基于 Rete 算法的多 Agent 的实时协同推理

冯 梅 王宏安 金 宏 刘海龙 戴国忠

(中科院软件所人机交互技术与智能信息处理实验室 北京100080)

The Real-Time and Cooperative Inference of Multi-Agent Based on Rete Algorithm

FENG Mei WANG Hong-An JIN Hong LIU Hai-Long DAI Guo-Zhong

(Lay of Intelligence Engineering, Institute of Software, Chinese Academy of Sciences, Beijing 100080)

Abstract In this paper, we first introduce the high efficient pattern matching algorithm named Rete. Second, we analyze the problem appeared in using Rete algorithm to develop expert system. But it develops a new way to realize Agents' real-time and cooperative inference. Then we put forward to embed the Rete algorithm into Agents and realize cooperative inference by Agents' communication. On these bases, we discuss the accuracy and real-time feature of cooperative inference in detail. Finally, we describe two Agent-communication performatives and a developing instance of multi- agent system to monitor oil tanks.

Keywords Rete algorithm, Rete net, Multi-Agent system, Real-time and cooperative inference, KQML

1 引言

多 Agent 系统是由若干具有一个或多个目标的 Agent 按照一定的信息关系和控制关系以及问题求解能力的分布模式组成的系统,它主要研究一组在逻辑上或物理上分离的 Agent 之间行为的协调。目前,多 Agent 系统已应用于诸如空中交通控制、电子商务、通讯网络管理和作业调度等生产实际领域[1~3]。Agent 技术应用到实际领域时映射的对象一般有两大类:物理实体和抽象过程[4]。物理实体指现实系统中可以独立发挥作用的设备、工具、存储地点等资源或它们的组合,及代表任务的被加工零件等以物质形态存在的客观实体,其对应的 Agent 是针对这些实体行为的控制功能模块。抽象过程是指那些具有动态性质的信息变换,如算法、事务处理过程和功能实现等,其对应的 Agent 是对这些过程进行封装和规范化处理的智能模块。文[5]将遗传算法封装到 Agent 当中,多个 Agent 之间通过协作完成 CIM 中 Job-shop 问题的求解;文[6]证明:生产控制中的各种方法(算法)都可以应用在多 Agent 系统的结构中。本文提出将模式匹配算法——Rete 算法封装到 Agent 当中,通过多个 Agent 间的合作实现实时协同推理。

Rete 算法是一种高性能的推理机,不少专家系统开发工具(Jess,OPS5)[8,9]均采用 Rete 算法。但用这些工具开发的专家系统一般只包含一个 Rete 网络集。要增加专家系统的功能,知识库势必包含更广泛更全面的事实和规则,Rete 网络集相应地会变得复杂庞大,专家系统的推理性能就会降低,对于实时性要求较高的专家系统来说,该问题显得比较严重;反之要提高专家系统的性能,有效方法之一是将一个专家系统分解为若干子专家系统,一个大的 Rete 网络集划分成若干子 Rete 网络集,但由于专家系统之间的相对孤立,Rete 网络之间也就不能进行交互,这样就局限了专家系统的适用范围,减弱了它的功能。为解决上述矛盾,充分发挥 Rete 算法的优势,减轻 Rete 网络的负担,我们提出将 Rete 算法嵌入到 Agent 中,这样既可使具有特定功能的 Agent 利用 Rete 算法实现快速推理,又可利用 Agent 良好的社会合作性实现 Rete 网络协同推理。

在基于 Rete 算法的多 Agent 系统中,每个 Agent 包含一个知识库,采用 Rete 算法进行推理。遇到超出推理能力的任务时,它将把任务交给能胜任该任务的 Agent(目标 Agent)。目标 Agent 接收到任务后将在尽量短的时间内按请求 Agent 的要求完成协同推理。我们力图使多 Agent 系统的实时协同推理所产生的结果不仅在逻辑上是正确的,而且在时间上也是正确的。目前,我们采用 Jess 开发了一个应用于罐区实时监控的多 Agent 系统原型。该系统包括罐区监控 Agent、报警 Agent、操作 Agent 和调度 Agent,它们之间能相互通信,必要时进行协同推理。该系统具有较强的主动性、协同性、及时性和可靠性的特征。

2 Rete 算法

2.1 Rete 网络的基本结构

C. L. Forgy 于七十年代发明的 Rete 算法[7]是进行多次模式匹配的一种特别有效的算法。很多专家系统的开发工具(例如 Jess,OPS5)均采用 Rete 算法作为推理机制来提高事实与规则的匹配效率。该算法的基础是一种被称为 Rete 网络的数据结构,Rete 网络是一种被指定了输入和输出结点的数据流图,它通过所有的产生式的左部建立起来。Rete 网络包含了两种结点:alpha 结点和 beta 结点,这两种结点类型实质上都是一种过滤器——它们对所有输入的"标识"进行测试,并将不能通过测试的"标识"过滤掉。Alpha 结点一般在 Rete 网络的起点出现,并对独立的条件进行测试。最后的 alpha 结点将它们的输出送到 beta 结点,beta 结点将进行一些交叉测试。所有的 beta 结点都有两个入口,这样,每一个 beta 结点都可以对两个事实集合进行小规模的匹配。例如:我们有以下规则:

```
(defrule r1
    (x) (y) ⇒ (action1))
(defrule r2
    (x) (y) (z)⇒ (action2))
```

由以上两个规则生成 Rete 网络结构如图1所示。

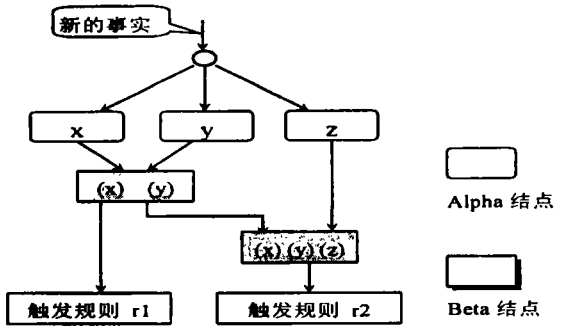

图1 Rete 网络的基本结构

Rete 网络有一个重要的特性——beta 结点可以保存本结点的当前工作状态。这样，当新的事实输入时，beta 结点可以根据以前的输入对它进行匹配。

2.2 Rete 算法对传统方法的改进

传统的知识规则的匹配方法一般是将规则链接成一个表，依次检查每一条规则的左项所包含的事实是否在知识库中，若存在，则执行规则的右项。其效率不高的原因是每次的循环检查都是从头重新开始。这种算法的复杂度是 $O(R * F^P)$，R 是规则数，P 是每条规则左项的平均元素数，F 是知识库中的事实数。在绝大部分的产生式系统中，知识库在每一个循环的变化都极少，这样绝大多数的检验是重复操作。

Rete 算法利用 Beta 结点将每一次循环中进行的匹配结果都保存下来，并在下一个循环中不再重复它们，每次只将新加入的知识与 Alpha 结点或 Beta 结点进行匹配检查。该算法的复杂度减小为 $O(R * F * P)$。

Rete 算法是在考虑到知识库相对稳定的情况下提出的，它适合于事实知识在每个周期间的变化不大的情况。如果每个循环之间的事实知识的变化显著，Rete 算法的效率就随之降低。同时，一个 Rete 网络还有它自身的负担，比如每一个 beta 结点都需要一定的存储空间，在存储空间中反复查询也要花费时间。结点越多，事实知识的变动越巨大，Rete 自身的负担也越大。

3 基于 Rete 算法的 Agent 结构

Agent 由基于 Rete 算法的 Rete 推理机、监视器、通讯构件、解释器、任务调度器和执行器等部件构成。其结构如图2所示。

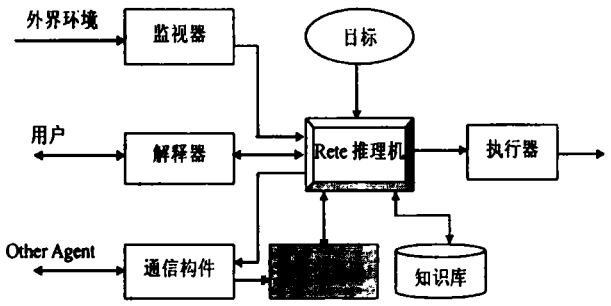

图2 基于 Rete 推理机的 Agent 基本结构

图中，监视器定时监测环境状态，发现异常情况，翻译后将其传给 Rete 推理机。通讯构件是 Agent 间的联系接口，在进行消息接收发送时使用消息队列机制，该构件是实现 Rete 网络协同推理的基础。解释器是 Agent 与用户进行对话的接口，通过它用户能够输入必要的数据，了解推理过程及推理结果等。Rete 推理机实现了 Rete 算法，它根据 Agent 的目标，对当前调度队列中的任务（监视器检测到的或来自其他 Agent 的请求），利用知识库进行推理并给出结论。任务调度器的主要功能是对于同一段时间内接收到其他 Agent 发送来的多个任务按照任务的重要性和优先级进行合理的调度，生成可行的任务调度队列，等待 Rete 推理机来调用。对一个任务进行调度时，我们根据任务的到达时间、就绪时间、运行时间、截止期及任务的重要性，采用启发式搜索算法[10]生成任务的执行队列。目标由一些约束组成，它能指导 Rete 推理机推理方向。执行器则根据推理机的推理结论执行具体的方法。

4 多 Agent 的实时协同推理

4.1 协同推理的原理

在一个大系统中，按功能划分的多个 Rete 网络并不是孤立存在的，它们之间也存在着联系。这种联系体现了 Rete 网络间的弱耦合（Rete 网络内部为强耦合），它的表现形式之一是在一个 Rete 网络中某一特定规则的执行会生成或改变另一个 Rete 网络的一些事实知识，从而导致另一个 Rete 网络中某些规则的执行。我们将能触发其他 Rete 网络的规则称为“临界规则”，从 Agent 的角度看，临界规则指明 Agent 在何种情况下与其他 Agent 发生联系。一般情况下，Agent 检测自身的 Rete 网络和知识库，进行规则匹配的推理工作。当临界规则被触发时，Agent 将从临界规则的右项所表达的信息中分析出目标 Agent 及其向目标 Agent 发送的信息，然后将该信息转换为一个任务按约定的通信格式通过通信构件发送给目标 Agent。目标 Agent 接收到源 Agent 发送来的任务后，判断任务的有效性，分析任务类型，对任务加以合理调度，然后启动自身的 Rete 推理机执行任务。依此类推，目标 Agent 必要时还可能与其他的 Agent 进行交互，从而实现多 Agent 的协同推理。在协同推理的过程中，我们采用一定的措施保证其推理的正确性和实时性。

4.2 协同推理的正确性

我们所设计的 Agent 具有一个知识库和一个 Rete 推理机，在规则定义正确的前提下，能否正确地进行逻辑推理取决于知识的正确性。知识的正确性不是简单的表达无误，而是其意义在特定的情景下是正确的有效的。导致知识无效的因素有多种，比如某个 Agent 中途取消合作，通信时延打乱相关知识的逻辑顺序等等。在 Agent 的交互过程中，相互交换的知识作为一个任务加以处理，因此无效知识的任务即为无效任务，本节将重点讨论如何判断由于通信时延而产生的无效任务并作出相应的处理。

对于同一个 Agent 在不同时间内发送的多个请求任务由于通信路径的不同，它们到达目标 Agent 的顺序可能发生变化。此时若不考虑任务顺序的改变，简单地按照“先来先服务”的原则对任务进行调度执行，就会产生推理的逻辑错误。为避免该类错误发生，必须在调度之前检查该任务的有效性，然后通过一定的调整措施使得无效任务发生时 Agent 仍能进行正确的推理。对任务有效性的判断，我们的办法是：采用时间戳机制验证源 Agent 针对同一个对象向目标 Agent 发出的多个具有一定的逻辑顺序的任务到达目标 Agent 时是否仍保持相应的逻辑顺序。时间戳包括源 Agent 发送任务的时间（send_time）和目标 Agent 接收到该任务的时间

(receive_time)。目标 Agent 具有任务的逻辑顺序表和每个任务的最近接收记录，当接收到一个新任务时，它参照逻辑关系表判断新任务与其相关联的任务的发送时间的顺序和接收顺序是否一致，若不一致说明该任务为无效任务，对于此类任务可根据其性质作不同的处理，比如不予调度。

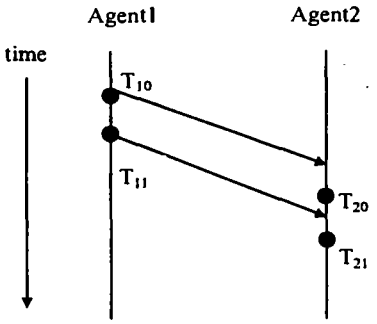

图3 Agent 间相关任务的一致顺序

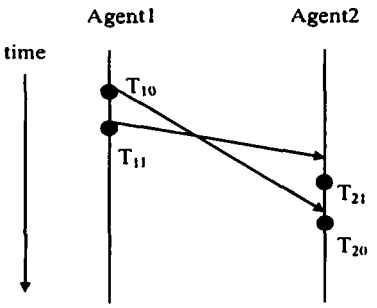

图4 Agent 间相关任务的不一致顺序

在图3中，Agent1发出任务的时间顺序是 T_{10}、T_{11}，Agent2接收到任务的时间顺序也是 T_{10}、T_{11}，并生成待调度任务 T_{20}、T_{21}。此时 Agent2利用时间戳机制检测到接收任务的顺序没发生变化，所以 T_{20}、T_{21}都是有效任务，应加以调度。在图4中，Agent1发出任务的时间顺序是 T_{10}、T_{11}，而 Agent2接收到任务的时间顺序却是 T_{11}、T_{10}，并生成待调度任务 T_{21}、T_{20}。这时 Agent2利用时间戳机制检测到接收任务的顺序发生改变，即在 T_{20}到来之前其后序任务 T_{21}已参与调度，这时 T_{20}变成无效任务，应不予调度。

4.3 协同推理的实时性

Agent 在同一个时间段内可能接收到来自不同 Agent 多个任务请求，Agent 如何在规定的时间内及时地响应所有的任务请求，在时间约束不能满足的情况下，如何对任务进行有效的组织安排，即实现协同推理的实时性是本节重点讨论的问题。为保证推理的实时性我们从两方面入手：1)采用高效的 Rete 推理机制(如前所述)，尽量减少推理花费的时间。2)对需要进行推理才能执行的任务进行合理的调度，尽可能满足各个任务的要求。

对一个任务进行调度时，我们需要考虑任务的到达时间、就绪时间、运行时间和截止期。到达时间是任务到达目标 Agent 时刻，该时间是可知的。就绪时间是指 Agent 接收到一个任务后，判断任务是否为有效任务的时间，该时间是可预测的。运行时间是指 Rete 推理机对任务进行推理执行的时间，在知识库变动不大的情况下，Rete 推理机的执行时间也是可预测的。截止期指明任务应该完成的最迟时间，它由源 Agent 指定。

在任务的到达时间和截止期已知，就绪时间和运行时间可预测的情况下，我们采用启发式搜索算法生成任务的执行队列。启发式搜索的基本做法是：搜索从搜索树的根结点，即一个空的局部调度开始，通过每次从待调度的任务集中选择一个任务来扩充当前调度而移动到搜索树下一层的某个结点上。这种搜索将一直持续，直到找到了任务集的可行调度。在这个搜索过程中，每次在对当前调度进行扩展前，都要进行可行性检查，以便确定当前调度是否为强可行。只有当这个局部调度为强可行的时候，调度才可以从这个结点进行扩充，同时将该任务从未被调度的任务集中删除。在进行可行性检查时，是针对当前未被调度的任务集进行的。此外，我们为每个任务定义了一个目标函数 H。这样，如果可行性检查表明当前调度是强可行的，则从未被调度的任务集中选择一个目标函数 H 值最优的任务来扩充当前调度。如果检查表明当前调度不是强可行的，调度便不再由此扩展，而是回溯到上一层的结点，在上一层的未被调度的任务集中重新选择一个目标函数 H 值次优的任务来扩充这个结点。

5 Agent 间的通信

实时协同推理的基础是 Agent 间能实现与 Rete 网络有关的通信。为此需要定义 Rete 网络间互操作信息的格式，它可用四元组来表示(sender，receiver，type，content)。假设有一条互操作信息为：sender：Tank_Agent；receiver：Alarm_Agent；type：update；content：(deffacts facts2 (Alarm (ID 6) (status full)))。它表示 Tank_Agent 要向 Alarm_Agent 的知识库中添加一条事实——(deffacts facts2 (Alarm (ID 6) (status full)))。向 Alarm_Agent 的知识库添加事实实际是向 Alarm_Agent 发出更新知识库的任务。对任务的描述采用三元组表示 Task(send_time，deadline，tast_value)，各项分别表示任务的发出时间、截止期及任务的重要性。有了 Rete 信息及任务的描述下一步的工作是将二者合起来映射为与 Rete 网络有关的一种 Agent 通信原语，并将其作为消息发送出去。目前应用较广泛的 Agent 通讯语言是 KQML 和 FIPA 提出的 ACL(Agent Communication Language)[11]。已定义的 KQML 通信原语有 require，accept，reject，inform，cancel，broadcast，ask_one 等等。在每个通信原语中包含若干参数，KQML 定义的保留原语参数有：content，force，in_reply_to，language，ontology，receiver，reply_with 和 sender。为实现 Agent 的实时协同推理，我们将 KQML 通信原语和其包含的原语参数进行有目的的扩展，添加 update 原语和 run_rete 原语，并为 update 和 run—rete 原语增添了一个原语参数 Task，它用来描述任务的发出时间、截止期及任务的重要性。update 和 run_rete 原语格式如下：

```
(update
   :sender Tank_Agent
   :receiver Alarm_Agent
   :language Jess
   :content (deffacts facts2 (Alarm (ID 6) (status full)))
   :Task (send_time, deadline, tast_value)
)
 (run_rete
    :sender Tank_Agent
    :receiver Alarm_Agent
    :language Jess
    :content (run)
    :Task (send_time, deadline, tast_value).
 )
```

当一个 Agent 需要与多个 Agent 进行交互时，它将产生多条消息。为此我们使用消息队列来对多条消息进行缓存，进

而实现异步通信。

6 开发实例

目前，我们采用 Jess 开发了一个应用于罐区实时监控的多 Agent 系统原型。该系统包括罐区监控 Agent、报警 Agent、操作 Agent 和调度 Agent，每个 Agent 包含与其职能相关的规则和知识，采用 Rete 算法进行推理。罐区监控 Agent 实时监控罐区的状态并动态显示给操作员，当它监测到非正常状态(罐空、罐将满、罐满)时，它通知报警 Agent。报警 Agent 根据油罐非正常状态类型，确定报警类型，给出报警信息，并将警报的解决方案通知调度 Agent。调度 Agent 根据报警 Agent 发来的信息，推理生成调度策略，通知操作 Agent。操作 Agent 具体执行对罐区对象的操作(如打开/关闭油罐的阀门、打开/关闭泵等)。这个过程就实现了四个 Agent 的实时协同推理，其过程如图5所示。

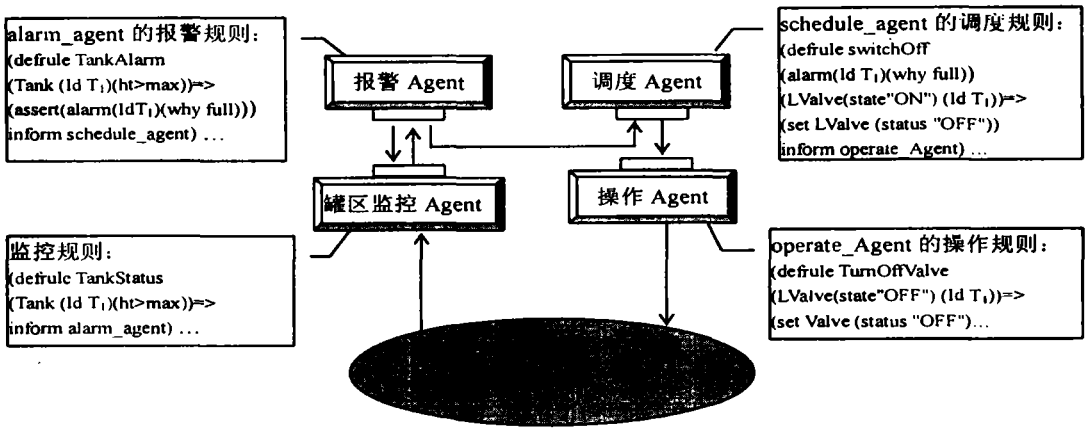

图5 多 Agent 协同推理

结束语 与传统的模式匹配算法相比，Rete 算法是一种高效的模式匹配算法，它非常适合于作为小型的知识库相对稳定的专家系统的推理机制。但是专家系统与专家系统之间相对孤立，存在交互性不强的弱点。为此，我们将 Rete 算法作为 Agent 的推理机，实现了 Agent 的实时推理，又通过 Agent 间的交互，实现了协同推理，从整体上实现了多 Agent 系统性能。同时为实时协同推理提供有效的底层通信服务，我们将 KQML 通信原语和其包含的原语参数进行有目的的扩展，添加 update 原语和 run_rete 原语，并为 update 和 run_rete 原语增添了一个反映协同任务性质的原语参数。

参 考 文 献

1 Bohnenberger T, Fischer K, Gerber C. Agents in Manufacturing: Online Scheduling and Production Plant Configuration. In: First Intl. Symposium on Agent Systems and Applications Third International Symposium on Mobile Agents, Palm Springs, California, Oct. 1999

2 DiPippo L, et al. A Real-time Multi-agent System Architecture for E-Commerce Applications. In: The Fifth Intl. Symposium on Autonomous Decentralized Systems With an Emphasis on Electronic Commerce (ISADS 2001), Dallas, Texas, U.S.A., March 2001

3 董军，潘云鹤. 路由选择的多 Agent 系统模型. 计算机学报，2000，23(1)

4 赵博，范玉顺. MAS 技术在生产调度研究中的应用. 控制与决策，2001，10(录用)

5 Morikawa K, et al. Evolution of CIM system with genetic algorithm [A]. IEEE Conference on Evolutionary Computation - Proceedings(2)[C]. IEEE, 1994. 746～749

6 Baker A D. Survey of factory control algorithms that can be implemented in a multi-agent heterarchy: Dispatching, scheduling, and pull [J]. Journal of Manufacturing Systems, 1998, 17(4):297～320

7 Forgy C L. Rete: A Fast Algorithm for the Many Pattern/ Many Object Pattern Match Problem. Artificial Intelligence, 1982, 19:17～37

8 http: // www. ilog. com // products/infrastructure/rules_wp6. htm, ILOG Rules white paper.

9 Ernest J, Friedman-Hill, Jess. The Java Expert System Shell, Version 6.0a8, SAND98-8206, Sandia National Laboratories, Livermore, CA (19 July 2001)

10 蔡自兴，徐光佑. 人工智能及其应用. 清华大学出版社

11 Finin T, et al. DRAFT Specification of the KQML Agent-Communication Language, The DARPA Knowledge Sharing Initiative External Interfaces Working Group, June 1993