BDI Agent 解释器的研究和改进*

路军[1] 王亚东[1] 王晓龙[1,2]

[1](哈尔滨工业大学计算机科学与工程系 哈尔滨 150001)

[2](香港理工大学计算系 香港)

E-mail: lujun@mlg.hit.edu.cn

摘要 现有的"信念-愿望-意向(BDI)"Agent 系统将它的 3 个主要心智状态(信念、愿望、意向)作为数据结构而不是模态算子来使用,导致信念-愿望-意向 Agent 的研究在理论和实践之间存在较大的差距.该文深入研究了过程推理系统(precedure reasoning system,简称 PRS)和扩展的多 Agent 推理系统(developed multi-agent reasoning system,简称 dMARs),针对 dMARs 系统的解释器,提出了一种改进的编程语言 AgentBDI,结合一个典型的分布式人工智能的例子,对它的语法和语义进行了描述,并给出了一个改进的 BDI Agent 的解释器,使 BDI Agent 能用一种类似于逻辑程序设计的方式编程实现.

关键词 "信念-愿望-意向"Agent,信念,意向,事件,规划.

中图法分类号 TP18

"信念-愿望-意向"Agent(简称 BDI Agent)系统是一个置身于变化环境中的系统,它不断地从感知器接收外界的输入,然后根据内部的心智状态产生动作来影响周围的环境.BDI Agent 系统有 3 个主要的心智状态:信念(belief)、愿望(desire)和意向(intention),它们分别代表了 BDI Agent 所具有的信息、动机和决策.

目前,在 BDI Agent 形式语义方面所做的工作,国际上主要是 Rao 和 Georgeff 在 1995 年提出的一系列 BDI 逻辑[1],这些逻辑都是分枝时间逻辑 CTL* 的扩展[2],用模态连接词来表示信念、愿望和意向等 Agent 的心智状态.国内学者毛新军等人 1999 年提出了一种计算的意愿理论[3].这些有关 BDI Agent 逻辑方面的研究,主要是对信念、愿望和意向之间可能发生的相互关系进行描述,分析和说明 Agent 的行为(behavior).

BDI Agent 系统的主要工作都是由它的解释器(interpreter)来完成的.本文在对 dMARs(developed multi-agent reasoning system)系统[4]和 PRS(procedure reasoning system)系统[5]进行了深入、细致的研究之后,提出了一种简化的、文本化的新型逻辑语言 AgentBDI,并将其作为 dMARs 系统的编程语言,给出了一个改进的 dMARs 系统解释器.dMARs 系统中有很多语言结构,使 AgentBDI 的编程任务更加简单.AgentBDI 语言的引入,使得 Agent 的行为可以靠编程来描述,信念、愿望和意向等 Agent 的心智状态也不再用模态公式表示,而是用 AgentBDI 语言编程描述.

1 一种新型逻辑语言 AgentBDI

信念、愿望和意向在 AgentBDI 中分别定义如下:信念是 Agent 当前的状态,是关于自身、周围环境和其他 Agents 的信息模型;愿望(目标)是系统期望的行为;意向是系统当前或将来某个时刻将要选择执行的任务.

AgentBDI 的字母表 α 为{变量,常数,函数符号,谓词符号,动作符号,连接词,量词,标点符号},其中的变量包括自由变量和约束变量,连接词除了常用的一阶连词外,还包括{!,?,;,←}."!"表示实现,"?"表示检测,";"

* 本文研究得到国家 863 高科技项目基金(No.863-306-ZD05-5)资助.作者路军,1971 年生,博士生,主要研究领域为分布式人工智能.王亚东,1964 年生,副教授,主要研究领域为专家系统,人工智能.王晓龙,1955 年生,教授,博士生导师,主要研究领域为自然语言理解,人工智能.

本文通讯联系人:路军,哈尔滨 150001,哈尔滨工业大学计算机科学与工程系

本文 1999-04-14 收到原稿,1999-08-31 收到修改稿

表示顺序执行,"←"表示隐含.该语言还使用了项的标准一阶定义、一阶公式和闭公式.

下面给出信念、目标、触发事件、动作和规划等基本概念在 AgentBDI 中的定义,并结合一个分布式人工智能的例子来说明这些定义的具体应用.

定义 1(信念). 如果 Bel 是谓词符号,$t_1,\ldots,t_n,t$ 是项,则 $\mathrm{Bel}(t_1,\ldots,t_n)$和 $\mathrm{Bel}(t)$称为原子信念,如果 $\mathrm{Bel}(t)$和 $\mathrm{Bel}'(t')$是原子信念,则 $\mathrm{Bel}(t)\wedge\mathrm{Bel}'(t')$和 $\neg\mathrm{Bel}(t)$称为信念.

例如,有两个机器人,将货车里的货物卸到仓库里.机器人必须搬运货物,并将其放入仓库中.在做此动作时,机器人之间必须按先后顺序搬运货物和进入仓库,并且不能同时出现在仓库内,以免互相碰撞,影响搬运进程.下面,我们为这些机器人编程.

本例中 Agent 的信念包含机器人和货物所在的位置以及机器人、货物与仓库之间的位置关系.信念 adjacent(X,cargo)表示机器人 X 和货物相邻,信念 adjacent(X,warehouse)表示机器人 X 和仓库相邻,信念 At(X,warehouse)表示机器人 X 已经在仓库中,信念 InHand(cargo,X)表示货物在机器人 X 手里.Agent 的原子信念集是信念原子的实例集,如{adjacent(*robot*1,cargo),At(*robot*1,warehouse)}等.

目标是 Agent 想要达到的状态,本文讨论两种目标:(1) 实现目标,用!Goal(t)表示,是 Agent 要达到的状态,在此状态下,Goal(t)为真信念;(2) 检测目标,用?Goal(t)表示,是指 Agent 检测公式 Goal(t)是否为真.在此例中,将货物放入仓库中可以写成一个实现目标,如!PutDown(X);查看仓库里是否还有其他机器人可以写成检测目标,如?At(X,warehouse).

定义 2(目标). 如果 Goal 为一个谓词符号,$t_1,\ldots,t_n,t$ 为项,则!Goal($t_1,\ldots,t_n$),!Goal(t),?Goal($t_1,\ldots,t_n$)和?Goal(t)称为目标.其中!Goal($t_1,\ldots,t_n$)和!Goal(t)称为实现目标;?Goal($t_1,\ldots,t_n$)和?Goal(t)称为检测目标.

当 Agent 产生一个新目标,或者获悉周围环境的一个变化时,就会被触发并对自身的目标和信念进行增减,新目标的产生或者环境的变化称为触发事件.增加、减少信念和增加、减少目标是常见的 4 种触发事件.增加用算子"+"表示,减少用算子"−"表示.本例中的两个触发事件如下:发现仓库里没有机器人记为+(¬At(X,warehouse));获得目标并将货物放入仓库记为+(!PutDown(X)).

定义 3(触发事件). 如果 Bel(t)为一个信念原子,!Goal(t)和?Goal(t)为目标,则+Bel(t),−Bel(t),+!Goal(t),−!Goal(t),−?Goal(t),−?Goal(t)称为触发事件.

Agent 的职能就是感知其所在环境的变化,并根据自身的目标,执行相应的动作.通过这些动作来改变周围环境的状态.例如,假设 move 是一个动作符号,机器人 X 走进仓库是一个动作,记为 move(X,warehouse),此动作的结果是使机器人 X 走入仓库.

定义 4(动作). 如果 Act 是一个动作符号,$t_1,\ldots,t_n,t$ 为一阶项,则 $\mathrm{Act}(t_1,\ldots,t_n)$或 $\mathrm{Act}(t)$为一个动作.

Agent 用规划作为手段(means)来完成目标.规划包含规划头(head)和规划体(body).规划头中包含触发事件(trigger event)和上下文(context),它们之间用冒号":"分开.触发事件直接导致规划的触发,如信念或目标的增减;上下文是当规划被触发时,Agent 原子信念集中应当具有的信念.规划体是一系列的目标和动作,它描述一个 Agent 应该实现或检测的目标以及 Agent 应当执行的动作.

例如,若编写一个规划,当仓库里没有其他机器人时,触发此规划.如果机器人与仓库相邻,它将拾起货物(动作),走进仓库(目标),接着执行初始动作,将货物放入仓库.这个规划可如下编写:

```
+(¬At(X,warehouse)):adjacent(cargo,Y)&
                    adjacent(Y,warehouse)
                        <—PickUp(cargo);
                        !At(Y,warehouse);
                        PutDown(Y).                    (规划 1)
```

下面给出机器人改变自身位置的规划.如果机器人产生一个目标,想要移动到仓库里,而它已经在仓库里,它将不必做任何事情,规划体此时为 true;如果规划头的内容表明没有其他机器人在仓库里,而且机器人和仓库相邻,它将执行初始动作,进入仓库.这两个规划可如下编写:

+! A:(X,warehouse);At(X,warehouse) <—true. (规划 2)

+! At(X,warehouse):(¬At(Y,warehouse))&.
(not(X=Y))&.
adjacent(X,warehouse)
←move(X,warehouse) (规划 3)

定义 5(规划). 如果 e 为触发事件,Bel(1),...,Bel(m)为原子信念,$f_1,...,f_n$ 为目标或动作,则 e:Bel(1) ∧... ∧Bel(m)←$f_1;...;f_n$ 是一个规划.

2 BDI Agent 解释器的非形式化讨论

dMARs 系统的解释器循环执行以下 5 个步骤:

(1) 感知周围环境和 Agent 内部状态的变化,并将获悉的事件加入事件序列 E 中;

(2) 找出触发事件 e 与事件序列 E 中的事件相匹配的规划,从而产生一些新的愿望(匹配的规划集);

(3) 从匹配的规划集中选择一个规划(意向)执行;

(4) 判断事件是否为子目标,将意向压入已经存在的意向集或新的意向集中(意向集在这里被看作是规划的堆栈);

(5) 选择一个意向集,取出顶层规划(意向),执行这个规划的规划体:若为动作,则执行相应的动作;若为子目标,则将这个子目标送入事件集中.

基于 AgentBDI 的一个 Agent 模型包含一个信念集 B,一个规划集 P,一个意向集 I,一个事件集 E,一个动作集 A 和 3 个选择函数 Se,Sp,Si.

下面,我们非形式化地给出基于 AgentBDI 语言的 dMARs 系统解释器的操作语义.

Step 1(产生触发事件). 当 Agent 感知到环境的一个变化,或者外部用户请求 Agent 执行一个目标时,相应的触发事件就产生了,这些触发事件将异步地加入事件集 E 中.

Step 2(选择相关规划). 选择函数 Se 从事件集 E 中选择一个事件 Ei 进行处理,并将这个事件从 E 中去掉,然后与规划集 P 中的规划进行匹配,能与其匹配的规划称为相关规划,相匹配的匹配元称为相关匹配元.

Step 3(选择可用规划). 每一个事件都可能有许多相关规划,选择函数 Sp 从这些规划中选择出一个规划,把相应的匹配元用在所选的规划中,形成对触发事件作出反应的可用规划(意向),每个意向集都是由可用规划或其他意向组成的堆栈.如果触发事件是外部事件,此意向将产生一个新意向,加入到意向集 I 中;如果触发事件是内部事件,比如增加一个新目标,则此意向将被压入触发此内部事件的那个意向集的栈顶.

Step 4(选择意向执行). 选择函数 Si 选择一个意向执行.当 Agent 执行一个意向时,首先执行顶端意向体的第 1 个目标或动作.执行一个实现目标等同于产生一个内部事件,将此目标加入当前意向;执行一个检测目标,相当于寻找此目标的一个替换,使其成为原子信念集 B 的一个逻辑推论.当找到此替换后,将检测目标从顶端意向体中去掉,目标的替换被用到余下的顶端意向体中;执行一个动作的结果,是使此动作加入到动作集 A 中,并将其从顶端意向体中去掉.

Step 5(返回事件集). Agent 又返回到事件集 E 中,整个循环继续直到 E 中没有任何事件存在或是没有可运行的意向为止.

3 形式化语义描述

3.1 形式化模型及其相关概念

定义 6(Agent 模型). 基于 AgentBDI 的 Agent 模型是一个八元组〈B,P,I,E,A,Se,Sp,Si〉,B 为一个原子信念集,P 为一个规划集,I 为一个意向集,E 为一个事件集,A 为一个动作集.选择函数 Se 从事件集 E 中选择一个事件,选择函数 Sp 从可用规划集中选择一个可用规划,选择函数 Si 从意向集 I 中选择一个意向.

如上所述,我们已经给出了 Agent 模型中信念 B、规划 P 和动作 A 的定义,下面,我们给出意向 I 和事件 E 的定义:

定义 7(意向). 集合 I 是一个意向集.意向是规划的堆栈,用$[P_1|\ldots|P_z]$表示,这里,P_1 为栈底,P_z 为栈顶.栈间的元素用|隔开.

定义 8(真意向). 意向[+! true:true←true]称为真意向,用 T 来表示.

定义 9(事件). 集合 E 是事件的集合.每个事件 Ei 是一个二元组$\langle d,i\rangle$,这里,d 是触发事件,i 是意向.如果 i 是真意向,则此事件为外部事件;否则,此事件为内部事件.

定义 10(相关匹配元). 一个从事件集 E 中选出的触发事件 d 与规划集 P 中所有规划的触发事件进行匹配,两者之间最通用的匹配元称为相关匹配元.

定义 11(相关规划). 选择函数 $Se(E)=Ei\cong\langle d,i\rangle$,一个规划 p 为 e:Bel(1)∧...∧Bel(m)←f_1;...;f_n.如果存在一个相关匹配元 z,使得 $dz=ez$,则规划 p 称为事件 Ei 的相关规划.

在上例中,假设从 E 中选出的触发事件为+At(X,warehouse),则在相关匹配元为{X/*robot*1}时,规划 1 和规划 2 对于此事件是相关的.

如果存在一个相关规划的替换,当它和相关匹配元组合并应用到规划的上下文中时,是原子信念集 B 的一个逻辑推论,则一个相关规划也是可用规划.也就是说,一个相关规划的上下文必须是 B 的一个逻辑推论,才能成为一个可用规划.

定义 12(可用规划). 一个规划 p 为 e:Bel(1)∧...∧Bel(m)←f_1;...;f_n,如果存在一个相关匹配元 z 和规划 P 的一个替换 Ψ,使得所有(Bel(1)∧...∧Bel(m))$z\Psi$ 是 B 的逻辑推论,则规划 p 称为事件 Ei 的可用规划,$z\Psi$ 称为 e 的可用匹配元.

然后,我们来看前面的例子,Agent 的原子信念集为{adjacent(*robot*2,warehouse),At(*robot*1,warehouse),InHand(cargo,*robot*2)},这时的可用匹配元是{X/*robot*2,Y/*robot*1},所以只有规划 3 是可用的.

3.2 解释器中的相关定理

下面,我们给出若干相关定理,用以形式化地描述意向的采纳和执行等 Agent 解释器的相关工作.

根据事件类型的不同(内部事件或外部事件),意向也有所不同.如果此事件是一个外部事件,则系统将选择那个事件的可用规划,把可用匹配元应用到规划体中,产生一个新的意向,并将其加入到意向集 I 中.

定理 1. 选择函数 $Sp(Oe)=p$,其中 Oe 是事件 $Ei=\langle d,i\rangle$所有可用规划的集合,规划 p 为 e:Bel(1)∧...∧Bel(m)←f_1;...;f_n.对于 i 为真意向的事件 Ei(即 Ei 为外部事件),可用规划 p 为意向,当且仅当存在一个可用匹配元 z,使得[+! true:true←true|(e:Bel(1)∧...∧Bel(m)←f_1;...;f_n)z]∈I.

定理 1 给出了当事件 Ei 为一个外部事件时,规划成为意向的充要条件.

在本例中,仅有的可用规划:规划 3 将成为意向,意向 I 现在可以描述为

```
| At(robot2,warehouse):(¬At(robot1,warehouse))&
                       not(robot2=robot1))&
                       adjacent(robot2,warehouse)
                            ←move(robot2,warehouse);
```

否则,如果此事件是一个内部事件,则意向将被压入触发此内部事件的意向集的栈顶.

定理 2. 选择函数 $Sp(Oe)=p$,Oe 是事件 Ei 所有可用规划的集合,这里,$Ei=\langle d,[p_1|\ldots|f:C_1\wedge\ldots\wedge C_y\leftarrow!\,\mathrm{Goal}(t);f_2;\ldots;f_n]\rangle$,$p$ 为+!Goal(s):Bel(1)∧...∧Bel(m)←k_1;...;k_j. 对于事件 Ei,规划 p 为意向,当且仅当存在一个可用匹配元 z,使得[p_1|...|f:C_1∧...∧C_y←! Goal(t);f_2;...;f_n|(+! Goal(s):Bel(1)∧...∧Bel(m))z←(k_1;...;k_j)z;(f_2;...;f_n)z]∈I.

定理 2 给出了当事件 Ei 为一个内部事件时,规划成为意向的充要条件.

与逻辑程序设计语言不同的是:目标 Goal 是通过产生一个事件而间接产生的,这就使得 Agent 能改变自身注意的"焦点",必要时可采纳并执行其他意向,从而更好地控制目标.从这一点上看,可以把 AgentBDI 程序设计看作是多线程、可中断的逻辑程序设计.

当一个意向栈被选择执行时,首先执行栈顶意向,意向体的第 1 个公式可能有 4 种情况:(1) 实现目标;(2) 检测目标;(3) 一个动作;(4) 真值.

下面的定理3～定理6分别给出了在4种不同情况下,意向已经执行的充要条件.

定理3. 选择函数 $Si(I)=i$,其中 i 为 $[p_1|\dots|f:C_1\wedge\dots\wedge C_y\leftarrow \mathrm{Goal}(t);f_2;\dots;f_n]$,则意向 i 是已经执行的,当且仅当 $\langle +!\ \mathrm{Goal}(t),i\rangle\in E$.

上面的定理表明,如果意向体的公式为实现目标,意向是已经执行的,当且仅当系统产生一个事件来执行它.

定理4. 选择函数 $Si(I)=i$,其中 i 为 $[p_1|\dots|f:C_1\wedge\dots\wedge C_y\leftarrow \mathrm{Goal}(t);f_2;\dots;f_n]$,则意向 i 是已经执行的,当且仅当存在一个替换 Ψ,使得所有 $\mathrm{Goal}(t)\Psi$ 都是 B 的一个逻辑推论,并且 i 被意向 $[p_1|\dots|(f:C_1\wedge\dots\wedge C_y)\Psi\leftarrow f_2\Psi;\dots;f_n\Psi]$ 所替代.

上面的定理表明,如果意向体的公式是一个检测目标,意向是已经执行的,当且仅当系统在Agent的原子信念集中找到了一个可以与此目标相匹配的相关匹配元,并将这个相关匹配元用于意向堆栈的其余部分.

定理5. 选择函数 $Si(I)=i$,其中 i 为 $[p_1|\dots|f:C_1\wedge\dots\wedge C_y\leftarrow \mathrm{Act}(t);f_2;\dots;f_n]$,则意向 i 是已经执行的,当且仅当 $\mathrm{Act}(t)\in A$,且 i 被意向 $[p_1|\dots|f:C_1\wedge\dots\wedge C_y\leftarrow f_2;\dots;f_n]$ 所替代.

上面的定理表明,如果意向体的公式是一个动作,意向是已经执行的,当且仅当系统将此动作加入动作集 A 中,并将它从意向栈中移走.

定理6. 选择函数 $Si(I)=i$,其中 i 为 $[p_1|\dots|p(z-1)|!\mathrm{Goal}(t):C_1\wedge\dots\wedge C_y\leftarrow \mathrm{true}]$,$p(z-1)$ 为 $e:\mathrm{Bel}(1)\wedge\dots\wedge\mathrm{Bel}(x)\leftarrow !\ \mathrm{Goal}(s);f_2;\dots;f_n$,则意向 i 是已经执行的,当且仅当存在一个替换 Ψ,使得 $\mathrm{Goal}(t)\Psi=\mathrm{Goal}(s)\Psi$,并且 i 被意向 $[p_1|\dots|p(z-2)|(e:\mathrm{Bel}(1)\wedge\dots\wedge\mathrm{Bel}(x))\Psi\leftarrow(f_2;\dots;f_n)\Psi]$ 所替代.

上面的定理表明,如果意向体的公式是一个真值,意向是已经执行的,当且仅当栈顶意向和实现目标从意向堆栈中被移走,并将替换用于意向体的其余部分.

继续上面提到的例子,使用规划3,执行 I,根据定理5,将{move(*robot*2,warehouse)}加入到动作集 A 中,并将 I 改变为

$$+\ \mathrm{At}(robot2,\mathrm{warehouse}):(\neg\,\mathrm{At}\ (robot1,\mathrm{warehouse}))\&\ \mathrm{not}(robot2=robot1))\&\ \mathrm{adjacent}(robot2,\mathrm{warehouse})\leftarrow \mathrm{true}$$

在下一个循环里,当机器人移动到仓库里时,系统产生一个外部事件(信念修正事件),用以将机器人的位置修改为warehouse,即将信念 At(*robot*2,warehouse)加入到信念集 B 中,并将事件 $+\mathrm{At}(robot2,\mathrm{warehouse})$ 加入到事件集 E 中.由于此举没有相关的规划,系统将选择最上面的意向执行,执行的结果产生一个意向增加事件并加入到事件集 E 中;换句话说,E 为 $\{\langle +!\ \mathrm{At}(robot2,\mathrm{warehouse}),i\rangle\}$,$i$ 是与以前相同的意向.根据定理2,在此情况下,相关规划为带有匹配元 $\{X/robot2\}$ 的规划1,此规划也是可用的,相关匹配元是相同的,由于此规划体为真值,意向被满足并且事件集为空,于是循环终止,直到有下一个事件加入到集合 E 中.

4 相关定理的证明

首先给出定理的证明理论:一个标签转换系统,然后给出相应的证明规则.这些证明规则定义了Agent从一种状态向另一种状态的转换.这些转换和Agent解释器的操作语义直接相关,所以,在BDI Agent的解释器和相关定理的证明理论之间能够建立一种一一对应的关系.

定义13(BDI Agent转换系统). BDI Agent转换系统是一个二元组 $\langle\Gamma,\vdash\rangle$,其中 Γ 是BDI的状态集,$\vdash$ 是一个二元转换关系集,且 $\vdash\subseteq\Gamma\times\Gamma$.

定义14(BDI Agent的状态). BDI Agent的状态是一个五元组 $\langle Ei,Bi,Ii,Ai,i\rangle$,其中 $Ei\subseteq E$,$Bi\subseteq B$,$Ii\subseteq I$,$Ai\subset A$,i 是转换标签.

在以下的证明规则中,我们假定规划集 P 为常量,目标Goal为Agent所采纳的意向.

4.1 意向集转换的证明规则

下面给出在Agent模型中意向集从一种状态转换为另一种状态的证明规则.

首先给出定理 1 的证明规则 I-End:

$$\text{(I-End)}\quad \frac{\langle\{\ldots,\langle+!\ \text{Goal}(t),T\rangle,\ldots\},Bi,Ii,Ai,i\rangle}{\langle\{\ldots\},Bi,Ii\cup\{[PZI\Psi]\},Ai,I+1\rangle},$$

其中 $P=+\text{Goal}(s)$;$\text{Bel}(1)\wedge\ldots\wedge\text{Bel}(m)\leftarrow f_1;\ldots;f_n\in P$,$Se(E)=\langle+!\ \text{Goal}(t),T\rangle$,$\text{Goal}(t)Z=\text{Goal}(s)Z$,且 $\forall\ (\text{Bel}(1)\wedge\ldots\wedge\text{Bel}(m))\Psi$ 是 Bi 的逻辑推论.

证明规则 I-End 说明:对于 Se 函数选中进行处理的外部事件,Agent 的意向集 I 是如何发生变化的.

定理 2 的证明规则 I-Means 与 I-End 相似,只不过 I-Means 将可用规划集压入意向集的顶端:

$$\text{(I-Means)}\quad \frac{\langle\{\ldots,\langle+!\ \text{Goal}(t),j\rangle,\ldots\},Bi,\{\ldots,[P_1|\ldots|P_x],\ldots\},Ai,i\rangle}{\langle\{\ldots\},Bi,\{\ldots,[P_1|\ldots|P_x|PZ\Psi],\ldots\},Ai,i+1\rangle},$$

其中 $P_x-f;C_1\wedge\ldots\wedge C_y\leftarrow\text{Goal}(t);f_2;\ldots;f_n$,

$$P=+!\ \text{Goal}(s);\text{Bel}(1)\wedge\ldots\wedge\text{Bel}(m)\leftarrow K_1;\ldots;K_x,Se(E)=\langle+!\ \text{Goal}(t),j\rangle,$$

$j=[P_1|\ldots|P_n]$,$\text{Goal}(t)Z=\text{Goal}(s)Z$,$\forall\ (C_1\wedge\ldots\wedge C_y)\Psi$,且是 Bi 的逻辑推论.

证明规则 I-Means 说明:对于 Se 函数选中进行处理的内部事件,Agent 的意向集 I 是如何发生变化的.

4.2 意向执行的证明规则

下面,我们给出意向执行的证明规则.选择函数 Si 从意向堆栈中选出一个意向执行,根据意向体中公式的不同,可以制定出意向执行的 4 个证明规则.

对于定理 3,执行实现目标的证明规则 Exec! 为

$$\text{(Exec!)}\quad \frac{\langle Ei,Bi,\{\ldots,[P_1|\ldots|P_{x-1}|f;C_1\wedge\ldots\wedge C_y\leftarrow!\ \text{Goal}(t);f_2;\ldots;f_n],\ldots\},Ai,i\rangle}{\langle Ei\cup\{\langle+!\ \text{Goal}(t),j\rangle\},Bi,\{\ldots,[P_1|\ldots|P_x],\ldots,\},Ai,i+1\rangle},$$

其中 $Si(Ii)=j=[P_1|\ldots|P_x]$,且 $P_x=f;C_1\wedge\ldots\wedge C_y\leftarrow!\ \text{Goal}(t);f_2;\ldots;f_n$.

对于定理 4,执行检测目标的证明规则 Exec? 为

$$\text{(Exec?)}\quad \frac{\langle Ei,Bi,\{\ldots,[P_1|\ldots|f;C_1\wedge\ldots\wedge C_y\leftarrow!\ \text{Goal}(t);f_2;\ldots;f_y],\ldots\},Ai,i\rangle}{\langle Ei,Bi\cup\{\text{Goal}(t)\Psi\},\{\ldots,[P_1|\ldots|(f;C_1\wedge\ldots\wedge C_y)\Psi\leftarrow(f_2\Psi;\ldots;f_n\Psi],\ldots\},Ai,i+1\rangle},$$

其中 $Si(Ii)=j=[P_1|\ldots|P_x]$,且 $P_x=f;C_1\wedge\ldots\wedge C_y\leftarrow?\ \text{Goal}(t);f_2;\ldots;f_n$.

对于定理 5,执行一个动作的证明规则 ExecA 为

$$\text{(ExecA)}\quad \frac{\langle Ei,Bi,\{\ldots,[P_1|\ldots|f;C_1\wedge\ldots\wedge C_y\leftarrow!\ \text{Act}(t);f_2;\ldots;f_n],\ldots\},Ai,i\rangle}{\langle Ei,Bi,\{\ldots,[P_1|\ldots|f;C_1\wedge\ldots\wedge C_y\leftarrow f_2;\ldots;f_n],\ldots\},Ai\cup\{\text{Act}(t)\},i+1\rangle},$$

其中 $Si(Ii)=j=[P_1|\ldots|P_x]$,且 $P_x=f;C_1\wedge\ldots\wedge C_y\leftarrow?\ \text{Act}(t);f_2;\ldots;f_n$.

对于定理 6,执行一个真值的证明规则 ExecT 为

$$\text{(ExecT)}\quad \frac{\langle Ei,Bi,\{\ldots,[P_1|\ldots|P_{x-1}|!\ \text{Goal}(t);C_1\wedge\ldots\wedge C_y\leftarrow\text{true}],\ldots\},Ai,i\rangle}{\langle Ei,Bi,\{\ldots,[P_1|\ldots|P_{x-2}|(e;\text{Bel}(1)\wedge\ldots\wedge\text{Bel}(x))\Psi\leftarrow f_2\Psi;\ldots;f_n\Psi],\ldots\},Ai,i+1\rangle},$$

其中 $Si(Ii)=j=[P_1|\ldots|P_x]$,$P_x=\text{Goal}(t);C_1\wedge\ldots\wedge C_y\leftarrow\text{true}$,$P_{x-1}=e;\text{Bel}(1)\wedge\ldots\wedge\text{Bel}(x)\leftarrow\text{Goal}(s);f_2;\ldots;f_n$,$\text{Goal}(t)\Psi=\text{Goal}(s)\Psi$.

5 BDI Agent 解释器的改进

5.1 一种改进的 BDI Agent 解释器

根据前面对 Agent 解释器的形式化描述和相关定理,下面给出一个改进的基于 AgentBDI 语言的 BDI Agent解释器,见表 1.

其中,函数 top 返回意向堆栈的栈顶;函数 head 返回被采纳为意向的规划的规划头;函数 body 返回被采纳为意向的规划的规划体;函数 first 和 rest 分别表示序列的第 1 个元素和序列中除第 1 个元素以外的其他元素;函数 push 取一个意向和一个意向堆栈,并将此意向压入意向堆栈的顶端;函数 pop 取一个意向堆栈,返回栈顶.

Table 1 Improved BDI Agent interpreter
表 1 改进的 BDI Agent 解释器

```
Interpreter()
Initiate();
Do Se(E)=Ei=(d,i);
    E=E/Ei;
    Oe={pΨ}                                   /*Ψ为事件Ei和规划p的可用匹配元*/
  if External-event (Ei)
      then I=I∪[Sp (Oe)];
  else push (Sp(Oe)z, i);                     /*这里z是e的可用匹配元*/
  case first (body(top(Si(I))))=true
    m=pop(Si(I));
    push (head(top(Si(I)))Ψ←rest(body(top(Si(I))))Ψ,Si(I))
                                  /*Ψ为可用匹配元,且mΨ=head(top(Si(I)))Ψ*/
  case first (body(top(Si(I))))=! Goal(t)
      E=E∪(+! Goal(t),Si(I))
  case first (body(top(Si(I))))=? Goal(t)
      pop(Si(I));
      push(head(top(Si(I)))Ψ←rest(body(top(Si(I))))Ψ,Si(I))
  case first (body(top(Si(I))))=Act(t)
      pop(Si(I));
      push (head(top(Si(I)))←rest(body(top(Si(I)))),Si(I));
      A=A∪{Act(t)};
Until E=0.
```

5.2 与原解释器的比较

90 年代初,以 Anand. Rao 为代表的澳大利亚学者为了缩短 BDI Agent 理论与实践之间的差距,在 BDI Agent方面做了一系列工作,在 1991 年提出了 PRS 系统,该系统至今还被认为是最完备的 BDI Agent 系统. PRS 系统的原解释器见表 2.

Table 2 The abstract agent interpretcr of original PRS
表 2 原 PRS 系统的抽象的 Agent 解释器

```
Interpreter()
Initiate();
option-generator(trigger-events)
option:={};
for trigger-event? trigger-events do
  for plan? plan-library do
    if matches(invocation(plan),trigger-event) then
      if provable(precondition(plan),B) then
        option:=options∪{plan};
return(option).
deliberate(options)
if length(options)<=1 then return(option);
else metalevel-options:=option-generator(belief-add(option-set(options)));
  select-options:=deliberate(metalevel-options);
  if null(selected-options) then
      return(random-choice(options));
else return(selected-options).
```

与我们给出的基于 AgentBDI 的解释器相比,原 PRS 系统理论上使用模态算子来刻画信念、愿望、规划等心智状态,而在实际系统的实现中却使用数据结构来描述它们,这就使得 PRS 系统的解释器过于抽象,不能说明

模态理论、模态证明理论与抽象解释器之间具有一一对应的关系.

而我们给出的 AgentBDI 语言,是与 dMARs 系统解释器的操作语义直接相关的,因此,在 AgentBDI 理论、解释器的证明理论和实际的解释器之间建立了一一对应的关系.

6 结束语

BDI Agent 的说明、设计、修改和应用近年来倍受关注.到目前为止,有许多 BDI Agent 的模型已成功地应用于许多重要的领域[6].这些应用系统都有一个共同的特点:即将信念、愿望和意向构造成相应的数据结构,而不是像理论上那样用模态或时态算子来表示.这些系统编码的复杂性说明,这些应用系统都缺乏一个有效的理论基础,事实证明,这些表达逻辑(express logic)对实际问题的用处不大.

而 BDI Agent 研究的长期目标是用一个理性的、实用的和可描述的语言来表明三者之间的一一对应关系[7].因此,近几年对 BDI 解释器的研究又热了起来,在这方面,澳大利亚学者 Weerasooriya 等人做了一些尝试性的工作[8],但由于 Agent 的心智状态(mental statement)非常复杂,使得 Agent 系统的解释器也相当复杂.因此,今后对 Agent 的研究应将重点放在将面向对象技术引入 Agent 的心智状态上来,以研究信念、愿望、意向的继承和封装,使 BDI Agent 的解释器变得更加简单和实用.

参考文献

1 Rao A S, Georgeff M P. BDI Agents: from theory to practice. In: Victor Lesser ed. Proceedings of the 1st International Conference on Multi-Agent Systems (ICMAS-95). San Francisco, CA: AAAI Press/MIT Press, 1995. 312~319

2 Rao A S, Georgeff M P. Formal models and decision procedures for multi-agent systems. Technical Report 61, Melbourne: Australian Artificial Intelligent Institute, 1995

3 Mao Xin-jun, Wang Huai-min, Chen Huo-wang *et al*. The computing intention theory of agent in multi-agent system. Journal of Software, 1999,10(1):43~48
(毛新军,王怀民,陈火旺等. Agent 在多 Agent 系统中计算的意愿理论. 软件学报,1999,10(1):43~48)

4 d'Inverno M, Kinny D, Luck M *et al*. A formal specification of dMARS. Technical Report 72, Melbourne: Australia Artificial Intelligent Institute, 1997

5 Rao A S, Georgeff M P. Modeling rational agents within a BDI architecture. In: Nebel B, Rich C, Swartout W eds. Proceedings of the 2nd International Conference on Principles of Knowledge Representation and Reasoning. San Mateo, CA: Morgan Kaufmann Publishers, Inc., 1991. 473~484

6 Lu Jun, Wang Ya-dong, Wang Xiao-long. The studies and developments of BDI Agent. Computer Science, 1999,26(2):48~51
(路军,王亚东,王晓龙. 信念-愿望-意向 Agent 的研究与进展. 计算机科学,1999,26(2):48~51)

7 Shoham Y. Agent-Oriented programming. Artificial Intelligence, 1993,60(7):51~92

8 Weerasooriya D, Rao A S, Ramamohanarao K. Design of a concurrent agent-oriented language. Technical Report 64, Melbourne: Australia Artificial Intelligent Institute, 1995

Research and Improvement of the Interpreter of BDI Agent

LU Jun[1] WANG Ya-dong[1] WANG Xiao-long[1,2]

[1](*Department of Computer Science and Engineering Harbin Institute of Technology Harbin* 150001)
[2](*Department of Computing Hong Kong Polytechnic University Hong Kong*)

Abstract The implemented BDI (belief, desire and intention) Agent systems used the three major mental attitudes as data structures, rather than as model operators. So there still remains a large gap between theory and practice. In this paper, the authors present an improved programming language——AgentBDI, concerning the interpreter of dMARS (developed multi-agent reasoning system) after in-depth studying PRS (procedure reasoning system) and dMARs, and provide AgentBDI's grammar and operational semantic using a classical example of DAI. The authors also provide an improved interpreter of BDI Agent based on AgentBDI. AgentBDI can be viewed as an abstraction of dMARs, and allows BDI Agent programs to be written and interpreted in a manner similar to the logical programs.

Key words BDI (belief, desire and intention) Agent, belief, intention, event, plan.