doi:10.13582/j.cnki.1672-7835.2022.05.005

基于改编命题动态逻辑的 Agent 交互协议推理

张晓君,邱君
(安徽大学 哲学学院,安徽 合肥 230039)

摘　要:Agent 交互协议是“基于消息交流的 Agent 之间的”协调说明,用于提供交互的背景。交互协议是说明“所有参与 Agent 都应该遵守”的交互规则的公共文件。对命题动态逻辑进行改编得到的 L_α 逻辑,可以对多 Agent 系统的交互协议进行建模。为此,需要建立第一类协议说明语言,该语言是“把行为限制为由信道发送的消息的”行动语言,既可对协议结果进行表示和推理,也可对消息之外的行动进行推理,其约束语言用于说明消息和状态;每条消息被显式地定义为“对共享社会状态的处理”。

关键词:第一类协议;约束;Agent;交互协议
中图分类号:B81　　**文献标志码**:A　　**文章编号**:1672-7835(2022)05-0035-10

Agent(主体或智能体)是人类智能、动物智能和机器智能的统一模型①,是通过传感器感知环境,并且借助执行器作用于环境的实体,可定义为从感知序列到实体动作的映射②,具有自寻优、自适应、自解释和自学习的功能,符合人工智能、互联网和万维网中的并行处理和分布式处理等技术(如分布式人工智能)的要求,因此,Agent 和多 Agent 系统的研究成为人工智能等相关领域研究的重点。

为了发挥 Agent 的智能性并增强其适应性,对交互协议(interaction protocols)进行建模,使得运行主体之间的交互协议可以相互引用、检查、组合、共享和调用,这类协议叫作“第一类协议(first-class protocol)”。本文将在 Harel 等(2000)③、Miller 与 McBuney(2007④、2011⑤)、郝一江(2022)⑥等的基础上,利用改编的命题动态逻辑(Propositional Dynamic Logic,简称 PDL)——L_α 逻辑,建立 Agent 第一类交互协议推理模型。

一　问题的提出

对多 Agent(主体)系统交互协议的研究主要集中在交互协议的文档研究,主要描述主体参与协议的可能交互集合,并且使用协议说明对主体交互进行硬编码,其主要缺点有:(1)强行将主体及其使用的协议结合;协议一旦改变,主体代码必须随之改变;(2)主体只能使用设计时已知的协议进行交互;(3)主体在运行时不能对协议进行组合,无法实现更复杂的交互。这三点都有悖于“对主体的自寻优、自适应和自学习的”要求。

如果主体可以不使用硬编码的决策机制来选

收稿日期:2022-06-03
基金项目:国家社会科学基金后期项目(20FZXB037);安徽省高校科研重点项目(ZK2021A0023)
作者简介:张晓君(1970—),女,四川南充人,博士,教授,博士生导师,主要从事 Agent 理论和人工智能逻辑的研究。

①张晓君:《等级 BDI 逻辑:关于行为表征的柔性逻辑》,《哲学动态》2013 年第 1 期。

②蔡自兴,徐光佑:《人工智能及其应用(第 4 版)》,清华大学出版社 2010 年版,第 313 页。

③Harel D, Kozen D, Tiuryn J. *Dynamic Logic*. Cambridge: The MIT Press, 2000.

④Miller T, Mcburney P. “Using constraints and process algebra for specification of first-class agent interaction protocols”. *Engineering Societies in the Agent World VII*, *Vol.* 4457 *of LNAI*, Berlin/Heidelberg: Springer, 2007: 245-264.

⑤Miller T, Mcburney P. “Propositional dynamic logic for reasoning about first-class agent interaction protocols”. *Computational Intelligence*, 2011, 27(3): 422-457.

⑥郝一江:《动态逻辑的双重性程序与结构化》,《贵州工程应用技术学院学报》2022 年第 2 期。

择下一步,而是在运行时通过检查协议说明来选择下一步,就可以增加其灵活性。这要求第一类协议存在于多主体系统的文档中,而硬编码协议仅作为参与者发送的消息中的抽象存在。协议说明不只是任意的标记序列,每条消息还显式地表征"对共享社会状态的处理"①,比如:主体之间的一组义务。

过程代数②和 π-演算③已经被用来模拟过程及其交互。而过程的结合可以形成协议说明的基础,但是这些语言没有状态的概念,因此不能表示协议的意义,而意义对于面向目标的主体而言是很重要的。有几种语言已经成功用于第一类协议说明,这些语言的基础各不相同,例如:基于 Petri 网④、基于声明性说明语言⑤、基于轻量协调演算⑥,等等。Eijk 等(2003)⑦使用约束系统(constraint system)和过程代数对多主体系统进行建模,其验证框架侧重于主体之间的交互,其逻辑与命题动态逻辑(PDL)相似,支持对结果的推理。

已经有一些学者利用命题动态逻辑 PDL 表示交互协议。例如:Paurobally 等(2005)⑧使用信念和意图模态算子对 PDL 进行扩展,从而定义了一种"能够对主体交互进行建模的"语言 PDL-BI。Brak 等(2004)⑨把 PDL 直接用于协议说明语言。Miller 与 McGinnis(2011)利用 PDL 对 RASA 语言构造的协议进行表示和推理,其中协议定义本身提供所允许的行为。

利用改编的命题动态逻辑(PDL)——L_α 逻辑,可以对第一类协议结果进行表示和推理。本文只讨论关于终止协议的推理,即有穷 L_α 逻辑。L_α 逻辑具有可靠性和完全性,而且该逻辑适用的模型限制为共享的社会状态,适用的程序限制为第一类协议,这样得到的极小系统的主体可以验证第一类协议的性质。对主体共享的社会状态而不是主体的心智状态进行建模,可以验证主体是否按照协议规则行事⑩。主体能够对其发送和接收的消息的效果进行推理,设计者执行面向目标的主体时就可以选择最能实现其目标的行为方式。能够对协议进行推理的主体可以在运行时学习新协议,从而使这些主体具有更强的适应性。

主体交互协议是"基于消息交流的 Agent 之间的"协调说明,用于提供交互的背景。这些协议定义了冲突规则,用于说明以下问题:谁能发送信息?发送内容是什么?什么时候发送?第一类协议是可引用的、可共享的、可操作的实体,在多主体系统中作为运行时值(runtime value)而存在。这类协议需要说明发送信息的主体、消息内容、发送信息的时间、发送消息的先决条件及其后置条件(即结果)。根据第一类协议说明,参与主体只需要知道该语言的句法和语义,就可以检查学习规则的说明和协议效果。交互协议是说明"所有参与主体都应该遵守"的交互规则的公共文件。可以把协议视为规则,并将主体的规划视为在这些规则内最好地实现其目标的策略。

消息的意义是指对共享社会状态的某种操作。例如,协议规则声明:接受签署专著出版合同,发送此消息意味着您承诺支付出版费用。共享状态表示先决条件,例如,主体只能"在该专著

①Miller T, Mcburney P. "Propositional dynamic logic for reasoning about first-class agent interaction protocols". *Computational Intelligence*, 2011, 27(3),p.422.

②Milner R. *A Calculus of Communicating Systems*. Secaucus: Springer-Verlag, 1980.

③Milner R. *Communicating and Mobile Systems: The Pi-Calculus*. Cambridge, UK: Cambridge University Press, 1999.

④Silva L P D, Winikoff M, Liu W. "Extending agent by transmitting protocols in open systems". Proceedings of the Challenges in Open agent Systems Workshop, Melbourne, Australia, 2003.

⑤Desai N, Singh M P. "A modular action description language for protocol composition". Proceedings of the 22nd Conference on Artificial Intelligence (AAAI), Menlo Park, CA: AAAI Press, 2007: 962-967.

⑥Robertson D. "Multi-agent coordination as distributed logic programming". Proceedings of the International Conference on Logic Programming, Vol. 3132 of LNCS, Berlin: Springer, 2004: 416-430.

⑦Eijk R M V, et al. "A verification framework for agent communication". Autonomous agent and Multi-agent Systems. 2003, 6(2): 185-219.

⑧Paurobally S, Cunningham J, Jennings N R. "A formal framework for agent interaction semantics". Proceedings of the 4th International Joint Conference on Autonomous agent and Multi-agent Systems, New York: ACM Press, 2005: 91-98.

⑨Brak R L, Fleuriot J D, McGinnis J. "Theorem proving for protocol languages". Proceedings of the European Union Multi-agent Systems Workshop, Barcelona, Spain, 2004.

⑩Singh M P. "A social semantics for agent communication languages". *Issues in Agent Communication*, *Vol.* 1916 *of LNCS*, Heidelberg: Springer-Verlag, 2000: 31-45.

不侵权的情况下”承诺出版。每个消息都有一个显式的先决条件和后置条件,这样主体就可以确定发送消息的结果;多个消息的结果可以组合地进行计算。为了使主体能够进行推理,必须以一种主体能够理解的方式加以说明。

为此,需要定义构成第一类协议说明语言①的如下四个性质。(1)形式化性质:形式化语言可以消除协议歧义,允许主体以值的形式传递和存储协议定义;(2)意义明显性:消息的意义必须由协议进行显式说明;(3)可检查性或机器可读性:主体在运行时能够对协议进行推理,通过对比不同协议的规则和效果以确定发送哪些消息才可以最好地实现目标;(4)动态可组合性:主体在运行时可以组合新的第一类协议来实现目标。

二　第一类协议说明语言

第一类协议说明语言是“可以对表示的信息进行约束的”行动语言,是作为所需要的算子最少的语言来设计的,这里给出其句法和外延语义。

(一)信息模型

为了使第一类协议说明语言尽可能广泛地应用,对消息(messages)中的信息(information)进行建模和对共享社会状态的操作,不受特定语言的限制。使用不同的基础语言(如描述逻辑②或约束语言③),利用约束就可以对信息进行建模。

定义 1(约束系统):一个约束系统是一个完备的代数格$\langle \Gamma, \looparrowleft, \uparrow, \min, \max, \Delta, ⇣\rangle$,其中,Γ是原子命题集,$\looparrowleft$是消息来源算子,↑是最小上界算子,min 是Γ中的最小元素,max 是Γ中的最大元素,Δ是变元的可数集,⇣是变元隐藏算子。

使用$\mathcal{L}$来表示该语言以及该语言中的所有约束组成的集合。蕴涵算子$\looparrowleft$对约束系统中的元素定义了一个偏序,使得 $a \looparrowleft b$ 表示 b 中的信息来源于 a,其中 $a, b \in \mathcal{L}$。a=b 是“$a \looparrowleft b$ 且 $b \looparrowleft a$”的缩写。在本文中,若无特别说明,a 和 b 表示信息,Δ表示变元的可数集,v 表示变元,φ和 ψ 表示约束,α和β表示协议,“$=_{def}$”表示“等号左边被等号右边定义”。

一个约束具有如下三种形式之一:(1)原子命题,例如 p=1,其中 p 是命题变元;(2)(最小上界的)合取,例如$\varphi \uparrow \psi$,其中φ和 ψ 都是约束;(3)隐藏约束$⇣_v \varphi$,其中φ是约束,$v \in \Delta$是变元。

允许对消息来源算子的右边进行否定:$c \looparrowleft \neg d$ 等值于$\neg(c \looparrowleft d)$,否定表示失败。其他命题算子的定义如常。使用 vars(φ)来表示出现在约束φ中的自由变元,即在φ中没有使用变元隐藏算子⇣来隐藏的变元。$⇣_{\tilde{v}} \varphi$表示:除了变元 v 之外,在约束φ中出现的所有变元都被隐藏,即$⇣_{\tilde{v}} \varphi = ⇣_{vars(\varphi) \backslash v} \varphi$。引入更名算子“/”,φ[v/u]表示:用 v 替换φ中的所有 u,即$\varphi[v/u] = ⇣_u((u=v)\uparrow\varphi)$。

(二)协议模型

第一类协议说明语言是一种行动语言,跨信道(across channel)发送的消息是“可以操纵共享社会状态的”行动。当主体通过信道发送消息时,利用协议中消息的定义,来通知其他参与者如何更新其状态副本。

定义 2(句法):第一类协议说明语言的句法定义如下。

(1)$\alpha ::= \varphi \rightarrow \in | \varphi \rightsquigarrow_{\psi}^{c(i,j)} \psi | \alpha; \beta | \alpha \cup \beta | var_v^{\psi} \cdot \alpha | \mathbb{D}(v)$;

(2)$\psi ::= a | \varphi \cup \psi | ⇣_v \psi$;

(3)$\varphi ::= \neg\varphi | \psi$。

其中 $a \in \mathcal{L}$是任意原子约束,α是协议,ψ 是非终止约束,而φ是约束和否定约束。由于约束系统将否定视为失败,因此,否定只有在先决条件下才被允许,而且不能用来断定社会状态。

使用形式为 $\psi \rightsquigarrow_{\psi_m}^{c(i,j)} \psi'$的原子协议可以说明信息,其中约束 ψ 表示要发送的消息在当前社会状态下必须保持的先决条件,c(i, j)表示从参与主体 i 到参与主体 j 的信道,约束 ψ_m 表示消息模板,约束 ψ′是“说明该状态下消息结果”的后置条件。约束 ψ_m 中的下标 m 是用于标记“ψ 是表示消息的约束”,其他与此类似,如φ_m。

后置条件计算新状态需要使用惯性(inertia)

①Miller T, Mcburney P. “Propositional dynamic logic for reasoning about first-class agent interaction protocols”. *Computational Intelligence*, 2011, 27(3), p.424.

②Baader F, et al. *The Description Logic Handbook: Theory, Implementation, Applications*. Cambridge, UK: Cambridge University Press, 2003.

③Boer F S D, et al. “Proving concurrent constraint programs correct”. *ACM Transactions on Programming Languages and Systems*, 1997, 19(5): 685-725.

的概念,即:后置条件中的任意变元都由后置条件定义,而状态中的任意其他变元都不变。允许主体发送消息φ_m,使得$\varphi_m \leftrightarrow \rho\ \psi_m$,从而约束消息的值;因此,$\psi_m$ 是消息模板。

实例 1:原子协议 $\min \rightsquigarrow^{c(S,A)}_{present_quote(book,\ fee)} cost(book,\ fee)$表示出版商 S 向作者 A 提供出版一本书(book)的报价(quote)。出版商希望作者接受这一费用(即变元 fee),如 5(单位是万元)。出版商 S 会发送消息 prensent_quote(bool, fee)↑fee=5 给作者 A,通过添加进一步的信息(即报价 5)就可以约束消息模板,这会涉及发送这条信息的结果状态。本文中的 min 是原子命题集Γ中的最小元素。

空协议 ε 是原子协议的特殊类型:$\psi \rightarrow \varepsilon$,其中 ε 是一个常元,表示如果 ψ 在当前社会状态下成立,则不需要发送消息。这类似于命题动态逻辑 PDL 中的测试算子,记为"$\psi?$",这里 ψ 是约束,而在 PDL 中,ψ 可以是动态逻辑公式。使用简写 ε 来表示"$\min \rightarrow \varepsilon$"。

复合协议可以利用算子由原子协议和空协议建构。如果α和β是协议,那么α; β表示α和β的序列合成,α∪β表示α和β之间的不确定性选择,$var_v^{\psi} \cdot \alpha$表示:除了局部变元 v 之外,在α的整个执行过程中,在 v 上的约束 ψ 始终得以保持。

在实例 1 中,要想增加变元 v 的整数值,就需要给 v 指派的值比消息发送之前大 1,而且不能使用原子协议来表示;后置条件仅表示状态变元之间的约束,而且不能提及前一个状态的值(即 5)。这可以定义如下变元声明算子(variable declaration operator)$\mathcal{D}$ 来模拟这种行为:$\mathcal{D} =_{def} \mathcal{B}_{(v_0 = v) \setminus v_0}(v < 5) \rightsquigarrow^{c(S,A)}_{inc(v)} v = v_0 + 1$,其中 inc(v)是增加变元值的消息模板。对局部声明变元的约束在其整个辖域内保持不变,因此,后置条件对 v_0 的约束等值于前置状态对 v 的约束。例如,在状态 v=5 时发送此消息,则后置条件为:$\downarrow_v (v = 5 \uparrow v_0 = v) \uparrow v = v_0 + 1$。这时 $v_0 = 5 \uparrow v = 6$,后置状态 v=6。

定义 3(协议说明):协议说明是格式为 $N(v_1, v_2, \cdots, v_n) =_{def} \alpha$的协议定义的聚合(collection),其中 N 是协议名称,α是协议,$(v_1, v_2, \cdots, v_n) \in vars(\alpha)$是协议α中的自由变元。协议可以通过引用其他协议名称而得到,因此允许递归定义。

实例 2:给出实例 1 中的作者请求对专著出版进行报价的运行实例。出版商 S 可以决定是否同意出版,并给出一个出版价格。作者 A 可以接受或拒绝这个价格。如果作者接受这个价格(用 p 表示该命题);作者可以先支付,然后出版,也可收到出版的书后就马上支付(用 q 表示该命题)。

出版商和作者的操作状态是一个条件承诺(Conditional Commitments,简称 CC),其形式为 CC(S, A, p, q),并规定:如果命题 p 为真,那么作者 A 将承诺实现出版商 S 的命题 q。如果 p 为真,则写为 C(S, A, q)。命题 request(book)表示:作者已要求对所出版的书进行报价;publish(book)表示:出版商已经出版该书;pay(fee)表示:作者已经支付了出版费(fee)。

承诺 CC(S, A, publish(book), pay(fee))表示:若作者 A 已经收到(accept)书,则承诺支付出版费,缩写为 cc_accept(book, fee)。承诺 CC(S, A, cc_accept(book, pay), publish(book))表示:一旦作者 A 承诺支付费用,出版商就承诺出版该书,缩写为 cc_promise(book, fee)。

为了启动交互协议,作者请求对出版费进行报价(request_quote)。其先决条件是:一旦报价,作者和出版商都不能更改报价。"-"表示不关心"出版商为其他作者提供的报价情况",而且任何一方都不会因此做出任何承诺,即:

(1) $request\_quote =_{def} \neg request(book) \uparrow \neg cc\_promise(book, -) \rightsquigarrow^{c(S,A)}_{request\_quote(book)} request(book)$

现在,出版商 S 发送报价(present_quote),并承诺以报价为作者出版该书。或者,在没有作者要求提供报价的情况下,出版商可以向潜在作者发送报价,即:

(2) $present\_quote =_{def} \neg cc\_promise(book, -) \rightsquigarrow^{c(S,A)}_{present\_quote(book,\ fee)} cc\_promise(book, fee)$

作者可以拒绝报价,也可以接受报价(accept_quote),并承诺该书一旦出版就付款,即:

(3) $accept\_quote =_{def} cc\_promise(book, fee) \rightsquigarrow^{c(S,A)}_{accept\_quote(book,\ fee)} cc\_accept(book, fee)$

作者 A 可以发送电子支付订单(EPO),从而履行其支付出版费的承诺,即:

(4) $pay =_{def} cc\_promise(book, fee) \rightsquigarrow^{c(S,A)}_{send\_EPO(A,\ book,\ fee)} pay(fee)$

最后,出版商可以把书交付给作者,即:

(5) $send\_book =_{def} cc\_accept(book, fee) \rightsquigarrow^{c(S,A)}_{deliver(book)} deliver(book)$

制定协议时都要求:主体发送消息之前必须满足特定的先决条件。例如,协议约定:一旦出版商同意出版,作者只能支付出版费。需要显式地对"参与者使用第一类协议组合算子发送消息的顺序"进行约束,即:有必要对上述这些消息进行显式排序,以降低由原子协议构建复合协议的复杂性。

在实例 2 中,在(1)—(5)中使用这 5 个算子不仅可以引入 5 个约束,而且可以替换多个先决条件,也可以仅仅把先决条件作为补充约束。使用算子的优点是:更容易计算"主体偏好决定行为"的搜索空间。

顶层协议的定义很直接。例如,在实例 2 中的交易大致分为两部分:报价和最终结果(即付款和交付出版的书),即:transaction $=_{def}$ quotation; finalize。一旦给出报价,出版商就承诺以该报价出版,这可用 present_quote 子协议来说明。作者可以选择接受或拒绝报价;如果接受报价,则承诺支付报价,即:

(6) quotation $=_{def}\{$request_quote$\}$; present_quote; (accept_quote $\cup$ reject_quote)

其中花括号$\{\}$表示 request_quote 是可选的,即:不必执行。定义$\{\alpha\}=\varepsilon\cup\alpha$,因此要么出现协议$\alpha$,要么什么都不出现(即空协议 ε)。

实例 2 中的最终结果(finalize)是交付出版的书和付款,可以先交付出版的书后付款,也可以先付款后交付出版的书;如果交易双方没有达成协议,不接受报价的承诺表示为($\neg$cc_accept),那么协议终止时就是空协议 ε,即:

(7) finalize $=_{def}$ (pay; send_book) $\cup$ (send_book; pay) $\cup$ ($\neg cc_{accept(book,\ fee)}\rightarrow\varepsilon$)

实例 2 说明:从(7)逆序至(1),就可以由原子协议和第一类协议组合算子自底向上构造协议的组合结构。当然,不同的主体,对协议的组合方式可能不同。

(三)外延语义

现在给出第一类协议说明语言的外延语义,进而建立需要的逻辑。其外延语义定义为一组跟踪(trace),而且每个跟踪定义了由主体系统发送的消息序列及其结果状态①。

定义 4(组合外延语义):第一类协议的语义定义为:包含协议允许的所有跟踪组成的集合。跟踪定义为三元组,其第一个元素表示跟踪的前置状态(prestate),第二个元素表示跨信道交流的序列,第三个元素表示结果状态。

语言$\mathcal{L}$的所有消息交流序列的集合表示为$I_{\mathcal{L}}$。语言$\mathcal{L}$的所有可能跟踪组成的集合(简称跟踪集)定义为$\wp(\mathcal{L}^{+}\times I_{\mathcal{L}^{+}}\times\mathcal{L}^{+})$,其中$\wp$是幂集函数,$\mathcal{L}^{+}$表示$\mathcal{L}-\{false\}$。

把第一类协议的语义定义为函数$f_{\check{D}(\alpha)}: f_e\rightarrow\wp(\mathcal{L}^{+}\times I_{\mathcal{L}^{+}}\times\mathcal{L}^{+})$,其中 $\check{D}$ 是"对协议α赋值"的声明集(set of declarations);f_e 是在环境 e 中从名称 N 到跟踪集的函数,$f_e: N\rightarrow\wp(\mathcal{L}^{+}\times I_{\mathcal{L}^{+}}\times\mathcal{L}^{+})$,用于把协议引用名称映射到跟踪集。因此,第一类协议的外延语义就是"代入协议定义和环境,得到由协议定义的跟踪集"的函数。当不使用 $\check{D}$ 和环境 e 时,就将其省略。

空协议 $\psi\rightarrow\varepsilon$ 定义为一个包含单个跟踪的集合,其中没有消息,而且前置状态和后置状态都是相同的,即:$[\![\psi\rightarrow\epsilon]\!]=_{def}\{(\varphi,\langle\rangle,\varphi)\mid\varphi\leftarrow\!\!\rho\,\psi\}$。

原子协议定义为如下跟踪集:其第一个元素满足先决条件,第二个元素是消息,第三个元素是结果状态。消息是约束 ψ_m 或约束φ_m,而且φ_m比 ψ_m 包含更多的信息,结果状态是$\varphi\blacklozenge(\varphi_m\Uparrow\varphi')$,其中 $\blacklozenge$:$(\mathcal{L}\times\mathcal{L})\rightarrow\mathcal{L}$ 是覆盖函数(overriding function),定义为$\varphi\blacklozenge\psi'=_{def}\psi'\Uparrow\!\!\downarrow_{vars(\psi)}\varphi$。结果状态是一个新的约束,使得$\varphi$中任意自由变元的值都被后置条件中的新值覆盖,而其他的变元保持不变。即,原子协议语义定义如下:

$[\![\psi\rightsquigarrow^{c(i,j)}_{\psi_m}\psi']\!]=_{def}\{(\varphi,\langle c(i,j),\varphi_m\rangle,\varphi\blacklozenge(\varphi_m\Uparrow\varphi'))\mid(\varphi\leftarrow\!\!\rho\,\psi)\wedge(\varphi_m\Uparrow\varphi'\leftarrow\!\!\rho\,\psi_m\Uparrow\psi')\}$

允许主体限制消息会对结果状态产生影响,规定:在一个消息中增加的任意附加信息φ_m都不能改变结果状态。例如,在实例 1 的原子协议 $\min\rightsquigarrow^{c(S,A)}_{present_quote(book,\ fee)}cost(book,\ fee)$中出版商提供一个出版费的报价,若发送方"在具有 fee = 5 的消息中"约束出版费,那么后置条件就共享此信息,以便确定变元 fee 的值。第一类协议说明语言的外延语义中,通过把消息作为后置状态的一部分来执行该条件:$\varphi\blacklozenge(\varphi_m\Uparrow\varphi')$。在实例 1 中,

①Miller T, Mcburney P. "Propositional dynamic logic for reasoning about first-class agent interaction protocols". *Computational Intelligence*, 2011, 27(3): 429-431.

唯一解决方案是 fee = 5,因此,后置状态是 cost(book, fee)↑fee = 5,这可以化归为 cost(book, 5)。虽然原子协议的执行是非确定性的,即:发送主体可以在不同的信息约束之间进行选择;但是,结果状态的计算是确定性的,即:结果是由消息、后置条件以及来自适合前置状态的约束决定的,这些前置状态包含了后置条件中未引用的变元。

第一类协议序列合成α; β可以通过"把α的后置状态作为β的前置状态"来定义:

$[\![\alpha;\ \beta]\!]=_{def}\{(\varphi,\ s_1\frown s_2,\ \varphi'')\mid \downarrow\varphi'\cdot(\varphi,\ s_1,\ \varphi')\in[\![\alpha]\!]\wedge(\varphi',\ s_2,\ \varphi'')\in[\![\beta]\!]\}$

其中 $s_1\frown s_2$表示序列 s_1 和 s_2的串联。

第一类协议的不确定性选择α∪β可以通过"从α和β得到的所有跟踪的并"来定义:$[\![\alpha\cup\beta]\!]=_{def}[\![\alpha]\!]\cup[\![\beta]\!]$。

现在讨论 $\mathrm{var}_v^{\psi}\cdot\alpha$的期望语义。首先,期望α正常执行,但在语境(context)中有一个受协议 ψ 约束的新变元 v。若 v 是协议状态下的自由变元,则必须隐藏 v 的这类出现。在执行α期间和执行α之后,不希望局部变元 v 上的约束发生变化。最后,一旦执行α,希望 v 的局部出现被隐藏,且之前的自由变元 v 及其约束将恢复到其前置状态中的值,其定义如下:

$[\![\mathrm{var}_v^{\psi}\cdot\alpha]\!]=_{def}\{(\varphi,\ s,\ \downarrow_v\varphi'\uparrow\downarrow_v\varphi)\mid(\downarrow_v\varphi\uparrow\psi,\ s,\ \varphi')\in[\![\alpha]\!]\wedge\downarrow_v\varphi'=\downarrow_v(\downarrow_v\varphi\uparrow\psi)\}$

此定义中部的$(\downarrow_v\varphi\uparrow\psi,\ s,\ \varphi')\in[\![\alpha]\!]$表示:在序列 s 中,执行来自状态$\downarrow_v\varphi\uparrow\psi$ 的子协议α,如果 v 是约束φ中的变元,那么 v 的任意自由出现都被隐藏;一旦执行了α,就剩下了后置状态φ′。此定义后部的条件$\downarrow_v\varphi'=\downarrow_v(\downarrow_v\varphi\uparrow\psi)$表示:后置状态 v 上的约束等于前置状态 v 上的约束;最后,计算整个协议的新后置状态。$\downarrow_v\varphi'$表示:隐藏局部变元 v 及其约束;但要重新引入之前的变元 v,因此,要与$\downarrow_v\varphi$进行结合,即有:$\downarrow_v\varphi'\uparrow\downarrow_v\varphi$。$\downarrow_v\varphi$表示:除了变元 v 之外,在前置状态φ中出现的所有变元都需要隐藏,即$\downarrow_v\varphi=\downarrow_{vars(\varphi)\setminus v}\varphi$。这样就可以将变元 v 恢复到它在前置状态中的值。

允许协议递归定义,即:协议可以自己引用自己。协议引用(protocol reference)的语义的递归定义如下:

$$[\![\check{D},N(v)]\!](e)=_{def}\begin{cases}[\![N(u)]\!](e)[v/v] & \text{当 } N(u)=_{def}\alpha\in\check{D}\text{ 且 }v\neq u\text{ 时;}\\ e(N) & \text{当 } N\in dom(e)\text{时;}\\ \mu G & \text{当 } N(v)=_{def}\alpha\in\check{D}\text{ 且 }N\notin dom(e)\text{时。}\end{cases}$$

其中 Ď 是"对协议α进行赋值"的声明集,N 是协议名称,u 和 v 都是变元,e 是环境,μG 是相对于偏序⊆的函数 G 的最小不动点①。

在此定义中,假定名称引用的是 Ď 中的有效协议名称。在第一种情况下,引用协议的名称 N 在声明集中,但是变元是不匹配的;把 v 的所有引用都重命名为 u,这样 N(v)的行为等价于 N(u)。在第二种情况下,N 在环境 e 中,因此,从该环境 e 返回到 N 的语义(记为 e(N))。在第三种情况下,N(v)是在具有匹配变元的定义中,而不是在环境 e 中;构造函数 $G(Z)=[\![\check{D},\ \alpha]\!](e\{N\mapsto Z\})$,用于给出"与 e 相同的环境"的协议α,而且 e(N)被 Z 覆盖。G 相对于偏序⊆的最小不动点(记为 μG)是协议 N 的表示值。对于任意 N 的后续引用,其语义由第二种情况 e(N)定义。

来自动态逻辑②的迭代算子α∗表示"零次或多次对子协议α进行迭代",即:$[\![\check{D},\ \alpha*]\!]=_{def}[\![\check{D}',\ N]\!]$,其中 $\check{D}'=\check{D}\cup\{N=_{def}\varepsilon\cup(\alpha;\ N)\}$,而且 ε 是空协议。即,α∗等值于协议 N,其中 N 被定义为:在空协议与"被 N 的递归调用跟随的"α的序列合成之间的不确定性选择。

三　协议蕴涵

命题动态逻辑 PDL 是能够对程序进行推理的模态逻辑。程序是 PDL 中的显式结构;程序由原子程序和复合算子组成,原子程序通常是编程语言中的语句,复合算子(如序列合成算子和选择算子)用于由原子程序构造更复杂的程序,因此,PDL 中的程序具有组合性。把复合程序分解

①Boer F S D, et al. "Proving concurrent constraint programs correct". *ACM Transactions on Programming Languages and Systems*, 1997, 19(5): 685-725.

②Harel D, Kozen D, Tiuryn J. *Dynamic Logic*. Cambridge: The MIT Press, 2000.

成原子程序以及证明它们的性质时,都会用到程序的组合性。

本文给出的关于第一类协议说明语言的逻辑,是对 Harel 等(2000)提出的命题动态逻辑 PDL[①] 进行改编而得到的,称之为 L_α 逻辑。通过改编,用第一类协议代替 PDL 中的抽象程序,用约束代替"PDL 中对命题进行解释所依赖的模型";在 PDL 中用于构造复杂程序的算子对应于 L_α 逻辑中的协议组合算子。

(一)L_α 逻辑的句法

L_α 逻辑可以对协议的结果进行推理,其语言是建立在定义 1 的约束系统语言$\mathcal{L}$之上,因为:如果任意约束$\varphi\in\mathcal{L}$,那么$\varphi\in L_\alpha$。为了区分,在 L_α 中使用表示协议性质的"命题",使用"约束"表示语言$\mathcal{L}$中的公式。虽然φ和ψ都可以表示命题和约束;但是带有数字下标的φ和ψ严格表示语言$\mathcal{L}$中的约束,不在 L_α -$\mathcal{L}$中。

定义 5(句法):L_α 逻辑的合式公式集定义如下。

$$\varphi ::= \varphi_0 | \varphi\wedge\psi | \neg\varphi | [\alpha]\varphi$$

每个公式在一个协议状态内进行赋值,即:每个公式都是一个约束;其中φ_0是语言$\mathcal{L}$中的约束;如果φ_0包含在协议状态中,则φ_0为真。$\wedge$和$\neg$的定义如常。命题$[\alpha]\varphi$表示:如果执行协议α,那么在协议α的每个结束状态(即终止状态)中,φ成立,这称之为"协议蕴涵(protocol entailment)",即:执行该协议就意味着一个特定命题成立。协议蕴涵允许表示协议性质,并对协议的可能未来结果进行推理。与"仅仅对协议的单一状态进行推理"相比,协议蕴涵具有更多的灵活性和更强的表达力[②]。

(二)L_α 逻辑的语义

L_α 逻辑建基于命题动态逻辑 PDL,因此可以使用 PDL 中的标准术语,对 L_α 逻辑的结构进行命名。命题只有在模型中进行赋值时才有意义。L_α 逻辑的模型是序对(F, ψ_0),其中 F 是框架,ψ_0表示协议的共享状态。L_α 逻辑的框架是一个序对$(\mathcal{L}, \check{D})$,其中$\mathcal{L}$是所有可能状态(即 L_α 逻辑的所有约束)组成的集合,$\check{D}$ 是被命名协议的定义集,即"对协议进行赋值"的声明集。在 PDL 中,$\check{D}$ 是意义函数,它把意义指派给原子程序。因为原子协议的意义可以直接由意义函数的定义得到,所以意义函数仅仅是从被命名协议到其定义的函数。$F, \psi_0\models\varphi$表示:ψ_0 在框架 F 中满足公式φ。在 F、$\mathcal{L}$和 $\check{D}$ 明了的情况下,可以省略它们。如果对于每个共享状态 $\psi_0\in\mathcal{L}$,公式φ在每个框架 F 中都为真,就记为$\models\varphi$,就说φ是有效的。$\psi_0\not\models\varphi$表示:φ在状态 ψ_0 中不成立。

定义 6(L_α 逻辑的语义定义):L_α 逻辑的语义定义如下:

(1)$\psi_0\models\varphi_0$(其中$\varphi_0\in\mathcal{L}$),当且仅当,$\psi_0 \leftrightarrow\!\!\!\!\rho\, \varphi_0$;

(2)$\psi_0\models\varphi\wedge\psi$,当且仅当,$\psi_0\models\varphi$且 $\psi_0\models\psi$;

(3)$\psi_0\models\neg\varphi$,当且仅当,$\psi_0\not\models\varphi$;

(4)$\check{D}, \psi_0\models[\alpha]\varphi$,当且仅当,$\forall(\psi_0, s, \psi_0')\in[\![\check{D}, \alpha]\!](\varnothing)\cdot\check{D}, \psi_0'\models\varphi$。

现在对定义 6 加以说明。其中的(1)表示:为了使协议φ_0在状态 ψ_0 中为真,根据"$\leftrightarrow\!\!\!\!\rho$"关系可知,$\psi_0$ 蕴涵φ_0。(2)表示:$\varphi\wedge\psi$ 在一个状态中为真,当且仅当,命题φ和 ψ 在该状态下都为真。(3)表示:$\neg\varphi$为真,当且仅当,φ不为真。(4)表示:$[\alpha]\varphi$为真,当且仅当,从状态 ψ_0 开始,对于协议α的每个结束状态 ψ_0' 而言,在框架 F 的协议声明集 $\check{D}$ 中,命题φ成立,即:对于协议α的任意跟踪,命题φ在该跟踪的每个结束状态中都成立。给定一个状态 ψ_0 以及约束φ_0和φ_0',$\psi_0\models\varphi_0\wedge\varphi_0'$等值于 $\psi_0 \leftrightarrow\!\!\!\!\rho\, \varphi_0\uparrow\varphi_0'$,且 $\psi_0\models\neg\varphi_0$等值于"$\psi_0 \leftrightarrow\!\!\!\!\rho\, \varphi_0$不成立"。

与命题逻辑一样,利用算子$\neg$和$\wedge$,可以对析取算子$\vee$、蕴涵算子$\rightarrow$和等值算子$\leftrightarrow$加以定义。与动态逻辑一样,利用算子[ ],在 L_α 逻辑中可定义其对偶算子$\langle\rangle$:$\langle\alpha\rangle\varphi\leftrightarrow\neg[\alpha]\neg\varphi$。$\langle\alpha\rangle\varphi$表示:如果进入协议$\alpha$,那么命题$\varphi$至少在协议$\alpha$的一个结束状态中成立。

实例 3:在实例 2 中,假定常元 self 是对实例 2 中第一类协议进行推理的主体 A 的 ID。如果主体接受报价,那么在交付出版的书时,将承诺支付,即:

$$C = self\rightarrow[accept\_quote]CC(self, A, deliver$$

①Harel D, Kozen D, Tiuryn J. *Dynamic Logic*. Cambridge: The MIT Press, 2000, pp. 165-255.

②Miller T, Mcburney P. "Propositional dynamic logic for reasoning about first-class agent interaction protocols". *Computational Intelligence*, 2011, 27(3): 422-457.

(book), pay(fee))。

在报价协议的情况中,并不是每个结果都是如此。但至少有一种"主体 A 接受报价的"情况,即:C = self → ⟨quotation ⟩CC (self, A, deliver (Item), pay(fee))。

如果主体 A 的目标是交付出版的书,就可以将上述特性与其目标相匹配,并使用此协议要求出版商 S 报价。收到报价后,主体 A 可以在 accept_quote 子协议上进行计算;如果接受报价,那么将承诺为此书支付报价费用。

四　有穷 L_α 逻辑的证明系统

为了证明第一类协议的性质,现在给出 L_α 逻辑的证明系统。首先给出 L_α 逻辑的公理和推理规则,然后证明该系统相对于前面给出的语义是可靠且完全的。这需要定义一个有穷论域中的"全部终止协议"证明系统——有穷 L_α 逻辑。"全部终止协议"是指:该协议终止所有执行;"有穷论域"是指:基础约束语言在有穷论域之上加以定义,如所有 32 位(bit)整数。

现在给出有穷 L_α 逻辑的证明系统。$\langle \check{D} | \psi \vdash \varphi \rangle$ 表示:命题φ在命题 ψ 成立的所有模型中都是可证的,其中 $\check{D}$ 是相关协议的声明集。语境清楚时可以省略 $\check{D}$。$\vdash\varphi$表示:在所有模型中,φ是可证的(即φ是一个定理)。

(一)有穷 L_α 逻辑的公理化

有穷 L_α 逻辑的基础约束语言 $\mathcal{L}$ 满足格(lattice)的性质[①],而且语言 $\mathcal{L}$ 中的任意公理都是有穷 L_α 逻辑证明系统中的公理。$\mathcal{L}$ 上有几个公理(例如 De Morgan 律和双重否定律),而且语言 $\mathcal{L}$ 是可靠且完全的[②]。

定义 7(L_α 逻辑的公理):除了语言 $\mathcal{L}$ 的公理外,有穷 L_α 逻辑的证明系统还包括如下 7 个附加公理模式:

① $[\alpha](\varphi\wedge\psi)\leftrightarrow[\alpha]\varphi\wedge[\alpha]\psi$(合取公理)

② $[\psi_0\rightarrow\varepsilon]\varphi\leftrightarrow(\psi_0\rightarrow\varphi)$(空协议公理)

③ $[\psi_0 \rightsquigarrow^{c(i,j)}_{\psi_m} \psi_0']\varphi\leftrightarrow(\psi_0\rightarrow(\psi_m\wedge\psi_0'\rightarrow\varphi)[v'/v])$(原子协议公理)

其中 $v = vars(\psi_m \uparrow \psi_0')$ 且 v′是新变元。

④ $[\alpha;\beta]\varphi\leftrightarrow[\alpha][\beta]\varphi$(序列合成公理)

⑤ $[\alpha\cup\beta]\varphi\leftrightarrow[\alpha]\varphi\vee[\beta]\varphi$(选择公理)

⑥ $[var_v^{\psi_0}\cdot\alpha]\varphi\leftrightarrow(v_0 = v\wedge\psi_0\rightarrow[\alpha](v_0 = v\rightarrow\varphi))$(局部变元公理)

其中 v_0 是新变元,v 在φ中不自由。

⑦ $\langle\check{D}|[N(v)]\varphi\leftrightarrow[\alpha[v/w]]\varphi\rangle$,其中 $N(w) =_{def} \alpha\in\check{D}$(协议名称公理)

除了公理③和⑥之外,其他公理都可以由其定义直接得到。公理③的意思是:原子协议达成结果φ,当且仅当,其先决条件成立时,该协议的所有后置状态蕴涵φ。将 v 重命名为新变元 v′,是为了在证明关于 v 的新值的性质φ时,前置状态值不受到干扰。公理⑥的意思是:变元声明达成φ,当且仅当,子协议 α 在"局部变元 v 上的约束成立"的所有结束状态中达成φ。在 box 算子[ ]左边的 $v_0 = v$ 表示:v_0 等值于 v 的前置状态值。因为 v_0 是新变元,所以它不会被α改变,因此只需考虑:v 通过α保持不变的结束状态,因此只需要证明:在结束状态下,φ仅仅在 $v_0 = v$ 的那些状态中成立。公理⑦表示:在 $N(w) =_{def} \alpha\in\check{D}$ 时,在声明集 $\check{D}$ 中宣告协议名称 N;否则,证明失败。

L_α 逻辑的证明系统中省略了 PDL 的几个公理,这是因为:由于第一类协议说明语言中存在名称引用,这些省略的公理可以从 L_α 逻辑的最小公理集和推理规则推导出来。即:命题动态逻辑 PDL 的所有公理都是 L_α 逻辑中的有效定理,PDL 的任意可证定理也是 L_α 逻辑的可证定理。即:可以使用大量现有的命题动态逻辑 PDL 的成果来研究第一类协议[③]。

有穷 L_α 逻辑的推理规则是分离规则,即:若φ且(φ→ψ),则 ψ。

(二)有穷 L_α 逻辑证明的自动化

L_α 逻辑定义了"可以对递归定义的协议进行推理的"一个命题动态逻辑证明系统。此系统分为两个部分:一部分是关于终止协议的推理,另一

①Boer F S D, et al. "Proving concurrent constraint programs correct". *ACM Transactions on Programming Languages and Systems*, 1997, 19(5): 685-725.

②Miller T, Mcburney P. "Propositional dynamic logic for reasoning about first-class agent interaction protocols". *Computational Intelligence*, 2011, 27(3): 422-457.

③Miller T, Mcburney P. "Propositional dynamic logic for reasoning about first-class agent interaction protocols". *Computational Intelligence*, 2011, 27(3): 456-457.

部分是关于非终止协议的推理。关于终止协议的证明可以自动化地加以证明,而关于非终止协议的证明在某些情况下是无法自动化地加以证明的。关于受限的第一类非终止协议的证明可以转化为关于终止协议的证明,从而使其证明自动化。简言之,有穷 L_α 逻辑中的证明可以通过公理和基础约束系统的蕴涵算子来自动完成。

定理 1:令φ是有穷 L_α 逻辑中的任意命题,使用相关公理可以将φ化归为基础约束系统语言$\mathcal{L}$中的约束φ_0,使得$\models\varphi$当且仅当$\models\varphi_0$。

证明:施归纳于φ的结构即可得证。假设有穷 L_α 逻辑中的公理是有效定理。归纳基础有两种情况:$[\psi_0\rightarrow\varepsilon]\varphi$和$[\psi_0 \overset{c(i,j)}{\rightsquigarrow_{\psi_m}} \psi_0']\varphi$。分别从左到右应用公理②和③,可以删除[ ]算子,把归纳假设应用于φ,就可以得到语言$\mathcal{L}$中的对应结果。

对于序列合成、选择、变元声明和名称引用的情况,可以分别从左到右应用公理④—⑦。协议定义形成树结构,这些公理将每个证明分解成它们的子树,直到只剩下空协议和原子协议(即基本情况)。唯一的例外是递归定义的协议,将展开(unfold)无穷多次。

现在讨论关于名称的无穷展开。对于有穷 L_α 命题,只有当名称为 N 的协议(简称协议 N)的先决条件成立时,才能通过应用公理⑦来终止这种展开。例如,证明 $\psi_0\rightarrow[N]\varphi$。如果 ψ_0 不满足 N 的先决条件,则不必考虑φ,该命题也成立。如果协议 N 是全部终止的,那么必然有:在每次递归调用 N 之前,状态发生变化,最终使得状态将不满足 N 的先决条件。形式$[N]\varphi$的证明可以构建$(\psi_0\vee\cdots\vee\psi_n)\rightarrow[N]\varphi$这样的结构,其中$(\psi_0\vee\cdots\vee\psi_n)$是 N 的先决条件,$\psi_i$(其中$0\leqslant i\leqslant n$)是整个协议的唯一路径的先决条件。因为假定基础约束系统是有穷的,那么先决条件的数量是有穷的。在展开前通过检查 N 的先决条件是否是可满足的,该证明即可终止。证毕。

定理 1 表明:给定有穷 L_α 逻辑的每个命题,都可以化归为基础约束系统语言$\mathcal{L}$中的一个约束,使用约束求解器(constraint solver)可以证明约束。这类似于情境演算中的回归①,它把关于未来状态的查询转换为关于当前状态的查询。定理 1 可以用于自动证明,即:主体只使用公理和约束求解器就可以证明协议蕴涵的性质,并且可以推导出它学到的任意新协议的性质。

(三)有穷 L_α 逻辑的可靠性和完全性

如果一个逻辑中的任意可证命题都是真的,那么该逻辑就是可靠的。如果一个逻辑中的任意真命题都是可证明的,那么该逻辑就是完全的。

定理 2:有穷 L_α 逻辑是可靠且完全的,即:若φ是 L_α 逻辑中的任意命题,则$\vdash\varphi\Rightarrow\models\varphi$且$\models\varphi\Rightarrow\vdash\varphi$。

证明:先证明可靠性。即证明:如果φ是 L_α 逻辑中的任意命题,那么$\vdash\varphi\Rightarrow\models\varphi$。由于一个逻辑中的任意可证命题都是由其公理和推理规则得到的,因此,只需要"通过证明每个公理是有效的"来证明可靠性。对于有穷 L_α 逻辑,如果证明了定义 7 中的每个公理都是有效的,那么可靠性就得以证明。根据定理 1 即可证明可靠性。

为了证明完全性,必须证明:若φ是 L_α 逻辑中的任意命题,则$\models\varphi\Rightarrow\vdash\varphi$。根据定理 1 可知,有穷 L_α 逻辑中的每个命题都可以化归为语言$\mathcal{L}$中的一个约束,而且基础约束系统语言$\mathcal{L}$是完全的,可以"通过把φ化归为一个约束"来构造φ的证明,并使用语言$\mathcal{L}$中的蕴涵算子来证明约束。该逻辑的可靠性保证了这种化归的正确性,因此该证明也是正确的,故有穷 L_α 逻辑是完全的。证毕。

L_α 逻辑的有穷演绎证明系统已在 Prolog 中实现,而且该证明系统建基于 CLP 边界求解器(bounds solver)②,即:一个带有变元的整数约束求解器。这一证明的实现利用了定理 1 的结果,并且把形式$[\alpha]\varphi$的命题化归为约束。

L_α 逻辑的可靠性和完全性也可以通过嵌入(embedding)的方法加以证明。因为 L_α 逻辑是由命题动态逻辑 PDL 改编得到的,通过改编,用第一类协议代替 PDL 中的抽象程序,用约束代替 PDL 中对命题进行解释所依赖的模型;在 PDL 中用于构造复杂程序的算子对应于 L_α 逻辑中的协议组合算子,因此,可以构造从 L_α 逻辑到 PDL 的

①Levesque H, Pirri F, Reiter R. "Foundations for the situation calculus". *Electronic Transactions on Artificial Intelligence*, 1998, 2(3-4): 159-178.

②Miller T, Mcburney P. "Propositional dynamic logic for reasoning about first-class agent interaction protocols". *Computational Intelligence*, 2011, 27(3), p.446.

翻译函数,根据 PDL 逻辑的可靠性和完全性①,就可以得到 L_α 逻辑的可靠性和完全性。具体证明细节类似于"通过嵌入的方法,利用 PDL 的可靠性和完全性证明信念—愿望—意图逻辑的可靠性和完全性"②。

五　结论与未来的工作

本文的主要工作和结论如下:(1)第一类协议说明语言是一种"把行为限制为由信道发送的消息的"行动语言,既可对协议结果进行表示和推理,又可对消息之外的行动进行推理,其约束语言用于说明消息和状态,每条消息都显式地表征"对共享社会状态的处理";(2)关于第一类协议说明语言的逻辑是命题动态逻辑 PDL 的改编逻辑(即 L_α 逻辑),并给出了有穷 L_α 逻辑的证明系统,证明了其可靠性和完全性;(3)有穷 L_α 逻辑的每个命题,都可以化归为基础约束系统语言$\mathcal{L}$中的一个约束,使用约束求解器可以证明这一约束。

编程语言中没有显式的先决条件,本文提出的方法是否适用于具有递归、无参数过程的编程语言,还有待进一步研究。通过把交互协议处理为第一类实体,第一类协议说明语言允许在交互中不断变化的参与主体,进行动态检查、引用、组合、共享和调用协议。因此,协议选择和调用的任务可以由主体而不是主体设计者执行,在运行时而不是在设计时执行。这样的框架能否用于多主体系统的全视角研究,还有待考察。

Reasoning about Agent Interaction Protocols Based on Adapted Propositional Dynamic Logic

ZHANG Xiao-jun & QIU Jun

(School of Philosophy, Anhui University, Hefei 230039, China)

Abstract: An agent interaction protocol is the specification that coordinates message-based communication between agents, and is used to provide a context of interaction. An interaction protocol is a public documents that explains the interaction rules by which all participating agents should abide. The logic L_α adapted from propositional dynamic logic can model the interaction protocol of multi-agent system. Therefore, it is necessary to design a first-class of protocol specification language, which is an action language that restricts actions to be messages sent by a channel, which can not only represent and reason about outcomes of protocols, but also reason about actions other than messages, and its constraint language is used to specify messages and states. Each message is explicitly defined as a manipulation of a shared social state.

Key words: first-class of protocol; constraints; agent; interaction protocol

(责任校对　唐尧)

①Harel D, Kozen D, Tiuryn J. *Dynamic Logic*. Cambridge: The MIT Press, 2000: 203-209.

②张晓君,林颖,周昌乐:《智能主体的等级 BDI(信念、愿望和意图)模型》,《计算机科学》2016 年第 7 期。