具体方法分开;以类的形式封装,便于同步代码的模块化和独立性;提供的绑定机制使得同步模式的重用具有更多的灵活性.因此,该方法满足了前面所说的分离问题、分解问题和独立问题三方面的要求,使得同步代码在更大的范围内达到重用的效果,比以前的方法具有更大的灵活性.关于该方法的具体讨论详见文[9]。

结语 解决并发面向对象语言中的继承异常是一个困难的问题.本文主要对现有的研究工作加以分析,并提出了一种解决此问题的新的思想和方法.进一步的工作在于将上述方法进一步的细化,并融入并发面向对象规约语言和并发面向对象程序设计语言之中.这部分内容将另文讨论.另一方面,目前关于并发面向对象中继承问题的讨论主要集中于被动对象类中同步代码的继承问题,而对于主动对象及其协同代码的继承问题的讨论未见涉及.值得开展进一步的讨论.

参 考 文 献

[1] 徐家福等.对象式程序设计语言.南京大学出版社,1992年

[2] Barbarab. Wyatt et al. Parallelism In Object-Oriented Languages: A Survey. IEEE Software, Nov. 1992

[3] Yasuhiko Yokote, et al, Concurrent Programming in Concurrent Smalltalk, In Object-Oriented Concurrent Programming, (ed) Akinori Yonezawa et al, MIT Press, 1987

[4] Yasuhiko Yokote, et al, POOL-T: A Parallel Object-Oriented Language, Same to [3]

[5] Dennis G. Kafura, et al, Inheritance in Actor based concurrent object-oriented languages, In Proc. of E-COOP'89, Cambridge University Press, 1989

[6] Staoshi Marsuoka, et al, Analysis of Inheritance Anomaly in Object-Oriented Concurrent Programming Languages, In Research Directions in Concurrent Object-Oriented Programming, (ed) Gul Agha, et al, MIT Press, 1993

[7] Gregory R. Andrews, et al, Concepts and Notations for Concurrent Programming, Computing Surveys, 15(1)1983

[8] Satoshi Marsuoka et. al, Highly Efficient and Encapsulated Re-sue of Sychronization Code in Concurrent Object-Oriented Languages, ACM SIGPLAN NOTICES, 28(10)1993

[9] 张鸣等.基于类层次结构的继承异常处理方法,第七届全国计算机青年工作者会议(上海 NCYCS'98)录用

(上接第39页)

同步/异步的互操作和阻塞/非阻塞的消息传递机制,只要在 KQML 消息中置上合适的参数即可。

KQML 可使用任何通讯协议作为自己的通讯机制(http,smtp,TCP/IP 等).由于 KQML 消息的内容是透明的,故其内容可用任一语言来书写.Facilitator 提供了在大型网络上发现知识的能力,能与其它 WWW 上发现知识的应用程序进行互操作。

关于 KQML 的安全性,应当做在传输协议层还是语言层仍没有定论.如果做在语言层,则应当引入新的行为原语和消息参数来对消息的内容或对整个消息加密[3]。

综上所述,KQML 是一种新型的 agent 通信语言,它能满足 agent 间通信的绝大部分需要.将来的发展是使其在工业界标准化并加上安全性能。

参 考 文 献

[1] Y. Labrou et al, A Proposal for a New KQML Specification. ARPA Knowledge Sharing Initiative, External Interfaces Working Group Working Paper, February 1997

[2] Y. Labrou et al, A Semantics approach for KQML-a general purpose communication language for software agents. Third Intl. Conf. on Information and Knowledge Management (CIKM'94), November 1994

[3] C. Thirunavukkarasu, et al, Secret Agents-A Security Architecture for the KQML Agent Communication Language. CIKM'95 Intelligent Information Agents Workshop, Baltimore, December 1995

[4] T. Finin, et al, KQML as an agent communication language. The Proc. of the 3rd Intl. Conf. on Information and Knowledge Management (CIKM'94), ACM Press, November 1994

[5] J. Mayfield, et al, Desiderata for agent communication languages. Proc. of the AAAI Symposium on Information Gathering from Heterogeneous, Distributed Environments, AAAI-95 Spring Symposium, Stanford University, Stanford, CA. March 1995

计算机科学1998Vol. 25№. 6

Agent 的通信语言——KQML

Using KQML in Agent Communication

陈冠岭 樊晓聪 徐殿祥 郑国梁

（南京大学计算机科学与技术系 南京210093）

摘 要 This paper introduces the communication language of agent——KQML. The structure of KQML, several models using KQML in agent communication and the logical framework of KQML-speaking agents are provided. Finally, a formal semantics of KQML and the evaluation of KQML are presented.

关键词 KQML, agent, facilitator, performative

KQML（Knowledge Query and Manipulation Language）是一种用于交换信息和知识的语言和协议，为表达消息和处理消息提供了标准的格式，用于支持 agent 之间的实时知识共享。KQML 可以用于应用程序和智能系统之间的交互，也可以用于两个或多个智能系统之间的知识共享来达到协同处理问题的目的。

KQML 包含了一系列可扩充的行为原语（performative）。行为原语定义了 agent 对知识和目标的各种操作，在其上可以建立 agent 互操作的高层模型。另外，KQML 通过 facilitator 提供了一种知识共享的基本结构，便于协调其他 agent 之间的互操作，构成功能复杂的多 agent 社会。

1. KQML 语言的规约

1.1 KQML 消息的语法

一条 KQML 消息也称为一条行为原语，此行为原语用 ASCII 串表示，行为原语的参数由关键字标识，与顺序无关。一条典型的 KQML 消息如

```
(ask-one
        :sender joe
        :content(PRICE IBM?price)
        :receiver stock-server
        :reply-with ibm-stock
        :language LPROLOG
        :ontology NYSE-TICKS)
```

1.2 保留的行为原语参数

KQML 使用了一些关键字作为保留的行为原语参数，它们的意义如表1所示。

参数：content 的值是一表达式，它必须符合参数：language 所指定的语言的语法，其中的常量必须在参数：ontology 指定的概念模式中有定义。

由于 agent 采用异步方式收发消息，故使用参数：in-reply-to 和：reply-with 来匹配发出的询问和收到的回答。即回答的：reply-with 的值应与询问的：in-reply-to 的值是一样的。

表1 保留的参数关键字及其意义

关键字	意义
:sender	行为原语的实际发送者
:receiver	行为原语的实际接收者
:from	用 forward 转发时，其：content 内行为原语的发送者
:to	用 forward 转发时，其：content 内行为原语的接收者
:in-reply-to	对前条消息的 response 内的关键字（其值应与前条消息的：reply-with 的值一致）
:reply-with	对本条消息的 response 内的关键字
:language	:content 的表示语言的名称
:ontology	:content 参数使用的实体集（如，术语定义集）的名称
:content	实际交换的信息，行为原语对其表达了一种态度（情感）

1.3 保留的行为原语

KQML 还保留了一些行为原语，规定了它们的意义和使用方法，这些保留的行为原语从功能上分为三类：

1）会话型　此类行为原语从语言学的角度考虑与言语行为理论尽可能地接近，常用于两个 agent 之间信息和知识交换的语境中。（见表2，表中 S 为 sender，R 为 receiver，下同）。

2）干预和监控型　此类行为原语被用于干预会话的正常进程。会话的正常进程如下：agent A 发送

表 2

ask-if	S 想知道 :content 是否在 R 的虚拟知识库（VKB）内
ask-all	S 需要其 :content 对 R 为真的 R 的所有实例
ask-one	S 需要其 :content 对 R 为真的 R 的一个实例
stream-all	ask-all 的多个回答
eos	对于多个回答（stream-all）的流结束标识
tell	语句处于 S 的虚拟知识库内
untell	语句不在 S 的虚拟知识库内
deny	语句的否定式处于 S 的虚拟知识库内
insert	S 要求 R 将 :content 加入其虚拟知识库内
uninsert	S 希望 R 反向操作前一 insert 动作
delete-one	S 希望 R 从其虚拟知识库内删除一条匹配语句
delete-all	S 希望 R 从其虚拟知识库内删除所有匹配语句
undelete	S 希望 R 反向操作前一 delete 动作
achieve	S 希望 R 在其物理环境内将 :content 置为真
unachieve	S 希望 R 反向操作前一 achieve 动作
advertise	S 告诉 R 它能够并愿意处理类似于 :content 内的消息
unadvertise	S 告诉 R 它取消前一 advertise 动作，以后不再处理类似于 :content 内的消息
subscribe	S 希望更新 R 对某个 performative 的回答

表 3

error	S 认为 R 的前一消息有错
:sorry	S 能够理解 R 的消息，但无法提供具体回答
standby	S 希望 R 声称它能对 :content 内的消息提供回答
ready	S 准备回答从 R 收到的前一消息
next	S 希望收到 R 对 S 发送的前一消息的下一个回答
rest	S 希望收到 R 对 S 发送的前一消息的所有余下的回答
discard	S 不再需要 R 对 S 发送的前一消息的所有余下的回答

表 4

register	S 通知 R 它的存在及其符号名称
unregister	S 希望 R 反向操作前一 register 动作
forward	S 希望 R 将消息转发给 :to Agent（可以就是 R）
broadcast	S 希望 R 将消息转发给它所知道的所有 Agent
transport-address	S 将其符号名称与一个新的传输地址相联系
broker-one	S 希望 R 找到一个对（performative）的回答（某些非 R 的 Agent 可以提供此回答）
broker-all	S 希望 R 找到所有对（performative）的回答（某些非 R 的 Agent 可以提供此回答）
recommend-one	S 希望知道一个可以对（performative）回答的 Agent
recommend-all	S 希望知道所有可以对（performative）回答的 Agent
recruit-one	S 希望 R 找到一个可以对（performative）回答的 Agent
recruit-all	S 希望 R 找到所有可以对（performative）回答的 Agent

一个 KQML 消息(开始一个会话)，agent B 不论自己能否回答都要响应 agent A。这类会话或者提前结束一个会话(error、sorry)或采用缺省的处理方式(如 standby，ready，next、rest 和 discard)。(见表3)。

3)facilitation 和网络型　从纯理论的角度来看此类行为原语不属言语行为理论。它们主要用于使 agent 找到能处理自己请求的其它 agent。虽然非 facilitator 的 agent 也可以处理此类行为原语，但这并不合适，因为 facilitation 行为原语依赖于 advertise 消息且只有 facilitator 具有广播 advertise 消息的能力，(见表4)。

2. KQML 在 agent 通信中的应用

2.1　KQML 支持的基本通信协议

KQML 具有多种内置的信息交换协议，最简单的一种是，client 端的 agent 发一查询到 server 端的 agent 并等待回答(如图1中的 A 和 B)。Server 的回答可能包含一个或多个答案。另一种情况是，server 的回答并不是完整的(如图1中 A 和 C)，而是一个指向答案的句柄，使 client 能一次取一个回答，常见的例子如一个 client 查询关系数据库。虽然这种交换要求 server 保持某种内部状态，但独立的事务与以前一样，包括 agent 间的同步通信。另外一种情况是 client 查询 server 并得到无限的异步回答(如图1中的 A 和 D)。client 并不知道回答的消息什么时候到达，它也许正忙着执行其它任务。

还有很多其它的通信协议。消息可能并不发给某个特定的主机，而采用广播的方式。同步或异步返回的消息将被检查并与相应的查询绑定。

2.2　Facilitator

Facilitator 是一种提供通信服务的 agent，它处理关于信息服务的知识和其它 agent 的请求，并提供以下服务：如维护服务名称的注册(registering)、转发消息到一个服务(forwarding)，根据内容来转发消息(content-based routing)，为消息提供者和 client 提供匹配(brokering)，以及提供调节和翻译服务。

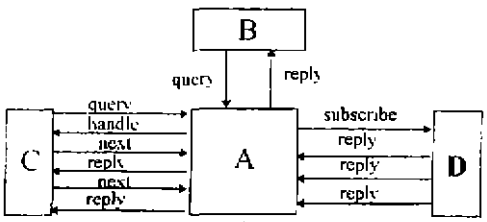


图1　KQML 支持的基本通信协议

例如，agent A 想知道句子 X 是否为真，agent B 的知识库可能有 X 且具有 facilitator F。如果 A 知道可以将关于 X 的查询发给 B，那它就可以使用简单的点到点协议并把查询直接发给 B，如图2a 所示。

如果 A 不知道 B 的存在，或它不知道谁的知识库内有 X，或它无法与那些 agent 建立连接，这种情况下有多种解决途径，图2b 显示 A 使用了 subscribe 行为原语来要求 F 检查 X 的真实性，如果 B 随后通知 F 它认为 X 为真，则 F 再通知 A。

图2c 显示了一种略为不同的情况。A 要求 F 找能处理 ask(X)行为原语的 agent，假设 B 已经通知 F 它愿意接收匹配 ask(X)的行为原语，则一旦 F 收到 A 的消息，F 就将查询发给 B，取得回答后再转发给 A。

在图2d 中，A 使用了不同的行为原语来通知 F 它对 X 的真实性感兴趣。Recruit 行为原语请求接收者找到一个愿意接收并处理其内嵌行为原语的 agent。那个 agent 的回答将直接传送给 A，虽然图2c 和图2d 的差别对于简单的查询是很小的，但当内嵌的行为原语是 subscribe(ask-all(X))时，就存在很大的效率差别。

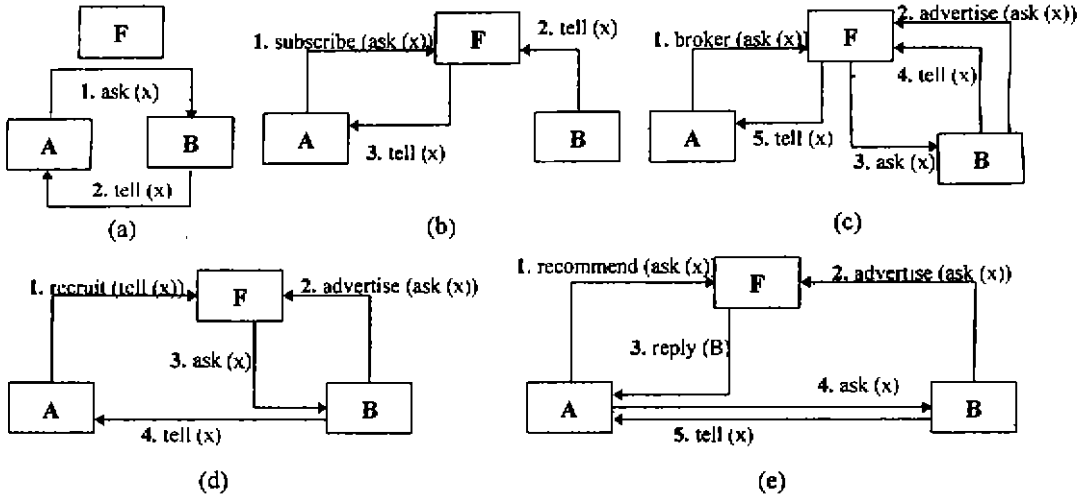
最后，图2e 中 A 要求 F“推荐”一个能处理行为原语 ask(X)的 agent。一旦 F 知道 B 愿意接收 ask(X)行为原语，它将 B 的名称传送给 A，然后 A 就可以与 B 对话并找到需要的答案。

在这些例子中，我们可以发现 facilitator 的一个主要作用是帮其它 agent 找到合适的 client 和 server。严格来说，agent 如何找到 facilitator 不是 KQML 的问题且有多种可能的解决途径。

目前基于 KQML 的应用使用了两种简单的技术。如在 PACT 项目中，所有的 agent 都使用一个中央的通用 facilitator，当 agent 启动时，此 facilitator 的位置作为一个参数被初始化。在 ARPI 应用中，找到并建立与本地 facilitator 的连接是 KQML API 的一个功能，当每一个 agent 启动时，其 KQML 路由器模块向本地 facilitator 声明使自己在本地数据库中注册。当应用退出时，路由器发另外一条 KQML 消息给 facilitator，将此应用从数据库中删除。一般地，facilitator 应运行于主机，并监听标准的 KQML 端口。

3. 采用 KQML 的 agent 的逻辑结构

使用 KQML 作为通信语言的 agent 从逻辑上可以分为四个部分(如图3)。



(a)如果 A 知道可以将关于 X 的查询发给 B，则使用点到点的协议。
(b)A 请求 F 监测其知识库的变化。
(c)用 broker 行为原语请求 F 找到能处理指定行为原语，取得并转发其回答。
(d)recruit 行为原语用于请求 F 找到合适的 agent 并将内嵌的行为原语发给它。所有回答直接传送给原来的 agent。
(e)recommend 行为原语用于请求 F 传回能处理特定行为原语的 agent 的名称。

图2

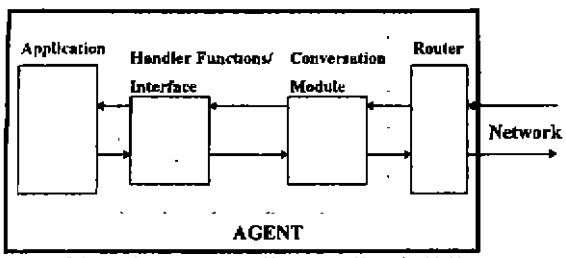


图3 应用 KQML 的 agent 的逻辑结构

应用程序模块在需要外部信息时，将询问发给那些有能力回答的 agent。关键在于如何知道哪些 agent 能够回答自己的询问。KQML 提供以下几种解决方案：1)如果应用程序员知道各个 agent 所具备的能力，则当程序模块需要外部信息时，自然知道应该向哪一个 agent 询问。2)如果应用程序处于一个开放系统，则它可以请求 facilitator 转发自己的询问。3)应用程序还可以通过与 facilitator 及其它 agent 的"讨论"，来获取必要的信息，再由自己决定发送给谁。

功能接口模块必须提供处理各种行为原语的功能。例如对于行为原语 ask-if 的处理，要先检查其 content 参数内的表达式对于本应用是否为真，然后通过 tell 或 deny 行为原语将结果传回提出询问的 agent。这就要求应用程序员知道各种行为原语的精确语义以及使用方法(参见第4节)。

对话模块处于路由模块和功能接口模块之间。每一条发出和收到的 KQML 消息都要经过此模块。例如：应用程序模块发出 ask-if 询问，收到的却是一条 delete 行为原语，则对话模块应当判断出这不是当前对话的有效回答。反之也是如此，收到其它 agent 的 ask-if 询问，对话模块应将结果通过 deny 或 tell 告诉对方，而不是发一条 delete 回去。

路由模块处理所有收到和发出的 KQML 消息(不必去考虑 KQML 消息的内容，只负责本 agent 与网络的通讯功能)，它既提供 Client 也能提供 Server 的功能并管理与其它 agent 的多条连接。

4. KQML 的语义

一个正式的语义有助于确切地定义该语言及其合适的用途。虽然 KQML 已部分实现并开始应用，但它缺少一个形式化的语义，只有一个基于对行为原语的自然语言基本描述。T. Finin[2]采用 speech act 理论简单地给出了 KQML 语义的形式化描述。

4.1 KQML 语义的框架

为了描述行为原语、前置条件、后置条件和完成条件，首先需要定义 agent 的认知状态(采用一阶谓词逻辑)。

1. Bel，bel(A，P)表示对 A 来说 P 为真，其中 P 是用 A 应用的语言写成的表达式。也可以说 P"存在"于 agent A 的知识库内(或虚拟知识库 VKB)。

2. Know，know(A，P)表示 A 认可 P 的知识状态。

3. Want，want(A，P)表示 agentA 期望用 P 描述的事件或状态发生。

4. Intent，intend(A，P)表示 A 愿意做 P。

简单地说，know，want 和 intend 分别表示知识的精神状态，需求和意愿。只有 bel 谓词中，P 是用 agent 的实现语言写的表达式。在其它三个谓词中，P 是结合了其它谓词的表达式，并代表了一个事件或状态。例如，"know(A，bel(B，foo(a，b)))"是正确的，但"know(A，foo(a，b))"是不正确的。

语义是通过 KQML 设计者提供的"对话策略"和应用程序程序员提供的"句柄函数"来实现的。对话策略表示什么样的行为原语可以跟在某个特定的行为原语后面，这样 agent 才能进行有意义的对话。对话策略是语义的一个部分且与前置条件、后置条件和完成条件相一致。例如，当 ask-if 发生后，它只能跟一个 tell 或 deny。句柄函数用于处理应用程序收到的消息且应与本语义一致。句柄函数是与应用程序无关，只与语言有关，这样所有使用同一种语言的应用程序都可共享同样的句柄函数。

4.2 KQML 行为原语的语义

KQML 行为原语的语义通常包含如下六个部分：

1. 行为原语直觉意义上的自然语言描述。

2. 用于描述意会行为(illocutionary act)的逻辑表达式，实际上，这是自然语言的形式化表示。

3. 用于表示 agent 发送，接收和处理行为原语的必须状态的前置条件。

4. 用于表示发出行为原语后或收到行为原语后但还未回答的状态的后置条件。

5. 用于表示发送者的最终状态(发送行为原语的意图得到满足)的完成条件。

6. 任何有利于对行为原语理解的自然语言评论。

如果接收者有非空的前置条件，表明对此行为原语的响应是对建立其前置条件的行为原语的某种响应，但对收到发起对话的行为原语的接收者来说，前置条件是不需要的。

在一次对话中，发送者的后置条件应该是消息接收者的前置条件的一个子集。当一条行为原语发出后结束对话时，完成条件是其后置条件的一个子集，此类行为原语只要成功发出并被对方处理就完成了。

4.3 KQML 行为原语形式化语义的举例

ask-if(A，B，X)

1. A 希望知道 B 关于 X 的真值状态。

2. want(A，know(A，Y))。其中 Y 可能是：bel(B，X)，bel(B，NOT(X))或 NOT(bel(B，X))。因此 Pre(A)表示为 want(A，know(A，bel(B，X)))或 want(A，know(A，bel(B，NOT(X))))或 want(A，know(A，NOT(bel(B，X))))。

3. Pre(A)：want(A，know(A，Y))。(也可包含条件 NOT(know(A，Y)))。

Pre(B)：无。

4. Post(A)：intend(A，know(A，Y))。

Post(B)：know((B，want(A，Know(A，Y)))。

5. Completion(A)：know(A，Y)。

6. NOT(bel(A，Y))与 bel(A，NOT(Y))是两个不同的概念(在某些特定的系统中是一样的)。

5. KQML 语言的综合评价与讨论

由于 KQML 消息是线性的字符流，类似于 Lisp 的语法，使消息易于处理和转换成其它格式。它的语法简单，允许通过增加新的参数来进行扩充。KQML 消息不关心所携带的消息内容是什么，故携带非 ASCII 的二进制流也可以。关于 KQML 的语义还在讨论，第4节只作了初步尝试。

现在对 KQML 的两个实现(Lockheed KQML API 和 UNISYS KQML API)都提供了一个与内容无关的消息路由器和 facilitator。Facilitator 是特殊的 KQML agent，它维护本域内的 agent 的信息和能力(现在的版本只有简单的注册服务)。应用程序必须提供对行为原语的处理功能，可根据 agent 的能力来实现行为原语的一个子集。

KQML 支持两个 agent 的点对点通信，广播和请求 facilitator 来转发自己的消息。KQML 还支持

(下转第18页)