第16卷 第2期
2006年2月

计算机技术与发展
COMPUTER TECHNOLOGY AND DEVELOPMENT

Vol. 16 No. 2
Feb. 2006

扩充KQML以实现合同网

陈业斌,周建钦,方木云
(安徽工业大学 计算机学院,安徽 马鞍山 243002)

摘 要:KQML目前已成了agent通信语言的事实标准,并得到广泛应用。文中在对KQML的语义分析的基础上指出它不利于Agent间任务级的高级复杂交互,不能实现合同网,为此对现有的KQML执行原语集合进行了扩充,以期支持合同网中存在的各种协商策略。文章给出了11条扩充原语的定义和语义及其在合同网上的应用。

关键词:智能代理;MAS;KQML;合同网;原语
中图分类号:TP311.52 **文献标识码**:A **文章编号**:1005-3751(2006)02-0053-03

Extension of KQML for Realization of Contract Net

CHEN Ye-bin, ZHOU Jian-qin, FANG Mu-yun
(School of Computer Science, Anhui University of Technology, Ma'anshan 243002, China)

Abstract: KQML has been accepted as a standard of ACL in fact and widely used at present. This paper points out that KQML is not well suited for agents task-level complex interaction and can not build contract net. For this, analyzes the set of KQML performers in the hope of supporting as much kinds of strategies of contract net as possible. The definitions, semantics and application to contract net of the eleven extended performative are provided.

Key words: agent; MAS; KQML; contract net; performative

0 引 言

Agent通信语言(ACL)作为充分发挥Agent潜力的关键所在,日益受到研究人员的重视。最著名的ACL是由ARPA下属的KSE外部接口小组制定的KQML语言(Knowledge Query and Manipulation Language)[1]。标准的KQML语法基于Lisp语言中的S-表达式,其语义模型以言语行为理论为基础,其目的是支持在分布式的、异构的、动态的、含大量自主节点(Agent)环境下知识和信息的共享、重用。

协商是一种以通信方式或通过推测其它Agent的意图和目标的方式解决冲突的过程。在开放的多Agent系统中,矛盾无处不在,但同时这种矛盾往往无法用逻辑演绎的方法解决,因此,协商便成为处理MAS(Multi Agent System)中各种矛盾的必不可少而又切实可行的办法。合同网是一种广为使用的协商协议[2]。综合考虑到KQML应用的广泛性和合同网的重要性,主要针对现有专家系统存在的知识不足、推理速度不快等缺陷,对现有的KQML执行原语集合进行了扩充,以期支持合同网中存在的各种协商策略。

收稿日期:2005-05-30
基金项目:国家自然科学基金资助项目(60473142)
作者简介:陈业斌(1971—),男,安徽全椒人,讲师,研究方向为人工智能软件、分布式数据库;周建钦,教授,研究方向为人工智能软件、网络安全;方木云,副教授,研究方向为计算机自动推理、计算机网络。

1 合同网模型

文中采用如下合同网,该合同网综合了经典的合同网模型[3]及刘大有等的合同网改进模型[4],能较好地模仿招标-投标的商业活动。使用合同网进行交互的基本协商机制为[5]:

(1)招标(Task_announcement):使用合同网进行协商的初始阶段,即通过广播声明任务。

(2)投标(Bid):收到招标书的结点,将标书加入队列并按兴趣程度及标书的处理排序,当该节点已完成正在执行的任务时,从标书队列中选择一个最感兴趣的任务投标。

(3)发标(Award):任务管理员结点在标书截止期限内,从收到的投标书选择一个最满意的结点来完成任务。若找不到,则可根据任务的重要程度决定是否重新招标。

(4)报告(Report):用于向任务管理员报告任务进程或结果。

(5)中止(Terminate):当任务管理员觉得任务没有执行下去的必要时,可以向任务承揽者发中止消息以中止任务的执行。

2 扩充KQML支持合同网

2.1 扩充的缘由

经典合同网的协商策略只考虑单任务、单回合、单中

标者的情况,我国研究人员刘大有等对合同网的协商策略进行了改进,解决了其中存在的一些问题。由于 KQML 产生的最初原因是为了共享和重用知识,所以目前其定义的绝大多数执行原语都是为了满足这一需要,其语义含义指出对话者对待知识的态度,认为是真还是假(Bel 或者¬ Bel)。但是,从语义层的角度上讲,Agent 在协商时交互的是对待某个活动(任务)的态度。很显然活动与知识不同,它没有真假性,也就是说,Agent 在协商时交互的态度不是用 Bel 或者¬ BBel 就可以说明的,它的态度应该使用想不想做(Want 或者¬ Want),愿意承诺还是不愿意承诺做(Int 或者¬ Int)来表达。因此使用现有的执行原语既无法体现出 Agent 在协商时对待活动的态度,同时又不能满足它们原有的语义含义。所以,有必要扩充现有的执行原语集合以适应具体的应用需要。

2.2 扩充 KQML 执行原语

根据本系统的实现需要,此处对 11 条 KQML 的执行原语集合进行了扩充,以支持各种形式的协商策略。

扩充的执行原语的的语义如下:

1)Announce(A,B,X):A 向 B 就任务 X 招标。

a.自然语言解释:Agent A 想知道 Agent B 对待活动 X 的期望。

b.形式化表示:Want(A, Know(A, S)),其中 S 指 Want(B,X)或者 Want(B,X)。

c.发送者和接收者的前提:Pre(A):Want(A, Know(A,S)) Pre(B):None

d.成功发送及接收该消息的后继状态:Post(A):Int(A,Know(A,S)) Post(B):Know(B, Want(A, Know(A, S)))

e.表明该消息已实现的条件:Completion:Know(A,S)

2)Bid(A,B,X):A 向 B 就任务 X 投标。

a.A 想做活动 X。

b.Want(A,X)

c.Pre(A):Know(A, Want(B, Know(B, S))) ∧ Want(A,X) Pre(B):Int(B,Know(B,S))

d.Post(A):Know(A, Know(B, Want(A, X))) Post(B):Know(B,Want(A,X))

e.Completion: Know(B,Want(A,X))

3)Refuse(A,B,X):A 拒绝就任务 X 向 B 投标。

a.A 不想做活动 X。

b.¬ Want(A,X)

c.Pre(A):¬ Want(A,X) ∧ Know(A, Want(B, Know(B,S))) Pre(B):Int(B,Know(B,S))

d.Post(A):Know(A,Know(B,¬ Want(A,X)))) Post(B):Know(B,¬ Want(A,X))

e.Completion:Know(B,¬ Want(A,X))

4)Aword(A,B,X):A 向 B 委托任务 X。

a.A 想知道 B 对待活动 X 的意图。

b.Want(A,Know(A,S')) S':Int(B,X),¬ Int(B,X)

c.Pre(A):Know(A,Want(B,X))) ∧ Want(A,Know(A,S')) Pre(B):Want(B,X)

d.Post(A):Int(A,Know(A,S')) Post(B):Know(B,Want(A,Know(A,S')))

e.Completion:Know(A,S')

5)Acknowladge(A,B,X'):A 拒绝向 B 委托任务 X。

a.A 有意图 X'。

b.Int(A,X') X'为终止协商的活动。

c.Pre(A): Know(A,Want(B,X))) ∧ Int(A,X') Pre(B):Want(B,X)

d.Post(B):Know(B,Int(A,X'))

e.Completion:Know(B,Int(A,X'))

6)Accept(A,B,X):A 接受 B 委托的任务 X。

a.A 有意图 X。

b.Int(A,X)

c.Pre(A):Know(A,Want(B,Know(B,S'))) ∧ Int(A,X)Pre(B):Int(B,Know(B,S'))

d.Post(A):Know(A,Know(B,Int(A,X)))Post(B):Know(B,Int(A,X))

e.Completion:Know(B,Int(A,X))

7)Reject(A,B,X):A 拒绝接受 B 委托的任务 X。

a.A 没有意图 X。

b.¬ Int(A,X)

c.Pre(A):Know(A,Want(B,Know(B,S'))) ∧¬ Int(A,X)Pre(B):Int(B,Know(B,S'))

d.Post(A):Know(A,Know(B,¬ Int(A,X)))Post(B):Know(B,¬ Int(A,X))

e.Completion: Know(B,¬ Int(A,X))

8)Report(A,B,X):A 向 B 汇报任务 X 进行的状态。

a.A 想要 B 知道状态 R。

b.Want(A,Know(B,R))

c.Pre(A):Want(A,Know(B,R)) ∧ Know(A,Know(B,Int(A,X)))

d.Post(A):Int(A,Know(B,R)) Post(B):Know(B,Want(A,Know(B,R)))

e.Completion:Know(B,R)

9)Continue(A,B,X):A 收到 B 的中间汇报,通知 B 继续。

a.A 想要 B 知道 A 接收到中间汇报。

b.Want(A,Know(B,Know(A,Interim)))

c.Pre(A):Want(A,Know(B,Know(A,Interim))) Pre(B):Int(B,Know(A,Interfim))

d.Post(A):Int(A,Know(B,Know(A,Interfim))) Post(B):Know(B, Know(A,Interim))

e.Completion:Know(A,Interim)

10)Thank(A,B,X):A 收到 B 的最终汇报,通知 B 任务结束。

a.A 想要 B 知道 A 接收到最终汇报。

b. Want(A, Know(B, Know(A, Final)))

c. Pre(A): Want(A, Know(B, Know(A, Final))) Pre(B): Int(B, Know(A, Final))

d. Post(A): Int(A, Know(B, Know(A, Final))) Post(B): Know(B, Know(A, Final))

e. Completion: Know(A, Final)

11) Terminate(A,B,X): A取消对B的任务委托X,即终止合同。

a. A要做活动X′。

b. Int(A,X′) X′为撤销合同。

c. Pre(A): (Know(A, Int(B, X))) ∧ Int(A, X′)) ∨ (Know(A,R))) Pre(B): (Int(B,X)) ∨ (Int(B, Know(A, R)))

d. Post(A): Know(A, Know(B, Int(A, X′))) Post(B): Know(B, Int(A, X′))

e. Completion: Know(B, Int(A, X′))

以上原语的语义实际上是给出了各条原语在使用时的逻辑先后关系及对Agent的状态的改变。如果一条原语的Completion条件不是该原语的Post条件的子集,则表明一个完整的会话还没完成,须进一步对话。

3 应用实例

现以分布式专家系统——枪械设计DES-FD(Distributed Expert System-Firearms Design)为例说明KQML扩充原语在合同网上的应用。DES-FD是一个以Agent为基础的基于Internet的分布式实验系统,以协助用户完成枪械或其部件的设计任务为目的,它由5种Agent组成: AppletAgent, UserAgent, ManagerAgent, ApplicationAgent和ServerAgent。各类Agent通过网络连接形成分布式系统,各司其职,利用KQML语言进行交流、合作,共同完成用户请求的设计任务。

基于合同网的协商方法是通过多回合的招标-投标-评标过程,最终确定任务的承揽者。限于篇幅,下面只给出在DES-FD中,用扩充原语完成的ManagerAgent与ApplicationAgent之间为进行协商所进行的对话。

●ManagerAgent发招标书:

(Annouce
:sender ManagerAgent1 :receiver ApplicationAgent1 :conversation MA-1
:reply-with MA-1-1-1 :language Prolog :ontology RiffleDesign
:content "tasksub(autometh, request(para(parafile), price(150), time(0,250), cf(0.6), rr(0.7)))")

●ApplicationAgent回投标书:

(Bid
:sender ApplicationAgent1 :receiver ManagerAgent1 :conversation MA-1
:reply-with MA-1-1-2 :in-reply-to MA-1-1-1 :language Prolog
:ontology RiffleDesign
:content "tasksub(autometh, request(para(parafile), price(200), time(0,200), cf(0.789), rr(0.76)))")

ManagerAgent当然可能给其他具有相应能力的Agent发同样的标书。在这种情况下ManagerAgent收到所有对同一个任务的投标书后,将会进行评标以选择合适的ApplicationAgent承揽任务。在这个过程中可能会进行多次的招投标过程,其中可能还包含讨价还价的过程,这与ManagerAgent所采取的具体的协商策略有关。当ManagerAgent为相应的任务选定一个合适的ApplicationAgent后,将会向其委托任务:

●ManagerAgent委托任务:

(Award
:sender ManagerAgent1 :receiver ApplicationAgent1 :conversation MA-1
:reply-with MA-1-1-3 :in-reply-to MA-1-1-2 :language Prolog
:ontology RiffleDesign
:content "tasksub(autometh, request(para(parafile), price(200), time(0,200), cf(0.789), rr(0.76)))")

●ApplicationAgent接受任务:

(Accept
:sender ApplicationAgent1 :receiver ManagerAgent1 :conversation MA-1
:reply-with MA-1-1-4 :in-reply-to MA-1-1-3 :language Prolog
:ontology RiffleDesign
:content "tasksub(autometh, request(para(parafile), price(200), time(0,200), cf(0.789), rr(0.76)))")

●ApplicationAgent在完成任务之后,返回求解结果:

(Report
:sender ApplicationAgent1 :receiver ManagerAgent1 :conversation MA-1
:reply-with MA-1-1-5 :in-reply-to MA-1-1-4 :language Prolog
:ontology RiffleDesign :content "result(automethresult)")

●如果是最终结果,则ManagerAgent回答:

(Thank
:sender ManagerAgent1 :receiver ApplicationAgent1 :conversation MA-1
:reply-with MA-1-1-6 :in-reply-to MA-1-1-5 :language Prolog
:ontology RiffleDesign :content "result(automethresult)")

●如果是中间汇报,则ManagerAgent回答:

(Continue
:sender ManagerAgent1 :receiver ApplicationAgent1 :conversation MA-1
:reply-with MA-1-1-4 :in-reply-to MA-1-1-3)

(下转第58页)

位于通信网络这一侧，通过 PARLAY Server，第三方的业务提供者能够利用和控制运营商的网络设备资源，Client 端位于业务提供者这一侧，通过 PARLAY Client，运营商的应用服务器可以回调(Call－back)大量第三方提供的丰富的业务程序。

PARLAY 组织的目标是定义一套分布式的、与具体实现技术无关的易于扩展的 API 集合，因此 PARLAY API 采用了抽象的面向对象的 UML(Unified Model Language)语言来描述 API 相关的语义、事件与参数，这些抽象的 API 可以通过 DCOM(Distributed Component Object Model，分布式组件对象模型)、CORBA(Common Object Request Broker Architecture，公共对象请求代理体系)或其他技术"翻译"成具体的实现[4]。

(2)JAIN。

Java 是近年来得到快速发展的一种跨平台计算技术。Java 平台已经在包括通信产品在内的许多系统中得到广泛应用，如在 PBX、呼叫中心上得到应用的 JTAPI(Java Telephony API)。在开发下一代开放式 API 时，Java 平台的面向对象及平台无关特征使得其成为一个非常不错的选择，以 Sun 微系统公司为主的七十多家产商联合发布的 JAIN API 就是这样一个基于 Java 平台的开放式 API 集合。

与 PARLAY 不同的是，JAIN 不仅定义了业务提供者与网络环境之间的 API，同时还定义了网络环境内部安全管理、呼叫控制、协议相关 API，因此 JAIN 是一个更加全面的 API 集合，同时由于它是基于具体开发语言(Java)的 API，可以直接在实际系统中得到应用。所以，JAIN 与 PARLAY 并不是两个完全不同的 API，只不过定义的范围不同，目标与定位也不尽相同。

PARLAY 与具体实现技术以及编程语言无关，更为抽象，层次更高一些；JAIN 则是一个范围更大的"PARLAY API"的 Java 翻译版本[5]。

4 结束语

开放式 API 是以下一代网络为目标，着眼于建立开放体系结构、融合不同网络及实现业务的快速、自由开发部署的新的技术。它的出现将把计算机工业中存在的业务开发及相关人才、技术方面的优势与通信网络基础设施优势结合起来，使得下一代网络的业务开发应用前景呈现令人激动的局面。从某种意义上也可以说，开放式 API 的出现代表了计算机工业界对于开放通信网络的一种迫切需求，或者说是计算机工业对于通信产业的一次"入侵"。而通信产业界应该敞开胸怀来拥抱这样的"入侵"，从而在下一代网络巨大的业务市场中形成通信业界和计算机业界双赢的局面。

参考文献：

[1] 石友康. 下一代网络的核心——软交换技术[J]. 电信科学，2002(1)：39－44.

[2] Mueller S M. APIs and Protocols for Convergent Network Services[M]. USA：McGraw Hill，2002.

[3] Falen D. To PARLAY or not to PARLAY，that is the Question[EB/OL]. 8th PARLAY Member Meeting，http：//www.PARLAY.org/，2000.

[4] PARLAY Group. PARLAY APIs version 2.1[EB/OL]. http：//www.PARLAY.org/specs/，1999.

[5] Lozinski. PARLAY：the Next Steps[EB/OL]. Closing Address，10th PARLAY Member Meeting，http：//www.PARLAY.org/，2002.

(上接第 55 页)

●ManagerAgent 毁约：

```
(Terminate
  :sender ManagerAgent1 :receiver ApplicationAgent1 :conversation MA－1
  :in－reply－to MA－1－1－5:language Prolog
  :ontology RiffleDesign:content"Cancel(autometh)")
```

由上例可以看出，扩充的 KQML 原语能满足建立高层次的任务级的协作，给予了协商双方充分的协商余地，具有较大的灵活性。系统采用面向对象的 Java 语言作为开发工具，可运行于不同平台。该系统框架作为一种通用的、基于 Agent 的分布式体系结构，可以推广应用于各种形式的专家系统以及其它分布式系统。

4 结束语

文中主要针对现有专家系统存在的知识不足、推理速度不快等缺陷，引入分布式人工智能领域的多 Agent 系统技术，采用跨平台的方法开发实验性多 Agent 系统。该系统对 Agent 通信、协商等方面进行了研究和尝试性实现。由于水平以及实验条件所限，系统尚有不尽人意的地方。例如缺乏有效的感知机制对系统动态变化的环境进行判断，采用过于简单的任务规划方法等等，这些都有待于今后的改进。

参考文献：

[1] Labroou Yannis，Finin Tim. A Proposal for a new KQML Specification[M]. Cambridge，MASS：AAAI Press，1997.

[2] Yanis Labrou，Tim Finin，Yun Peng. Agent Communication Language：The Current Landscape[Z]. IEEE Internet Computing，1999.

[3] Smith R G. The Contract Net Protocol：High－Level Communication and Control in a Distributed Problem Solver[J]. IEEE Transaction in Computers，1999(12)：1104－1113.

[4] 刘大有，杨 鲲，陈建中. Agent 研究现状与发展趋势[J]. 软件学报，2000，11(1)：67－69.

[5] 陈武华，王立春，李红兵，等. 扩充 KQML 以实现合同网[J]. 计算机工程与应用，2000，22(4)：106－109.