摘　要

算法程序设计自动化是一种利用计算机程序来自动化生成算法的技术。它主要用于解决程序设计中的瓶颈问题，通过自动化生成算法来提高程序设计的效率和可靠性。随着人工智能领域的不断发展，它在计算机科学领域得到了广泛的研究和扩展，然而算法程序设计依然存在一些挑战和限制。首先需要耗费大量的时间和精力。其次算法程序设计需要处理大量的数据和复杂的问题，人为的疏忽和错误也会导致程序设计中存在一定的错误率。因此，寻求一种高效、可靠的算法方法是实现程序设计自动化的迫切需要，同时也是本文的研究目标。

本文基于所在研究团队在多项国家级课题连续资助下自主研发的PAR方法，提出了一种针对存在常规解问题的高效算法程序设计理论和框架，并将其与专家系统的技术融合起来以实现算法程序的自动构造；选择Radl作为目标语言，开发了Radl规约到Radl算法程序自动生成系统，并对系统的可靠性进行了理论分析。最后，通过多个案例验证其支撑工具RSPEtoRALG生成系统的自动转换能力。结果表明，本文提出的方法具有一定的研究意义，并且可以为程序开发人员提供一定的参考依据。

本文主要创新性如下：

（1）基于PAR方法提出了一套新的高效算法程序设计理论方法和问题的泛型算法结构，针对程序设计中常见的存在常规解的问题进行了处理。基于知识表示的技术对PAR方法中程序设计过程中的知识进行了分类、提取和管理，提炼出Radl规约程序分析规则、Radl算法程序生成规则，将其转化为计算机可理解的形式构建了算法程序知识库。

（2）基于提出的高效算法理论与知识表示及知识推理技术相结合，面向精准智能这一目标设计了一个Radl算法程序自动化系统RSPEtoRALG，实现Radl规约到Radl算法程序的自动转换，并对该系统进行了测试和评估，极大的提高了程序开发效率和PAR平台的自动化程度。

关键词：PAR；Radl；专家系统；自动化；

Abstract

Algorithmic programming automation is a technique that uses computer programs to automate the generation of algorithms. It is mainly used to solve bottlenecks in programming and to improve the efficiency and reliability of programming by automating the generation of algorithms. As the field of artificial intelligence continues to grow, it has been extensively researched and expanded in the field of computer science, however, algorithmic programming still has some challenges and limitations. Firstly, it requires a lot of time and effort. Secondly algorithmic programming needs to deal with a large amount of data and complex problems, and human negligence and errors can also lead to a certain error rate in programming. Therefore, seeking an efficient and reliable algorithmic method is an urgent need to automate programming and is also the research goal of this dissertation.

In this dissertation, based on the PAR method developed by our research team under the continuous funding of several national projects, we propose a theory and framework for efficient algorithmic programming for problems with conventional solutions, and integrate it with the techniques of expert systems to realize the automatic construction of algorithmic programs; we choose Radl as the target language, develop a Radl protocol to Radl algorithmic program automatic generation system, and The reliability of the system is analyzed theoretically. Finally, the automatic conversion capability of its support tool RSPEtoRALG generation system is verified by several cases. The results show that the method proposed in this paper has some research significance and can provide some reference basis for program developers.

The main innovations of this dissertation are as follows:

(1)A new efficient theoretical approach to algorithmic programming and a generalized algorithmic structure of problems based on the PAR method is proposed for problems with conventional solutions commonly found in programming. The knowledge of the programming process in the PAR method is classified, extracted and managed based on the technique of knowledge representation, and the Radl statute program analysis rules and Radl algorithm program generation rules are refined, which are transformed into a computer understandable form to construct a knowledge base of algorithmic programs.

(2) Based on the proposed efficient algorithm theory combined with knowledge representation and knowledge inference techniques, a Radl algorithm program automation system RSPEtoRALG is designed for the goal of precise intelligence to realize the automatic conversion of Radl statute to Radl algorithm program, and the system is tested and evaluated, which greatly improves the program development efficiency and the automation degree of PAR platform.

Key words: PAR; Radl; Expert System; Automation;

目　　录

中文摘要 I
英文摘要 II
目　　录 IV
第 1 章 绪论 1
1.1 研究背景 1
1.2 研究现状 2
1.3 研究内容 3
1.4 本文组织 3
第 2 章　相关研究及技术简介 5
2.1 形式化方法及 PAR 方法 5
2.2 人工智能与专家系统 7
2.3 专家系统技术和方法 10
2.3.1 知识获取 11
2.3.2 知识表示 11
2.3.3 知识推理 13
2.3.4 产生式系统 16
2.3.5 RETE 算法 19
第 3 章　泛型算法程序语言 Radl 21
3.1 Radl 总体结构 21
3.2 数据类型和变量说明 22
3.3 泛型程序设计 26
3.4 量词表示方法 27
3.5 Radl 语言算法程序开发实例 28
第 4 章　高效算法程序设计 29
4.1 存在常规解问题算法程序设计理论 29
4.2 存在常规解泛型算法程序结构 29
4.3 高效算法的应用实例 30
第 5 章　Radl 算法程序的生成系统开发 33
5.1 生成系统的目标 33
5.2 总体设计框架 33
5.3 用户界面模块功能设计与实现 35

5.4 知识库模块的设计与实现 36
5.4.1 Radl 规约分析规则 37
5.4.2 Radl 算法生成规则 37
5.4.3 Radl 规约到 Radl 算法转换规则 38
5.5 事实库模块的设计与实现 39
5.7 推理机模块的设计与实现 40
5.7.1 推理机模块的功能 42
5.7.2 推理机的处理流程 42
5.6 控制策略模块的设计与实现 40
5.8 系统 RSPEtoRALG 可靠性分析 44
第 6 章 系统运行效果展示 45
6.1 RSPEtoRALG 的功能说明及使用 45
6.2 实例分析 46
6.2.1 最长连续上升子序列问题 46
6.2.2 冒泡排序问题 47
6.2.3 单源最短路径问题 49
6.2.4 最大和问题 50
6.2.5 只用加法计算数组元素下标的四次方问题 53
第 7 章 总结与展望 57
7.1 研究成果总结 57
7.2 后续工作及展望 57
参考文献 59
致 谢 63
在读期间公开发表论文(著)及科研情况 64

第1章 绪论

1.1 研究背景

随着计算机硬件的不断升级和应用场景的不断扩展，算法程序设计的自动化变得越来越重要[1]。1999 年图灵奖获得者 Jim Gray 曾在他的文章“The Next Big Thing”中指出，自动化程序设计将是计算机科学领域的一个重要发展趋势。他认为，未来的程序员应该更加注重设计和管理计算机系统，而不是过多地花费时间在机械性的编程工作上。自动程序设计及其成功标准也被列为新世纪计算机科学需要解决的 12 个问题之一，应引起重视[2]。Jim Gray 指出了一些挑战，这些挑战有一定的逻辑关系。首先，为了提高程序自动化程度和准确性，需要设计更加高效的程序自动化算法和技术。其次，为了能够自动生成高质量的代码，自动化程序设计需要使用更高级的编程语言。最后，为了确保生成的代码符合规范并且具有高质量，需要更好的测试和验证工具。尽管国内软件产业已经发展了多年，但与国际先进水平相比，在程序自动化技术方面仍有差异。具体来说，部分国内企业的软件开发过程仍然比较传统，缺乏自动化流程和工具支持，因此需要大量手工和重复工作[3]。此外，一些公司在软件开发领域主要依赖国外提供商提供的技术和工具，难以实现自主创新和技术自主可控性。同时，安全性问题还没有得到足够的关注和保障。此外还缺乏统一的程序自动化标准和规范。形式化方法是一种可行的途径来实现算法程序自动化，它可以将算法程序设计规范化成一组明确的规则和标准[4],确保程序设计的一致性和正确性。通过将算法程序的设计和实现转换为数学模型，形式化方法为算法程序设计自动化方面发挥了重要作用。这种方法提供了一种系统化的方式来规范算法程序的设计和实现，方便进行分析和验证，进而提高程序的精确性和可信性。

专家系统（ES）是人工智能技术的重要组成部分，其在程序设计领域的应用将有助于推动人工智能技术在该领域的发展和应用[5]。专家系统可以模拟人类专家的知识和经验，为用户提供决策支持和问题解决的服务。专家系统技术能够将程序设计中的知识和经验进行建模和知识表示，能够对程序设计中的决策进行分析和评估，提供决策支持。专家系统将知识和推理两个组成部分分别独立开来，这使得人们能够更加清晰地对它们进行研究、分析和设计，方便在以后的程序设计中进行重复利用，提高程序设计的可重用性。通过对知识和推理的分离，专家系统能够更加灵活地处理不同领域中的复杂问题。同时，这种分离也使得专家系统具备更好的可扩展性和可维护性，能够更加方便地进行知识库和推理引擎的更

新和维护。因此，专家系统的这种设计思想得到了广泛的应用和推广。如今，专家系统已经在各个领域普及，并取得了进一步的发展[6]。斯坦福大学教授费根堡姆(E. A. Feigenbaum)等人开发的有机化学结构分析系统 DENDRAL 以及血液传染病诊断和抗菌素处方系统MYCIN等计算机智能系统的开发和盈利成为了一种独特的商业运作模式，随后涌现出了许多软件和硬件公司。在这一人工智能发展阶段，仅专家系统研发行业的产值就高达数亿美元。专家系统是人工智能最重要和最广泛的应用领域之一，在过去 40 多年中获得很大发展，对国民经济各个部门和科学技术各个领域以及人民文化生活与物质生活的方方面面做出不可磨灭的贡献[7]。

本文旨在利用专家系统的方法和技术结合分划递推的高效算法，实现算法程序设计的自动化。使用了 PAR 方法中的 Radl 语言作为规约语言和算法程序语言，Radl 是 PAR 方法的一部分，是一种基于递推关系的算法程序语言[8]。本文研究将通过对 PAR 方法中最优化问题的知识进行建模和表示，结合高效算法实现算法程序的自动生成。本研究成果有助于解决传统程序设计方式所面临的问题，提高程序设计的效率和质量，推进计算机技术的发展和应用。

1.2 研究现状

计算机和人工智能技术的不断发展，使得计算机程序作为支撑高新技术的主要载体之一变得越来越复杂，这也导致人们对程序自动化的需求日益迫切[9]，许多高校和科研机构正在开展相关的研究。其中，北京大学和中国科学院计算技术研究所等知名机构在算法程序设计自动化领域积累了丰富的经验和技术，并取得了一定的成果[10]。在欧洲，德国和英国等国家也在该领域开展了一系列研究。例如，德国图灵奖得主马丁·格罗斯等人提出了基于搜索的程序设计自动化方法，可以自动合成高效的程序代码[11]。目前，很多学者认为将人工智能技术应用于算法程序的自动生成和设计是加速智能化进程、实现 AI 完全自主构造算法程序的有效途径[12]。算法程序形式化是实现算法设计自动化的基础和前提条件，在算法程序自动化研究中，通常会与算法程序形式化相结合进行研究。算法程序形式化和自动化目前处于发展初期，国内外众多学者使用了不同的技术进行了研究。尽管部分研究工作较少探讨算法设计自动化，但其研究方法、途径和成果均构成了算法设计自动化研究的基础[13]。

在国内外的研究中，首先，针对算法程序设计自动化的模型和算法方面，国际上已经开始利用机器学习和深度学习等技术来实现算法程序自动化设计。其中，自动编程领域的研究成果比较突出，例如利用遗传算法、神经网络等方法来实现自动编程，这些方法能够自动化生成程序代码并不断优化,提高编程效率和代码质量[14]。其次，针对算法程序设计自动化的应用方面，国内外的研究人员已将其

应用于多个领域。最后，针对算法程序设计自动化的挑战和未来研究方向，国内外的研究者正在探索如何进一步提高自动化设计工具的效率和可靠性。例如，如何将自动化设计工具应用于更加复杂的算法程序设计中，如何提高自动化设计工具的智能化程度等问题都是当前需要解决的难题[15]。未来，算法程序设计自动化还将与更多新技术结合，如区块链、云计算等，以实现更加智能化的算法程序设计自动化。总体而言，自动化算法程序设计是一个快速发展、备受关注的研究领域。随着人工智能、机器学习等技术的不断发展，算法程序设计自动化的发展前景将更加广阔。

人工智能面临的核心难题之一便是让机器能够自我学习，从现有程序中快速生成新程序，并在一定条件下自动执行，以解决不同任务。人工智能不断设计和优化执行这些程序的算法的能力是至关重要的。基于目前人工智能在算法程序自动化领域的研究，自动算法程序生成领域的专家系统已经成为一般人工智能的一个重要途径，并且可以加速人工智能的算法程序设计自动化。

1.3 研究内容

本文以分划递推法的高效算法和专家系统的技术为理论指导，将展开以下几方面的工作：

（1）概述了程序设计自动化、形式化方法和专家系统的相关研究。

（2）针对存在常规解的最优化问题提出了一种新的算法程序设计理论和新的算法程序表示方法用以解决常规解问题。

（3）基于专家系统的技术和方法，结合 PAR 中已有的 Radl 规约与 Radl 算法程序总结出了排序类和最优化问题中 Radl 规约和 Radl 算法之间的规律，提炼出 Radl 规约程序分析规则，Radl 算法程序生成规则，将 Radl 规约程序通过分析、转换、综合三阶段生成 Radl 算法程序。

（4）基于提出的高效算法并结合知识表示和知识推理的技术，以实现精准智能这一目标，设计并实现 Radl 算法程序的自动生成系统 RSPEtoRALG。

（5）给定存在常规解的五个问题的 Radl 规约，运行 Radl 算法程序的自动生成系统 RSPEtoRALG，并对该系统在 PAR 平台上进行了测试和评估，极大的提高了软件开发效率和 PAR 平台的自动化程度。

1.4 本文组织

本文的组织方式分为七章，分别为：

第 1 章 绪论。简要阐述本文的研究背景，说明了程序设计自动化方面相关研究的现实意义和理论意义，明确了具体的研究内容，并概述了全文的整体组织

框架。

第 2 章 相关研究及技术简介。主要介绍了不同领域的研究现状和发展现状，包括形式化方法的概述、专家系统的发展现状和理论方法以及专家系统的技术和方法。

第 3 章 泛型算法程序语言Radl。详细介绍了形式化规约和算法程序语言Radl，对Radl语言的组成和用法进行总结，并选择Radl语言作为形式化规约和算法设计的描述语言。

第 4 章 高效算法程序设计。针对存在常规解的问题结合PAR方法提出了一种新的算法程序设计方法和新的表示方法以及问题的泛型算法结构。

第 5 章 Radl算法程序的生成系统开发。结合知识表示和知识推理的技术和方法，使用高效算法实现Radl算法程序设计自动化，以提高程序设计的效率和质量。通过将Radl规约通过分析、转换、综合三阶段生成Radl算法程序。在此基础上，进一步阐述了如何设计和实现Radl算法程序设计自动化系统RSPEtoRALG的各组成部分。

第 6 章 系统运行效果展示。通过展示五个实例来评估RSPEtoRALG的性能，最终基于测试结果对该系统进行了分析和总结。

第 7 章 总结和展望。对全文进行了回顾和总结，同时对接下来的研究方向做出展望和建议。

第 2 章　相关研究及技术简介

2.1 形式化方法及 PAR 方法

（一）形式化方法概述

形式化方法以逻辑系统为基础，能够提供对计算系统进行严格规约、建模和验证的支持[16]。在 60 年代末，Floyd 的归纳断言法以及 Hoare 的公理系统[17]。各自对程序正确性证明进行了系统夺的描述，为形式化方法的发展奠定了基础。形式化方法早期主要用于安全关键系统的研究，典型的案例有伦敦机场空中交通管理系统，巴黎地铁列车信号系统，NATO 指挥和控制系统[18],[19]等。随着软件系统规模的不断增长，人们发现形式化方法也能够为各种大型系统提供非常有益的帮助。人们对形式化方法一直存在许多争议，如认为其抽象数学理论难以掌握，影响用户理解和表达、影响开发效率等。实际上，形式化规范可以帮助用户更清晰地理解软件系统；将形式化方法应用于软件开发被广泛认为是一种革命性的途径，可以克服软件危机[20]，提高软件可靠性和生产效率。自上世纪 80 年代以来，形式化方法的研究取得了一系列进展，涌现出了 Z、VDM、XYZ、RAISE、B 等一系列方法和工具，特别是从上世纪 90 年代开始，形式化方法受到了西方发达国家政府和军方的高度重视。随着支持工具的丰富和完善，应用形式化方法可以有效提高软件开发的效率。如 Cai 和 Paige 的研究显示：应用程序变换技术的算法程序开发效率是传统方法的 5 到 10 倍。经过对面向对象软件系统形式化开发案例的分析，结果表明，通过在系统的关键部分引入形式化方法，可以用相对较小的投入获得质量显著提升的效。当然形式化方法也存在自身的局限性，如形式规约和初始需求之间的一致性难以保证、模型解释上可能存在二义性等。从价值软件工程的视角出发，应根据应用需求合理适度地使用形式化方法，并在形式化和非形式化之间寻找最佳平衡点，这样才能最有效地支持软件开发。形式化方法的研究和发展，也依赖于代数、逻辑、认识论、人工智能等其他相关学科的发展，形式化方法也需要与其他技术和方法进行更充分的结合。

（二）PAR 方法概述

PAR 方法和 PAR 平台（简称 PAR）[21]- [23]是一种基于形式化开发的程序设计综合开发环境平台，是一种比较统一的算法实现过程和设计理论研究以及证明方法[24]。建模语言以及可形式化推理验证的形式规则组成了 PAR 方法，涵盖了新的算法表示和建模语言，泛型程序设计方法和规约变换规则。它研究设计了从 SNL 模型到 Radl 模型再到 Apla 模型再到 C++, Java 等可执行语言程序的自动转

换平台[25]，提出了在软件模型设计和变换阶段实现泛型算法和程序设计的方法，和基于图形化的 MDE 和 UML 相比，PAR 具有便于软件形式化开发和验证的优点。基于这些理论和方法有些很难的算法和程序设计问题就会变得十分简单，并在建模语言中增加了新的泛型程序设计机制[26]显著提高了软件模型的简单性、抽象性和可靠性，使算法设计和程序设计的效率显著提高，提出了基于形式化方法的模型驱动软件开发方法并构建了模型自动转换平台。

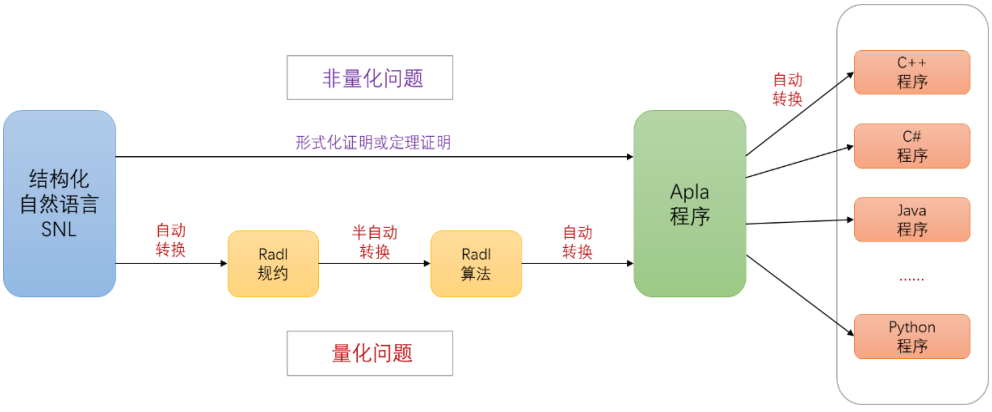

图 2-1 PAR 平台架构图

PAR 平台的架构如图 2-1 所示，从 Radl 规约到 Radl 算法的自动化转化目前仍然是一个难以解决的问题。

一般来说，开发问题的算法程序可以分为以下几个步骤：

1.用结构化自然语言构造问题的形式规约，基于形式语言精确描述算法所要做的工作。

2.将大问题分解为多个结构相同但规模更小的子问题，重复这个过程直到所有子问题都能够被直接解决。

3.构建一个问题求解序列的递推关系 Si＝F(Sj)，需要首先确定初始条件和涉及到的变量和函数。然后，将所有递推关系和初始条件组合成一个算法，使得每次迭代都能以前一次迭代的结果为输入，并输出下一次迭代的结果，直到求解出最终结果。

4.采用新的编写循环不变式的策略，需要开发适用于循环结构的不变式。这意味着需要使用递推关系来确定程序设计中必须使用的各种变量，并解释这些变量的规律和功能，以便在循环的每一次迭代中都能保持不变式的成立。

5.根据得出的算法和循环不变式，编写 Apla 算法程序。接着，使用自动化工具将 Apla 算法程序转换为等效的可执行程序，如 Java、Python 等。

相比于目前国内外普遍使用的基于图形化建模语言 UML 的软件模型驱动方法，该方法和平台更为简单、方便。研究成功的形式化模型驱动开发平台 PAR 是一种基础软件，具有广泛的应用范围。该平台已被用于开发许多难度和复杂度很大的算法程序和数据库应用软件，并在我国军方多项高可信软件开发中得到了广泛应用。上述研究成果中，多项属于原创性和突破性成果，具有重要的理论和实

际应用价值，对社会和经济产生了显著的影响。

2.2 人工智能与专家系统

人工智能（Artificial Intelligence，AI）是一种模拟和扩展人类智能的技术和科学，涵盖了机器学习、计算机视觉、自然语言处理、知识表示和推理、智能控制等多个方面[27]，目标是使机器能够像人类一样思考、学习和解决问题。自 2017 年以来，中国政府就开始大力推动人工智能技术的研究和应用，目前国内在人工智能领域的研究已经取得了很多进展：在深度学习领域已经取得了很多成果，例如阿里巴巴的 DAMO Academy、腾讯 AI Lab、百度研究院等在该领域拥有很高的研究实力；在人机交互研究领域也比较活跃，例如阿里巴巴的人机交互实验室、华为的用户体验中心等，这些机构通过研究人机交互技术，提高用户的体验，使人工智能更好地服务于人类；在机器人研究领域，例如中科院自动化所、清华大学机器人研究所等。这些机构在机器人智能、视觉感知、运动控制等方面取得了很多进展，为未来智能机器人的发展奠定了基础。在国外，谷歌、Facebook、IBM、微软等大型科技公司都在深度学习领域进行了大量的投入和研究。未来，随着技术的不断进步和应用场景的不断拓展，人工智能技术将会更加普及和深入，为人类带来更多的便利和机遇。

专家系统是实现人工智能的一项重要实践成果。它是一种基于专业领域知识和人类专家经验的计算机智能系统，旨在解决只有领域专家才能解决的复杂问题。它模仿了人类专家的知识和推理过程来解决特定领域的问题。它的工作原理是将专家的知识和经验储存到一个数据库中，然后使用推理引擎来模拟专家的决策过程，最终输出一个答案或建议。通常情况下，专家系统对于知识密集型的任务能够表现出色，这意味着只需耗费较少的成本来开发专家系统，就可以模拟人类专家解决特定实际问题的能力从而大大降低了开支[28],[29]。其核心在于知识表示与推理。与传统的基于数学模型的计算机程序不同，专家系统通过获取和组织领域专家的知识，使用推理机制进行逻辑推断，从而实现对决策问题的解决[30]。中国科学院院士、首届国家最高科学技术奖得主，他指出：“随着计算机的发展，机器将取代或减轻人类的脑力劳动。”专家系统已经对人类的物质文明和精神文明做出了重要的贡献，并将继续不断发展，为人类社会做出更多的贡献。因此，专家系统将成为 21 世纪人类智能管理和决策的有力工具和可靠的智能助手。

（一）专家系统实现人工智能的 4 个步骤：

1.知识表示和存储：将专家的知识和经验表示为规则或其他形式的语句，并将它们存储在专家系统的知识库中。

2.推理引擎：专家系统的推理引擎使用规则、逻辑、概率等技术来推理，以解决问题的建议。

3.用户接口：专家系统需要一个用户接口，以允许用户向系统提供问题和接收答案或建议。

4.学习能力：专家系统还可以具有学习能力，它可以通过分析已知数据来自动更新和改善其知识库和推理能力。

总的来说，专家系统通过模仿人类专家的知识和推理过程，实现了一定程度的人工智能，可以在一些特定领域中自动化处理决策和问题解决。

（二）根据其不同的实现方式和应用场景，可以将专家系统分为以下几类：

1.基于规则的专家系统：基于规则的专家系统是一种人工智能应用程序，它可以通过预先定义的规则来模拟人类专家的知识和推理能力。这些规则被设计为解决特定问题或领域中的特定任务。基于规则的专家系统由三个基本组件组成：知识库、推理机和用户接口。知识库是一个包含专家知识的数据库，其中包括问题、答案和规则。推理机是一个程序，它能够根据知识库中的规则进行推理，并输出相应的答案或建议。用户接口则是专家系统与用户交互的接口，它可以接收用户输入的问题或信息，并将推理机的结果反馈给用户。在基于规则的专家系统中，规则是最基本的元素。规则是一个条件和结论的组合，用于描述问题和答案之间的关系。例如，一个简单的规则可以是：如果有人说“我感到头疼”，那么推理机可以得出结论，这个人可能生病了。在知识库中，可以包含许多这样的规则，用于解决特定的问题，已经广泛应用于医疗、金融、法律和工业等领域。虽然它们可以通过事先定义的规则来解决问题，但也存在一些限制，例如它们可能无法处理复杂的非线性关系或大量的数据。此外，基于规则的专家系统也需要一定的人工干预和更新。由于规则库通常需要不断更新和修改以保持最新的知识，因此这需要专业人员进行定期的维护和更新。然而，基于规则的专家系统仍然具有许多优点。它们可以在较短的时间内提供高质量的答案和建议，并且可以自动执行繁琐的任务。此外，基于规则的专家系统也能够帮助初学者和非专业人员更好地理解领域知识和相关问题。随着人工智能技术的发展，基于规则的专家系统已经逐渐演变为更先进的技术，如基于知识图谱的专家系统、深度学习和自然语言处理技术等，这些新技术可以更好地应对复杂的问题和数据。如图 2-2 呈现了基于规则专家系统的完整架构：

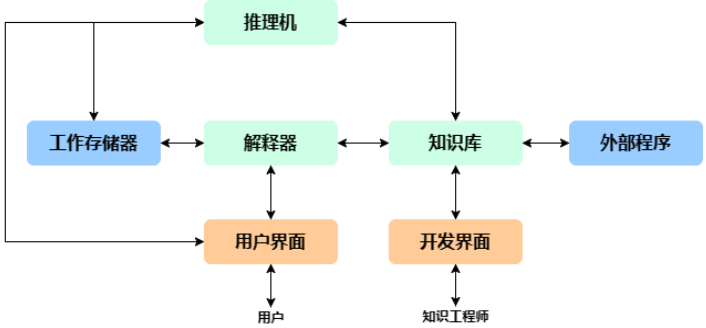

图 2-2 基于规则专家系统的结构

基于规则的专家系统是一种应用人工智能技术来模拟人类专家知识和决策的系统。如 CLIPS[31]是一个开放源代码的基于规则的专家系统开发工具，用于开发专家系统和决策支持系统；MYCIN 是一个早期的基于规则的医学专家系统，用于诊断疾病；PROSPECTOR[32]是一个用于地质勘探的基于规则的专家系统，用于发现矿藏和油气资源；京东研发的 AIX(AI 云决策)是一个基于规则的人工智能决策系统，用于处理订单异常问题等。这些国内外知名的基于规则的专家系统。它们被广泛应用于医学、金融、航空、制造等各个领域，可以帮助人们自动化和提高决策的准确性和效率。综上所述，基于规则的专家系统是一种重要的人工智能应用程序，它可以帮助人们快速地解决特定领域中的问题。虽然它们有一些限制，但随着技术的发展，基于规则的专家系统将继续发展和演变，为我们提供更加智能化和高效的解决方案。

2.基于框架的专家系统是一种基于规则的人工智能系统，其目的是模拟人类专家的决策过程，为用户提供决策支持。该系统使用事先定义的框架来组织知识，并使用推理机制来推断最优解决方案。基于框架的专家系统的工作原理如下：

a.知识获取：该系统需要获取与特定领域相关的知识，知识可以从人类专家、文献或其他来源中获取。

b.知识表示：获取到的知识需要以一种计算机可读的方式进行表示，例如规则、决策树或产生式等。

c.框架定义：系统需要定义一些框架，以便组织知识，并确定每个问题需要哪些知识。

d.推理机制：该系统使用推理机制来推断最佳解决方案。当系统接收到用户输入的问题时，它将基于框架和先前获取的知识进行推断。

e 用户界面：系统必须提供一个用户界面，以便用户可以与系统进行交互并获得决策支持。

基于框架的专家系统的优点在于，它可以处理复杂的领域，并提供可靠的决策支持。它还可以帮助减少错误，提高效率，并减少人工专家的需求。然而，它的缺点在于它需要大量的人工工作来获取和表示知识，以及定义框架和规则。

3.基于模型的专家系统：它可以模拟复杂系统的行为和决策过程，以预测和优化系统的性能[33]。与基于规则的专家系统不同，基于模型的专家系统使用数学模型来描述问题和决策过程，而不是预先定义的规则。基于模型的专家系统由两个基本组件组成：数学模型和推理机。数学模型是一个包含各种变量、参数和方程式的数学表达式，用于描述问题和决策过程。推理机则是一个程序，它可以根据数学模型和用户输入的数据进行推理，并输出相应的结果或建议。数学模型是最关键的元素。数学模型可以是线性模型、非线性模型、概率模型等等，其选择取决于特定的问题和领域。例如，在金融领域，可以使用基于时间序列的模型来

预测股票价格变化；在工程领域，可以使用基于流体力学的模型来模拟流体的运动和特性，应用于各种领域。它们可以通过模拟复杂系统的行为和决策过程来预测和优化系统的性能，并为用户提供相应的建议和解决方案。此外，基于模型的专家系统还可以自动学习和优化模型，从而不断提高自身的性能和效率。

4.基于 Web 的专家系统：基于 Web 的专家系统是一种能够在 Web 平台上运行的人工智能应用程序，它利用专业知识和推理技能来帮助用户解决复杂的问题。这些系统通过与用户进行交互，收集问题相关的信息，然后使用知识库中的规则和逻辑来提供准确的解答。基于 Web 的专家系统具有以下几个特点：可以通过 Web 浏览器在任何地方访问：基于 Web 的专家系统不需要用户安装任何软件，用户只需要通过浏览器访问相关网站即可使用系统；根据用户输入提供个性化的解答：系统通过与用户进行交互，了解用户的问题，然后根据用户提供的信息来给出个性化的解答。知识库可以随时更新：由于基于 Web 的专家系统的知识库存储在服务器上，因此系统管理员可以随时更新知识库中的规则和逻辑，从而提高系统的准确性和实用性。

5.基于案例的专家系统：这种专家系统主要使用案例库来存储和利用领域知识。它通过将先前的解决方案（即案例）存储在案例库中，并在需要解决新问题时使用类似的案例来进行推理。

6.基于知识图谱的专家系统：这种专家系统主要使用知识图谱来表示和推理知识。它通过将知识转化为一组节点和关系来表示，并使用图形结构来实现知识的推理。

7.基于神经网络的专家系统：这种专家系统主要使用神经网络来表示和推理知识。它通过使用神经网络来建立输入和输出之间的映射关系，并通过神经网络进行推理和决策。

2.3 专家系统技术和方法

基于规则的专家系统是一种常见的专家系统类型，其主要依赖于一组规则来表示和推理知识。在构建和应用基于规则的专家系统时，可以使用以下技术和方法[34]：

1.规则获取技术：规则获取是构建基于规则的专家系统的第一步，它涉及到如何从专家或其他知识源中获取规则。常用的规则获取技术包括：面向领域的知识工程方法、知识抽取技术、机器学习方法等。

2.知识表示方法：在基于规则的专家系统中，规则是主要的知识表示方式。为了有效地表示和利用知识，需要选择合适的规则表示方法。常用的规则表示方法包括：IF-THEN 规则、产生式规则、框架规则、决策树等。

3.规则推理技术：基于规则的专家系统的核心是规则推理，即根据已知事实

和规则推导出新的结论。在规则推理过程中，需要使用合适的推理技术，如前向推理、后向推理、双向推理、证明搜索等。

4.知识库管理技术：基于规则的专家系统中，知识库是存储和管理知识的重要组成部分。为了方便知识库的维护和更新，需要使用合适的知识库管理技术，如知识表示语言、知识库编辑器、知识库维护工具等。

5.系统测试和评估方法：为了保证基于规则的专家系统的正确性和可靠性，需要使用合适的测试和评估方法。常用的测试和评估方法包括：黑盒测试、白盒测试、模拟测试、专家评估等。

2.3.1 知识获取

知识获取是指从各种信息来源获取知识和信息的过程。这些信息来源可以包括人类专家、文献、数据库、实验数据等。知识获取的目的是将这些知识和信息转化为计算机程序能够理解和使用的形式，以支持计算机系统的自动化推理和决策。在人工智能领域中，知识获取通常是构建专家系统和其他智能系统的重要步骤之一。知识获取可以通过人工采访和分析、自动化数据挖掘和机器学习等方式实现。

知识获取的过程可以被分为以下几个步骤：

1. 确定知识获取目标：在开始知识获取之前，需要明确知识获取的目标和范围。这可以包括需要获取的领域知识、知识获取的时间和预算等。

2. 确定知识来源：确定从哪些信息来源获取知识和信息，例如人类专家、文献、数据库、实验数据等。

3. 收集和整理知识：收集和整理从不同信息来源获取的知识和信息，以便将其转化为计算机程序可以理解和使用的形式。

4. 知识表示：将收集到的知识和信息表示为计算机程序可以使用的形式，例如规则、本体、知识图谱等。

5. 知识验证和评估：对表示为计算机程序可以使用的知识进行验证和评估，以确保其准确性和可靠性。

知识获取是一个迭代的过程，需要根据系统的实际需求和反馈进行调整和改进。同时，知识获取过程中的人类专家是非常重要的资源，他们的知识和经验对于构建高效的专家系统和其他人工智能系统至关重要。

2.3.2 知识表示

知识表示是人工智能领域的一个重要问题，旨在将人类知识以计算机可处理的方式进行表示和存储。符号主义时期是知识表示的起始阶段，主要采用逻辑和语义网络等符号化方法来表示知识。20 世纪 70 年代到 80 年代初期，人工智能领域的先驱们提出了一些经典的知识表示方法，如产生式系统、框架系统、语义

网络、谓词逻辑等。这些方法均是以符号为基础，通过对符号进行组合、推理、匹配等操作来实现对知识的表示和处理。20 世纪 80 年代后期到 90 年代初期，随着神经网络和深度学习等连接主义方法的发展，知识表示进入了一个新的阶段。连接主义方法的核心思想是通过学习来获取知识和表示能力，而不是像符号主义方法那样人工定义符号和规则。这一时期的代表性工作包括：Kohonen 的自组织映射、Boltzmann 机和 Hopfield 网络等。这些方法不仅可以进行知识表示，还可以进行知识学习和推理。随着人工智能技术的不断发展，知识表示在混合智能时期得到了更为广泛的应用。在这一时期，研究人员将知识表示与其他人工智能技术相结合，如自然语言处理、知识图谱、机器学习等。这一时期的代表性工作包括 BERT、GPT、ELMo 等预训练语言模型，这些模型可以自动学习和表示语言知识，广泛应用于自然语言处理任务中。知识表示不仅在理论研究上有了很大的发展，也在实际应用中得到了广泛的应用。例如，在智能问答、推荐系统、金融风险管理等领域，知识表示技术都得到了广泛的应用。知识图谱也是一种广泛应用知识表示技术的形式，它将实体、关系和属性等知识以图谱的形式表示出来，被广泛应用于智能搜索、智能问答、智能客服等领域。当前，知识表示在人工智能领域中仍是一个活跃的研究方向，不断涌现出新的理论模型和算法。未来，随着深度学习、知识图谱、自然语言处理等技术的不断发展，知识表示也将在更多的应用场景中得到广泛的应用。

专家系统中的知识表示指的是如何表示和组织领域知识，以便计算机程序能够理解和利用这些知识来解决特定问题[35]。知识表示是专家系统中的一个关键问题，它涉及到如何将领域知识以计算机能够理解和处理的形式表示出来。

（一）常见的知识表示方法有[36]：

1.规则表示法：规则是专家系统中最基本的知识表示形式。规则表示法通常使用 IF-THEN 形式的规则来表示知识，其中 IF 部分称为前提，THEN 部分称为结论。例如，一个简单的规则可以表示为：IF A THEN B，其中 A 是前提，B 是结论

2.框架表示法：是一种用于描述对象、事物或概念的结构化知识表示方法。它将对象描述为一组属性和值的集合，其中属性描述对象的特征，而值描述这些特征的具体信息。

3.语义网络表示法：是一种基于节点和边的知识表示方法，其中节点表示对象或概念，边表示它们之间的关系。例如，一个语义网络可以表示为：汽车→有→引擎，其中“汽车”和“引擎”是节点，“有”是边，表示汽车具有引擎。

4.产生式表示法：产生式表示法是一种类似于规则表示法的知识表示方法，它使用一组 IF-THEN 规则来描述问题领域中的知识。不同之处在于，产生式规则中的 IF 部分通常更加复杂，可以包含多个条件和子规则

5.语言表示法：语言表示法是一种基于自然语言的知识表示方法，它通过对自然语言进行语义分析和形式化处理来表示知识。语言表示法的优点是易于理解和表达，但它也存在歧义和不确定性问题。

6.本体表示法：本体是一种描述特定领域中概念、实体和关系的结构化模型，本体表示法是一种用于表示本体的知识表示方法。本体表示法通常使用描述逻辑语言来表示本体，包括 OWL、RDF 和 RDFS[37]等。

7.产生式网络表示法：产生式网络是一种基于产生式规则的知识表示方法，它使用产生式规则来描述问题领域中的知识，同时将这些规则组织成一个有向图的形式。产生式网络可以通过节点之间的连接来实现推理和推断。

除了上述常见的知识表示方法外，还有许多其他的知识表示方法，如基于案例的表示法、神经网络表示法等。选择何种知识表示方法需要根据具体应用场景和领域特点进行选择和权衡。

（二）在专家系统中，知识表示不仅仅是将领域知识转换成计算机可处理的形式，还包括如何组织、存储和检索知识。以下是几种常见的知识表示技术：

1.知识库：知识库是专家系统中存储知识的数据结构，它通常包含一个或多个知识库文件，每个文件都包含一组相关的知识。知识库可以采用不同的格式进行表示，如文本、XML、JSON 等。

2.本体库：本体库是专家系统中存储本体的数据结构，它通常采用 RDF 和 OWL 等描述逻辑语言来表示本体。本体库可以帮助专家系统在不同领域和系统中实现知识共享和重用。

3.语义网：语义网是一种基于本体的知识表示和组织方式，它可以帮助专家系统在 Web 上进行知识共享和集成。语义网使用 RDF 和 OWL 等描述逻辑语言来表示知识，同时使用 URI 和 XML 等标准技术来实现知识的访问和交换。

4.数据库：是一种存储和管理数据的软件系统，它可以用来存储专家系统中的知识。数据库通常采用关系型数据库或 NoSQL 数据库来存储数据，可以提供高效的数据检索和管理功能。

5.索引技术：是一种用来加快数据检索速度的技术，它通过建立索引来记录数据的位置和关键字等信息。在专家系统中，可以采用全文索引、倒排索引和基于向量空间模型的索引等技术来实现知识的快速检索。

在知识表示方面，需要根据专家系统的应用场景和需求，选择合适的知识表示技术和存储方式。同时，还需要考虑知识的组织和分类、知识的版本控制和更新等问题。

2.3.3 知识推理

在人工智能发展的早期阶段，知识推理主要采用符号主义的方法进行研究。20 世纪 50 年代末到 70 年代初，研究人员主要关注如何用逻辑和符号表示知识，

并使用规则推理等方法实现推理。这一时期的代表性工作包括：Newell 和 Simon 的逻辑理论、McCarthy 的 LISP 语言[38]和规则推理系统以及 Woods 和 Schmolze 的语义网络等。20 世纪 80 年代后期到 90 年代初期，随着神经网络和深度学习等连接主义方法的发展，知识推理进入了一个新的阶段。连接主义方法的核心思想是通过学习来获取知识和推理能力，而不是像符号主义方法那样人工定义规则。这一时期的代表性工作包括：Rumelhart 和 McClelland 的反向传播算法、Hopfield 的自适应神经元模型和 Kohonen[39]的自组织映射等。21 世纪初，人工智能领域进入了一个混合智能时期。在这一时期，知识推理不再是单独的研究方向，而是与其他人工智能技术相结合，如机器学习、自然语言处理、大数据等。这一时期的代表性工作包括：基于规则的机器学习方法、知识图谱的构建和应用、深度学习与推理相结合的方法等。知识推理技术经历了从符号主义到连接主义再到混合智能时期的发展演变过程[40]。未来，随着人工智能技术的不断发展和应用场景的不断拓展，知识推理技术也将不断创新和发展。知识推理不仅在理论研究上有了很大的发展，也在实际应用中得到了广泛的应用。例如，在医疗诊断、工业自动化、金融风险管理等领域，知识推理技术都得到了广泛的应用。

在专家系统中，知识推理是一种基于已知事实或规则，从中推导出新结论或知识的过程[41]。它可以用于问题求解、决策制定、自动推理等领域，将领域知识应用到具体问题上，以产生新的知识或解决问题的过程。

（一）知识推理分为以下几种：

1.前向推理：前向推理也称为数据驱动推理，它是从已知事实出发，根据规则系统地推导出新的结论或知识的过程。前向推理通常从前提开始，逐步推导到结论。如果已知某些事实或条件，那么可以根据已知事实和规则推导出新的结论，直到得到最终结论或无法再推导为止。

图 2-4 展示了前向推理的推理示意图，其中包含三个规则：A1、A2 和 A3，而 R1、R2、R3 和 R4 则可以是命题或谓词。在 A1 规则中，R1 能够推出 R2；在 A2 规则中，R2 能够推出 R3；在 A3 规则中，R3 能够推出 R4。通过这三条规则形成的推理链，在给定 R1 的事实的情况下，系统能够依次推出 R2、R3 和 R4。

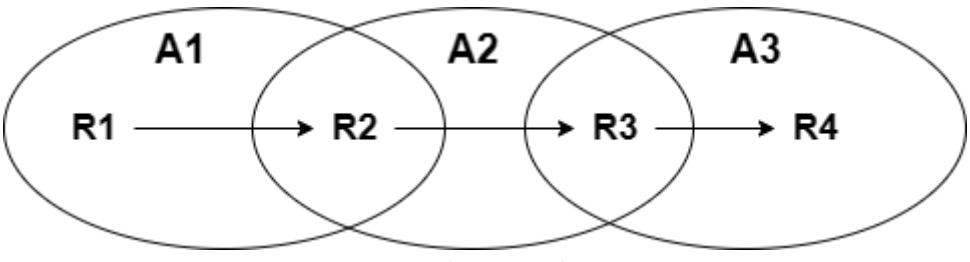

图 2-4 正向推理示意图

2.后向推理：后向推理也称为目标驱动推理，它是从目标或结论出发，根据

规则系统地推导出支持目标的前提或条件的过程。后向推理通常从结论开始，逐步推导到前提。如果已知一个目标或结论，那么可以根据规则逆向推导出支持目标的前提或条件，直到得到能够满足目标的前提或条件为止。

后向推理的过程中，使用了规则 A1、A2 和 A3，以及谓词或命题 R1、R2、R3 和 R4。为了证明 R4 成立，首先从 R4 出发，使用规则 A3 并假设 R3 成立，得到 R3 作为新的子目标。如果 R3 存在，则推理结束，因为 R4 也成立。如果 R3 不存在，则继续使用规则 A2，假设 R2 成立，验证 R2 是否存在。如果 R2 存在，则推导出 R3 成立，从而 R4 也成立，推理结束。如果 R2 不存在，则使用规则 A1，假设 R1 成立，验证 R1 是否存在。如果 R1 存在，则推导出 R2 成立，然后 R3 成立，从而 R4 也成立，推理结束。如果 R1 不存在，则无法验证 R2 和 R3，因此也无法验证 R4。图 2-5 展示了这个逆向推理的过程。

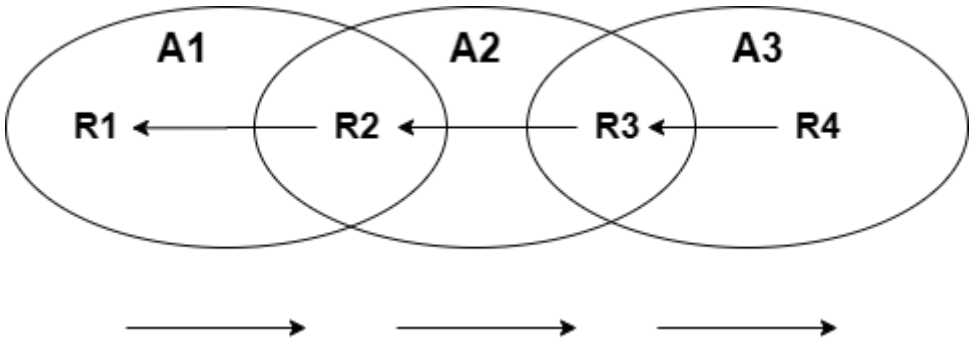

图 2-5 逆向推理示意图

3.混合推理：混合推理是前向推理和后向推理的结合，它根据具体问题和知识的性质，灵活地选择前向推理和后向推理的方法进行推理。混合推理可以充分利用前向推理和后向推理的优势，同时避免它们的缺点，从而提高推理的效率和准确性。

4.基于规则的推理：是专家系统中常用的推理方式之一，它基于一组规则和已知的事实，通过匹配规则来推导出新的结论或知识。规则通常由条件和结论组成，当条件满足时，就可以得到结论。基于规则的推理可以用于分类、推理、推荐等领域。

5.基于案例的推理：是一种类比推理方法，它通过比较一个新问题和已有的案例，从而产生新的结论或知识。基于案例的推理通常包括四个步骤：获取案例、案例比较、结论生成和结论评估。

6.基于模型的推理：是一种基于数学模型的推理方式，它通过建立模型来描述问题和知识之间的关系，并用模型进行推理和预测。基于模型的推理通常需要采用统计学、机器学习等技术进行模型建立和参数估计。

总之，不同的知识推理方式适用于不同的问题和应用场景，需要根据具体情况进行选择。同时，还需要考虑知识的表示和存储方式、推理策略和推理效率等

因素，以便优化专家系统的推理性能且知识推理是专家系统的核心功能之一，它可以帮助专家系统解决具体问题、产生新的知识和发现隐藏的关联性等，对于提高专家系统的应用价值和实用性具有重要意义。

（二）知识推理的过程：

1.收集和表示知识：首先需要将相关的事实、规则和知识收集起来，并用一种适当的形式进行表示和存储，例如语言、图形或逻辑表达式。

2. 统一和归纳知识：将收集到的知识进行分类、归纳和整理，以便更好地理解和应用。

3. 推理过程：根据已有的知识和规则，应用逻辑推理、数学推理、概率推理等方法，从中推导出新的结论或知识。

4.验证和评估：对推导出的新结论或知识进行验证和评估，确保其正确性和可靠性。

知识推理是一种重要的人工智能技术，可以帮助人们解决各种问题和决策，提高工作效率和准确性。此外，知识推理也是构建智能系统的关键技术之一。例如，在智能对话系统中，需要根据用户输入的语句和已有的知识库，进行语义理解和推理，生成合适的回复；在自动驾驶系统中，需要根据传感器采集到的数据和路况信息，进行决策和规划，实现自动驾驶。需要注意的是，知识推理的效果和准确性往往取决于所使用的知识和规则的质量和完备性。如果知识库中存在错误、不一致或不完备的情况，可能会导致推理结果不准确或不可靠。因此，建立高质量的知识库和规则，是实现有效知识推理的关键。

2.3.4 产生式系统

产生式系统最早可以追溯到 20 世纪 50 年代。当时，约翰·麦卡锡（John McCarthy）等人在人工智能领域中提出了“程序员可以编写能够自主学习的程序”这一观点，而产生式系统就是其中的一种重要实现方式。1960 年代初期，另一位人工智能领域的重要人物马文·明斯基（Marvin Minsky）在其著作《Steps toward Artificial Intelligence》中首次提出了产生式系统的概念，标志着该技术开始正式被人工智能研究者所关注和应用。20 世纪 60 年代末期和 70 年代初期，产生式系统得到了快速的发展，出现了很多经典的产生式系统模型，如 MYCIN、DENDRAL、OPS5 等。其中，MYCIN[42]是一个专门用于诊断疾病的产生式系统，能够根据输入的病例信息自动判断病情并推荐治疗方案。DENDRAL 是一个专门用于分析化学分子结构的产生式系统，可以通过分析化学分子的质谱信息和 NMR 谱图等信息，自动推断出分子的结构。OPS5 则是一种通用的产生式系统，可广泛应用于领域专家系统、机器学习和自然语言处理等领域。产生式系统在过去的几十年中已经得到了广泛的应用，其应用领域包括医学、化学、机器人、控制系统、金融等各个领域。产生式系统在医学诊断中的应用已经被证明是非常成

功的，例如在冠心病的诊断中，产生式系统的准确率可以达到 96%以上。产生式系统也被广泛应用于人工智能领域的其他子领域中，如机器学习、自然语言处理、机器人等。虽然产生式系统在很多领域都取得了成功，但它也存在一些局限性。其中最大的一个局限性就是其处理效率和推理能力的限制[43]。由于产生式系统中规则的匹配和推理是基于符号层面进行的，因此当知识库中规则和事实的数量达到一定程度时，系统的推理效率就会急剧下降。此外，由于产生式系统中只能处理符号层面的知识表示，因此其在处理涉及到数值和实数的问题上也存在一定的局限性。为了解决产生式系统存在的局限性，研究者们提出了一些改进方法，如 Rete 算法、物化视图等。此外，近年来深度学习技术的兴起也使得产生式系统的发展受到了一定的影响。一些研究者将产生式系统与深度学习技术相结合，提出了基于深度学习的产生式系统，用于处理更加复杂和抽象的问题。总的来说，随着人工智能领域的不断发展，产生式系统作为一种经典的知识表示和推理方法仍然具有重要的地位和应用价值，其发展也将受到新技术和新方法的不断推动和促进。

产生式系统是一种基于规则的推理机制，它是专家系统的一种重要形式。它的核心思想是基于一系列规则和事实，通过匹配规则和推理过程，从而产生新的结论或知识。

（一）产生式系统的组成

产生式系统通常由三部分组成：产生式规则库、总数据库和控制策略。如图 2-6 所示：

1.产生式规则库包含了一系列的规则，每个规则通常由前提和结论两部分组成，前提是一组逻辑表达式，用来描述已知事实的条件，结论是根据前提得出的结论。

2.总数据库用来存储已知事实，它是一个动态的数据结构，可以根据问题和规则动态地变化。

3.控制策略则是用来匹配规则和执行推理过程的核心组件，它负责对工作内存中的已知事实和规则库中的规则进行匹配，从而产生新的结论或知识。

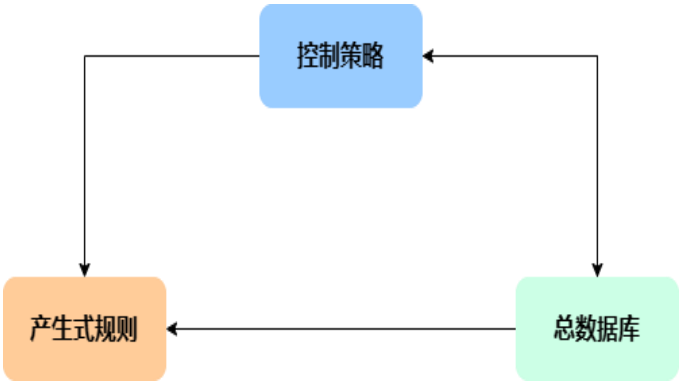

图 2-6 产生式系统的主要组成图

（二）产生式系统的推理过程

产生式系统的推理过程通常分为两个阶段：匹配阶段和执行阶段。

1.在匹配阶段，推理机会根据工作内存中的已知事实，逐一匹配规则库中的规则，找出满足条件的规则。

2.在执行阶段，推理机会根据匹配到的规则，从工作内存中获取相应的事实，执行规则的结论部分，并将新的结论加入到工作内存中。随着推理的进行，工作内存中的已知事实和新的结论会不断地增加和变化，直到得到最终的结论或无法再产生新的结论为止。

（三）产生式系统具有以下特点：

1.灵活性：由于规则库和工作内存都是动态的，因此产生式系统非常灵活，可以根据具体问题和应用场景进行配置和调整。

2.可扩展性：产生式系统支持增加和修改规则库，因此可以方便地扩展和更新领域知识。

3.可读性：产生式系统中的规则通常由条件和结论组成，因此易于理解和调试。

4.高效性： 由于规则匹配和执行都是基于规则的，因此产生式系统具有较高的推理效率。

（四）在实际应用中，产生式系统还可以进一步扩展和改进，以满足不同应用场景和问题的需求。以下是一些常见的产生式系统扩展和改进方式：

1.带有优先级的规则：为了解决规则匹配的冲突问题，产生式系统可以引入规则优先级的概念，规则按照优先级顺序执行。这样可以确保高优先级规则得到优先执行，从而提高推理效率。

2.带有权重的规则：为了更好地处理不同规则之间的关系，产生式系统可以引入规则权重的概念，权重反映了规则的重要性和可信度。在规则匹配和执行时，可以根据规则的权重进行调整。

3.反馈机制：为了更好地处理复杂的推理问题，产生式系统可以引入反馈机制，通过将产生的新结论重新放回工作内存中，实现新的规则匹配和推理过程。这样可以使系统更加灵活和智能。

4.带有模糊逻辑的规则：为了更好地处理模糊和不确定性的问题，产生式系统可以引入模糊逻辑的规则，使用模糊量化的方式表示规则的条件和结论，从而更好地处理模糊问题。

总之，产生式系统是专家系统中一种重要的知识表示和推理机制，它具有灵活性、可扩展性、可读性和高效性等特点。在实际应用中，可以根据具体问题和应用场景进行扩展和改进，以满足不同需求。它在许多领域都有广泛的应用，如医疗诊断、机器人控制、工业控制等。

2.3.5 RETE 算法

RETE（Rete Algorithm）算法是一种用于高效执行基于规则的专家系统推理的算法。该算法最早由美国卡内基梅隆大学的 Charles L. Forgy 于 1974 年[43]提出，被视为规则匹配算法的重大进展，是基于规则的专家系统领域中最为著名的推理引擎之一。1980 年代，RETE 算法开始被广泛应用于专家系统和决策支持系统等领域。此时，RETE 算法还存在一些问题，比如不支持动态添加规则等。1990 年代，研究人员提出了一些改进 RETE 算法的方法，比如 TREAT 算法和 Rete-II 算法，以解决 RETE 算法存在的一些问题。2000 年以后，随着大数据和人工智能技术的发展，RETE 算法也得到了更广泛的应用，同时，研究人员也提出了一些新的规则匹配算法，比如 Leaps 算法和 Rete-NT 算法等。在国内，RETE 算法的研究和应用起步较晚，大部分应用都是在近几年逐渐增加的。目前，RETE 算法已经被广泛应用于金融、电力、医疗等领域。在国外，RETE 算法的研究和应用也非常活跃，除了传统的专家系统和决策支持系统等领域，RETE 算法也被广泛应用于人工智能、机器学习、自然语言处理等领域。此外，研究人员也提出了一些基于 RETE 算法的变种，比如 Rete-UL[58]算法、Rete-MT 算法等。RETE 算法的发展历程充满了不断创新和应用的过程。未来，随着人工智能技术的不断发展，RETE 算法的应用场景也将会更加广泛，同时也会有更多的研究人员对其进行改进和优化。

（一）RETE 算法的核心思想

RETE 算法的核心思想是构建一个规则匹配网络，通过将规则集中的规则表示为一个网络结构，来高效地进行匹配和推理。RETE 算法的网络结构主要由两种节点组成：测试节点和连接节点。其中，测试节点对应规则中的条件部分，连接节点对应规则中的动作部分。连接节点将满足测试节点的数据传递给下一个测试节点或连接节点。

（二）RETE 算法的优缺点

RETE 算法的优点在于它可以高效地处理大规模的规则集合，而且能够在规则集合发生变化时快速地更新网络结构[44]。此外，RETE 算法还支持深度优先搜索和广度优先搜索等不同的匹配策略，使得它可以应用于不同类型的问题。然而，RETE 算法也存在一些缺点，例如需要大量的内存来存储网络结构，网络结构的复杂度会随着规则集合的增加而增加。此外，RETE 算法在处理规则集合较小的情况下可能不如其他算法高效。总的来说，RETE 算法是一种非常重要的基于规则的专家系统推理算法，具有广泛的应用前景。

（三）RETE 算法的实现包含两个主要的阶段[45]：构建阶段和匹配阶段。

1.构建阶段：在这个阶段中，将规则集合转化为一个规则网络。这个过程可以分为以下步骤：

a.将规则条件部分表示为测试节点，并将测试节点连接到动作部分对应的连接节点。

b.对于多个规则中相同的条件部分，将它们合并为一个测试节点，这样可以避免重复的计算。

c.对于一些常用的测试节点，例如比较相等或不相等等，可以预先创建它们并存储在一个节点库中，这样可以提高匹配效率。

2.匹配阶段：在这个阶段中，将输入数据与规则网络进行匹配，找到满足条件的规则并执行相应的动作，匹配过程如图 2-7 所示。

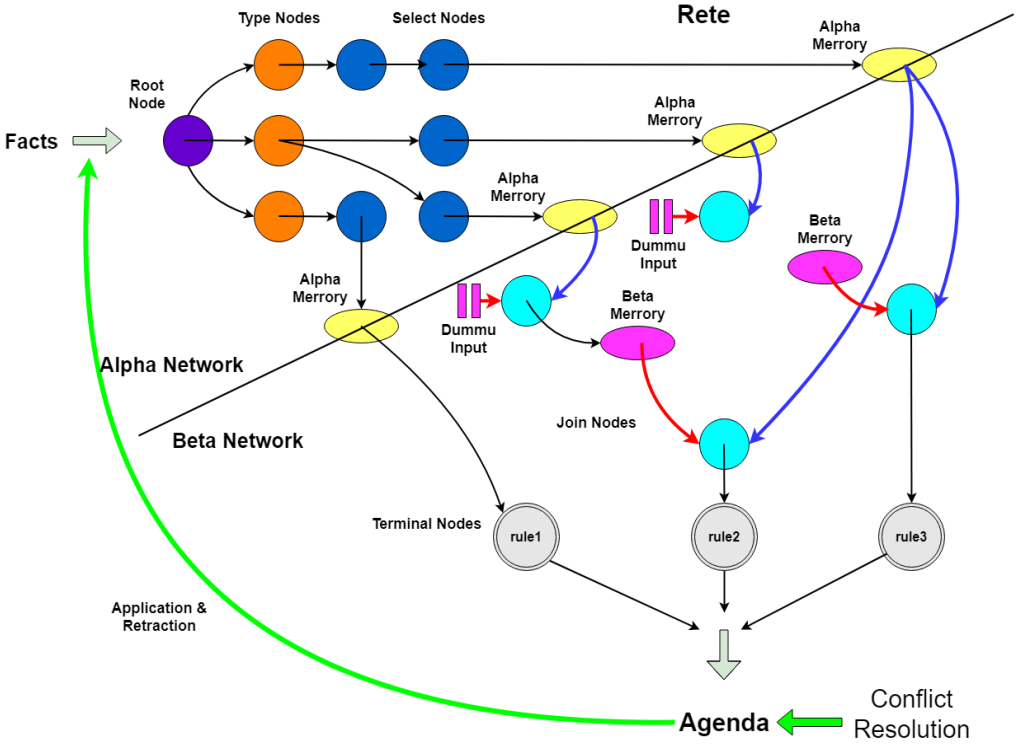

图 2-7 模式匹配算法图

这个过程可以分为以下步骤：

a.输入数据到达测试节点，测试节点对数据进行测试并传递到连接节点。

b.连接节点将测试节点的数据与之前存储的数据进行比较，如果满足条件，则将数据传递给下一个测试节点或连接节点。

c.如果一个连接节点接收到了满足其所有前置条件的数据，则该连接节点将触发相应的规则动作，即执行规则中定义的操作。

d.当规则集合发生变化时，可以快速地更新网络结构，这可以通过缓存一些中间结果和利用前面匹配过程中的剪枝策略来实现。

总之，RETE 算法通过建立网络来减少规则匹配的次数，RETE 算法可以大大提高规则引擎的性能。

第 3 章　泛型算法程序语言 Radl

3.1 Radl 总体结构

Radl 语言[46]-[49]是一种基于递推关系的算法设计语言，本文用 Radl 描述问题规约、规约变换规则和算法描述。它支持快速原型和直接执行，并且具有引用透明性，便于算法的形式推导。Radl 和 Apla 语言的格式完全一致，目前主要适用于高可靠软件部件的开发。本章介绍了 Radl 语言的总体结构、数据类型和变量声明，还给出了 Radl 算法设计开发实例。对于问题规约和算法描述方面，Radl 语言是一种强大且灵活的工具[50]。

（一）规约程序结构：

算法规约由三个主要部分组成[51]：

|[标识符说明]:定义规约中出现的变量和函数的类型，以及辅助变量的定义

<前置条件>:指定了算法在执行之前必须满足的条件

<后置条件>:指定了算法在执行之后必须满足的条件。

（二）算法程序结构[52]：

通过研究实践证明，通常的算法可以被表示的递推关系组结构为：

Algorithm: <算法名称>

{<算法规约>}

Begin: <递推控制变量和递推函数初始化>

Termination: <递推终止条件>

Recur: <递推关系式组>

End.

（三）规约程序转换规则

在 PAR 方法中要得到问题的递推关系，要用到以下推导规则[53]。规约转换规则大多是量词性质,我们用 θ 表示满足交换律和结合律的二元运算符,

即有: (θ i : r(i) : f(i))

在对函数 f(i)在 r(i)的范围上进行运算后所得的量，具有以下性质，即结合律和交换律成立，其量词 θ 也满足以下条件：

1.交叉积性质：

(θ i, j : r(i)∧s(i, j) : f(i, j)) = (θ i : r(i):(θ j : s(i, j) : f(i, j)))

2.范围分裂性质：

(θ i : r(i) : f(i)) = (θ i : r(i)∧b(i) : f(i)) θ (θ i : r(i)∧¬b(i) : f(i))

3.广义结合律和交换律：

(θ i : r(i) : s(i) θ f(i)) = (θ i : r(i) : s(i)) θ (θ i : r(i) : f(i))

3.2 数据类型和变量说明

（一）标准数据类型和变量说明：

标识符重载思想体现在算术运算符和关系运算符中。这些运算符的含义取决于它们作用的对象。表 3-1 列出了标准数据类型和变量说明。

表 3-1 标准数据类型和变量说明

符号	类型	描述
Integer	整型	+，—，*，mod,div
Real	实型	+，—，*，/
Char	字符型	
Boolean	逻辑型	∧，∨，┐，=，Cand，Cor
String	字符串型	
通用关系运算符		≠，<，>，=，≤，≥

（二）自定义类型:

语法:type <类型名>=<类型说明>

1. 记录类型:

```
type  student=record
        number:1..99999;
        name:seq;
        score:0..100;
        s:student;
  end.
```

则记录分量可用 s.number，s.name，s.score 来表示。

2.枚举类型

type<类型名>={e_1,e_2,…, e_n};

e_1,e_2,…, e_n 代表不同的标识符可以是不同类型。

枚举类型可以有赋值，关系等运算，还可计算 pred，succ，ord 等函数值。

3.预定义类型：

经过对大量复杂算法程序进行抽象描述和形式化开发的实践，我们定义了以下类别的抽象组合数据类型。每个类型都包含一组基本操作，这些基本操作可以

结合起来实现更为复杂的操作。在类型定义中，size、基类型、元素类型和结点值类型都是类别参数，用户可以根据需要自行定义。

a.数组类型

创建一个名为“数组类型名”的数组类型，其下标类型为<下标类型>，元素类型为<元素类型>。下标类型可以是子界类型、枚举类型等。

数组的元素定义为：数组变量名[下标]。

例如，定义一个数组类型 Ar，其下标类型为 0 到 n 之间的整数，元素类型为实数： type Ar = array [0..n, real]; 定义两个数组变量 A 和 B，都属于类型 Ar：var A, B: Ar; 可以用 A[i]和 B[j]来表示数组元素。

b.集合类型和包类型

一个集合是由一组无重复元素的数据构成的。与之不同的是，包允许有重复元素。我们可以定义一个集合类型，称为 set，其基类型为<基类型>。同时，size 是一个可选项，如果省略，则表示集合元素的数量没有限制。类型定义如下：

操作符集的使用和具体描述，如表 3-2 所示：

表 3-2 操作符集说明

操作	描述
#(A)	表示 A 中元素的个数
A∩B	表示 A 和 B 的交集
AUB	表示 A 和 B 的并集
A - B	使 A 中保留不在 B 中出现的元素
X∈A	若 x 在 A 中取值为 true，否则为 false
A⊃B	若 B 是 A 的真子集，取值为 true，否则为 false
A = B	若 A 和 B 完全相同取值为 true，否则为 false
A := B	A 被 B 替换，若 B 为空，则 A 被置空
Set_iterate(A, I, ops)	集合 A 的集成迭代算子

上面定义的集合类型及其上的操作符全部适用于包，但有些操作的实现算法有所不同。因此在实现包类型时，必须考虑集合和包的区别。

c.序列类型

操作符集的使用和具体描述，如表 3-3 所示：

表 3-3 操作符集说明

操作	描述
#(S)	表示 S 中元素的个数
S[i..j]	表示 S 的子序列，h≤i ，j≤t
S ↑ R	序列 S 的尾和 R 的头连接，形成一条新序列，操作被表示为“S+R”，“+”不遵循交换律，可利用进行插入和 删除操作
S[i]←→S[j]	S[i] 和 S[j] 进行交换
S = R	若 S 和 R 完全不同取值为 false，否则为 true
S[i..j] := R[k..k+j-i]	S[i..j]被 R[k..k+j-i]替换
S := R	若 R 包含值，用 R 替换 S，否则相反
Choose(x, S, C)	如果条件 c 成立，则从集合 S 中随机选择一个元素存储到变量 x 中，并且可以通过执行语句 S := S−x 中去将 x 删除，否则，x 不会被赋值或保持不变

显然，若元素类型为 char，则 S 是一字符串；若 t 固定不变，对序列的插入和删除由 h 决定，则 S 是一堆栈；若删除元素由 h 决定，插入元素由 t 决定，则 S 为一队列。

d.树类型

由于二叉树是常用的树结构，我们将研究二叉树的类型定义及操作。

操作符集说明如表 3-4 所示：

表 3-4 操作符集说明

操作	描述
#(T)	表示树 T 中结点个数
T.d	二叉树的根结点值部件
T.l	二叉树的左子树
T.r	二叉树的右子树
n + T	加结点 n 到 T 使之成为和 T 结构相同的新树使用这一操作可以建立一颗二叉树
T - n	从 T 中删除结点 n 使之成为和 T 结构相同的新树
$T.d_1 \leftarrow\rightarrow T.d_2$	将树两结点的值 d_1 和 d_2 交换

续表 3-4 操作符集说明

操作	描述
$T_1=T_2$	若 T_1 和 T_2 完全相同取值为 true，否则为 false
$T_1 := T_2$	T_1 被 T_2 替换。若 T_2 为空，则 T_1 被置空；
Made(T_1, n, T_2)	将二叉树 T_1，T_2 和结点 n（作为根结点）连成一新的 二叉树
Choose(n, T, C)	从 T 中选一满足条件 C 的结点存入 n 中
Transform(T_1, T_2)	转换多叉树 T_1 成为二叉树 T_2
Pre_iterate(T, i, ops)	先遍历二叉树 T
In_iterate(T, i, ops)	中序遍历二叉树 T
Post_iterate(T, i, ops)	后序遍历二叉树 T
Lever_iterate(T, i, ops)	层次遍历二叉树 T

e.带权有向图类型

有向图是一种图形结构，可以通过将每条无向边拆分成两条有向边来表示。它可以用邻接矩阵或邻接表两种方式表示。对于带权有向图，可以表示为一个二元组 G=(V,E/W)，其中 V 表示顶点集，E 表示边集，W 表示每条边的权重集合，E/W 是带权边的集合，即 E/W={(u,v)/w|(u,v)∈E,w∈W}。对于一个顶点 a，它的入弧集合可以表示为 in(a)，出弧集合可以表示为 out(a)。通常情况下，一个顶点的入弧集合和出弧集合不相等，可以用 in-degree(a)表示顶点 a 的入度（即入弧数），用 out-degree(a)表示顶点 a 的出度（即出弧数）。邻接集合 Adj(a)可以表示顶点 a 的所有邻接顶点组成的集合，即 Adj(a)={v∈V|(v,a)∈E}。

操作符集说明如表 3-5 所示：

表 3-5 操作符集说明

操作	描述
e.w	边 e 的权值，e.w 的值可以赋值语句改变
e.h	e 的头顶点
e.t	e 的尾顶点
e.f	若 f 为 1，e 已被访问；f 为 0，e 尚未被访问。

续表 3-5 操作符集说明

操作	描述
v.d	顶点 v 的值或标号，可用赋值语句改变。
v.f	若 f 为 1，v 已被访问；f 为 0，v 尚未被访问。
v = u	可值可用赋值语句改变。
v := u	顶点 v(可以是 e.h 或 e.t)被 u 替换
e1 = e	判断两条边是否完全相同
e1 := e	边 e1 被 e 替换
G.V	G 的顶点集
G.E	G 的边集
#(G.V)	G 的顶点集
#(G.E)	G 的边数
In(u)	由顶点 u 射出的边的终点集
out(u)	图 G 的一条路径 p 的长
Length(p)	从 G 中选一条满足条件 C 的顶点存入 v 中
Choose(v, G, C)	从 G 中选一条满足条件 C 的边存入 e 中
Dfs iterate(V, i, ops)	Dfs 方式遍历图 G 顶点集 V 的集成迭代算子
Dfs_iterate(E, i, ops)	Dfs 方式遍历图 G 边集 E 的集成迭代算子
Bfs_iterate(V, i, ops)	Bfs 方式遍历图 G 顶点集 V 的集成迭代算子
Bfs_iterate(E, i, ops)	Bfs 方式遍历图 G 边集 E 的集成迭代算子

3.3 泛型程序设计

泛型程序设计是一种高度抽象的程序设计方法，它可以显著提高程序开发效率和可靠性。我们定义的 Radl 和 Apla 语言具有丰富和灵活的泛型机制，可以支持类型作为参数和程序作为参数。然而，Java 缺乏清晰的泛型程序设计机制，这给使用 Java 进行泛型程序设计带来了很大的困难，并阻碍了将泛型程序设计语言 Apla 转换成对应的 Java 程序。因此，我们探索了利用 Java 提供的语言机制，通过泛型参数化和多态实现操作（子程序）的参数化来解决这个问题。我们还提供了一种将 Apla 程序自动转换成泛型 Java 程序的方法和实现模型。实践证明，

这个构想在 Java 泛型程序设计实践中得到了成功应用，可以显著提高 Java 程序设计的效率和可靠性。在 Apla-Java 程序自动转换器的构建中，我们也获得了成功。

Radl 语言提供了丰富的泛型程序设计机制，主要包括两种：类型参数化和操作参数化。其中，类型参数化使用关键字“sometype”，可以将类型作为参数传递；操作参数化使用关键字“somefunc”，可以将操作（子程序）作为参数传递。这些机制使得 Radl 语言的泛型程序设计更加简便。

例如：可以定义数组类型：

type array=array(0:n,sometype elem);

我们定义了一个特定类型的数组类型，其元素类型仅限于一些特定的类型。这是因为在对数组元素进行排序时，只有具有相同类型的元素才能进行比较运算。因此，我们需要将数组类型限制在一些常见的类型中，以便可以进行元素比较。我们使用关键字 sometype elem 来表示这个特定的元素类型。需要注意的是，anytype 的取值范围取决于 elem 出现的上下文环境，这个限制条件比较复杂，我们仅约定 sometype elem 表示 elem 是一个类型变量。

3.4 量词表示方法

在 Radl 语言中，所有的量词都以（Qi:r(i):f(i)）的统一形式呈现。

1.全称量词：∀

“在 m≤i≤n 范围中，对于所有的 i，若 Ei 为真，则可使用逻辑表达式(∀i:m≤i≤n:Ei)。这里使用的是全称量词，它的一般化形式是二元运算“∧”。

2.存在量词：∃

表示“在 m≤i≤n 范围中至少存在一个 i,使得 Ei 为真”的逻辑表达式为(∃i:m≤i≤n:Ei)。

3.求最大值量词:MAX

“要表示在 m≤i≤n 范围内求所有 Si 的最大值，可以使用逻辑表达式 (MAX i:m≤i≤n:Si)。这里的 Si 为数值型表达式，求最大值量词是二元运算“max”的一般化形式。”

4.求最小值量词:MIN

“要表示在范围 m≤i≤n 中求所有 Si 的最小值，可以使用逻辑表达式(MINi:m≤i≤n:Si)。这里的 Si 为数值型表达式，求最小值量词是二元运算“min”的一般化形式。”

5.求和量词：∑

“要表示在 m≤i≤n 范围内对所有的 Si 求和，可以使用逻辑表达式(∑i:m≤i≤n:Si)。这里的 Si 为数值型表达式，求和量词是二元运算“+”的一般化形

式。”

7.计数量词:N

“要表示在范围 m≤i≤n 中求使 Ei 为真的 i 的个数，可以使用逻辑表达式(Ni:m≤i≤n:Ei)。这里的 Ei 为布尔表达式。此外，∩表示求交运算符，表示多个集合的公共部分；∪表示求并运算符，表示多个集合的合并。”

8.求积量词：∏

“表示在 m≤i≤n 范围内对所有的 Si 求积的逻辑表达式为 (∏i:m≤i≤n:Si)。这里的 Si 为数值型表达式，求积量词是二元运算“*”的一般化形式。”

3.5 Radl 语言算法程序开发实例

以 1 个常见问题为例，使用 Radl 语言描述他们的问题规约和算法程序。

例 1 只用加法计算数组元素下标的四次方

1.精确描述问题的程序规范可以帮助我们获得问题的规约模型，该模型的前置断言 Q 和后置断言 R：

Q: 给定空整型数组 b[0..n-1];

R:{j: 0≤j<n: b(j)=j^4};

2. 通过将原问题划分为若干与原问题相同的子问题，并利用递推关系构建算法，我们可以设计 Radl 算法程序：

```
Algorithm: biquadrate
|[b: array[0..n-1,integer],j,c,d,e,f,g,h,k: integer;]|
Q: True
R: {j: 0≤j<n: b(j)=j^4}
Begin:j=0++1;b(j)=0;c(j)=0;d(j)=0;e(j)=0;f(j)=0;g(j)=0;h(j)=0;k(j)=0;
Termination: j=n;
Recur:
b(j+1)=b(j)+c(j)+g(j)+k(j)+1;c(j+1)=c(j)+d(j)+f(j)+4;
d(j+1)=d(j)+e(j)+12;e(j+1)=e(j)+24;
f(j+1)=f(j)+12;g(j+1)=g(j)+k(j)+6;
h(j+1)=h(j)+12;k(j+1)=k(j)+4;
End
```

Radl 算法由 Algorithm、Begin、Termination、Recur 和 End 这几个关键部分构成。其中，Algorithm 标识算法的名称；Begin 部分给出了递推关系中标识符的初始值；Termination 部分给出了算法的终止条件；Recur 和 End 部分则反映了算法的递推关系成分。