描述说明性知识的形式逻辑及其计算机中的推理系统。

事实上，所有数学均可用一定形式系统来表达，例如，Zermelo-Fraenkel集合论。不仅形式数学如此，而且许多别的技术领域也可这样表达。一阶逻辑是一特别简单和易于理解的语言，许多问题，用一阶逻辑表达就足够了，它甚至可以表达 von Neumann-Bernays-Gödel集合论。在一阶逻辑中，这种集论不弱于Zermelo-Fraenkel集合论。我们将精力集中于一阶逻辑，其它逻辑做为补充。

在人工智能界，关于逻辑在人工智能中的作用问题已有很多争论。大家公认，要处理常识推理，需要扩充古典逻辑，这种扩充包括非单调逻辑，知识和信念的逻辑以及时态逻辑等。但是，人们提出更一般性的问题，逻辑可否做为人工智能的基础？

我们的立场是，公理化的推理有其独特的趣味，不管它是否是人工智能的基础。大量的现代技术知识能够在数学或逻辑中形式化，并且大量的现代研究尝试，包含公理化推理，甚至，大量人工智能问题能够在古典逻辑中形式化，有些可能在上述提到的古典逻辑的扩充中进行。

推理的自动化特别诱人，因为用现代计算机进行快速准确推理已成可能，目前可以做到每秒钟做数千次推理的速度。现在的问题不在速度而在控制，计算机以同样的灵活性产生相关的和不相关的结果。推理自动化亦是需要的，因为人们作出详细证明是乏味的和易出错的。事实上，数理逻辑的推动力之一就是想使推理尽可能机械化，从而构成的逻辑系统一般来说适合于计算机。

在计算机推理系统中，指导搜索的可能方法之一是人机交互。人给出一般的命令，计算机具体执行，并检查准确性。这种结合是有吸引力的，无疑将会继续被使用。然而，本文将集中讨论相当自动化的方法。这种方法除自身有用外，在人机推理系统中，

*)八六三计划软件生产自动化课题的部分资助项目

还可以减轻人的负担。

定理证明器的可能应用包括专家系统，规划，常识推理，证明验证，教育以及帮助人类数学家。事实上，可以想象，哪里用到推理，哪里就用到定理证明器。我们可将一专家系统看成一定理证明器。如果有效的搜索策略取得进展，那么通过加进更多的一般公理和推理能力，熟知的专家系统的"脆性"(brittleness)会得到缓和。一些这样的应用需要非单调逻辑或别的古典逻辑的扩充，这些领域仍是活跃的研究领域。一阶逻辑中许多熟知的方法仍可应用于这些扩充了的逻辑。可以将一定理证明器看做是最终的说明性程序设计语言，并且，自动推理用来实现一程序设计语言，或者用来实现一程序设计语言的高层控制。为使这些应用成为事实，从本质上推进机器定理证明技术的现状，是非常必要的。

自动定理证明系统的许多应用早已存在。例如，程序验证器已应用于验证相当复杂的程序，这些验证器包括SRI的STP验证器(1982)，Stanford的验证器(1975)，Texas大学的Gypsy验证器(1975)。另外，Cornell的Nuprl系统(1986)是一真正带有说明性和过程性解释的逻辑系统，在其中可以保证构造正确的程序，这系统具有一定的自动推理特征，但是，主要是通过人的交互来指导搜索。一些专家系统具有证明器功能，ART专家系统用来证明一阶逻辑中有价值的定理。

当然，不可否认，机械推理系统的应用到目前还很少。主要原因是定理证明技术还不够先进。另一理由是，许多定理证明器，不含有人工智能中已得到发展的启发式搜索技术。同样，在任何特定领域，通过写出一服务于那个领域的专用程序来获得较好的执行是可能的。然而，如果使用一般搜索，势必导致在每个领域搜索技术中的重复建造工作，同时也导致在专家系统中已看到的脆性。问题的部分原因是对定理证明器的可用性缺乏信任，或不熟悉逻辑。定理证明器可被视为Horn子句逻辑程序设计的下一步骤，并且作为Prolog的扩充被开发出来，用户们渐渐习惯定理证明和公理化推理。除了不熟悉逻辑外，另一问题是在许多定理证明器中，用户界面是初等的，并且程序设计语言特征，如继承性没有包含在逻辑中。当定理证明器具有较好特征和界面变得灵活了，当技术水平提高了，当各种理论公理参数化了的组件(modules)变得广泛可利用时，上述情况亦随时间的推移而改变。

使用定理证明器的一个问题是保证赋给系统的公理是正确的。因为相同的公理被重复使用，构造库会有很大用处。这样的经过良好测试的公理库会帮助减少错误。要保证通过定理证明器发现的证明的正确性是另一问题，因为定理证明程序大而复杂，很难保证它们的正确性。然而，因为一定理证明器是一个一般的程序，它可重复地用来解决许多不同的问题，因而值得将它测试完好。要求定理证明器一点错误也没有是没有必要的，我们仅仅需要由此得到的证明是正确的。这一点可通过证明检测器来运行证明。构造的大多数逻辑系统其证明检测是简单的。因此，似乎可以写出简明的证明检测器来证明正确性。因此，一个证明可通过许多用不同语言写的不同的证明检测器以及在分布广泛的地方运行不同的机器来检测。这样，对于很长证明的正确性可得到确信。赋给证明器的初始规范的正确性问题仍是很现实的。证明一个定理，通常由标准公理加上问题的规范组成。规范没有确切把握用户的直觉是有可能的。进一步，一些领域，如准备商业报告，似乎难以公理化。能够做到的最好的办法是找一种方法来帮助用户检查问题规范是否准确反应了他的意图。最后，有一些哲学的及实际的问题，是关于逻辑是否原则上对一些领域是足够的。许多人工智能的研究人员觉得不是如此。如果不花更多的时间，研究和发展机械推理系统使之有能力将

形式方法用于实际，关于这些讨论，是不会有结果的。

二、一般问题

现在我们讨论关于定理证明器的一般问题，诸如，搜索策略、化简、语义、抽象和类比、发现相关公理以及相关的一般软件工程问题。必须讨论的第一问题是完全性。一个定理证明器不完全是不是大问题？人类数学家可以不完全，但不成为问题。进一步，一些不完全的特殊方法在它们的应用中比一般方法要好。因而，看起来特殊方法有一席之地，甚至对那些应用领域还不十分清楚的方法也是如此。一般不刻意追求一定理证明器的完全性，但是，不完全证明器对做简单问题也会失败，就令人为难了。

另一问题是，形式逻辑是否大体上足够刻画关于现实世界的推论？在AI理论方面，存在一些关于常识推理问题，但是，一点不知道这理论，人们似乎活得很好。进一步，形式主义者往往强调简单的、周密的方法，而忽视启发式方法，后者在实际应用中对取得成功是必要的。到目前为止，关于世界的大部分表达可以用逻辑项来表示，而且逻辑演绎本身也很有趣。因此，努力改进形式方法似乎是合理的，放弃逻辑其实是放弃对目前正在讨论东西的精确理解，这似乎花费的代价太大了。

1.搜索策略

现在，我们对搜索策略做些评论，许多例子表明了带存贮的向后链法的重要性。但一些问题用向前链法比用向后链法更好，因而以某种方式平衡向前链法与向后链法是有意义的。为了使向后链法完全，必须使用宽度优先搜索、最好优先搜索、深度优先迭代加深(DFID)等等搜索。深度优先迭代加深比起别的搜索有很好的存贮效率。与此同时，在一定条件下，宽度优先有时间效率上的优越性。我们知道，与DFID连用的存贮的重要性，然而，真正做好存贮是不容易的。为知道哪个解对应哪个子目标，必须得做某种簿记('book keeping')。对DFID的另一问题是怎它与优先顺序系统结合。在最好优先搜索中，每一步容易选择期望值最高的子目标并执行它，对深度优先迭代加深并不容易。基于一般化的某种解释，对反向链法是有用的，能够保证其解尽可能一般化。

对反向链法的另一问题是以智能方式对子目标重新排序。我们发现这一点对某些问题非常重要。设计一个好的子目标排序方法是困难的，我们一般喜欢先做大的子目标。但是，我们发现，子目标比目标具有更多的函数符号，会引起麻烦，因为反向链能导致函数符号增殖。

2.重写

重写和化简对许多问题是重要的。重写(调解)允许将表达式以等价的但更简单的表达式替代。例如，x-x能用0替代。化简的概念亦可扩展到子句上。例如，如果not(L_1)已导出，则一子句$L_1 \vee L_2 \vee \cdots \vee L_n$能化简为$L_2 \vee L_3 \vee \cdots \vee L_n$。如果比not($L_1$)更一般的一文字被导出，这个简化也是可行的，因为在这样的化简中，可用单位子句，所以，调整适合单位子句生成的搜索是有意义的。这样的单位化简在重写中可自动进行，如Hsiang 1985年提出的方法。删除被包含的子句可看作是化简的另一方法。另外还有一种化简方法是用谓词的定义替换谓词。例如，谓词subset(x,y)可用公式$(\forall z)((z \in x) \rightarrow (z \in y))$来替代。系统地使用这样的替换，接着利用布尔等价式化简所得公式，许多简单的集合论问题可以几乎不用搜索得到完全的自动的证明。例如，证明$P(x \cap y) = P(x) \cap P(y)$，这里$P(x)$表示x的幂集，可以用这种方法来证明。在一归结定理证明器中，这不可能直接去做，因为定义包含显量词，归结用于不含量词的子句形式。不幸的是，显量词使一致化算法更加复杂，所以，一般什么是最好方法并不清楚。如果集合论公理直接赋给一归结定理证明器，这样，简单集合论恒等式常常是难处理的，因为在大量的Skolem函数

中，失去了通过其定义替换谓词的自然思想。在不带量词框架中，部分地克服这问题的一种方法由Potter及Plaisted于1988年给出。与此同时，用其定义替代谓词并不总是最好的，因为这给搜索增加了不必要的复杂度。

3.优先性

使用优先性来控制搜索也是重要的。一般短子句比长子句优先，所谓长短是指出现在其中符号的个数。与此同时，容易被产生的子句一般先于难产生的子句。可在Argonne定理证明器中发现前述思想的一种好的实现，其中每一步选择最短的可用子句并与用户所选策略允许的其它所有子句进行归结。这样使用优先性有助于解释这些定理证明器的一般好的行为。当然，修正优先性，使一些函数符号的权超过别的，使一些子项的权超过别的，这总会更好些。Argonne证明器通常允许用户指定这些权值。另一个好的思想是对含函数符号的子句和出现在定理中的谓词优先，因为它们似乎更相关。

分层演绎证明器(Wang〔1987〕)对优先性结构有特别兴趣。它通过考虑变量在一致化中如何受限来测试一个证明的复杂度。这给出了证明复杂度的一相当自然和直观的测度，它能使该证明器自动求解某些相当困难的问题。这个证明器主要是锁归结和支架集的结合，且可视为普通归结的一种精化。在P-laisted修改过的问题演绎格式证明器中，用类似Wang的优先性测度也得到好的结果。

4.语义

因为人们在证明定理中广泛地使用语义(模型)，所以计算机的定理证明器也自然会这样做。这个方向，已有一些成功尝试，例如，Gelernter〔1963〕的几何定理证明器。但到目前为止，这种方法用在别的证明器中似乎没有得到什么。使用语义对应人们用以决定所考虑的引理是否为真的例子或图形。人类数学家试图证明一定理之前，先确信在语义层，该定理是真的。使用语义和有穷模型在非标准逻辑定理证明器(McRobbie〔1988〕)中是有益的。McRobbie的证明器使用一问题归约格式和Gentzen型证明系统。Wang的分层演绎证明器也使用语义，但如果离开语，义似乎也做得很好。使用语义有一些明显的方法，只需对各种谓词选择正负号(即是否带连接词¬)，而不考虑自然的和基真实世界解释的语义。另一与使用语义密切相关的工作是使用例子，见Ballantyne，A.〔1982〕。

5.抽象

一个定理证明器从经验中学习似乎是有用的。一种方法是通过类比来学习。当试图证明一定理时，过去已证明过的类似定理在指导搜索中提供有用信息。这领域的早期工作见Kling〔1971〕。Austin小组在这个领域做了一些最新工作〔1988〕，但到目前为止，使用类比还相当受限，对搜索时间的改进也不大。抽象的思想与类比密切相关：为求解问题P，我们抽象P得到Q，求解Q，将抽象过程反过来，将Q的解返回到P的解。抽象过程，一般抛弃P中一些信息，例如，抽象可能忽视一个行为的某些前置条件。即使对Q求得解，Q的解返回不到P的解也是可能的。尽管如此，这个思想仍是有吸引力的。这方面的早期工作由Sacerdoti，E.〔1974〕及Plaisted〔1981〕所做。抽象的另一方法见P-laisted〔1986〕。最近，Plummer，D.用抽象来帮助决定哪些定义在证明一个定理中是有用的，在这一点上，他取得了一定的成功。使用超过一层的抽象或者在同一时刻，使用多个抽象均是可能的。例如，我们抽象P_1到P_2，抽象P_2到P_3，然后求解P_3，将其解映回P_2的解，然后再映回P_1的解。这个思想，实际上已由Greenbaum，Plaisted在Illinois大学实现了。尽管在搜索时间方面某些化简得到了，但是，表演通常令人失望。最引人注目的成功是，当输入出现错误时，抽象证明器迅速失败。因此，Plaisted所试验的抽象特别适用于快速测试何时一个证明不能被发现。直接应用归结不能测试这

一点，反而为试图寻找一个证明产生许多子句。使用多层抽象和同一时刻使用多个抽象结合起来是可能的，这样得到一个抽象"网络"来指导搜索。另一思想是放宽抽象空间和初始空间的对应，我们可用抽象空间来修正优先性，以便我们推荐初始空间的子句映到抽象空间的有用子句，而不是要求初始空间中每一推理对应到抽象空间中一推理。实际上，这个领域有许多需要人们去尝试，但已经做的却很少。

6.推理步的大小

自动演绎的另一问题是推理步的大小。用较大的推理步有一个明显优点，即可省去一些中间引理，搜索空间也不会增长得过快。超归结在一定程度上是这样做的。另一个这样的方法是连接(linked)推理。Stickel的归结理论也实现了这一想法。

7.相关测试

对一个机械定理证明器来说，选择相关公理亦是非常重要的。为了一证明，可能有成百上千个事实可用于搜索，一推理系统要有能力选择可能与之相关的断言。目前的证明器有一个优点，即大多数情况由人来选择。支撑方法集倾向于选择相关事实，因为它们仅仅执行那些直接或间接依赖定理的推理。通过观察过去已证明的定理，选择似乎相关的事实是可能的。由 Jefferson 实现的Plaisted的相关准则，是另一可能的途径。在两个常识推理问题中，这个方法用来从数百个输入子句集中发现相关子句，取得了惊人的成功，在其中，证明相当简单。其思路是，如果两个子句C、D分别包含文字L、M，且满足not(L)与M是可一致化的，则C、D是密切相关的。如果C和D是密切相关的，D和E是密切相关的，则C和E是次密切相关的。以这种方式，我们能够度量两个子句何等密切相关。一子句集称为全匹配的，是指对该集中的每一子句C，C中每个文字L，均存在该集中一子句D和D中的一文字M满足L和not(M)可一致化。可以证明，如果从一子句集S存在反驳，那么，由用于反驳的子句所构成的集必与定理密切相关且是全匹配的，这里，子句的密切相关测度依赖于反驳的长度。这个思想，可用来在多项式时间内过滤某子句集，找出对一简短证明有用的子句集；还可用来检测和产生一子句集的相关实例。企图将这一方法用到带较长证明的数学问题没有取得成功。当一证明相当短，且有许多输入子句时，基于相关方法(如Jefferson和Plaisted的方法)的图似乎是最有用的。

8.一般软件工程问题

定理证明器也是计算机程序，其设计自然会出现一般的软件工程问题。良好的输入、输出界面和精心选择数据结构是很重要的。阻止证明器对难解问题使用过多的存贮，"结构共享"的思想是重要的。一种"判别网"可用来快速查出潜在的可一致化文字对。这方法成功地用在Greenbaum和Plaisted的Franz Lisp归结证明器中。与此同时，McCune的Otter定理证明器也用了此方法。并行的使用可提高证明器的速度。对一证明器来说，具有良好的缺省开关设置是重要的，这样可使一些低水平用户也能得到好的结果。在定理证明器的发展中，经验测试是有价值的。这对识别用什么样的机制处理一问题领域是充分的，同时对揭示什么样的因素被忽略是有帮助的。在测试一证明器过程中，人们常常发现一简单问题成功，而一复杂的问题在成功地执行了几小时后失败了。在这种情况下，人们想知道，策略是否完全，搜索策略是否需要改进以及程序是否有缺陷。在这样一种情况下，一种有效的方法是使简单问题一步一步逼近复杂问题，直到出现失败，从而正确地指出问题之所在。比起修改复杂问题以希望发现一证明来说，这方法是非常节省时间的。自动演绎搜索的一个优点是相同定理可在许多证明器上运行，这样可以比较不同证明器的效率。这在人工智能的其它领域中没有这么简单，不同程序将使用不同

的体系。

9.扩展

我们现在考虑对一阶逻辑的各种扩展，这些扩展在定理证明器中有用。扩展之一是类和序类代数的使用。使用类，大大地改进定理证明器的效率，同时允许比较简单地进行公理刻画，尽管序类逻辑要求更复杂的一致化算法。Walther, C. 和Andrews, P. B.分别做了不少技术和理论工作。另一扩展是数学归纳法。一些定理离开了数学归纳法是不能证明的。例如，在一阶逻辑中，仅仅通过后继关系定义的加法公理是不能证明加法可交换的。即使等式x+y=y+x的所有基例可以证明，等式仍不可证明，因为在一阶逻辑中，不能确定一个元素是否有穷。使用归纳法的最著名的证明器是Boyer-Mooer证明器。这个证明器需要得到用户的指导，这些指导为一系列逐渐通向定理的引理，这些引理必须细心挑选，使得程序真正用上这些启发式信息。在这个证明器中，定理是重写规则，很好地限制了搜索。一般存在量词也不能在此证明器中使用。用在TABLOG系统中的非子句定理证明器亦允许数学归纳法。另一扩展是高阶逻辑，高阶逻辑允许对函词和谓词使用量词，这样，一些定理的叙说比起用一阶逻辑来更自然。带有一定高阶逻辑能力的证明器之一是诞生在Carnegie-Mellon大学的著名Andrews证明器。这个证明器使用了Huet的一致化算法，能够对高阶表达式一致化。λ-演算允许表达高阶函数，Nadathur, G.等人的λ-Prolog系统允许使用λ-演算表达式。这是有价值的，因为它对有界变量提供了一体系，这个体系对逻辑和计算机程序的形式化是有用的。最后指出，集合论本身在证明器中有用，它能够直接指称元素组成的集合以及方便地证明它们的性质。Boyer, R.等人将集合论嵌入一阶逻辑，但是，这种编码是不自然的，也许存在更自然的编码方法。

对定理证明器来说，存在这么多的扩展及其长处，如何以好的方式将它们结合起来，成为一个问题。这个问题比起基础研究中的问题说来，更象是工程问题。定理证明如此复杂是令人吃惊的，因为定理证明所依托的数学形式体系是如此简单。许多定理证明器忽视了上面提及的不少重要特性，诸如优先性和反向链。现在还不清楚，是否有一种方法对所有问题都执行良好。简单地将许多方法放到一定理证明程序中，并交替使用它们，可以增加许多"智能"。这似乎是由人来使用的方法。什么方法适用于什么问题的一类元知识在这里是非常重要的。

参考文献

1. Chang, C.and Lee, R., Symbolic Logic and Mechanical Theorem Proving, 1973
2. Loveland, D., Automated theorem proving: a quarter century review, in Automated Theorem Proving:After 25 Years, W.Bledsoe and D. Loveland, eds, 1984, pp.1-45
3. Pelletier, F.J., Seventy-Five Problems for Testing Automatic Theorem Provers, Journal of Automated Reasoning 2 (1986), 191-216
4. Plaisted, D.A.,Mechanical Theorem Proving, Formal Techniques in Artificial Intelligence, R.B.Banerji, eds, 1990, 269-320
5. Wos, L.Automated Reasoning: Basic Research Problems, DE 86 009214, 1986

刘叙华，定理机器证明与自动推理，智能技术与系统基础，高庆狮主编，1990,pp.30-66