24-29 产生式系统 人工智能 程序语言 ⑤

计算机科学1993 Vol. 20№.4

产生式系统的两个发展方向

TP18

——实时产生式系统和协同产生式系统

李明树 唐朔飞

（哈尔滨工业大学计算机系 哈尔滨150006）

摘 要

产生式系统是当今最负盛名的一种人工智能程序设计语言，然而它在实时处理、协同求解方面的应用却已被证明比较困难。本文则在大量相关工作基础上，详细讨论了实时产生式系统、协同产生式系统实现的可能途径和有关方法。

一、引言

目前，产生式系统，也称为基于规则系统，已成为当今最负盛名的一种人工智能程序设计语言。然而，专家系统技术正越来越多地开始转向诸如航空航天、过程控制，医疗监测及机器人学等复杂和实用领域。尽管其它一些基于知识方法得到迅速发展，但传统产生式系统在实时处理、协同求解方面的应用却已被证明比较困难。因此，讨论实时产生式系统、协同产生式系统方法实属必然。

二、产生式系统在实时处理、协同求解方面应用的不足

关于“实时”的概念有多种定义，比如，“处理过程中使用起来足够快”[1]；“不考虑采用算法，系统产生应答有一个严格的时间界限”[2]；“在一个（确定域下）固定时间之后，系统保证产生一个应答的能力”[3]；及“（系统的）操作具有良好反应措施”[4]等等。

实际上严格来说，尽管速度是实时性能最根本的要求，但只有速度还不是实时。一个系统的实时性能应该从四方面考察：（1）速度：任务执行的速率；（2）应答：系统对到来事件处于准备状态的能力；（3）及时：系统通过首先处理最紧急任务来满足时间期限要求的能力；（4）适应：系统根据工作载荷或现存资源的变化重排任务优先级的能力。

协同问题求解是指分布在多个处理器上的、平等的、松散耦合的多个知识源互相协作，共同解决同一问题的一种问题求解方法。这里，平等的知识源指的是没有一个知识源对其它知识源有控制权，也不存在控制全局的知识；松散耦合的知识源指的是每个知识源用于计算的时间远远大于用于通信的时间；而知识源的互相协作指的是这些知识源中没有一个具有解决整个问题的足够知识，因此为了得到问题的解答，必须允许多个知识源共享知识。事实上，协同问题求解最突出的特征之一就是，各知识源只有局部的、不完全的知识。因此，困难就在于：基于局部的、不完全知识的知识源，其动作的聚合如何保证全局的协同。

收到日期：92-10-14

协作有两种方式，即任务共享和结果共享。任务共享是指分布在不同处理器的多个知识源通过执行分配给它们的子任务而共同分担计算任务；结果共享则指多个知识源通过共享部分计算结果而完成整个问题的求解。

一般来说，产生式系统表示形式单一且用户易于理解，因此成为最重要的知识表示方法。但同时它也有诸如规则间的相互关系不明显，知识的整体形象难以把握，难于管理和维护；处理效率低；推理缺乏灵活性，容易出现控制饱合问题；与真正专家的知识结构不同等等不足[5,6]。影响产生式系统在实时处理、协同求解方面应用的两个主要问题是：

1．执行速度低

产生式系统处理实时任务时，其搜索/匹配时间要占全部计算时间的90%。通常，一个确定问题域中的产生式系统包含成百上千个产生式规则和数据元素，大量计算要求出现在PM（产生式存储器）中的产生式规则与WM（工作存储器）中的数据元素之间的匹配过程中并成为影响产生式系统运行效率的主要瓶颈。在一些动态变化环境下，产生式系统的推理速度尤其不能满足要求。

2．对外部事件不敏感

多数产生式系统处理输入数据的唯一方式是在执行过程中执行选定规则RHS动作的"读"语句，这就带来两个局限性：(1) 外部数据只有通过执行一个确定的读语句才能输入。产生式系统仅能在程序运行期间的某一特定时间处理高确定性数据，却不能接受和处理具有随机和突然特征的实时数据。(2) 外部数据变化不能直接影响产生式系统程序控制。产生式系统控制流唯一地决定于在一给定时间下WM的状态（决定满足匹配条件的产生式规则集）及系统冲突消解策略（决定选择其中一个产生式规则执行），而WM只能由RHS动作操纵，因此外部数据源不能直接控制程序运行。这样如果求解复杂问题，采用分别只具有局部的、不完全的知识的不同产生式系统来完成，就不容易实现全局的协同求解。

三、实时产生式系统的实现方法

近年来，人们在开发实时产生式系统方面做了大量工作，并有许多成功的实践[7]。象采用OPS5实现、执行计算系统活动控制和提供计算机操作建议的实时专家系统YES/MVS[8]；实时控制应用混合专家系统HEXSCON[9]；用于实时过程控制系统PICON[10]；具有优先权实时产生式系统PRIOPS[11]；用于实时信号处理和解释的HOPES[12]，等等。概括起来，实现实时产生式系统的方法主要包括相关的两个侧面：

1．加快产生式系统执行速度

(1) 开发新的快速匹配算法。发展象Rete[13]一样的快速匹配算法是一种采用软件技术加快产生式系统匹配过程的有效方法。Rete算法效率较高是因为用空间换时间及多条规则共享相同模式匹配测试结果之故，但它在知识库改变很快的实时环境中还需要进一步优化，从根本上需要在此基础上修改结构和功能。

(2) 改进推理执行机制。传统产生式系统以顺序方式执行模式匹配、冲突消解和动作执行这三个推理步骤，这样的推理机制妨碍产生式系统的开发执行，因此影响其推理执行效率。实际上，产生式系统蕴含三种级别的并行性：一是产生式级（又可分为产生式间并行匹配及产生式间并行点火）；二是元素级（又可分为条件元素间并行匹配及动作元素间并行处理）；三是元素内部（指元素内部的项之间并行匹配）。

目前正在开发的并行产生式系统程序设计语言有两种方式：一是基于操作系统中进程方式的并发产生式系统，二是基于通信顺序进程的面向过程的产生式系统。进一步正在进行的并行产生式系统机的研究则着手于并行匹配算法、产生式系统的划分与分布式处理、并行点火三个方面，已经开发和正在研制的并行产生式系统机有：Columbia大学

的DADO[14]及NON-VON[15]、CMU的PSM[16]及Oflazer设计的机器[17]、Honewell计算机科学中心的PESA-1[18]、ITT的CAP[19]、庆应义熟大学的MANJI[20]等。

(3) 在数据驱动的产生式系统程序设计语言中借鉴和引入其它思想，特别是传统命令驱动的控制流思想具有潜在的前景，OPS83可算是一个有益的尝试[21]。

(4) 把产生式系统映射到神经元结构。人工神经网络特别适合于解决模式识别和分类中的大量问题，目前也越来越多地应用到产生式系统中以期建立有效并行处理机制，提高处理效率。

在各种神经网络类型中，三层环结构反馈网络 (three layers of ring-structured feedback network)[22]比较适于产生式系统的映入。映入后的结构包括三层：条件层、规则层、动作层，以及三个关联存储器：AM_1、AM_2、AM_3。

条件层包括代表当前WM状态的工作存储单元集的n个神经元，每个神经元的取值对应于相应的工作存储单元是否存在于WM；规则层代表一个规则集，有p个神经元，每个神经元代表规则的几个例示之一，所有可能的例示规则都表示于此层，一个神经元激活则表明相应规则被例示；动作层保存对WM如何及进行什么样操作的工作存储单元集。

关联存储器AM_1联系条件层和规则层，保存工作存储单元集与一个特定规则集间相关性的必要信息，给定一组工作存储单元则有一组规则被例示，这种激活信息即存于AM_1；AM_2联系规则层和动作层，保存一组动作集和一个特定规则的一组工作存储单元之间相关性的信息，一个规则被选择和点火，动作层工作存储单元集即被改变，这种激活信息存于AM_2；AM_3联系动作层和条件层，保存WM原有及下一阶段相关信息，为了保证当规则点火、工作存储单元集改变时，动作层神经元正确影响当前状态，AM_3要保存原始状态使规则层神经元能正确作用于新状态。

已经证明，在三层间使用三个关联存储器的方法显著减少了匹配时间。

通过上述几大类方法从软、硬件两方面着手，有助于改善产生式系统匹配、推理速度太慢的事实。

2．建立实时异步环境

这种方法旨在通过创新数据结构、执行策略及骨架结构使产生式系统能够实现异步数据的快速合成和实时反应，这样就形成了异步产生式系统 (APS) 的概念[23]。APS保留了传统产生式系统的基于产生式规则结构，因此仍具备其优点及可用之处，但对其数据结构和执行机制做了根本修改，从而能够提供对实时激励的优先生成、事件驱动反应。

APS增加了一组称为外部输入单元的变量集，分别对应于APS外部的每个动态实时数据源。外部输入不能由RHS动作操作，而只受相应外部数据源变化影响，并直接激活匹配模块，因此系统具备同时监测环境，接受意外情况并优先执行处理的能力。意外事件处理之后，系统通过使用可编程恢复知识来决定与原任务相关的动作过程，并回到例行状态。

APS推理机包括匹配、选择、执行三个并发异步模块，三个模块通过外部输入、WM、CS (冲突集)、被选规则四个全局数据集进行通信，每个模块独立于其它模块执行，可以分别在一个多机环境下的分立处理器上实现，关键之一在于有效实现一个模块与外部全局数据集间的联系。

匹配模块当WM (因为执行RHS动作) 或外部输入 (因为实时数据输入) 之一改变时即被激活，被满足产生式规则的新集进入冲突集CS，不再匹配其数据集元素的原有满足例示从CS中被删除。在所有激活产生式被处理并在CS中生成适当变化后，匹配模块停止。

选择模块只要CS中有变化即被激活。这种变化可以是CS中一个现存例示被删除，也可以是一个新例示被加入。该模块的功能是应用一定的冲突消解策略选择可执行的单个规则。选择模块通过返回被选规则做为输出而停止。

执行模块则因为可能发生执行竞争而明显有别于传统认知循环动作执行阶段。执行竞争发生是由于现存被选规则全部RHS动作被执行完之前，选择模块又选择了一个新的规则准备执行。

APS执行模块接受被选规则做为其输入并处理它的RHS动作，这些动作可以是不同类型的计算、改变WM、I/O操作和调用外部过程执行特定任务等等。改变WM和多数I/O操作时间开销不大，对实时性能影响不显著，用户定义的外部过程则不同，因此在接受实时优先数据时有必要抑制这种过程的执行。另外，APS执行模块只能在被选规则的全部RHS动作成功被执行之后才正常停止，而当发生执行竞争时，执行模块甚至会在完全处理完现存被选规则之前就处理新选规则，所以应考虑增加类似于中断保护的恢复功能。

四、协同产生式系统的实现方法

协同问题求解也被称为布式问题求解，它的提出是由于以下几方面原因[24]：

（1）协同具有不同但互补（可能有重叠）的专门知识的多个专家系统，为那些不能由单一专家系统求解的问题提供一个解答；

（2）重用性：每一个小的独立专家系统可以分别是不同分布专家系统的组成部分；

（3）并行执行带来高效性：相似任务可由一组专门处理器执行；

（4）分布处理是控制计算复杂性的一个基本策略，分布式人工智能是求解问题本身就有分布特性的最佳方法；

（5）计算的分布性能增强对系统某部分发生故障或输入数据衰减时的处理能力，控制的分布性则可以避免"瓶颈"。

（6）分布对开放系统来说是更"自然"的方法，随着系统增大及复杂性增加，分布式模型可以方便地改变比如向上扩张性；

（7）分布式人工智能系统能够解决对集中式系统而言规模太大的问题。

黑板模型是一种典型的协同问题求解模型，它是CMU开发声音理解系统HEARSAY-Ⅱ过程中提出的[26]。它开创了协同式问题求解的先例，使人们开始注意到协同问题求解在AI领域中的地位和重要性。

黑板模型相当于分割的工作存储器和与之相关的分离的多个产生式存储器构成的系统。该模型的主要特点是：黑板是一种全程的层次性数据库，不同的知识源访问黑板的特定层次，多个知识源通过对黑板的读写进行通信，通过控制和调度机构进行系统的控制操作。

黑板模型的主要问题是：某个知识源只能访问黑板的某几个层次，知识源之间的通信也通过黑板间接实现，因此黑板模型不能做为协同问题求解的通用模型，只适合于结果共享方式。

还有一些协同产生式系统问题求解的方式和方案，如基于进程概念的并发产生式系统[26]，基于Hoare的通信顺序进程的分布式产生式系统[27]，等等。

协同问题求解的方法突破了传统的问题求解框架，给问题求解注入了新的活力。然而也出现了这样一种局限性，即过于强调平等的知识源之间的协作性。因此，需要研究一种新的方法，可以将一般问题求解和协同问题求解统一起来。

通过仔细观察和认真思考，〔28〕提出了一种人类思维过程的"有限关联团块假说"。它说明人类的思维过程是由有限个具有联系的团块组成，可以抽象为一个广义结点有向图。因此可以构造一种模型，该模型只需给定广义结点的描述和广义结点间的联系，而

无需考虑是用于协同式问题求解还是一般性问题求解，从而将两者统一起来。成员系统(member system, MS)就是这样一种模型，它是一个四元组：$MS=(M_S, I_{MS}, R_{MS}, A_{MS})$。其中，$M_S$是成员的非空有限集合，成员是可以表示为广义结点有向图的一个广义结点并能模拟一个团块功能的程序模块，它相当于AI问题求解中的一个物理或概念上的实体，它与某个知识源或智能处理单元具有直接对应的关系，是并行处理的一个基本单位；I_{MS}是成员间传递的信息集合；R_{MS}是成员间关系的集合；A_{MS}是成员特性的集合。

成员系统的执行表现为成员对信息的并行归约关系。成员系统又可分为确定性成员系统和不确定性成员系统。我们已经证明，确定性成员系统的特例与项重写系统等价，不确定性成员系统的特例与产生式系统等价。这也从理论上说明成员系统模型是支持多种AI问题求解的比较理想的抽象模型。下面我们将具体讨论这样一种基于不确定性成员系统模型的协同产生式系统程序设计语言，称之为MS-1

1．MS-1概述

MS-1是在OPS5之上开发的，与OPS5兼容，是广义上的OPS5，也沿用了OPS5的习惯用语。其基本结构如图1所示，除了原有的PM(成员的产生式规则存储器)、WM(成员的局部工作存储器)，增加了异步消息接收存储器RCM(ReCeive Memory)、同步请求消息接受存储器RQM(ReQuest-accept Memory)、同步应答消息接受存储器RPM(Reply-accept Memory)，共同参与匹配，在成员特性MC的影响下进行冲突消解(CR)，根据成员的关系MR执行被选规则的右手部，依照成员间的通信协议CP进行成员间的消息传递。

MS-1中的每个被激活的成员按如下识别-动作循环周期方式工作：

①匹配：检查PM中每个产生式规则的

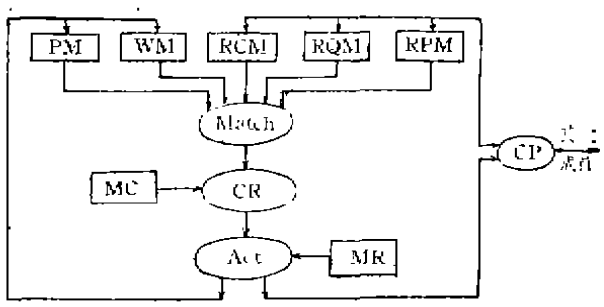

图1 MS-1基本结构

左部，找出被当前WM、RCM、RQM和RPM的内容所满足的产生式，并送入冲突集中。

②冲突消解：按照一定的策略从冲突集中选择一个产生式。若冲突集为空，则进入被动等待状态。

③动作：执行所选择的产生式右部动作。其中：若有同步请求动作，则进入主动等待状态；若有等待动作，则进入被动等待状态；若为结束动作，则进入结束状态；否则转①。

2．MS-1的实现

MS-1是在OPS5基础上采用IQ-LISP在PC机上实现的，其解释器由四大部分构成：

①编译部分：用于为成员系统中的每个成员构成一个运行环境，采用了对每个成员进行独立编译的思想，使用了高速匹配算法Rete。

②通信部分：用于成员间的通信。MS-1的通信方式分为同步和异步两种。同步方式是发送同步请求消息后，发送方进入主动等待状态，直到接收到对方的应答消息后才继续运行。异步方式则是发送异步消息后继续运行，而不等待对方的应答消息。

MS-1的通信原语分为动作原语和模式原语两类。动作原语是出现在产生式规则右部用于发送消息的动作，包括request、reply、send；模式原语则是出现在产生式右部用于接受消息的模式，包括req-accept、rep-accept、receive。它们是一一对应的。

③调度部分：用于协调不同成员占用处理机的顺序和时间、以及成员间的同步处理。

MS-1实现了两种调度算法：调度算法1是一种轮转法；调度算法2则是一种争用法，很适合成员系统在单机环境下的并发执行。

④解释部分：用于为每个成员的执行提供各种支持，包括匹配部分的解释执行、冲突消解及右手部动作的解释执行。

MS-1解释器的工作原理及过程如下：

①成员系统程序装入内存过程中进行编译，建立成员名索引表和每个成员的运行环境，其中包括对每个成员的产生式规则集建立左手部匹配网络Rete。

②在顶层上用start命令使该成员系统开始运行。

③按照一定的调度算法，选择一个可执行的成员。

④对该成员解释执行。

⑤若有通信动作，则通过通信原语与系统中其它成员通信，然后转向③。

⑥系统运行结束，可使用各种顶层命令。

通过运行典型例子（比如哲学家就餐问题）及开发一个实用研究程序（基于知识的空间站GaAs自动生产线协同故障诊断系统[28]），证明MS-1是一个比较成功的协同产生式系统程序设计语言。

五、结束语

迄今，在理论上比较完善并且在实践中获得广泛应用的人工智能程序设计语言首推以OPS5为代表的产生式系统。然而产生式系统一些固有的缺陷限制了它在复杂工程场合特别是实时处理、协同求解环境里的应用，使得一些人对产生式系统产生了怀疑，转而寻找其它问题求解的方法。可是为什么人工智能创始人Simon和Newell却将其做为人类认知心理学模拟人类思维的模型，并且有人已经通过心理学实验证实了产生式系统与人类思维方式相似呢？这说明产生式系统，或者确切地说，基于产生式系统的方法尚有许多潜力，这正是本文讨论目的之所在。述及的实时产生式系统方法、协同产生式系统方法代表了产生式系统的两个发展方向，也是挖掘产生式系统潜力，使其成为比较理想的人工智能理论模型及程序设计语言的诸多尝试之一。（参考文献共29篇略）

（接第页6）

参考文献

[1] Tony Hoare 著，周巢尘译，通信顺序进程，1989，北京大学出版社

[2] G.Agha,Supporting multiparadigm Programming on actor architectures. In Proceedings of Parallel Architectures and Languages Europe, Vol.Ⅰ,Espirit,Springer-Verlag, 1989

[3] G.Agha, Semantic considerations in the actor paradigm of concurrent computation. In Seminar on concurrency, Springer-Verlag, 1985

[4] G.Agha, The structure and semantics of actor languages. In Proceedings of the School/ Workshop on Foundations of Object-Oriented Languages. Springer-Verlag, 1992

[5] G.Agha, Concurrent object-oriented programming. CACM, 33(9), 1990

[6] W.Athas, Fine grain concurrent computations. Ph.D.dissertation, Computer Science Dept., California Institute of Technology, 1987. Also published as Tech. Rep. 5242:TR:87

[7] J. Backus, Can programming be liberated from the von Neumann style? a functional style and its algebra of programs. CACM, 21(8), 1978

[8] W.D.Clinger,Foundations of Actor Semantics. AI-TR-633, MIT Artificial Intelligence Laboratory, May 1981

[9] W.Dally,A VLSI Architecture for Concurrent Data Structures. Kluwer Academic Press, 1986

[10] P.Henderson,Functional programming, applications and its implementation. Addison Wesley, Reading, MA., 1983

[11] C.Hewitt and G.Agha, Guarded Horn clause languages, are they deductive and logical. In Proceedings of Fifth Generation Computer Systems Conference, ICOT, Tokyo, Dec. 1988