ISSN 1000-9825, CODEN RUXUEW
Journal of Software, Vol.19, No.7, July 2008, pp.1644−1653
DOI: 10.3724/SP.J.1001.2008.01644

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel/Fax: +86-10-62562563

Agent主动目标的形式化模型*

吴 骏[1,2+], 王崇骏[1,2], 骆 斌[1,2], 陈世福[1,2]

[1](南京大学 计算机软件新技术国家重点实验室,江苏 南京 210093)

[2](南京大学 计算机科学与技术系,江苏 南京 210093)

A Formal Model for Agent Active Goals

WU Jun[1,2+], WANG Chong-Jun[1,2], LUO Bin[1,2], CHEN Shi-Fu[1,2]

[1](State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing 210093, China)

[2](Department of Computer Science and Technology, Nanjing University, Nanjing 210093, China)

+ Corresponding author: E-mail: wujun80448@126.com

Wu J, Wang CJ, Luo B, Chen SF. A formal model for agent active goals. *Journal of Software*, 2008,19(7): 1644−1653. http://www.jos.org.cn/1000-9825/19/1644.htm

Abstract: In agent architecture, an active goal is a functionally self-contained entity with independent control flow. The related syntax of active goal and the operational semantics of active goal execution are presented. Furthermore, the BDI agent architecture driven by active goals is formally specified. Different from some former BDI agent architectures, goals are explicitly represented in the agent architecture as active entities. Parallel goals are supported in the architecture level by a very natural fashion, which is considered to be an important aspect of the rational behavior of agent. What's more, the explicit definition of goals provides convenience to the reconsideration of commitments for agents situated in dynamic environments.

Key words: intelligent agent; agent architecture; agent programming language; BDI model; goal

摘 要: 在 agent 结构中,主动目标是一个功能上自含且有自己独立控制流的实体.给出了主动目标相关的语法定义以及主动目标运行的操作语义,而且主动目标驱动下的 BDI agent 结构也被形式化地定义出来.区别于以往的一些 BDI agent 结构,目标不是隐含地表示而是作为实体显式地表示在 agent 结构中,使 agent 结构很自然地支持并行的目标,这被认为是 agent 理性行为的一个重要方面.此外,对目标的显式定义也为 agent 在动态的环境中对承诺的重新考虑带来了方便.

关键词: 智能 agent;agent 结构;agent 编程语言;BDI 模型;目标

中图法分类号: TP18 **文献标识码**: A

BDI模型的理论基础起源于Bratman的实用推理理论以及Dennett的意图理论.区别于早期agent结构领域所流行的理论推理,实用推理是直接针对行动的推理.实用推理实质上是对人的思维与行动的过程的模拟,即人类

* Supported by the National Natural Science Foundation of China under Grant No.60503021 (国家自然科学基金); the Natural Science Foundation of Jiangsu Province of China under Grant No.BK2005075 (江苏省自然科学基金); the High-Tech Research Program of Jiangsu Province of China under Grant No.BG2006027 (江苏省高技术研究计划)

Received 2006-11-07; Accepted 2007-09-27

某一时刻的思考可能决定今后的一系列行动过程.按照实用推理的观点,一个agent在从感知到行动的决策过程中应当首先进行慎思,即决定在今后的行动中要达到什么目的(ends)状态,然后再进行从目的到方法的推理(means-ends reasoning),即选择合适的方法来实现确定的目的,最后要对所选的目的和方法进行承诺[1].通俗地说,就是首先决定做什么,然后再确定怎么做,最后一直做下去.

承诺的引入同时也带来了对承诺的坚持问题.意图实质上是对目的的一种承诺,其引入在一定程度上保证了 agent 行动的连续性与可理解性,但是在动态的环境中必须用意图反思(intention reconsideration)的方法不断地重新考虑所承诺的目的和方法的有效性.对于意图反思在实现上很难确定反思的程度,就像 Wooldridge 在文献[2]中指出的那样,如果 agent 不停下来充分地重新考虑它的意图,常常会一直盲目地试图实现这个意图,尽管这些意图明显地不能实现,或者这些意图已经没有任何理由实现;如果 agent 不断地重复考虑它的意图,这将会没有足够的时间来实现这些意图,因此,面临不能真正实现这些意图的风险.同样,对方法的承诺也存在同样的问题,由于环境的改变,预先确定的方法可能变得不再适用,继续执行这样的方法甚至有可能导致 agent 进入某个不可恢复的状态.

在BDI模型理论的不同形式化方式中,目标(goal)是比较常见的一个心智成分[3−5].目标的引入是为了解释和说明agent的前摄性(proactive)的心理行为[6],目标驱动性就是一种前摄性.在这类系统中,agent的承诺表现在对目标的承诺和对实现目标的计划的承诺上.文献[7,8]中提到,agent能够同时追求多个目标的能力是其理性行为的一个重要方面.现有的BDI理论和系统都没有提供一种理论的或者结构的框架来决定目标之间如何交互以及agent如何决定追求哪一个目标.取而代之的是,它们以简单的理由假定agent总是追求一致的目标集合.通过忽略这种很重要的理性行为,目标慎思的问题被从agent结构层次移到agent编程层次,而且需要agent开发者以一种很容易出错的专门方式来处理.在文献[9]中,目标被分为两类,即声明性目标(declarative goals)和程序性目标(procedural goals).我们认为,在BDI结构中把目标理解为主动性的单元,而不是把它们看作固定的BDI推理循环中的声明性或程序性的单位将更加自然.这种方法的基本思想是:把目标形式化为功能上自含的实体,而且具有自己独立的控制流.在这一前提下,多个并行的目标很自然地被agent结构层次所支持.而且,目标的显式表示为承诺的再考虑提供了清晰而直接的对象;多个目标的并发执行也体现了一种分而治之的思想.

1 语 法

在定义形式系统的语法之前,我们首先作如下 6 条假定:

① 假定一种含有“非(negation)”和“与(conjunction)”的命题逻辑语言 $\mathcal{L}$,符号 $\vDash$ 表示 $\mathcal{L}$ 上的标准蕴涵关系(standard entailment relation).

② 假定一个信念查询语言 $\mathcal{L}_{\mathcal{B}}$,它的典型元素是 β.一个信念查询可以测试 agent 是否有某种信念.

③ 假定一个定义在信念库和信念查询语言之上的蕴涵关系,表示为 $\vDash_{\mathcal{L}_{\mathcal{B}}}$.

④ 假定所有可能的目标缓冲的集合为 $\mathcal{GB}$,它的典型元素是 $\mathcal{A}$.

⑤ 假定一个目标缓冲查询语言 $\mathcal{L}_{\mathcal{A}}$,它的典型元素是 α.一个目标缓冲查询可以用来测试 agent 在过去的一段时间内是否有某个目标信息.

⑥ 假定一个定义在目标缓冲和目标查询语言上的蕴涵关系,表示为 $\vDash_{\mathcal{L}_{\mathcal{A}}}$.

基于以上假定,可以建立主动目标和基于主动目标的 BDI agent 的形式化模型的语法定义.在后文中提到的目标都是主动目标.在假定④中所提到的目标缓冲是一个记录 agent 在过去的一段时间内所采纳的目标的执行状况的单元,在后文中对目标缓冲将会进行进一步的描述.

定义 1(目标状态). 目标状态标识一个目标的行为状态.令目标状态的集合为 *GoalState*,典型元素是 τ,定义为

$$\tau ::= INI \mid ACT \mid SUS \mid SUC \mid FAI .$$

INI 表示主动目标的初始状态,*ACT* 表示一个目标是活动的,*SUS* 表示目标处于等待状态,*SUC* 表示目标已经成功,*FAI* 表示目标已经失败.

定义 2(计划). 令 *BasicAction* 是所有基本行动的集合,典型元素是 $a;\varphi\in\mathcal{L}$ 表示一个需要达到的子状态; $\lambda(a)$ 是一个向 agenda 里插入一个基本行动 a 的操作,则令计划的集合 *Plan* 的典型元素是π,定义为

$$\pi ::= \lambda(a) \mid \varphi \mid \pi_1 : \pi_2 .$$

在上述定义中提到的基本行动是直接可以执行的行动,包括 agent 对外界的行动(例如"向某个方向移动一步")以及 agent 对外界的感知(例如"看当前位置四周有什么物体").agenda 是 agent 用来放置和调度执行将要执行的动作的单元,这将在下文给出定义.区别于以往的 BDI 结构,这里,计划是属于某个目标的,而不是属于 agent 的.一个计划由目标内的计划产生规则来产生,计划产生规则 R_{PG} 定义如下:

定义 3(计划产生规则). 计划产生规则的集合定义为

$$R_{PG} = \{\varphi : \beta \leftarrow \pi \mid \varphi \in \mathcal{L}, \beta \in \mathcal{L}_{\mathcal{B}}, \pi \in Plan\} .$$

$\forall r = \varphi : \beta \leftarrow \pi \in R_{PG}$ 都是一条计划产生规则,表示当β被信念库支持时,执行计划π可以实现状态φ.

由以上定义可知,一个计划并不是可以线性执行的,因为里面可能含有一些要达到的子状态.对于这些子状态的实现需要利用计划产生规则生成新的计划来实现,这实际上是一个树状的执行过程.一个子状态实际上就是执行树上的一个分支节点.对于这种执行方式,把子状态扩充为子目标能为执行提供方便,也就是把子状态、计划产生规则以及一个扩展标识封装为一个元组,子目标定义如下:

定义 4(子目标). 一个子目标是一个三元组 (φ, PG, ω), 其中 $\varphi\in\mathcal{L}$, $PG\subseteq R_{PG}$,ω是扩展标识,定义为 $\omega ::= T \mid F$.

由于计划是树状执行的,因此在目标中引入一个目标栈将为计划的执行带来方便.由计划产生规则生成的计划序列必须把其中的元素逐个转化为栈元素,再按顺序压入栈中.目标栈定义如下:

定义 5(目标栈). 令栈元素的集合为 *StackElement*,典型元素为 *e*,定义为

$$e ::= \lambda(a) \mid (\varphi, PG, \omega) .$$

目标栈的集合为 *GoalStack*,典型元素为 *S*,定义为

$$S ::= e \mid e.S \mid E .$$

至此,我们可以给出如下主动目标的定义:

定义 6(目标). 一个目标σ是一个四元组 $(\varphi, \upsilon, S, \tau)$,其中 $\varphi\in\mathcal{L}$ 是这个目标的目的状态, $\upsilon\subseteq R_{PG}$ 是这个目标可用的计划产生规则的集合, $S\in GoalStack$ 是一个目标栈用于执行计划,以及 $\tau\in GoalState$ 标识这个目标的当前状态.令 GOAL 是所有目标的集合.

在一个 agent 中,一个目标是否要在某时刻出现由预先定义的目标产生规则所决定.实际上,人类目标的产生不仅与当时的信念有关,而且与过去一段时间内所完成的目标有关.为了模拟这种机制,我们把目标产生规则定义如下:

定义 7(目标产生规则). 目标产生规则的集合 R_{GG} 定义为

$$R_{GG} = \{\beta \wedge \alpha \rightarrow \sigma \mid \beta \in \mathcal{L}_{\mathcal{B}}, \alpha \in \mathcal{L}_{\mathcal{A}}, \sigma \in GOAL\} .$$

$\forall r = \beta \wedge \alpha \rightarrow \sigma \in R_{GG}$ 是一条目标产生规则,表示当β被信念库支持,同时α被目标缓冲支持时,σ是一个可能的目标.

实际上,并不是所有的目标都可以并发执行,比如"向东走"和"向西走"的目标如果并发执行就会产生毫无意义的行为.因此必须定义目标优先规则来确定互相冲突的目标之间的优先关系,而不冲突的两个目标则没有优先关系.

定义 8(目标优先规则). 目标优先规则的集合 R_{GP} 定义为

$$R_{GP} = \{\sigma_a : \beta \wedge \alpha \succ \sigma_b \mid \beta \in \mathcal{L}_{\mathcal{B}}, \alpha \in \mathcal{L}_{\mathcal{A}}, \sigma_a, \sigma_b \in GOAL\} .$$

$\forall r = \sigma_a : \beta \wedge \alpha \succ \sigma_b \in R_{GP}$ 是一条目标优先规则,表示当β被信念库支持,同时α被目标缓冲支持时,目标 σ_a 优先于目标 σ_b,或者说目标 σ_a 抑制目标 σ_b.

如果一个目标集合里面不含任何冲突的目标,那么我们说这个目标集合是一致的.这种一致性可以通过使用目标优先规则不断地删除一个被抑制的目标来实现.设初始的目标集合为 *S*,最终达到一致的目标集合为 *S'*, 则被删去的目标的集合称为被抑制目标集合,记作:

$$sup(S) = S / S' .$$

定义 9(agenda). 令 $a \in BasicAction$, CLG 是清理已成功或已失败的目标的操作,INS 是向活动目标集合里插入所有可能的目标的操作,CLS 是清理活动目标集合中所有被抑制的目标的操作, $unlock(\varphi)$ 是一个唤醒一个目标的操作, $a\downarrow$ 是向产生这个行动的目标反馈这个行动的执行信息的操作,则令 agenda 的集合 $AGENDA$ 的典型元素为ρ,定义为

$$\rho ::= a \mid CLG \mid INS \mid CLS \mid BEG \mid unlock(\varphi) \mid a\downarrow \mid \rho_1.\rho_2 .$$

由以上 agenda 的定义可知,一个 agenda 中不仅可能含有由目标插入的基本行动,而且还含有 agent 的一些基本操作.实际上,agent 是通过顺序执行 agenda 中的元素来与外界交互以及控制 agent 内部的各个组成成分的.

定义 10(agent). 令 $\Sigma = \{\mathcal{B} \mid \mathcal{B} \subseteq \mathcal{L} \wedge \mathcal{B} \nvdash \bot\}$ 是信念库的集合, $\mathcal{A} \in \mathcal{GB}$ 是一个目标缓冲, $\mathcal{B} \in \Sigma$ 是一个信念库, $\mathcal{GS} \subseteq GOAL$ 是一个活动目标集合, $\rho \in AGENDA$ 是一个 agenda, $\Lambda \subseteq BasicAction$ 是一个基本行动库, $\Gamma \subseteq GOAL$ 是一个目标库, $\Omega_G \subseteq R_{GG}$ 是一个目标产生规则库, $\Omega_P \subseteq R_{GP}$ 是一个目标优先规则库, $\varsigma_{\mathcal{B}} : (BasicAction \times \Sigma) \to \Sigma$ 定义基本行动执行后信念库的更新方法, $\varsigma_{\mathcal{A}} : (GOAL^* \times \mathcal{GB}) \to \mathcal{GB}$ 定义目标执行后目标缓冲的更新方法,则一个 agent 定义为如下元组:

$$(\mathcal{B}, \mathcal{A}, \mathcal{GS}, \rho, \Lambda, \Gamma, \Omega_G, \Omega_P, \varsigma_{\mathcal{B}}, \varsigma_{\mathcal{A}}) .$$

配置的作用是用来表示计算中一个目标或者一个 agent 在任一时刻的状态,一个配置由计算中可能变化的元素组成.在我们的形式化系统中有两种运行,分别是目标层次运行和 agent 层次运行,因此我们定义如下两个配置来分别定义其操作语义.

定义 11(目标配置). 一个目标配置是一个二元组 $\langle S, \tau \rangle$,其中 $S \in GoalStack$ 是一个目标栈, $\tau \in GoalState$ 标识该目标的目前状态.如果 $(\varphi, \upsilon, S, \tau)$ 是一个目标,则该目标的初始配置是 $\langle (\varphi, \upsilon, F), INI \rangle$.

定义 12(agent 配置). 一个 agent 配置是一个四元组 $\langle \mathcal{B}, \mathcal{A}, \mathcal{GS}, \rho \rangle$, $\mathcal{B} \in \Sigma$ 是一个信念库, $\mathcal{A} \in \mathcal{GB}$ 用于记录过去一段时间所执行的目标的信息, $\mathcal{GS} \subseteq GOAL$ 是一个活动目标集, $\rho \in AGENDA$ 是一个 agenda.

2 语义及形式化模型描述

我们用转换系统[10]的方式来给出形式系统的操作语义.一个转换系统是一个语言,包含一个公理以及转换规则的集合来为这个语言产生转换.一个转换是一步计算中一个配置到另一个配置的变化.

在下文中,目标层次执行定义一个目标的基本执行过程,用目标配置 $\langle S, \tau \rangle$ 上的转换系统来定义其操作语义;而 agent 层次执行定义 agent 的基本执行过程,用 agent 配置 $\langle \mathcal{B}, \mathcal{A}, \mathcal{GS}, \rho \rangle$ 上的转换系统来定义其操作语义.

2.1 目标层次运行

首先我们定义目标栈的执行方式,在上文中已经提到目标栈是一个目标中用于执行计划的单位.目标栈的执行方式定义为:在一次目标栈的执行中,只有栈顶元素可被处理.需要注意的是,在以下转换规则中初始目标配置中目标状态标识一定是 ACT,否则将不产生任何转换.

定义 13(目标栈执行). 令 $e \neq E$, $\tau = ACT$

$$\frac{\langle e, \tau \rangle \to \langle S', \tau' \rangle}{\langle e.S, \tau \rangle \to \langle S'.S, \tau' \rangle} .$$

在目标栈的执行中,如果栈顶元素是一个插入操作,就直接执行这个操作;如果栈顶元素是一个子目标,就进行子目标扩展,随后的两条转换规则给出这两种操作的语义.

定义 14(插入操作).

$$\frac{e = \lambda(a)}{\langle e.S, ACT \rangle \to \langle S, SUS \rangle} .$$

以上转换规则说明,如果一个目标向全局 agenda 插入了一个基本行动,那么这个目标立即被置为等待状态.实际上,这个目标是在得到插入的这个基本行动执行完毕的反馈后才恢复到活动状态的.在被置为等待状态到恢复活动状态这段时间里,这个目标将不发生任何转换.

定义 15(子目标扩展).

$$\frac{\mathcal{B} \not\models \varphi \quad \exists r = \varphi : \beta \leftarrow \pi \in PG \quad \mathcal{B} \models_{\mathcal{L}_B} \beta \quad S' = \psi(\pi)}{\langle (\varphi, PG, \omega).S, ACT \rangle \rightarrow \langle S'.(\varphi, PG \setminus \{r\}, T).S, ACT \rangle}.$$

如果目标栈顶是一个子目标,那么该目标的下一步执行将导致新的栈元素被插入目标栈,我们把这个过程定义为子目标扩展.新的栈元素其实是由栈顶的子目标产生的计划转变而来的,实质上是把计划序列中的子状态封装为子目标即可.为方便起见,在以上定义中,我们用一个函数 ψ 表示这个过程.在栈顶子目标进行一次扩展后,产生这次扩展的计划产生规则必须被删去,这是为了使同样的扩展不被重复地进行.

一个子目标可能被实现,也可能失败.在接下来的两条转换规则中将给出子目标实现和失败的操作语义.

定义 16(子目标实现).

$$\frac{\mathcal{B} \models \varphi}{\langle (\varphi, PG, \omega).S, ACT \rangle \rightarrow \langle S, ACT \rangle}.$$

当目标栈顶出现一个子目标时,实际上首先是要对其封装的子状态进行测试.如果由信念库能够导出这个子状态,那么这个子目标已经被实现,这时直接把它从栈顶删除即可.

定义 17(子目标失败).

$$\frac{\mathcal{B} \not\models \varphi'' \quad \neg\exists (r = \varphi : \beta \leftarrow \pi \in PG \wedge \mathcal{B} \models_{\mathcal{L}_B} \beta) \quad \neg\exists (\varphi_0, PG_0, T) \in S'}{\langle (\varphi'', PG'', T).S'.(\varphi, PG', T).S, ACT \rangle \rightarrow \langle (\varphi, PG', T).S, ACT \rangle}.$$

当栈顶的子目标封装的子状态不能由信念库导出,且无法进行扩展时,我们称这个子目标已失败.此时,目标栈中的元素需依次出栈,直到遇到下一个扩展标识为 T 的子目标为止.可以证明,以这种方式出栈每次出栈的元素都是最后留在栈顶的子目标在同一次扩展中生成的.

定理 1. 目标栈中,设在同一次扩展中生成的子目标中,被扩展过的子目标数目为 N,则 $N \le 1$.

证明:对于任意处于栈顶的子目标,设其扩展中含有 m 个子目标并依次插入栈中,这些子目标从栈顶到栈底依次为 $\sigma_0,...,\sigma_m$.现证明,若 σ_k 在栈中存在时,$\sigma_{k+1},...,\sigma_m$ 一定都尚未被扩展.

假设 $\exists \sigma_i, k+1 \le i \le m$, 已被扩展.

∵ 按定义 13,只有栈顶元素可以被扩展.∴ σ_i 一定在之前得到某个时刻 t 出现在栈顶.

∴ 在 t 时刻之后,$\sigma_0,...,\sigma_{i-1}$ 已按定义 16 或者定义 17 的转换规则出栈.这与 σ_k 还在栈中矛盾(σ_k 再次进栈的操作无定义).

∴ 假设不成立.∴ 若 σ_k 在栈中存在时,$\sigma_{k+1},...,\sigma_m$ 一定都尚未被扩展.

∴ 只有最接近栈顶的一个子目标可能被扩展,因此一定有 $N \le 1$. □

定理 2. 当一个子目标失败时,与之在同一次扩展中产生的栈元素将全部出栈.

证明:由定义 17 的转换规则,若失败子目标为 ε_0,出栈前一个时刻目标栈为 $\varepsilon_0.S'.\varepsilon.S$.其中,$\varepsilon$ 已被扩展,且 $\neg\exists(\varphi_0, PG_0, T) \in S'$,则出栈的元素序列为 $\varepsilon_0.S'$.

∵ $\neg\exists(\varphi_0, PG_0, T) \in S'$. ∴ ε_0不是由 S' 中的子目标扩展生成的.

假设ε_0是由 S 中的一个子目标ε_s在前一个时刻t扩展生成的.

1° 若ε_0与ε在同一次扩展中生成,那么这与定理 1 的结论矛盾,故这种情况不可能.

2° 若ε_0与ε在不同的扩展中生成,那么由定义 13 可知,t时刻ε_s处于栈顶,且t时刻之后ε_0被压入栈.而ε出现在ε_0后方无定义支持.故这种情况也不可能出现.同理,S' 也与ε_0是在同一次扩展中生成的.

∴ $\varepsilon_0.S'$ 是由ε在同一次扩展中生成的.结论成立. □

由目标配置的初始化可知,目标栈中的最后一个元素实际上是一个子目标,且这个子目标中封装的子状态就是该目标的目的状态.当目标栈中的最后一个元素被推出时,此时如果由信念库能够导出其封装的子状态,那么称该目标实现了,否则称该目标失败了.一个目标完成了是指这个目标实现了或失败了.如以下两条转换规则所示.

定义 18(目标实现).

$$\frac{\mathcal{B} \vDash \varphi}{\langle(\varphi, PG, \omega), ACT\rangle \to \langle E, SUC\rangle}.$$

定义 19(目标失败).

$$\frac{\mathcal{B} \nvDash \varphi \quad \neg\exists(r = \varphi : \beta \leftarrow \pi \in PG \wedge \mathcal{B} \vDash_{\mathcal{L}_{\mathcal{B}}} \beta)}{\langle(\varphi, PG, \omega), ACT\rangle \to \langle E, FAI\rangle}.$$

至此,我们定义了主动目标运行的操作语义.在本节的下一部分,我们将定义支持主动目标的 agent 运行的操作语义.

2.2 Agent层次运行

在一个 agent 中,agenda 是一个用于放置和调度 agent 将要执行的操作的成分.实际上,这些 agenda 里的元素有两个来源,分别是 agent 的活动目标集中的目标以及 agenda 中元素的执行过程.如下的转换规则说明,在 agenda 中每次只有第一个元素可以被执行.

定义 20(agenda 执行).

$$\frac{\langle \mathcal{B}, \mathcal{A}, \mathcal{GS}, e\rangle \to \langle \mathcal{B}', \mathcal{A}', \mathcal{GS}', \rho'\rangle}{\langle \mathcal{B}, \mathcal{A}, \mathcal{GS}, e.\rho\rangle \to \langle \mathcal{B}', \mathcal{A}', \mathcal{GS}', \rho'.\rho\rangle}.$$

如以下定义所示,agenda 中的基本行动是 agent 活动目标集合中并发执行的目标所插入的.每个被插入的基本行动都被放置在 agenda 的队尾.对于活动目标集合中的多个并发的目标,agent 通过不断顺序执行 agenda 中的元素,实际上是一个不断趋近一个能满足所有主动目标的状态的过程.

定理 3. agent 在运行中同时趋近多个并发的目标.

证明:因为计划产生的基本行动都被插入到 agenda 中,由定义 14 的转换规则,一个目标插入一个基本行动后立即被置为等待状态.即一个目标不会一次插入所有要执行的基本行动.又因此而 agenda 中的元素是由定义 20 顺序执行的.所以多个目标产生的基本行动是在 agenda 中交替执行的,而且活动目标集是一致的.agent 在运行中同时趋近多个并发的目标. □

定义 21(基本行动插入).

$$\frac{\langle \lambda(a).S.(\varphi, PG, \omega), ACT\rangle \to \langle S.(\varphi, PG, \omega), SUS\rangle}{\langle \mathcal{B}, \mathcal{A}, \mathcal{GS} \cup \sigma, \rho\rangle \to \langle \mathcal{B}, \mathcal{A}, \mathcal{GS} \cup \sigma', \rho.a\rangle}$$

where $\sigma = (\varphi, \lambda(a).S.(\varphi, PG, \omega), \upsilon, ACT)$.

and $\sigma' = (\varphi, S.(\varphi, PG, \omega), \upsilon, SUS)$

一个基本行动的执行将导致两个元素依次从 agenda 的前端插入,按插入顺序分别是对产生这个基本行动的目标的反馈操作,以及清理活动目标集合中已经完成的目标的操作.

定义 22(行动执行).

$$\frac{\mathcal{B}' = \varsigma_{\mathcal{B}}(a, \mathcal{B})}{\langle \mathcal{B}, \mathcal{A}, \mathcal{GS}, a.\rho\rangle \to \langle \mathcal{B}', \mathcal{A}, \mathcal{GS}, CLG.a \downarrow .\rho\rangle}.$$

对于一个已经完成的目标,并不是简单地把这个目标清除出活动目标集.由于该目标的执行情况对以后目标的产生和目标优先关系都有指导意义,所以必须把它记录在目标缓冲之中,如以下转换规则.

定义 23(清理完成的目标).

$$\frac{\mathcal{DS} = \{(\varphi, S, R_{PG}, \tau) \in \mathcal{GS} \mid \tau = SUC \vee \tau = FAI\}}{\langle \mathcal{B}, \mathcal{A}, \mathcal{GS}, CLG, \rho\rangle \to \langle \mathcal{B}, \varsigma_{\mathcal{A}}(\mathcal{DS}, \mathcal{A}), \mathcal{GS}/\mathcal{DS}, INS.\rho\rangle}.$$

通过以上的转换规则可以清理出已经完成的目标,但是仅仅这样还不够,进一步的操作是插入新的目标.这由三步完成,首先插入所有可能的目标,然后清理被抑制的目标,最后激活新加入的目标.随后的两条转换规则分别定义这 3 个操作.为叙述方便起见,首先在这里定义两个二元关系如下:

$$(\mathcal{B},\mathcal{A}) \Rightarrow_s \sigma \quad \text{iff} \quad \mathcal{B} \in \Sigma \wedge \mathcal{A} \in \mathcal{L}_{GB} \wedge \sigma \in GOAL \wedge \exists(\beta \wedge \alpha \rightarrow \sigma) \in \Omega_G \wedge \mathcal{B} \models_{\mathcal{L}_B} \beta \wedge \mathcal{A} \models_{\mathcal{L}_A} \alpha ,$$

$$\mathcal{GS} \succ_d \sigma \quad \text{iff} \quad \mathcal{GS} \subseteq GOAL \wedge \sigma \in sup(\mathcal{GS}) .$$

定义 24(插入所有可能目标).

$$\frac{\mathcal{GS}' = \{\sigma \in \Gamma \mid (\mathcal{B},\mathcal{A}) \Rightarrow_s \sigma\}}{\langle \mathcal{B},\mathcal{A},\mathcal{GS},INS.\rho\rangle \rightarrow \langle \mathcal{B},\mathcal{A},\mathcal{GS} \cup \mathcal{GS}',CLS.\rho\rangle} .$$

定义 25(清理被抑制目标).

$$\frac{\mathcal{GS}_d = \{\sigma \in \mathcal{GS} \mid \mathcal{GS} \succ_d \sigma\}}{\langle \mathcal{B},\mathcal{A},\mathcal{GS},CLS.\rho\rangle \rightarrow \langle \mathcal{B},\varsigma_{\mathcal{A}}(\mathcal{GS}_d,\mathcal{A}),\mathcal{GS}/\mathcal{GS}_d,BEG.\rho\rangle} .$$

定义 26(激活初始目标).

$$\frac{\mathcal{GS}_i = \{(\varphi,\upsilon,S,\tau) \in \mathcal{GS} \mid \tau = INI\} \quad \mathcal{GS}_b = \{(\varphi,\upsilon,S,ACT) \mid (\varphi,\upsilon,S,INI) \in \mathcal{GS}_i\}}{\langle \mathcal{B},\mathcal{A},\mathcal{GS},BEG.\rho\rangle \rightarrow \langle \mathcal{B},\mathcal{A},(\mathcal{GS}/\mathcal{GS}_i) \cup \mathcal{GS}_b,\rho\rangle} .$$

从定义 14 描述的转换规则可知,一个目标在向 agenda 插入一个基本行动后马上会被置为等待状态,等待这个基本行动执行完毕后的反馈.在以下的一条转换规则中,我们定义了基本行动的执行对产生这个基本行动的目标的反馈.在这里我们把这种反馈仅仅定义为唤醒相应的目标.实际上向目标可以反馈更多的内容以利于计划的执行,比如这个行动执行的效用、代价等.同时应定义相应的机制来利用这些信息来辅助更高效地产生计划.这方面的工作将在我们今后的研究中进行探讨.

定义 27(反馈).

$$\frac{\langle S.(\varphi,PG,\omega),\tau\rangle \rightarrow \langle \lambda(a).S'.(\varphi,PG,\omega),\tau\rangle}{\langle \mathcal{B},\mathcal{A},\mathcal{GS},a\downarrow.\rho\rangle \rightarrow \langle \mathcal{B},\mathcal{A},\mathcal{GS},unlock(\varphi).\rho\rangle} .$$

唤醒目标的操作定义如下,如果在活动目标集中找到了这个目标,则把它的目标状态标识由 *SUS* 更新为 *ACT*;如果找不到相应的目标,则这个操作不产生任何效果.后一种情况可能在一个目标被新加入的目标抑制而被清理出活动目标集时出现,如定义 25 描述的转换规则所示.

定义 28(唤醒目标).

$$\frac{\exists\sigma = (\varphi,\upsilon,S,SUS) \in \mathcal{GS}}{\langle \mathcal{B},\mathcal{A},\mathcal{GS},unlock(\varphi).\rho\rangle \rightarrow \langle \mathcal{B},\mathcal{A},\mathcal{GS}' \cup \{\sigma'\},\rho\rangle} ,$$

where $\mathcal{GS} = \mathcal{GS}' \cup \{\sigma\}$ and $\sigma' = (\varphi,\upsilon,S,ACT)$

$$\frac{\neg\exists(\varphi,\upsilon,S,SUS) \in \mathcal{GS}}{\langle \mathcal{B},\mathcal{A},\mathcal{GS},unlock(\varphi).\rho\rangle \rightarrow \langle \mathcal{B},\mathcal{A},\mathcal{GS},\rho\rangle} .$$

可以证明在如上转换规则下,agenda 中执行的实际上是特定元组的序列.

定理 4. 令 $\Delta = \{\langle a,CLG,INS,CLS,BEG,a\downarrow,unlock(\varphi)\rangle \mid a \in BasicAction \wedge (\varphi,\upsilon,S,\tau)\ \ produced\ \ a\}$,则实际上在agenda中执行的是序列$act_0,act_1,\ldots,act_n, \forall act_i \in \Delta, 0 \le i \le n$.

证明:由定义 21 描述的转换规则得,任意目标只能从 agenda 队尾插入基本行动.

又由定义 20 的转换规则可得,只有 agenda 队头的元素可以被执行.且操作在执行定义 22 描述的转换规则后,定义 23~定义 27 的转换规则将依次地被调用,先后执行的是 $CLG,INS,CLS,BEG,a\downarrow,unlock(\varphi)$.

所以在agenda中执行的是序列$act_0,act_1,\ldots,act_n$. □

通过上文定义的语法与语义,一个支持主动目标的 agent 模型的形式化系统被精确地定义出来.下面我们将简要讨论这个形式化系统的性能特征.

3 模型分析

文献[1]中 Rao 和 Georgeff 把信念、愿望、意图定义为心智成分,表示为可能世界状态.一个 agent 的意图

是信念和愿望的子集,即一个agent通过执行行动趋进一个愿望上为真且信念上认为可能的世界状态.为了计算上的可行性,他们对这套理论进行了一些简化,最重要的一点是,只有信念被实际地表示出来.意图被简化为预先定义的计划模板处理的事件,而意图被简化表示为栈中将要被执行的计划.实际上,很多实际的系统都是以类似的方式进行处理的,如PRS[11],AGENTSPEAK[12],3APL[13],JACK[14]和CAN[9]等.在本节中将在与这类系统对比的基础上进行简要的分析.

3.1 执行效率

由定理 4 可知,实际上,每个基本行动 a 在执行中都是被扩充为序列 $\langle a,CLG,INS,CLS,BEG,a\downarrow,unlock(\varphi)\rangle$ 顺序执行的.agent 的执行效率由其实际用于执行基本行动上的时间比例来表征.最理想的情况是,agent 执行一个基本行动和执行下一个基本行动的过程之间没有时间间隔.令 t 为执行基本行动 a 的时间消耗,Δt 为执行控制操作 $CLG,INS,CLS,BEG,a\downarrow$ 和 $unlock(\varphi)$ 的时间,那么效率 E 可被定义为

$$E=\frac{t}{t+\Delta t}.$$

因为目标是agent中的并发成分,计划每一个目标是在目标运行层次并发进行的,其消耗的时间可以忽略不计.如果一个 agent 有很多目标,我们可以认为两个被不同目标插入的行动之间没有时间间隔,所以,计划的时间消耗没有被包含在Δt 中.我们认为这是文中提出的方法在提高 agent 的执行效率方面的一个优势.

Δt 主要由两个操作 INS 和 CLS 的执行时间组成.因为目标是显式表示的,我们也可以希望这两个操作在有限的时间内完成.

为了提高执行效率,可以设法加大 t 的值,即定义耗时长一点的基本行动.这是可能的,因为完全可以把一个行动分解为几个较小的行动,或者把几个行动合成为一个较大的行动.

3.2 对并发目标的支持

在本文提出的形式化系统中,目标被定义为一个元组 $(\varphi,\upsilon,S,\tau)$,封装了目的状态$\varphi$,计划产生规则库$\upsilon$,目标栈 S 以及目标状态τ.目标在功能上是自含的,确切地说,也就是一个目标能够利用自己的计划产生规则库υ和目标栈 S 来实现其目的状态φ,而实现的方式是不断地向 agenda 中插入基本行动,使 agent 通过执行的基本行动向自己的目的状态趋近.在目标层次运行中,定义了目标运行的操作语义,而且在 agent 层次可以对目标进行控制.对于任意可能的目标,都由定义 24 的转换规则被插入到活动目标集中,并由定义 25 的转换规则保持活动目标集的一致性.由定理 3 可知,agent 在运行中是趋向于一个满足所有目标的状态的,多个并发的目标实质上在 agent 结构层次被很好地支持.而在以往基于栈运行的系统中,目标经常是隐含表示的,因此并不好在结构层次上对目标的并发性进行控制.多个并发的目标往往是在实现层次进行讨论的,而没有清晰的语义定义.图 1 直观地描述了上述的这种区别.

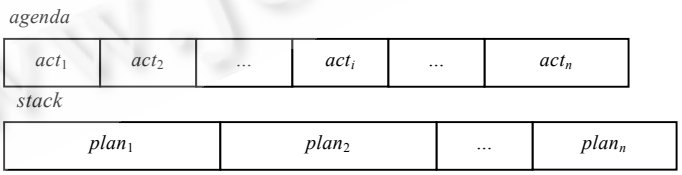

Fig.1 Comparison between agenda and stack

图 1 Agenda 与 stack 的结构比较

3.3 反应能力

反应能力包括两个方面:一方面是对动态多变的环境的反应能力;另一方面是对同时存在的不同的目标的反应能力.

在动态环境中,一个目标或者一个实现目标的计划都可能在执行过程中变得无效.此时 agent 应当有马上发

现和放弃无效目标或计划的能力.实际上,这是一种对过去所作承诺的反思能力.对计划的反思被移至目标层次运行由定义 17 表示的子目标失败处理来完成,而这个过程是可以并发执行的.由文中定义的操作语义,agenda 元素的执行是一个原子转换,即当一个 agenda 元素执行时,在 agent 运行层次没有任何其他转换有可能发生.对目标的反思是在任意一个被插入到 agenda 的基本行动被执行完后进行的.因此,在本文提出的模型中,agent 对动态环境的反应能力取决于 t.在非常动态的环境中,计划库中的基本行动应该被定义得短一些,即消耗时间短一点.与以往的系统相比,本文提出的系统在对环境变化反应能力方面展现出更多的可控制性.

Agent 对同时存在的不同目标的反应能力也取决于 t,因为只有一个目标插入 agenda 中的行动被执行时,这个目标才得以响应.具体地说,t 越小,则对不同目标的反应能力越强.实际上,$t+\Delta t$ 决定了不同目标的并发粒度.因此,为了提高对不同目标的反应能力,也要求定义耗时较短的基本行动.

综上所述,较长的基本行动可以提高 agent 的执行效率,而较短的基本行动可以使 agent 的反应能力较优.所以,应当根据具体的应用在基本行动的大小方面取得一个平衡.

4 结论与展望

本文提出了主动目标的概念与方法.一个主动目标是一个功能上独立的实体,它封装了目的状态、执行栈、计划产生规则和目标状态信息,并且有自己独立的控制流.主动目标的行为是产生基本行动然后把它们插入全局 agenda 供 agent 调度执行.基于这种思想,本文形式化地建立了一个支持主动目标的 agent 模型.在这个系统中有两种运行,分别是目标层次运行和 agent 层次运行,在本文中均以转换系统的方式定义其操作语义.

在提出的模型中解决了或者至少部分解决了如下几个问题:首先,目标不再是固定BDI推理循环中声明性或解释性的单元,而是活动目标集合中的主动成分.通过利用目标产生规则和目标优先规则更新活动目标库以及控制其一致性,并发的目标被agent结构层次所支持,而这被认为是理性行为的一个很重要的方面[7].其次,对于处于动态环境中的agent,对承诺进行反思是一个保证agent追求的目的以及执行的计划的有效性的重要方法,但是很难在过多的反思和反思不过之间找到平衡点,即很难在目标驱动性和反应能力两方面都有较好的表现.文中提出的结构提供了从另一个角度解决这个问题的方法,对目标的显式定义为对承诺的重新考虑提供了直接的对象,多个目标之间的并发运行也为克服反思带来的过量计算提供了一种可能的解决方案.最后,文献[15]中提到模块化是软件工程的一个中心问题,本文提出的形式化系统通过目标的封装性在一定程度上实现了基于目标的模块化.

今后的研究主要为以下两点:首先,研究模型的实现问题,并对其计算复杂性等作更为深入的分析.其次,结合其他一些形式化方法(如情景演算、π演算等)对本文的模型给出实例验证.

References:

[1] Rao AS, Georgeff MP. BDI Agents: From theory to practice. In: Proc. of the 1st Int'l Conf. on Multi-Agent Systems (ICMAS-95). San Francisco, 1995. 312−319.

[2] Wooldridge M. An Introduction To Multiagent Systems. New York: John Wiley & Sons, 2001. 78−79.

[3] Kang XQ, Shi CY. A BDI model for rational agents. Journal of Software, 1999,10(12):1268−1274 (in Chinese with English abstract).

[4] Zhang W, Shi CY. A formal semantics of agent organization structure design. Journal of Software, 2002,13(3):447−452 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/13/447.pdf

[5] Hu SL, Shi CY. An improved twin-subset semantic model for intention of agent. Journal of Software, 2006,17(3):396-402 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/17/396.htm

[6] van Riemsdijk MB, Dastani M, Meyer JC. Semantics of declarative goals in agent programming. In: Proc. of the AAMAS'05. Netherlands: ACM, 2005. 133−140. http://portal.acm.org/citation.cfm?id=1082473.1082494

[7] Pokahr A, Braubach L, Lamersdorf W. A BDI architecture for goal deliberation. In: Proc. of the AAMAS'05. Netherlands: ACM, 2005. 1295−1296. http://portal.acm.org/citation.cfm?id=1082473.1082740

[8] Pokahr A, Braubach L, Lamersdorf W. A flexible BDI architecture supporting extensibility. In: Proc. of the 2005 IEEE/WIC/ACM Int'l Conf. on Intelligent Agent Technology (IAT-2005). 2005. 379−385. http://doi.ieeecomputersociety.org/10.1109/IAT.2005.9

[9] Winikoff M, Padgham L, Harland J, Thangarajah J. Declarative & procedural goals in intelligent agent systems. In: Proc. of the KR 2002. Toulouse, 2002. 470−481. http://goanna.cs.rmit.edu.au/~winikoff/Papers/KR2002.pdf

[10] Plotkin GD. A structural approach to operational semantics. Technical Report, DAIMI FN-19, University of Aarhus, 1981.

[11] Ingrand FF, Georgeff MP, Rao AS. An architecture for real-time reasoning and system control. IEEE Expert: Intelligent Systems and Their Applications, 1992,7(6):34−44.

[12] Rao AS. AgentSpeak(L): BDI agents speak out in a logical computable language. In: Velde WV, ed. Agents Breaking Away (LNAI). LNAI 1038, Springer-Verlag, 1996. 42−55. http://www.springerlink.com/index/5x727q807435264u.pdf

[13] Hindriks KV, de Boer FS, van der Hoek W, Meyer JC. Agent programming in 3APL. Autonomous Agents and Multi-Agent Systems, 1999,2(4):357−401.

[14] Howden N, Ronnquist R, Hodgson A, Lucas A. JACK intelligent agents-summary of an agent infrastructure. In: Proc. of the 5th Int'l Conf. on Autonomous Agents. 2001. http://www.agent-software.co.uk/shared/resources/papers/JACK-infrastructure.pdf

[15] van Riemsdijk MB, Dastani M, Meyer JC, de Boer FS. Goal oriented modularity in agent programming. In: Proc. of the AAMAS 2006. Hakodate: ACM, 2006. 1271−1278. http://portal.acm.org/citation.cfm?id=1160864

附中文参考文献:

[3] 康小强,石纯一.一种理性 Agent 的 BDI 模型.软件学报,1999,10(12):1268−1274.

[4] 张伟,石纯一.Agent 组织结构设计的一种形式语义.软件学报,2002,13(3):447−452. http://www.jos.org.cn/1000-9825/13/447.pdf

[5] 胡山立,石纯一.Agent 意图的双子集语义改进模型.软件学报,2006,17(3):396−402. http://www.jos.org.cn/1000-9825/17/396.htm

吴骏(1981－),男,湖南郴州人,博士生,主要研究领域为多 agent 系统,计划识别.

骆斌(1967－),男,博士,教授,博士生导师,CCF 高级会员,主要研究领域为人工智能,多 agent 系统.

王崇骏(1975－),男,博士,副教授,主要研究领域为机器学习,多 agent 系统.

陈世福(1938－),男,教授,博士生导师,主要研究领域为人工智能,模式识别.