基于主 Agent 信念修正的推测计算及其资源协商*

王黎明[1,2+], 黄厚宽[1]

[1](北京交通大学 计算机与信息技术学院,北京 100044)

[2](郑州大学 信息工程学院,河南 郑州 450052)

Speculative Computation Based on Master Agent Belief Revision and Its Resource Negotiation

WANG Li-Ming[1,2+], HUANG Hou-Kuan[1]

[1](School of Computer and Information Technology, Beijing Jiaotong University, Beijing 100044, China)

[2](School of Information Engineering, Zhengzhou University, Zhengzhou 450052, China)

+ Corresponding author: Phn: +86-10-51683602, Fax: +86-10-51840526, E-mail: ielmwang@zzu.edu.cn, http://www.njtu.edu.cn

Received 2004-05-06; Accepted 2005-05-09

Wang LM, Huang HK. Speculative computation based on master Agent belief revision and its resource negotiation. *Journal of Software*, 2005,16(11):1920−1928. DOI: 10.1360/jos161920

Abstract: Abduction-Based speculative computation is a process which uses default hypothesis as tentative answer and continues computation when resources information cannot be obtained in time. In a computing process, if response is not consistent with belief, master Agent will revise its belief. To achieve goals, get the more accurate result of computation and reduce risks of decision under time constraints, the master Agent in speculative computation should make efforts to obtain more information via negotiation, so the negotiation is a key measure in reducing risks of decision. In this paper, basic theories of abduction and speculative computation are introduced. Based on these theories, a framework extended from the speculative computation with belief revision based on time constraints is presented, and the further negotiation framework based on time constraints and negotiation algorithm with belief revision are presented. The algorithm is embedded in speculative computation to make the master Agent revise its belief during negotiation. Finally, three experiments on goods transportation are given. The experimental results indicate that the algorithm can improve the accuracy of result of the speculative computation and reduce the risk of the result.

Key words: speculative computation; belief revision; abduction; negotiation dialogues; dialogue constraint; time constraint

摘 要: 基于假设推理(abduction-based)的推测计算(speculative computation)是在资源信息不能及时到达时,利用缺省假设进行计算的过程.在计算过程中,如果应答和信念不一致,则主 Agent 将修正它的信念.为了实现目标,在有限时间内使推测计算的结果更精确,主 Agent 要通过协商获得尽可能多的实际信息,协商是降低决策风险的主要途

* Supported by the National Natural Science Foundation of China under Grant No.60443003 (国家自然科学基金)

作者简介: 王黎明(1963－),男,河南浚县人,博士,副教授,主要研究领域为分布式人工智能,数据挖掘,机器学习;黄厚宽(1940－),男,教授,博士生导师,CCF 高级会员,主要研究领域为分布式人工智能,机器学习,数据挖掘,演化计算.

径.在介绍假设推理和推测计算的基本原理的基础上,提出了基于时间约束的推测计算扩展框架、基于时间约束的进一步协商框架和基于信念修正的协商算法,并将进一步协商框架和协商算法嵌入到推测计算的过程中,在协商过程中赋予主 Agent 更强的信念修正能力.最后,在货物运输领域的实验中,证实了基于信念修正的推测计算的有效性.

关键词: 推测计算;信念修正;假设推理;协商对话;对话约束;时间约束
中图法分类号: TP18 文献标识码: A

推测计算(speculative computation)[1−3]是在不能及时获得资源信息的情况下,利用缺省假设进行计算的一种计算过程.在一个模拟现实的主-仆(master-slave)多 Agent 系统中,主 Agent 为了实现其目的,需要通过和其他 Agent 协商对话(negotiation dialogue)[4,5]获得必要的资源.当它获得的应答和当前缺省假设一致时,计算继续,否则,它修正自己的信念,并调整相应计算.我们在推测计算中采用协商获得资源信息,并要求推测计算和资源协商在一定的时间内完成,所以时间是一个关键因素.

文献[1,3]定义了一个基本推测框架$\langle \Delta,P\rangle$,Δ表示 Agent 的信念,P 表示逻辑程序.推测计算就是以这个框架为基础进行的.文献[2]定义了一个类 Prolog(Prolog-like)过程,该过程将规划、动作执行、程序更新(program updates)和规划修正(plan modifications)集成在一起.推测计算利用这个过程,按照应答对规划和动作的执行进行修正.然而,这些文献并没有考虑计算的时间因素和进一步协商过程.实际上,在推测计算过程中,主 Agent 必须保证计算和协商在一定时间内完成.在时间允许的情况下,如果没有进一步的协商过程,主 Agent 就不能获得较多的资源信息,也就无法降低计算结果的风险.为此,本文的主要目的是将基本推测框架扩充为具有时间约束的框架,将基于时间约束的进一步资源协商框架嵌入到推测计算过程中,并赋予主 Agent 更强的信念修正能力.

1 推测框架的扩充

文献[1,3]提出了在信息不完全情况下,基于假设推理的推测计算框架.为了实现多 Agent 之间的协商对话,我们将基本推测框架扩充为七元组:$SF_{MS}=\langle \Sigma,\beta,\Delta,SR,L,Tsc,P\rangle$.

- Σ表示一个参与计算的 Agent 集,即$\Sigma=\{M,a_1,a_2,\ldots,a_n\}$,其中 M 表示主 Agent,$a_1,a_2,\ldots,a_n$ 表示仆 Agent.推测计算是主 Agent 的行为,M 为了实现其目的需要和仆 Agent 进行协商.在解决某些问题时,M 可以对未知资源提前设置一定的缺省值.
- β表示一个谓词集,即$\beta=\beta_{askable}+\beta_{response}$,其中$\beta_{askable}$ 表示一个可问谓词集,$\beta_{response}$ 表示应答谓词集.如果 Q 是用β中的谓词表示的关于资源的一个文字,那么 $Q\geqslant S$ 表示将 Q 发送给 S,并且由 S 来确认 Q,而 $Q\leqslant S$ 表示 Q 是来自 S 的结果,其中 $S\in\Sigma$.
- Δ表示一个缺省集,也称为信念集.Δ中的每个值都是用β中的谓词表示的.
- SR 表示 M 所需要的资源文字集,即 $SR=SR_M+SR_{un}$,其中 SR_M 表示 M 相信可以获得的资源与资源提供者所组成的文字集,SR_{un} 表示 M 相信不能获得的资源与资源提供者所组成的文字集.例如,文字 $R\leqslant S\in SR_M$ 表示 M 相信资源 R 能从 S 那里获得,文字 $R\leqslant S\in SR_{un}$ 表示 M 相信资源 R 不能从 S 那里获得.
- L 表示一个协商语言.在文献[1]中,M 和 a_i 之间的通信只是一对简单的问与答.若在推测计算中包含进一步协商过程就需要定义一个协商语言.在下一部分,我们将给出这个语言的形式定义.
- Tsc 表示推测计算所花费的时间.推测计算要在一定时间约束下完成决策过程.Tsc 采用倒计时的方式,即如果 Tsc 等于 0,则推测计算结束,此时,M 不再和其他仆 Agent 进行协商.
- P 表示逻辑程序.

例如,一个运输公司面对两个运输项目.大项目要求有 3 台卡车同时工作,小项目要求两台卡车同时工作.如果调度能提前知道 3 台卡车可以及时返回,公司就可以承接大项目,如果调度能提前知道两台卡车可以及时返回,则公司就可以承接小项目,否则,公司可能要放弃这两个项目.假定这个决策要在 3 天(72 小时)之内完成.

这个问题的推测框架 $SF_{MS}=\langle \Sigma,\beta,\Delta,SR,L,Tsc,P\rangle$.其中:$\Sigma=\{M,a,b,c\}$,$M$ 表示调度,a,b 和 c 分别表示 3 个卡车司机;$\beta=\{\text{free}\}+\{\text{-}\},\{\text{free}\}$ 是可问谓词集,$\{\text{-}\}$ 是应答谓词集,它们和协商语言 L 有关;$\Delta=\{\text{free}(a\_truck)\geqslant a,$

free(b_truck)$\geqslant b$, free(c_truck)$\geqslant c$};SR={$a\_truck\leqslant a$,$b\_truck\leqslant b$,$c\_truck\leqslant c$}+{},M 相信 3 台卡车可以从 3 个司机那里获得;Tsc=72,Tsc 表示 72 小时;P={规则}.

2 具有时间约束的资源协商框架的形式定义

协商是推测计算中的关键环节.在协商过程中,来自仆 Agent 的应答在 Tsc 之内是有效的,而迟到的应答不能再改变推测结果.例如,如果货运公司在 Tsc 之内已经决定承接小项目,那么,Tsc 以后的应答就不能改变这个决策结果.文献[5]定义了一个协商框架,但这个框架没有进一步协商过程.我们在这里扩充这个协商框架.

定义 **1**(对话移动(**dialogue move**)). 一个对话移动[4]是形式 send(A_{from},Content,A_{to},T)的一个实例,其中 A_{from} 是移动的发出者,A_{to} 是移动的接收者,Content 是移动的内容,T 是对话的剩余时间,随 Tsc 变化.

定义 **2**(内容原语模式). 对话移动中的内容原语是具有特定含义的基本动作.核心原语 give(R)表示提供资源 R,它是原语集的基本元素,即 MP={give(R)}.我们对原语集 MP 扩充如下:

ask(give(R))$\in MP$ 表示请求资源 R.propose(give(R'),give(R))$\in MP$ 表示资源提供者给资源需求者的一个资源替换建议,即用 R'代替 R.当资源请求被拒绝的理由充分时,exchange(R',give(R))$\in MP$ 表示资源需求者给资源提供者的一个资源替换建议,即用 R'代替 R.当资源请求被拒绝的理由不充分时,advise(give(R))$\in MP$ 表示资源请求者给资源提供者的一个劝告.

如果 content$\in MP$,则 accept(content)$\in MP$ 表示接受 content,refuse(content)$\in MP$ 表示拒绝 content,accept 和 refuse 被称为终止原语,why(content)表示为拒绝 content 请求一个理由,justify(content,reason)$\in MP$ 表示为拒绝 content 提供理由 reason,reason 是一个合取公式.图 1 给出了对话移动中原语之间的应答关系.$\forall$content$\in MP$, content 就是一个内容原语模式,它对应于从树根到某节点的一条路径.例如,ask-refuse-why-justify 是一条路径,则原语模式 justify(why(refuse(ask(give(R)))),reason)$\in MP$.另外,还有一个特殊的原语 urgency,该原语用于督促还没有应答的仆 Agent,但它没有应答原语.后面将给出一个督促算法.

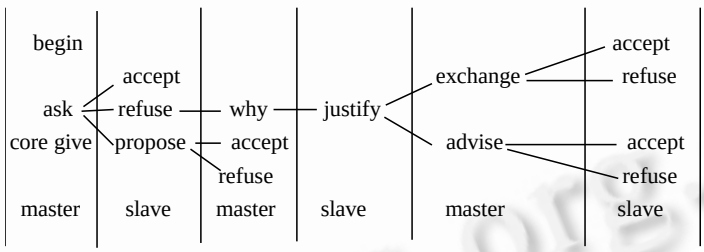

Fig.1 Responding relationship tree of dialogue move primitives

图 1 对话移动原语应答关系树

定义 **3**(资源协商语言). 资源协商语言 L 是一个对话移动的集合.对给定的 L,我们定义两个移动子集:$IMove$(L),$FMove$(L)$\subseteq L$,它们分别表示初始移动集和终止移动集.每个终止移动要么导致成功的对话,要么导致失败的对话.

L={ send(X,ask(give(R)),Y,T), send(X,accept(content),Y,T), send(X,refuse(content),Y,T),
send(X,why(content),Y,T), send(X,justify(content,reason),Y,T),
send(X,propose(give(R'),give(R)),Y,T), send(X,exchange(R',give(R)),Y,T),
send(X,advise(give(R)),Y,T), send(X,urgency,Y,T)}

$IMove$(L)={send(X,ask(give(R)),Y,T)}

$FMove$(L)={ //导致成功对话的对话移动
send(X,accept(ask(give(R))),Y,T), send(X,accept(propose(give(R'),give(R))),Y,T),
send(X,accept(exchange(R',give(R))),Y,T), send(X,accept(advise(give(R))),Y,T),
//导致失败对话的对话移动

send(X,content,Y,0) //表示一个超时对话

send(X,refuse(ask(give(R))),Y,T), send(X,refuse(propose(give(R'),give(R))),Y,T),

send(X,refuse(exchange(R',give(R))),Y,T), send(X,refuse(advise(give(R))),Y,T)}.

我们假设所有的 Agent 都使用这个协商语言,但是,从图 1 可以看出,M 和仆 Agent 具有不同的原语.send(A_{from},Content,A_{to},T)的一个实例可以表示为 $m_i(T)$,i 表示对话中移动的序号,一个协商对话是一个实例的序列,例如,$d=\{m_0(T_0),m_1(T_1),\ldots,m_n(T_n)\}$就是一个协商对话.为了方便,$Move^{in}(ag)$表示以 ag 为接收者的所有移动的集合,$Move^{out}(ag)$表示以 ag 为发出者的所有移动的集合.

定义 **4**(对话约束(**dialogue constraint**)). 给定一个 Agent 集Σ和一个协商语言 L,$\forall ag\in\Sigma$,对 ag 的一个对话约束是一个 if-then 规则:$m_i(T_1)\wedge C\Rightarrow m_{i+1}(T_2)$.其中:(1) $T_1>T_2$;(2) $m_i(T_1)\in Move^{in}(ag)$并且 $m_{i+1}(T_2)\in Move^{out}(ag)$;(3) $m_i(T_1)$的发出者是 $m_{i+1}(T_2)$的接收者,同样,$m_i(T_1)$的接收者是 $m_{i+1}(T_2)$的发出者;(4) C 是一个合取文字公式,如果 $K_{ag,T1}\wedge T_1>T_2|=C$,则约束满足,可进行 $m_{i+1}(T_2)$,其中 $K_{ag,T1}$ 表示 ag 在 T_1 时刻的知识库.

Agent ag 的一个对话约束表达了它和其他 Agent 进行交互的协议,其形式为 $m_i(T_1)\wedge C\Rightarrow m_{i+1}(T_2)$,它的直接意义是:在一个对话中,如果某个 Agent bg 在时间 T_1 向 ag 发出一个移动 $m_i(T_1)$,则 ag 的对话约束被启动,如果条件 C 在 ag 的知识库中成立,那么 ag 将在时间 T_2 向 bg 发出一个移动 $m_{i+1}(T_2)$.下面是一个对话约束的例子:

send(bg,ask(give(R)),ag,T_1)$\wedge$have(R,T_1)$\wedge\sim$need(R,T_1)$\Rightarrow$send(ag,accept(ask(give(R))),bg,T_2).

其中,C=have(R,T_1)$\wedge\sim$need(R,T_1).在两个 Agent 之间的对话中,相邻的两个对话移动必须满足这样的对话约束.

定义 **5**(终止对话(**termination of dialogue**)). 给定一个对话 $d=\{m_0(T_0),m_1(T_1),\ldots,m_n(T_n)\}$,其中 $n\geq 0$,$T_0>T_1>\ldots>T_n$.如果 bg 发出 $m_n(T_n)$,而 ag 没有移动可以发出,则对话 d 是一个终止对话.也就是说,终止对话满足条件:$m_n(T_n)\in FMove(L)\vee T_n=0\vee K_{ag,Tn}|\neq C$.

定义 **6**(对话协议(**dialogue protocol**)). 对话协议是一个对话约束集,表示为 DP.$\forall i\geq 0$,如果 Agent x 向 y 发出 $m_i(T_i)$,其中 $x\in\{a,b\}$,则 Agent y 向 x 发出 $m_{i+1}(T_{i+1})$,其中 $y\in\{a,b\}-\{x\}$.它们必须满足一个约束 c,即$\exists c\in DP$,$body(c)\Rightarrow head(c)$,使得 $K_y\cup\{m_i(T_i)\}|=body(c)$并且 $head(c)=m_{i+1}(T_{i+1})$.

3 推测计算中资源协商的实现

3.1 进程约简阶段和事实到达阶段

推测计算实际上是一个 APP(abductive proof procedure)[1−3].该过程包括两个阶段:进程约简阶段(process reduction phase)和事实到达阶段(fact arrival phase).进程约简阶段实际上是一个 Agent 对程序的执行阶段,它和 IFF 证明过程(IFF proof procedure)[6,7]是一样的.事实到达阶段是一个中断处理阶段.这两个阶段在 M 中构成了观察-思考-动作循环(observe-think-act Agent cycle)[8].在这个循环中,M 可以按照当前应答取消一些不可能的进程,也可以从悬挂进程集中恢复一些进程,或者建立一些新进程.推测计算重复执行这两个阶段.推测计算中有两种进程:活动进程(active process)和悬挂进程(suspended process).当应答没有到达,或应答和缺省一致时,活动进程就以缺省值进行工作.当应答和缺省不一致时,就会有一些不可能的进程被取消.如果一个基可问文字 F 的缺省值是“no”,并且还没有得到确认,则该文字对应的活动进程将被悬挂.为此,推测计算不采用逻辑程序设计中的“负则失败(negation as failure)”策略,而是采用“负则悬挂(negation as suspension)”策略.

进程约简阶段的步骤如下:

Step 0. 设置初始目标集和时间约束,初始化相关变量,启动包含初始目标集的活动进程.

Step 1. 如果 Tsc 等于 0,则整个计算结束.否则,当活动进程集中有一个进程的目标集为空时,则计算临时停止,等待来自仆 Agent 的应答,输出当前决策方案和未被确认的缺省值.新的应答会使计算进入下一个循环.M 会在临时等待时督促未应答的仆 Agent.

Step 2. 从活动进程集中选择一个活动进程,并从该进程的目标集中选择一个扩展文字(extended literal)F.

Step 3. 处理扩展文字 F

Case 1. F 是一个正文字.用 unfolding 规则[6−8]约简当前目标.

Case 2. F 是一个基非可问负文字(ground non-askable negative literal).产生以失败文字 fail(B)为目标的活动进程.该进程不会失败,但可能被悬挂.

Case 3. F 是一个 fail(B)文字.处理 B 中的正文字、基负文字和基可问文字,更新活动进程集.

Case 4. F 是一个基可问文字.在这里,M 和其他仆 Agent 协商来确认缺省值或否定缺省值,修正信念.

Endcase

事实到达阶段的步骤如下:

Step 0. 记录来自仆 Agent 的应答($Q\preccurlyeq S$).

Step 1. 设 $F=Q\succcurlyeq S$,如果$\sim F\in\Delta$,则执行下列步骤:

Step 1.1. 修正Δ.

Step 1.2. 更新活动进程集.

Step 1.3. 更新悬挂进程集.

3.2 主Agent信念修正过程

在推测计算框架中,Δ是一个缺省集,也称为初始信念.对于$\forall d\in\Delta$,d 只能是包含可问谓词的正文字或负文字,也称为信念文字.当应答和缺省不一致时,执行信念修正过程[1].假设 APS 表示活动进程集,SPS 表示悬挂进程集,GS 表示待证实的扩展文字集(目标集),TD 表示缺省为"真"的可问文字集,ANS 表示初始目标中变量代换集,FD 表示缺省为"假"的可问文字集.用三元组$\langle GS,TD,ANS\rangle$表示活动进程,四元组$\langle GS,TD,FD,ANS\rangle$表示悬挂进程.

信念修正过程(belief revision process,简称 BRP)如下:

说明:当接收应答 $Q\preccurlyeq S$ 时,F 表示基可问文字 $Q\succcurlyeq S$.如果$\sim F\in\Delta$,则执行下列算法.SR_M 和 SR_{un} 随Δ而变化.

算法 **1**.

Step 0. $\Delta=\Delta-\{\sim F\}\cup\{F\}$

Step 1. 处理资源文字集

如果 F 是正文字,则 $SR_M=SR_M\cup\{R\preccurlyeq S|R\preccurlyeq S\in SR_{un}\}$,$SR_{un}=SR_{un}-\{R\preccurlyeq S\}$

如果 F 是负文字,则 $SR_{un}=SR_{un}\cup\{R\preccurlyeq S|R\preccurlyeq S\in SR_M\}$,$SR_M=SR_M-\{R\preccurlyeq S\}$

//根据 F 的正负,从 SR_{un} 到 SR_M 转移资源文字或者从 SR_M 到 SR_{un} 转移资源文字

Step 2. $APS'=APS-\{\langle GS,TD,ANS\rangle\in APS|\sim F\in TD\}\cup\{\langle GS,TD\cup\{F\},ANS\rangle|\langle GS,TD,\{F\},ANS\rangle\in SPS\}$

//从 APS 中去掉和 F 不一致的进程,添加和 F 一致的进程

Step 3. $SPS=SPS\cup\{\langle GS,TD-\{\sim F\},\{\sim F\},ANS\rangle\ |\langle GS,TD,ANS\rangle\in APS$ and $\sim F\in TD\}-\{\langle GS,TD,FD,ANS\rangle\in SPS|F\in FD\}\cup\{\langle GS,TD\cup\{F\},FD-\{F\},ANS\rangle|\langle GS,TD,FD,ANS\rangle\in SPS$ and $FD\neq\{F\}\}-\{\langle GS,TD,FD,ANS\rangle\in SPS|\sim F\in TD\}\cup\{\langle GS,TD-\{\sim F\},FD\cup\{\sim F\},ANS\rangle|\langle GS,TD,FD,ANS\rangle\in SPS$ and $\sim F\in TD\}$

//悬挂 TD 和 FD 与 F 不一致的进程

Step 4. $APS=APS'$ //更新 APS

Step 1 按照Δ的变化更新资源集.Step 2 更新 APS,但更新内容暂存 APS'.Step 3 更新 SPS.在 Step 4 中,APS 获得新内容.

3.3 进程约简阶段中的资源协商算法

推测计算中协商的目的是为了获得更多的真实信息.这些信息可能和缺省信念一致,也可能不一致,在不一致的情况下,需要执行 BRP 过程.进程约简阶段是 M 的对话移动发出阶段,根据仆 Agent 的应答,有时需要作进一步协商.为了表达方便,我们引入如下符号:$Sject$ 表示 M 发出的对话移动所对应的可问文字集,即 $Sject=\{\text{Content}\succcurlyeq b|\text{send}(M,\text{Content},b,T)\in Move^{out}(M)\}$,$Rject$ 表示 M 接收的对话移动所对应的应答文字集,即 $Rject=\{\text{Content}\preccurlyeq b|\text{send}(b,\text{Content},M,T)\in Move^{in}(M)\}$,$Stime$ 表示形如 $T@Obj$ 的时间文字集,$T@Obj$ 表示 M 在时间 T 向 Obj 发出了一个对话移动,即 $Stime=\{T@Obj|\text{send}(M,\text{Content},Obj,T)\in Move^{out}(M)\}$.

在进程约简阶段,内容原语被看作意义相同的谓词,分别归属于$\beta_{askable}$和$\beta_{response}$,核心原语 give 称为核心谓

词.give(R)和~give(R)称为核心文字,信念文字主要由核心文字构成.

有两个时间表示符号:T 和&T.为了使所有 Agent 知道时间的变化,对话移动中使用时间符号&T.例如,如果 Agent X 发出 send(X,ask(give(R)),Y,&T)且 T=3,当 Y 接收到这个移动时,则 T<3 且 Y 能知道时间的变化.如果 Agent X 发出 send(X,ask(give(R)),Y,T)且 T=3,当 Y 接收到这个移动时,则 T=3,且 Y 不知道时间的变化.时间符号&T 的目的是使仆 Agent 能尽快地应答.实际上,如果通信延迟,那么当 Y 接收到移动时,时间符号 T 就不能准确地表示剩余时间.

算法中的一些符号说明:*QName* 表示 Q 中最外层谓词的名称,*QMove* 表示最外层谓词的内容参数,*Qgive* 表示 Q 中的核心文字,$RP1$ 和 $RP2$ 分别表示 Q 中的主资源和替代资源.

下面给出进程约简阶段 M 和仆 Agent 的协商算法.

说明:处理形如 $Q\succcurlyeq S$ 的文字.$\langle GS,TD,ANS\rangle$是当前进程,且 $APS=APS-\{\langle GS,TD,ANS\rangle\}$,$GS=GS-\{Q\succcurlyeq S\}$.*Tsc* 按照时钟进行倒计时.

算法 **2**.

Case 4. F 是基可问文字

Case 4.1. *QName* 是 give 谓词,处理核心谓词

如果 ask(Q)$\succcurlyeq S\notin$*Sject*,则 send(M,ask(Q),S,&*Tsc*),*Sject*=*Sject*$\cup\{$ask(Q)$\succcurlyeq S\}$,*Stime*=*Stime*$\cup\{$*Tsc*@$S\}$

如果 $F\in TD$,则 $APS=APS\cup\{\langle GS,TD,ANS\rangle\}$,否则 //和缺省一致,继续计算

如果 $F\notin TD$ 且 $F\in\Delta$,则 $APS=APS\cup\{\langle GS,TD\cup\{F\},ANS\rangle\}$,否则 //保持缺省和信念的一致,计算继续

如果~$F\notin TD$ 且~$F\in\Delta$,则 $SPS=SPS\cup\{\langle GS,TD,\{F\},ANS\rangle\}$ //和缺省及信念不一致,出现悬挂进程

Case 4.2. *QName* 是 accept 谓词,处理来自仆 Agent 的资源替换建议,对应 propose 谓词

test($RP2$)检查 $RP2$ 对 M 的合理性 //检查替换资源的合理性

如果 $K_M|$=test($RP2$)且 $Q\succcurlyeq S\notin$*Sject*,则

send(M,Q,S,&*Tsc*),$RF=RF\cup\{$give($RP2$)$\preccurlyeq S\}$,*Stime*=*Stime*$\cup\{$*Tsc*@$S\}$

Sject=*Sject*$\cup\{Q\succcurlyeq S\}$ //接受资源替换建议

如果~give($RP1$)$\in\Delta$,则 BRP,$SR_M=SR_M-\{RP1\preccurlyeq S\}\cup\{RP2\preccurlyeq S\}$ //信念修正和资源更新

如果 $K_M|\neq$*test*($RP2$)且 refuse(*QMove*)$\succcurlyeq S\notin$*Sject*,则

send(M,refuse(*QMove*),S,&*Tsc*) //拒绝资源替换建议

$RF=RF\cup\{$~give($RP1$)$\preccurlyeq S\}$,*Sject*=*Sject*$\cup\{$refuse(*QMove*)$\succcurlyeq S\}$,*Stime*=*Stime*$\cup\{$*Tsc*@$S\}$

如果 *give*($RP1$)$\in\Delta$,则 BRP

Case 4.3. 处理 why 谓词、exchange 谓词和 advise 谓词 // 做进一步协商处理

如果 $Q\succcurlyeq S\notin$*Sject*,则 send(M,Q,S,&*Tsc*),*Sject*=*Sject*$\cup\{Q\succcurlyeq S\}$, *Stime*=*Stime*$\cup\{$*Tsc*@$S\}$

Endcase

M 和仆 Agent a_i 之间的每个对话移动必须满足对话协议 *DP*,即对于给定的移动 $m\in$*Sject*,则 $\exists m'\in$*Rject*,$\exists c\in DP$,使得 $K_{ai}\cup\{m\}|=$*body*(c)且 *head*(c)=m'.在 Case 4.1 中,如果 F 的真值还没有确认,且~$F\in\Delta$,则悬挂相应进程.如果应答和悬挂文字一致,则悬挂进程被恢复为活动进程.在 Case 4.2 中,propose 是一个协商行为,仆 Agent 是理性的,它利用这个原语只对 ask(give(R))提出建议,不对 ask(~give(R))提建议,这个约束在 *DP* 中表达.如果 $RP2$ 对 M 是合理的,则 give($RP2$)$\preccurlyeq S$ 就是一个应答,否则,~give($RP1$)$\preccurlyeq S$ 是一个应答.从 SR_M 的变化,我们能看出 $RP1$ 和 $RP2$ 之间的交换.另外,一个对话可能在这里结束,所以,协商结果和信念不一致时要执行 BRP.在 Case 4.3 中,做进一步协商.当资源请求受到拒绝时,M 将选择合适的原语做进一步协商.在我们讨论的协商环境中,M 处于主控地位,仆 Agent 处于从属地位,从图 1 中也可以看到这一点.

3.4 事实到达阶段中的资源协商算法

M 在事实到达阶段接收来自仆 Agent 的应答,并且按照应答修正信念.

在事实到达阶段:

说明:处理形如 $Q\preccurlyeq S$ 的文字.RF'表示对核心谓词拒绝的负文字集,RF 表示对核心谓词的最终应答文字集.

算法 **3**.

$Rject=Rject\cup\{Q\preccurlyeq S\}$

Case 1. 处理 accept(*QMove*)或 refuse(*QMove*)

如果 refuse(*QMove*) 则

如果 *Qgive* 是正文字,则 //只对正文字的拒绝处理进一步协商

如果 $S\notin\{S|Q\preccurlyeq S\in RF'\}$ 则 $APS=APS\cup\{\langle\{\text{why}(Q)\succcurlyeq S\},TD,ANS\rangle\},RF'=RF'\cup\{\sim Qgive\preccurlyeq S\}$

$Q=\sim Qgive,RF=RF\cup\{\sim Qgive\preccurlyeq S\}$ //记录应答结果

否则

$Q=Qgive,RF=RF\cup\{Qgive\preccurlyeq S\}$

如果$\sim Q\in\Delta$ 则 BRP //信念修正

Case 2. 处理资源交换 propose(*QMove*)

$APS=APS\cup\{\langle\{\text{accept}(Q)\succcurlyeq S\},TD,ANS\rangle\}$ //为 propose(*QMove*)建立接收进程

Case 3. 处理对 *QMove* 拒绝的原因 justify(*QMove*,*s*),根据原因 *s* 做进一步协商

如果原因 *s* 是充分的,且存在一个内部资源 R',该资源不优于 $RP1$,则 $APS=APS\cup\{\langle\{\text{exchange}(R',\text{give}(RP1))\succcurlyeq S\},TD,ANS\rangle\}$,如果原因 *s* 不充分,则 $APS=APS\cup\{\langle\{\text{advise}(\text{give}(RP1))\succcurlyeq S\},TD,ANS\rangle\}$

Endcase

当 *M* 接收到 refuse(*QMove*),有两种处理方法:一种处理方法是接受这种拒绝,更新活动进程集和悬挂进程集.但这不是本文的目的.由于我们的协商环境是一个不平等环境,*M* 处于主导地位,所以,还有另一种处理方法,就是在时间允许的情况下做进一步协商.在图 1 中,从 ask 到 exchange 和 advise 有两条路径,这两条路径就是对 refuse 的进一步协商.为了进一步协商,第 1 次拒绝应答记录在 RF'中,并且修正Δ中相应的信念.另外,原语 exchange 和 advise 只用于正谓词,所以原语 why 只在正核心谓词受到拒绝时,才开始进入进一步协商过程.

设 $RF_{set}=\{Q\succcurlyeq S|Q\preccurlyeq S\in RF\}$,$\Delta-RF_{set}$ 是未被确认的信念集,$|\Delta-RF_{set}|/|\Delta|\times100\%$称为未被确认的信念率,缩写为 UNBR(UN-decided belief rate).很显然,UNBR 越高,推测计算结果的风险越大.RF 只记录对核心谓词的最终应答.当 ask(give(*RP*1))第 1 次被拒绝时,修正相应信念,建立一个新的活动进程以便做进一步协商.在 Case 2 中处理来自仆 Agent 的资源替换建议,在 Case 3 中,根据资源请求受到拒绝的原因做进一步协商.

3.5 资源协商中对仆**Agent**的督促算法

当 *M* 向某个仆 Agent 发出一个移动,并且在一定时间内没有得到应答,那么 *M* 将在空闲时间对所有没有应答的仆 Agent 执行督促算法.

下面给出处理对仆 Agent 的督促(processing urgency to urge slave Agents,简称 PUUSA)算法.

说明:δ是一个时间间隔(time interval),它的值在进程约简阶段的 Step 0 中给出.如果 *M* 发出一个对话移动给一个仆 Agent,δ过后,*M* 还没有接到来自那个仆 Agent 的应答,则 *M* 将发出 send(...,urgency,…,&*Tsc*)给该仆 Agent 以督促它.*M* 在临时等待时执行这个督促算法.*Uslave* 表示已经接收到一次督促的仆 Agent 集.

算法 **4**.

Step 0. 产生一个仆 Agent 集 $SlaveSet1=\{S|\forall Q\succcurlyeq S\in Sject \text{ and } T@S\in Stime \text{ and } T-Tsc\geq\delta\}$,从 *M* 在时间 *T* 向这个集合中的仆 Agent 发出资源请求开始,到此时(*Tsc*)的时间间隔已经大于或等于δ.

Step 1. 产生一个没有应答的仆 Agent 集合 $SlaveSet2=\{S|\forall S\in SlaveSet1 \text{ and } S\notin\{S|\forall Q\preccurlyeq S\in RF\} \text{ and } S\notin Uslave\}$.如果 $SlaveSet2=\varnothing$,退出算法.

Step 2. $\forall S\in SlaveSet2$,send(*M*,urgency,*S*,&*Tsc*),$Uslave=Uslave\cup S$.

M 在临时等待时执行 PUUSA.一旦有新的应答,终止 PUUSA,并启动事实到达阶段的算法.如果在这个临时等待时间内不能对全部没有应答的 Agent 督促,可以到下一个临时等待时间继续进行.每个未应答的仆 Agent 只被督促一次,如果已督促过的仆 Agent 仍没有应答,则视第 1 次拒绝为最终应答.

4 实 验

我们模拟了一个货物运输公司的调度环境,并在这个环境中对上述算法做了 3 个实验.在不同的时间随机抽取了 3 组卡车参与实验.前期工作平台[9]跟踪公司卡车的运行情况和它们的返回概率.以文献[9]的工作为基础确定参与实验卡车的信念.随机选择的第 1 组数据不满足进一步协商的条件,所以,实验结果和文献[3]的推测计算是一样的,如图 2 所示.从图 3 和图 4 可以看出,具有进一步协商的推测计算可以降低 UNBR,从而降低推测结果的风险,比文献[3]的结果要好.图 2~图 4 显示的结果分别是在 72 小时内完成的.如图 5 所示,Tsc 越大,UNBR 越低.在实验中,也遇到一些意外因素,需要人为来辅助.

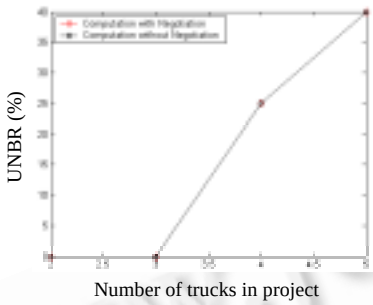

Fig.2 The first group of experiment

图 2 第 1 组实验

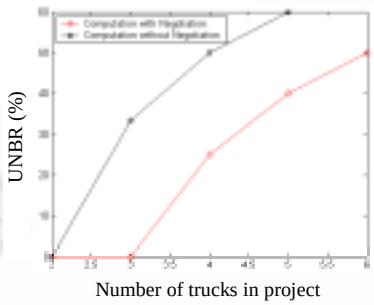

Fig.3 The second group of experiment

图 3 第 2 组实验

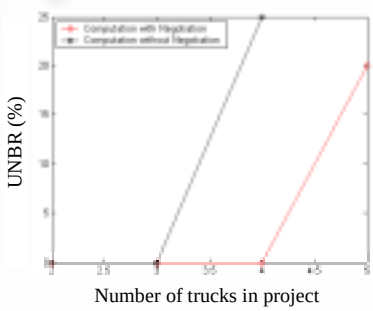

Fig.4 The third group of experiment

图 4 第 3 组实验

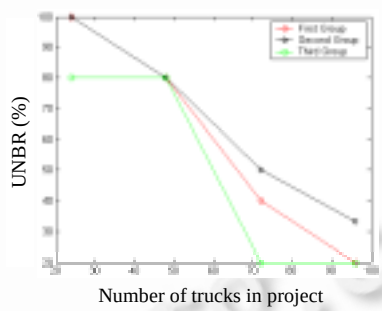

Fig.5 Variety of three groups of experiments with time

图 5 3 组实验随时间的变化

在实验中,我们还对比分析了保持信念不变的情况,即不进行信念修正时,进行进一步协商的过程.在这个过程中,我们引入了一个处理滞留应答的环节,就是说,当 *M* 对仆 Agent 提出的资源请求受到拒绝时,信念依然不变,但要记录被拒绝的结果,进入进一步协商过程.进一步协商的结果有 3 种:(1) 仆 Agent 同意提供资源;(2) 仆 Agent 依然不同意提供资源;(3) 仆 Agent 没有应答.对于第 1 次拒绝,*M* 将利用 why 原语向仆 Agent 询问原因,对原因进行分析,利用领域原因规则集来决定放弃(最终拒绝),或采用 exchange 交换资源,或采用 advise 规劝.在情况(1),撤销被拒绝的结果,保持当前决策方案;在情况(2),将拒绝结果作为最终应答,修正决策方案;在情况(3),将第 1 次拒绝结果作为最终应答,在处理滞留应答中修正决策方案.当 *Tsc*=0 时,处理滞留应答开始,事实到达阶段结束,这时由于不再接收外界的应答,所以处理滞留应答效率很高.从实验中可以发现,及时修正信念比保持信念不变有如下几个特点:(1) 及时修正信念保证了结果的时效性,但悬挂进程较多;(2) 及时修正信念增强了 M 的学习能力和协商能力,也提高了结果的准确性;(3) 在一个协商对话中,信念修正可以提高协商效率,缩短对话进程;(4) 及时修正信念允许仆 Agent 多次反悔,而保持信念不变只允许仆 Agent 一次反悔.

5 结 论

推测计算就是针对信息不完全的多 Agent 场景进行求解的技术.本文在文献[10]的基础上做了较深入的工作.在文中,我们为推测计算添加了时间约束,为主 Agent 增添了学习功能(信念修正),在推测计算的基本理论和方法的基础上,提出了在时间约束下的推测计算框架,并在这个框架中嵌入了具有时间约束的进一步协商算法.进一步协商框架包括进一步协商原语(例如 why,justify),资源交换原语(例如 propose,exchange)以及劝说原语(例如,advise)等.最后,在模拟货物运输公司的调度环境中做了实验.从实验结果可以看出,推测计算通过进一步协商可以降低计算结果的风险,比文献[3]的结果要好;实验也证明了资源协商算法的有效性;通过实验对比还可以看出,具有信念修正的推测计算结果具有很好的时效性和准确性,信念修正增强了 M 的学习能力和协商能力.

致谢 在此,我们向对本文工作给予支持的同行,尤其对郑州大学信息工程学院的柴玉梅老师表示衷心感谢.

References:

[1] Satoh K, Yamamoto K. Speculative computation with multi-agent belief revision. In: Arabnia HR. Proc. of the 1st Int'l Joint Conf. on Autonomous Agents and Multi-Agents Systems. New York: ACM Press, 2002. 897−904.

[2] Hayashi H, Cho K, Ohsuga A. Speculative computation and action execution in multi-Agent system. Electronic Notes in Theoretical Computer Science, 2002,70(5):1−14.

[3] Satoh K, Inoue K, Iwanuma K, Sakama C. Speculative computation by abduction under incomplete communication environments. In: Proc. of IEEE Int'l Conf. on Multi-Agent Systems. Boston: IEEE Press, 2000. 263−270.

[4] Sadri F, Toni F, Torroni P. Logic Agents, dialogues and negotiation: An abductive approach. In: Schroeder M, Stathis K, eds. Proc. of the Conf. on Artificial Intelligence and Simulation of Behavior (AISB 2001). York, 2001. 31−38.

[5] Sadri F, Toni F, Torroni P. Dialogues for negotiation: Agent varieties and dialogue sequences. In: Maher JJ, ed. Intelligent Agents VIII. LNAI 2333, Heidelberg: Springer-Verlag, 2002. 405−421.

[6] Sadri F, Toni F. Abduction with negation as failure for active and reactive rules. In: Lamma E, Mello P, eds. Proc. of the AI*IA 99, the 6th Congress of the Italian Association for Artificial Intelligence. LNAI 1792, Heidelberg: Springer-Verlag, 2000. 49−60.

[7] Fung TH, Kowalski RA. The IFF proof procedure for abductive logic programming. Journal of Logic Programming, 1997,33(2): 151−165.

[8] Kowalski A, Sadri F. From logic programming to multi-Agent systems. Annals of Mathematics and Artificial Intelligence, 1999,25(3-4):391−419.

[9] Wang LM, Chai YM, Huang HK. Model for distributed data mining based on multi-Agent. Computer Engineering & Application, 2004,40(9):197−199 (in Chinese with English abstract).

[10] Wang LM, Huang HK: Multi-Agent resource negotiation in speculative computation. Journal of North Jiaotong University. 2004,28(2):17−21 (in Chinese with English abstract).

附中文参考文献:

[9] 王黎明,柴玉梅,黄厚宽.基于多 Agent 的分布式数据挖掘模型.计算机工程与应用,2004,40(9):197−199.

[10] 王黎明,黄厚宽.推测计算中的多 Agent 资源协商.北方交通大学学报,2004,28(2):17−21.