面向多智能体的知识查询管理语言模型分析*

刘海龙[1,2], 吴铁军[1]

[1](浙江大学 工业控制技术国家重点实验室,浙江 杭州 310027);
[2](中国科学院 软件研究所 人机交互技术与智能信息处理实验室,北京 100080)
E-mail: hlliu.dai@163.com; tjwu@iipc.zju.edu.cn
http://www.zju.edu.cn

摘要: 在多智能体交互的研究中,KQML(knowledge query manage language)通信模型是最具代表性的.通过对KQML 通信模型的分析,为实现在知识水平上面向智能体编程所需的通信支持作了一定的阐述.首先,通过建立KQML 状态模型和 KQML 通信的转换模型,分别针对同步和异步通信模型分析了其实现的必要条件.其次,对KQML 模型在通信过程中将出现的死锁及资源匮乏问题,从 KQML 状态转换模型的角度分析了其产生的原因,并指出解决问题的根本途径.最后,指出 KQML 同步和异步通信的在多智能体交互中的优缺点和选用的一般原则.

关 键 词: 多智能体;KQML;交互
中图法分类号: TP18 文献标识码: A

在多智能体理论中,在实现知识共享和问题分布求解的研究中所采用的通信方式主要有两种:

(1) 公共存储[1]

通过建立公共存储区,每一个智能体均可对其进行访问,从而实现智能体之间的交互.黑板模型是公共存储的典型应用例子.

(2) 信文传递

信文传递是指在环境支持下各对象直接进行的信息交换.美国 ARPA 知识共享研究组定义的智能体通信语言 ACL[2,3]是其中的代表.

另外,Shoham[4]提出的面向智能体编程的 AOP 语言给出了智能体交互的元语,也属于信文传递通信模型.由于公共存储方式在异质的和物理上分散的多智能体间的实现较为困难[1],因而对信文传递方式的研究成为多智能体交互的热点.虽然,AOP 和 KQML 语言对通信原语句法和结构都有具体的描述,但是均未给出支持系统知识水平编程[5]的具体公式化的实现方法.Gapari[6]在为实现面向智能体编程所需的标号水平支持等方面作了有益的尝试,但是比较初步的.他给出了 KQML 同步和异步通信模型,并指出了同步通信在对面向智能体编程的支持中将产生资源匮乏和死锁问题,但并未分析产生的原因,也未给出解决方法.

本文通过建立智能体的状态模型和 KQML 转换模型,对上述问题进行了分析,并指出解决问题的根本途径.

1 智能体的状态模型和 KQML 转换模型

信息传递有同步和异步两种方式,同步方式是指信息传递过程中不存在缓冲区,接受方在信息到来之前

* 收稿日期: 2000-01-10; 修改日期: 2000-10-16
基金项目: 国家 863 高科技发展计划资助项目(9845-005)
作者简介: 刘海龙(1972－),男,山东单县人,博士,主要研究领域为人工智能,智能控制;吴铁军(1950－),男,江苏南京人,博士,教授,博士生导师,主要研究领域为模式识别与人工智能,大系统理论.

一直处于等待状态;异步方式则相反,由于具有缓冲区,所以可以利用空闲时间接受其他信息.KQML 语言的信息交互行为指令集较大,为方便论述,以 assert,tell,ask 这 3 条指令组成的子集为例,我们可以说明 KQML 语言在交互行为中的同步和异步通信的实现.

1.1 智能体的状态模型空间

若用 A 表示智能体集合,P 表示智能体工作进程的集合,STATUES::={IDLE,BUSY,WAIT(a)|$a\in A$}表示智能体的状态集合,则可定义智能体的状态模型空间.

定义 **1**. 智能体的状态模型空间定义为三元组空间Γ={<a,p,*statue*>|$a\in A$,$p\in P$,*statue*∈STATUES}. $\forall\ \gamma\in\Gamma$,被称为智能体的状态模型.

若用Γ_{KQML}表示 KQML 所描述的智能体状态模型空间,可对各类智能体状态模型描述如下:

$\Gamma_{\mathrm{KQML}}::=\Gamma_{\mathrm{IDLE}}\oplus\Gamma_{\mathrm{BUSY}}\oplus\Gamma_{\mathrm{WAIT}}$

$\Gamma_{\mathrm{IDLE}}::=\{<a>|a\in A\}$

$\Gamma_{\mathrm{BUSY}}::=\{<a,p>|a\in A,p\in P\}$

$\Gamma_{\mathrm{WAIT}}::=\{<a,p,\mathrm{wait}(a')|a\in A,a'\in A>\}$

其中"⊕"为集合构成操作符.

在对Γ_{IDLE}的描述中,因无工作进程项,故 p 被略去.同时若 p 不存在,则一定有 statue=IDLE,所以 statue 项也被略去.

1.2 KQML状态模型转换规则

定义 **2**. KQML 转换模型可表示为一个三元组(Γ_{KQML},L,{$\rightarrow^{l}$|$l\in L$}),其中 L 为指令集,$\rightarrow^{l}$:$\Gamma_{\mathrm{KQML}}\rightarrow\Gamma_{\mathrm{KQML}}$表示使 $\forall\ l\in L$ 都可被满足的一个公理和推理规则系统中的转换关系.

对于不同的公理和规则系统,就可以构造不同的 KQML 通信模型.表 1 和表 2 分别给出两组基于点对点通信的 KQML 模型系统的规则集,对应于表 1 的 L={assert(a),tell(a,a'),ask(a,a'),√},$\Gamma\subseteq\Gamma_{\mathrm{KQML}}=\Gamma_{\mathrm{IDLE}}\oplus\Gamma_{\mathrm{BUSY}}\oplus\Gamma_{\mathrm{WAIT}}$,而 dispatch 函数将接受到的信息映射到处理该信息的程序进程.

Table 1 Transition rules of synchronous system

表 **1** 同步系统转换规则集

(1) $<a,\mathrm{assert}(a').P>\oplus\Gamma\oplus<a'>\rightarrow^{\mathrm{assert}(a',a)}<a,P>\oplus\Gamma\oplus<a',P'>$

where P'=dispatch(assert(a'))

(2) $<a,\mathrm{ask}(a').P>\oplus\Gamma\oplus<a'>\rightarrow^{\mathrm{ask}(a',a)}<a,P,\mathrm{wait}(a')>\oplus\Gamma\oplus<a',P'>$

where P'=dispatch(ask(a',a))

(3) $<a,\mathrm{tell}(a').P>\oplus\Gamma\oplus<a',P',\mathrm{wait}(a)>\rightarrow^{\mathrm{tell}(a',a)}<a,P>\oplus\Gamma\oplus<a',P''>$

where P'=dispatch(tell(a'),P')

(4) $<a,\surd>\oplus\Gamma\rightarrow^{\surd}<a>\oplus\Gamma$

分析上述规则(1)和(2)可知,assert 和 ask 信息可被接受方立即获得,因为接受方处于 IDLE 状态.因此,接受方无须设置缓冲区.由规则(2)和(3)可知,一方发出 ask 信息后一直处于等待状态,这样,tell 信息也无须经过缓冲区就可被直接获取.这种交互过程由于不存在信息的缓冲存取进程,而且信息的发送和接受均是整块(blocking)地一次性完成,所以有下面的结论.

结论 **1**. 满足表 1 转换规则的 KQML 状态模型系统交互过程具有同步特性.

在上述定义的 KQML 状态转换模型下,KQML 程序可用 BNF 范式定义为

Program::=CommAction.Program|√|C

CommAction::=assert(a)|tell(a)|ask(a)

C=$^{\mathrm{def}}$Program

对应于表 2 的通信模型中,L={assert(a),tell(a,a'),ask(a,a'),get(a),√}.

另外,由于该规则集中的 ask 指令是无等待的数据操作,所以不包含Γ_{WAIT}状态元组,因而有:

$$\Gamma\subseteq\Gamma_{KQML}=\Gamma_{IDLE}\oplus\Gamma_{BUSY}.$$

$m(\bullet,\bullet)$表示系统中缓冲区的状态模型;“+”表示智能体状态模型与缓冲区的状态模型的组合.

Table 2 Transition rules of asynchronous system

表 2 异步系统转换规则集

(1) $\langle a,\text{assert}(a').P\rangle\oplus TS\rightarrow^{\text{assert}(a',a)}\langle a,P\rangle\oplus TS+\{m(a',\text{assert}(a'))\}$

(2) $\langle a,\text{ask}(a').P\rangle\oplus TS\rightarrow^{\text{ask}(a',a)}\langle a,P\rangle\oplus TS+\{m(a',\text{ask}(a',a))\}$

(3) $\langle a,\text{tell}(a').P\rangle\oplus TS\rightarrow^{\text{tell}(a',a)}\langle a,P\rangle\oplus TS+\{m(a',\text{tell}(a'))\}$

(4) $\langle a,\surd\rangle\oplus TS\rightarrow^{\surd}\langle a\rangle\oplus TS$

(5) $\langle a\rangle\oplus TS+\{m(a,msg)\}\rightarrow^{\text{get}(a)}\langle a,p'\rangle\oplus TS$

where $p'=\text{dispatch}(msg)$

分析规则(1)~(3)可知,智能体 a 在处理到关于 assert,ask,tell 指令的进程时可立即执行,而无须判别指令的作用对象所处的状态.规则(5)给出了智能体 a 接收系统中的信息流的规则,即 a 处于 IDLE 状态,且缓冲器中有指向智能体 a 的信息流 $m(a,msg)$.此规则集所支持的智能体系统交互过程,由于对指令的作用对象无显式的状态判断,所以为确保交互的正常操作,各智能体具有信息缓冲区是隐含的必要条件.于是有下面的结论 2.

结论 **2**. 满足表 2 转换规则的 KQML 状态模型系统交互过程具有异步特性.

在上述定义的 KQML 状态转换模型下,KQML 程序可用 BNF 范式定义为

Program::=CommAction.Program|√|C

CommAction::=assert(a)|tell(a)|ask(a)|get(a)

C=$^{\text{def}}$Program

1.3 状态转换的迹

定义 **3**. 设多智能体系统 A 初始状态均处于 IDLE,即$\gamma_0\in\Gamma_{IDLE}$,由指令χ^0起始的后续转换过程为$\gamma_0\rightarrow^{\chi^0}\gamma_1\rightarrow^{\chi^1}\gamma_2\ldots\rightarrow^{\chi^n}\gamma_{n+1}$,则状态转换的迹为转换指令序列$\{\chi^0,\chi^1,\ldots,\chi^n\ldots\}$,记为 $T(\gamma_0)_{\chi^0}$.

另外,我们用 $t_i\in T(\gamma_0)_{\chi^0}$ 表示迹 $T(\gamma_0)_{\chi^0}$ 的第 i 项;$t\downarrow a$ 表示 t 之后仅有智能体 a 参与的转换指令序列.$t\downarrow a$ 可能有两种情况:(a) 一个通信行为的无限序列;(b) 一个由内部结束程序为结尾的有限个通信行为序列.

状态转换的迹用来描述每一个参与交互的智能体在整个过程中的交互行为.它可用来分析无限序列的产生原因.

2 KQML 模型特性分析

在分析 KQML 模型特性之前,先给出几个与 KQML 模型有关的定义.

2.1 指令的使能空间与可激活性

定义 **4**. 设$\xi\in\Gamma_{KQML}$为 KQML 状态模型空间中的一个子状态模型空间,$\chi\in L$ 为指令集 L 中的一条指令,当指令χ对应的规则对于子状态模型空间ξ是使能的,即$\exists\xi'\in\Gamma_{KQML}$,使得$\xi\rightarrow^{\chi}\xi'$成立,则称在子状态模型空间$\xi$中,指令$\chi$作用下的转换是使能的,记为$\xi\vdash\chi$,或称$\xi$是$\chi$的使能子状态模型空间,简称使能空间.

另外,所有χ的使能空间组成的集合简称为使能空间集,记为 $\rho(\chi)$.同时,$\rho(\chi)$中所有使能状态模型经χ的映射得到的集合称为值空间,记为 $R(\chi)$.

在一个 KQML 的使能空间ξ中,使能的指令χ作用于ξ上,将使ξ转换为状态模型空间ξ',这时,指令χ可看做是对状态模型空间的函数映射关系,即$\chi:\Gamma_{KQM}\rightarrow\Gamma_{KQML}$.函数$\chi$的定义域为 $\rho(\chi)$,值域为 $R(\chi)$,且满足$\rho(\chi)\subseteq\Gamma_{KQML}$,$R(\chi)\subseteq\Gamma_{KQML}$.所以也可将指令$\chi$称为函数$\chi$.

定义 **5**. 设ξ为函数χ的使能空间,即$\xi\in\rho(\chi)$,若 a 是使能空间ξ中可执行指令χ的智能体,即 a 可触发指令χ,则称 a 在ξ空间中对χ是可激活的.

使能性描述的是函数χ的可作用空间,可激活性描述的是使能空间中智能体与指令的关系.下面可举例加以说明.

例 1:设有 3 个智能体 $a,b,c\in A$,它们当前的状态模型分别为 γ_a=<a,assert(b).pa>,γ_b=<b>,γ_c=<c,pc>,并令 $\xi=\{\gamma_a,\gamma_b,\gamma_c\}$.若采用同步 KQML 转换规则,在这样一个子状态模型空间中,指令χ=assert(b)所对应的规则前件是满足的,可在Γ_{KQML}中找到一个子状态模型空间$\xi'=\{<a,pa'>,\gamma_b=<b,pb'>,<c,pc>\}$,使得$\xi\rightarrow^{\mathrm{assert}(b,a)}\xi'$成立,那么指令$\chi$对应的规则 assert($a'$,$a$)在$\xi$中是使能的,$\xi'$就是$\chi$的使能空间.在$\xi'$中,若智能体 a 可以观测到 b 的状态是处于 IDLE 的,那么 a 便可触发指令χ,这时,称 a 在ξ'空间中对χ是可激活的.

2.2 同步KQML通信特性

在上例中,若智能体 a 无法观测到 b 的状态,将不能触发指令χ,这时 a 对χ是不可激活的,虽然它处于χ的使能空间.所以,在智能体通信中,某个智能体 a 发出通信指令χ必须具备以下两个条件:

- a 所在的状态空间是χ的使能空间;
- a 在χ的使能空间中对χ是可激活的.

在同步 KQML 模型通信方式中,由于智能体对指令的可激活性是建立在对指令的作用对象状态的观测之上,所以状态观测就成为同步 KQML 通信方式实现的必要条件.实质上,智能体对可激活性的判断是对所在的状态空间是否为使能空间的判断.所以就有了下面的结论.

结论 **3**. 设 A 为智能体集合,Γ_A 为 A 的状态模型空间,L 为通信指令集合,那么,同步 KQML 模型通信实现的必要条件是对 $\forall\ \chi\in L,\forall\ a\in A$

$$\Gamma_A\in\rho(\chi)\Leftrightarrow\kappa_a\Gamma_A\in\rho(\chi),$$

其中"κ_a"为模态逻辑的知道算子,$\kappa_a\Gamma_A\in\rho(\chi)$表示智能体 a 知道命题$\Gamma_A\in\rho(\chi)$为真.

在结论 3 中,$\Gamma_A\in\rho(\chi)\Rightarrow\kappa_a\Gamma_A\in\rho(\chi)$说明,任意一个智能体 a 对所处的状态模型空间为χ的使能空间这一事实是可获知的,从而保证了 a 对χ的可激活性;$\kappa_a\Gamma_A\in\rho(\chi)\Rightarrow\Gamma_A\in\rho(\chi)$说明,任意一个智能体 a 对所处的状态模型空间的认识都为真,从而保证了 a 对χ的激活是有效的.

2.3 异步KQML的通信特性

在异步 KQML 模型通信方式中,设有 $\forall\ \chi\in\{\mathrm{assert,tell,ask}\}$,$\chi$的使能空间 $\rho(\chi)$可表示为

$$\rho(\chi)=\{\xi|\xi\vdash\chi\}.$$

由表 2 可知,对 $\forall\ a\in A$,设 a 所处的状态模型空间为Γ_{KQML},则$\Gamma_{\mathrm{KQML}}\in\rho(\chi)$成立的充分必要条件为

$$\exists\ a'\in A\ \text{且}\ a\neq a',\text{使得}<a,\chi(a').P>\in\Gamma_{\mathrm{KQML}}.$$

所以,对 $\forall\ \chi\in\{\mathrm{assert,tell,ask}\}$,智能体 a 对χ的可激活性只与其自身状态有关,即下一段进程是否为$\chi(a')$,且 $a\neq a'$. 在此限定 $a\neq a'$,是为了避免智能体 a 欲与自身建立通信链路这种无意义的指令被激活.

与一般的异步通信机制一致,由于 ask 和 tell 这一对交互指令的通信链路不是实时建立的,这两条指令必须显式地给出识别指令次序的特征符.

异步 KQML 通信模式中包含显式的 get 指令,用于读取缓冲区中的信息流,这也正是异步通信与同步通信的关键区别.由表 2 可知,对 $\forall\ a\in A$,a 所处的状态模型空间为Γ_{KQML},则$\Gamma_{\mathrm{KQML}}\in\rho(\mathrm{get}(a))$成立的充分必要条件为

$$<a>+\{m(a,msg)\}\in\Gamma_{\mathrm{KQML}}.$$

所以,智能体 a 对指令 get(a)的可激活性只与 a 以及缓冲区的状态有关,即 a 是否空闲以及缓冲区中是否有发向 a 的信息流.

表 2 中没有限定缓冲区类型,可以是智能体独立占有的缓冲区,也可以是公用的缓冲区,但后一种类型必须标明信息流的流向.

3 资源匮乏和死锁问题

Gapari[5]指出,同步 KQML 通信存在着资源匮乏(starvation)和死锁(deadlock)的问题.

3.1 资源匮乏

资源匮乏问题一般是指,由于其他并发的激活过程持久地占有所需的资源,使某一个过程在可预测时间内

不能被激活.例如,有两个智能体 b 和 d,它们试图向同一个智能体 a 发出 assert 和 ask 指令,如下:

设:B=assert(a).B

D=ask(a).D

Dispatch(ask(a,d))=tell(d).√

Dispatch(assert(a,b))=√

另设有状态空间模型{<d,D>,<b,B>,<a>},这时,assert(a,b)和 ask(a,d)规则均是使能的:

$\{<d,D>,<b,B>,<a>\}\rightarrow^{\text{assert}(a,b)}\{<d,D>,<b,B>,<a,\surd>\}$

$\{<d,\text{D}>,<b,B>,<a>\}\rightarrow^{\text{ask}(a,d)}\{<d,D,\text{wait}(a)>,<b,B>,<a,\text{tell}(d).\surd>\}$...

若 ask(a,d)被激活,则会产生一个无限嵌套的进程,其状态转换的迹 $t\downarrow d$ 为{ask(a,d),ask(a,d),…}的一个无限指令序列,assert(a,b)将无法被激活,于是产生了资源匮乏问题.

资源匮乏问题产生的根本在于某个进程对相关资源的独占(如对 CPU,通信端口或链路等的独占),往往是由于进程的无限循环或无限次的嵌套而无终止点所造成的.所以说,资源匮乏问题与所采用的通信模式(如同步或异步)无关.进程对相关资源的独占应该在面向智能体的问题编程求解中有所避免,而在支撑面向智能体编程的标号水平通信层是无法预知和限制无终止进程的发生的.

3.2 死锁问题

死锁问题实质上是当前欲建立通信链路指令 χ 的智能体 a 所在的状态模型空间 Γ 不是使能空间,即 $\Gamma\notin\rho(\chi)$,而且在有限时间内也将不存在使能空间.它的发生一般有两种情况:一是发生在两个智能体同时以通信发起主体的身份希望与对方建立通信链路而又缺少退让造成的通信链路建立失败;二是发生在某一智能体陷入无限循环的进程中,欲与之建立通信链路的智能体因等待对方的 IDLE 状态而造成的无回应等待.

前一种情况由于缺少多智能体交互时的总体调度和相应的避退协调,使智能体在同时相互请求建立通信链路时处于相互等待状态.这是在同步通信中,ask 等请求指令无缓冲地按块传输所存在的普遍性问题[6].问题的原因是缺少调度和协调,那么,在智能体交互的标号水平通信中建立有效的调度和协调退让机制是保障同步通信顺畅的重要环节.

由结论 2 可知,KQML 模型通信实现的必要条件是智能体具有判断使能空间的能力.对使能空间的判断关键在于对智能体自身状态和相关智能体的状态的获知.例如,assert(b,a)规则的激活条件是 a 和 b 的状态模型分别为<a,assert(b).P>和<b>,即 a 的下一条指令为 assert(b),b 的状态为 IDLE.智能体通信层既然需要获知相关智能体的状态,也就可以在判别使能空间是否成立的同时作出是否处于死锁状态的结论.例如,智能体 a 和 b 的状态模型分别为<a,assert(b).Pa>和<b,ask(a).Pb>,当 a 和 b 分别准备同时执行 assert(b,a)和 ask(a,b)时,若已获知了对方的状态,就可知 a 和 b 正处于死锁状态,需相互间地协调和退让.关于智能体间的协调和退让也是多智能体领域中的研究热点,本文在此不作论述.

至于死锁问题的第 2 种情况则与资源匮乏问题类似,也是由于某一智能体执行进程的无限循环或无限嵌套所致.这种情况同样无法在通信层解决,而应该在面向智能体编程时设法消除.一般是通过提供静态强行限制的机制来防止死锁的发生.如禁止向其他智能体发送可能会引起无限循环计算的询问;禁止相互间的递归请求等[6].

在异步通信中,由于通信链路的建立无须等待对方的某个特定的使能状态,所以不会引起死锁问题的出现,这可以说明如下:

设某个时刻系统的状态模型空间为 Γ,智能体 a 的状态模型为<a,χ.P>,且 $\chi\in${ask,assert,tell},由表 2 可知:$\Gamma\in\rho(\chi)$成立,则指令 χ 可被智能体 a 激活,所以表 2 给出的异步通信模型不会产生死锁问题.

4 结 论

结论 2 指出,KQML 同步通信的实现必须有能够向参与协作的智能体提供它们所处的状态的机制.这就使得 KQML 同步通信的实现较之异步通信更为困难.

文献[7]中指出,异步通信的信息传递可以保证对更高的知识水平编程的支持.但是,异步通信在智能体问题的分布求解中,由于不能及时地将其他智能体最新的求解结果用于自己下一步的求解过程,因而降低了分布求解的功效.而对于同步通信,从实现机理上分析可知,它并不存在这个问题.

两种通信模式各有优势,至于采用哪一种则需根据实际对象来选取.一般说来,对于知识共享问题,由于对知识的及时性要求不是很高,为方便实现,可采用异步通信方式;而对于问题分布求解,正如上面所述,可考虑采用同步通信模式.

本文对 KQML 同步和异步通信的实现条件及其特性进行了分析,这对于建立 KQML 通信及实现过程中应予以考虑的问题具有一定的指导性意义.

References:

[1] Stefan, K. STRICT: a blackboard based tool supporting the design of distributed PPC application. Expert Systems with Applications, 1994,7(1):10~20.

[2] Genesereth, M.R., Ketchpel, S.P. Software agents. Communications of the ACM, 1994,37(7):48~54.

[3] Gruber, T.R. A translation approach to portable ontology specifications. Technical Report, KSL 92-71, Computer Science Department, Stanford University, 1992.

[4] Shoham, Y. Agent-Oriented programming. Artificial Intelligence, 1993,60:51~92.

[5] Gapari, M., Motta, F. Symbol-Level requirment for agent-level programming. Technical Report, UBLCS-94-2, Laboratory for Computer Science, University of Bologna, 1994.

[6] Newell, A. The knowledge level. Artificial Intelligence, 1982,18:87~127.

[7] Dalmonte, A., Gaspari, M. Modelling interaction in agent system. Technical Report, UBLCS-95-7, Laboratory for Computer Science , University of Bologna, 1995.

Analysis of the KQML Model in Multi-Agent*

LIU Hai-long[1,2], WU Tie-jun[1]

[1](*National Laboratory of Industrial Process Control Technology*, *Zhejiang University*, *Hangzhou* 310027, *China*);

[2](*Human Computer Interaction and Intelligent Information Processing Laboratory*, *Institute of Software*, *The Chinese Academy of Sciences*, *Beijing* 100080, *China*)

E-mail: hlliu.dai@163.com; tjwu@iipc.zju.edu.cn

http://www.zju.edu.cn

Abstract: The KQML (knowledge query manage language) is a representative model in multi-agent interaction. In this paper, some conclusions are given about the necessary communication support to the agent-oriented program on the knowledge level based on the analysis of KQML model. Firstly, the agent statue model and the transform model of KQML communication are founded to analyze the necessary conditions in performing interaction with the synchronal and asynchronous KQML model respectively. Secondly, the deadlock and starvation problems in the communication process with KQML model are analyzed from the KQML statue transform aspect, and the cardinal reasons and the solutions are given. At last, the advantages and the disadvantages of the synchronal and asynchronous KQML model are listed respectively, and the choosing principle is given.

Key words: multi-agent; KQML; interaction

* Received January 10, 2000; accepted October 16, 2000

Supported by the National High Technology Development 863 Program of China under Grant No.9845-005