AgentSpeak中意图生成过程的可靠性与完整性研究

杨 博 邵利平 覃 征
(西安交通大学电子与信息工程学院 西安 710049)

摘 要 意图生成是BDI型Agent为实现目标而产生动作序列的过程。验证软件Agent中意图生成的正确性是Agent编程语言中一个重要的研究问题。针对软件Agent中意图执行的正确性,以当前最流行的BDI型Agent编程语言AgentSpeak为例,证明了软件Agent意图执行的有效性。首先根据AgentSpeak的语法构造了一个解释系统,并给出了该解释系统的满足关系,从而得出了AgentSpeak的模型论语义。在该模型论语义的基础上,结合由Moreira和Bordini所给出的操作语义,证明了AgentSpeak的意图生成等价定理:AgentSpeak语言中模型论语义的意图等价于AgentSpeak程序操作语义的意图。由此可得出结论——AgentSpeak中的意图执行是可靠而完整的,从而验证了AgentSpeak中软件Agent意图完成目标的正确性。

关键词 AgentSpeak,意图生成,Agent编程语言,模型论语义,BDI Agent
中图法分类号 TP311 **文献标识码** A

Soundness and Completeness Proof of Agent Intention Production in AgentSpeak

YANG Bo SHAO Li-ping QIN Zheng
(School of Electronics and Information Engineering, Xi'an Jiaotong University, Xi'an 710049, China)

Abstract Intention production is the procedure that BDI Agent drafts an action sequence to realize a goal. However, it is also a big obstacle to verify the validity of Agent intention produced by the specification of Agent Programming Language(APL). In this paper, to prove the validity of intention execution in AgentSpeak, we constructed a model-theoretic semantics and a formal interpretation of Agent program's intention. Then, on the bases of the model-theoretic semantics and the operational semantics presented by Moreira and Bordini, we proved the equivalence theorem: the intention in model-theoretic semantics of AgentSpeak language is equivalent with the one in operational semantics of AgentSpeak program. According to the equivalence theorem, we can get the conclusion that intention execution of AgentSpeak is sound and complete.

Keywords AgentSpeak, Intention production, Agent programming language, Model-theoretic semantics, BDI Agent

1 引言

Wooldridge将当前多Agent系统的研究趋势总结为5个方面:无处不在、可交互、智能、代理和人性化[1]。所谓多Agent系统是包含一定数量Agent的系统,其中Agent是代表用户或拥有者利益的计算实体。Agent为满足设计目标而自治性地执行某些动作,因此不需明确地告知Agent在某个时刻该执行哪个(些)动作。

在多Agent系统的研究中,作为将抽象理论研究转换为具体应用的一项关键技术,面向Agent的编程(AOP, Agent-Oriented Programming)受到了越来越多的关注。AOP是由Yoav Shoham于1993年首先提出的[2]。这种程序设计方法的核心思想是以精神性和意图的概念来实现Agent理论家所提出的Agent属性,进而实现Agent。虽然面向Agent的编程目前还处于起步阶段,但在最近几年,多Agent系统得到了计算机科学家的极大关注。多Agent系统领域的研究人员普遍认为,该领域所涌现出的技术将会极大地影响动态不可预知环境下的应用程序设计模式。AOP将会成为计算机系统中一种主流的开发方法[3]。而所有这些,则有赖于Agent理论研究的发展和进步。

目前针对Agent理论研究可分成两个方面:①Agent技术的理论研究;②将Agent理论推向实际应用的研究。对Agent的理论研究又可分为两个方面:①基于Agent个体设计的研究,如Rao等针对Agent的理性决策所提出的BDI结构模型[4]和吴朝晖等提出的基于对策论的BDI主体模型等[5];②基于Agent社会性设计的研究,如Zambonelli等提出的面向多Agent系统分析与设计的Gaia方法[6]、DeLoach等提出的MaSE方法[7]、李超明等提出的基于智能的智能系统模型MIBIS[8]、张新良等提出的多Agent联盟结构动态生成算法[9]等。而将Agent理论推向实际应用的研究主要集中在Agent理论的技术实现方面,例如Jo等人提出的BDI Agent建模方法[10]、毛新军等提出的面向Agent软件开发方法学ODAM[11]等。一些研究者也提出了多种Agent编程语言(APL, Agent Programming Language),如Jason[12]和JADE[13],

到稿日期:2010-04-08 返修日期:2010-08-05 本文受国防"十一五"预研项目(60673024)资助。
杨 博(1978—),男,博士生,主要研究方向为人工智能、多Agent系统,E-mail:yangbo@xjtu.edu.cn;覃 征(1956—),男,博士,教授,主要研究方向为人工智能、信息处理与信息融合。

也为 Agent 的成功应用做出了巨大贡献。

以上两方面的研究都取得了可喜的进展，为 AOP 的实际应用做出了巨大的贡献。但是在注重这两方面研究的同时，面向 Agent 的研究却忽视了两者间的联系，缺乏开发和构造实际 Agent 系统的理论基础研究。各种 APL 所构建的软件 Agent 是否具有自治性，所执行的动作是否理性，是 Agent 编程语言能否达到设计要求的一个衡量标准，而对此进行的研究却鲜见报导。

本文以验证 APL 中软件 Agent 决策的制定是否理性为目标，研究了目前最流行的 APL 语言——AgentSpeak 中的意图制定过程。在该过程中，通过对软件 Agent 理性行为的形式化分析，证明了 AgentSpeak 语言意图制定中理性因素的可靠性与完整性。本文所提出的形式化分析过程不但可推广到验证其他软件 Agent 的理性行为，还可为开发新 Agent 编程语言提供一个研究思路。

本文第 2 节给出了 AgentSpeak 的语法结构和 AgentSpeak 中所使用的字符串集合；第 3 节构造了 AgentSpeak 的模型论语义，并给出了该模型论语义中通过执行意图而完成目标的可满足条件；第 4 节介绍了 Moreira 等人所给出的 AgentSpeak 中软件 Agent 的操作语义；第 5 节通过分析 AgentSpeak 模型论语义中意图完成目标与操作语义对应定义之间的等价关系，得出了 Agent 的意图实现目标的可靠性与完成性特性；最后对全文进行了总结。

2 AgentSpeak 语法

Anand S. Rao 提出的 AgentSpeak 语言[14]是一种用于实现 BDI 型 Agent[4]的受限一阶逻辑编程语言。所谓 BDI 结构[15]是由 Bratman 提出的一种实现“智能”或“理性”Agent 的方法。BDI Agent 系统是一个置身于变化环境中的系统，依靠感知器接收外界的输入，然后根据内部的心智状态执行动作来影响周围的环境[16]。BDI Agent 系统有 3 个主要的心智状态：信念(belief)、愿望(desire)和意图(intention)，分别代表了 BDI Agent 所具有的信息、动机和决策。

在 AgentSpeak 中，Agent 由一个基础信念集和一个规划集组成。信念集是包含一个或多个信念原子的集合，信念原子或者信念原子的否定被称为信念文字(literal)。初始的信念集是落实的，即集合中的信念原子不包含未赋值的变量。所谓规划是 Agent 可执行的基本动作序列，规划中的动作以一阶谓词定义，并通过特殊的谓词符号(动作符号)区分其他谓词。规划由触发事件、上下文条件以及规划主体组成。其中，触发事件是触发整个规划执行的事件；上下文条件是执行规划必须满足的条件，触发事件和上下文条件合称规划头；规划主体是一个由动作和(子)规划构成的序列。意图是 Agent 为处理特定事件而提交动作的过程，是一个部分实例化的规划栈。触发事件中的事件，可进一步划分为外部事件(由于对外部环境的观测而对自身信念的增加和删除)和内部事件(Agent 执行自身规划而生成的事件)。在内部事件生成时，这个事件将会与生成它的意图捆绑在一起。而外部事件则会产生新意图，表示 Agent 对环境改变所产生的不同反应。

2.1 AgentSpeak 语言执行流程

图 1[12]给出了 AgentSpeak 语言的执行流程。图中的图形符号含义如下：矩形表示集合(信念、事件、规划和意图)，菱形表示选择(从一个集合中选择一个元素)，圆形表示处理过程(包括 AgentSpeak 程序的解析)。

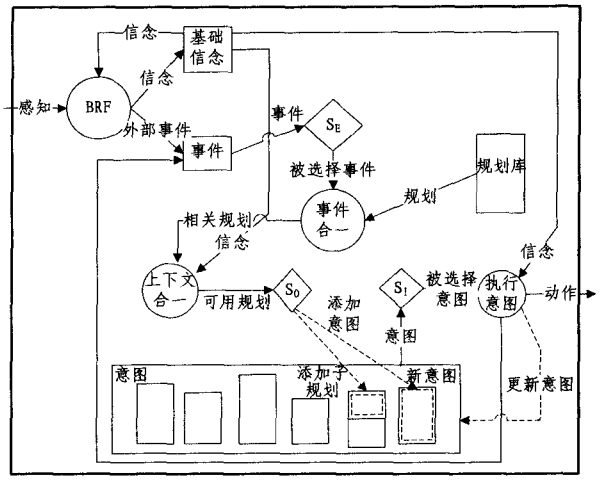

图 1 AgentSpeak 语言的执行流程示意图

在 Agent 程序的每个解析循环中，Agent 都会更新事件列表。这个更新动作可能是由于对外部环境的观测而产生的(外部事件)，也可能是由于自身意图的执行(内部事件)而产生的。假设信念是通过观测环境而更新的，并且不管 Agent 的信念在什么时间发生了改变，都需要立即在事件集中插入新事件，以更新整个信念集。信念转换函数(图中的 BRF 组件)并不是 AgentSpeak 解析器的一部分，但却是 Agent 结构中不可或缺的组件。

在事件选择函数 S_E 选择事件之后，Agent 程序会对所选事件与规划集中所有规划的触发事件进行合一，生成一个关于所选事件的相关规划集(所选事件可以触发的规划集合)。然后，AgentSpeak 解析器通过检查相关规划中的上下文条件是否符合 Agent 的信念，确定一个可用规划集(在处理所选事件时可以使用的规划集合)。此后，规划选择函数 S_O 会从可用规划集中选择一个可用规划作为处理所选事件的“下一步打算”，或者将规划放到已经存在的意图栈的顶部(如果所选事件是一个内部事件)，或者在意图集中创建一个新的意图(如果所选事件是一个外部事件)。

最后一步要做的就是从意图栈中选择一个意图开始执行。意图选择函数 S_I 负责从意图栈中选择意图。在执行意图时，如果要执行的操作是一个基础动作(基础动作的执行会对外部环境做出改变)或者测试型目标(测试型目标的执行会产生内部事件)，就需要更新 Agent 的意图集。如果要执行的操作是一个测试型目标，将会在信念集中搜索一个满足条件的信念原子，搜索条件就是要求该信念原子与测试型目标中的谓词可以合一。如果搜索成功，该合一产生的变量实例化将会代入到包含测试型目标的规划中(并且测试型目标本身也会从意图栈中移除)。如果选择了一个基础动作，对意图集的必要更新操作只是从意图中移除这个动作。当规划主体中所有的操作都移除以后(也就是在执行了规划主体中的所有动作之后)，整个规划从意图中移除，同时要移除生成这个规划的达成型目标。这将会结束一次循环的执行，然后 Agent 程序会继续开始下一次循环。

2.2 AgentSpeak 语法结构

为在下文中对 AgentSpeak 构造进行形式化描述，本小节简要介绍 Bordini[12]等提出的 AgentSpeak 的语法结构。

图 2 给出了 AgentSpeak 的语法结构。其中，Agent 由一个信念集 bs（Agent 的初始信念库）和一个规划集 ps（Agent 的规划库）组成。AgentSpeak 中的原子公式 at 实际上是一个谓词，其中 P 是谓词符号，$t_1,\cdots,t_n$ 是一阶逻辑的标准项。注意，bs 中的 at 必须是落实的，即 at 中不存在变量。

$$
\begin{aligned}
&ag::=bs\ ps\\
&bs::=at_1,\cdots,at_n \qquad (n\geqslant 0)\\
&at::=P(t_1,\cdots,t_n) \qquad (n\geqslant 0)\\
&ps::=p_1\cdots p_n \qquad (n\geqslant 0)\\
&p::=te:ct\leftarrow h\\
&te::=+at\mid -at\mid +g\mid -g\\
&ct::=\text{true}\mid l_1\&\cdots\&l_n \qquad (n\geqslant 1)\\
&h::=\text{true}\mid f_1;\cdots;f_n \qquad (n\geqslant 1)\\
&l::=at\mid not\ at\\
&f::=A(t_1,\cdots,t_n)\mid g\mid u \qquad (n\geqslant 0)\\
&g::=!\ at\mid ?\ at\\
&u::=+at\mid -at
\end{aligned}
$$

图 2 AgentSpeak 语法

AgentSpeak 中的规划通过上面的 p 给出，其中 te 是触发事件，ct 是规划的上下文条件，h 是一个动作、目标或者信念更新的序列；$te:ct$ 是规划的头，h 是规划主体。Agent 的规划集通过一个规划列表 ps 来定义。每个规划的头部都包含一个公式 ct 来定义规划能够执行的先决条件。如果一个规划是可用的，那么上下文条件 ct 必须是 Agent 信念集的逻辑结果。

触发事件 te 可能是对 Agent 的基础信念集的增加和删除（分别是 $+at$ 和 $-at$），或是对目标的增加和删除（分别是 $+g$ 和 $-g$）。动作、目标和信念的更新序列 h 定义了规划的主体。动作是 Agent 可自由使用的用来改变外界环境的基本操作。动作的定义方式与普通谓词相同，只是使用动作符号 A 代替了普通谓词符号 P。目标 g 可以是一个达成型目标（$!\ at$），也可以是一个测试型目标（$?\ at$）。$+at$ 和 $-at$ 表示对基础信念集的更新（u）操作，分别代表增加和删除 at。

2.3 AgentSpeak 的字符串集合

本小节以 Bordini[17] 提出的形式化语义为基础，构造 AgentSpeak 语义中所使用的字符串集合。为方便本文证明，对 Bordini 的形式化语义进行了修改和扩充。下面给出该形式化语义。

AgentSpeak 语义所使用的词汇表是一个六元组 $\langle V,C,F,P,A,\Delta\rangle$，其中：

1）V 为无限变量符号集合；

2）C 为无限常量符号集合；

3）F 为有限函数符号集合；

4）P 为有限谓词符号集合；

5）A 为有限动作符号集合；

6）$\Delta=\{!,?,;,\leftarrow,\rightarrow,\hat{},(,)\}$，为 AgentSpeak 语义使用的连词符号集合；

7）满足 F,P,A,Δ 两两不相交。

在上述词汇表的基础上，所构造的 AgentSpeak 语义中使用的字符串集合为：

1）无限项集 TS，项的序列缩写为 $t\cdots$，其产生规则为 $t\cdots\rightarrow C\mid V\mid F(t\cdots)$；

2）有限落实项集 $GTS\subseteq C$。其中项的缩写为 $\tau\cdots$，产生规则为 $\tau\cdots\rightarrow C\mid F(\tau\cdots)$；

3）无限谓词集 $PRS=\{p(t\cdots)\mid p\in P,t\cdots\in TS\}$，其中 $PRS\subseteq P\times TS^*$；

4）无限谓词否定集 $NPS=\{\rightarrow p(t\cdots)\mid p\in P,t\cdots\in TS\}$，也就是 PRS 的否定形式；

5）无限落实谓词集 $GPS=\{p(\tau\cdots)\mid p\in P,\tau\cdots\in GTS\}$，其中 $GPS\subseteq P\times GTS^*$；

6）无限落实谓词否定集 $NGPS=\{\rightarrow p(\tau\cdots)\mid p\in P,\tau\cdots\in GTS\}$，其中 $NGPS\subseteq\{\rightarrow\times P\times GTS^*\}$；

7）闭合世界中的有限信念集 $BS=GPS$ 和开放世界中的有限信念集 $BS=\{\rightarrow\times P\times GPS^*\cup GPS\}$；

8）无限动作集 $AS=\{\alpha(t\cdots)\mid\alpha\in A\}$，其中 $AS\subseteq A\times TS^*$；

9）有限落实动作集 $GAS=\{\alpha(\tau\cdots)\mid\alpha\in A\}$，其中 $GAS\subseteq A\times GTS^*$；

10）有限达成目标集 $AGoal=\{!\ p\mid p\in GPS\}$；

11）无限查询目标集 $QGoal=\{?\ p\mid p\in GPS\}$；

12）$IAS=\{+p\mid p\in BS\}\cup\{-p\mid p\in BS\}=ADD\cup DEL$，无限内部动作集；

13）无限触发事件集 $TES=IAS\cup\{\pm q\mid q\in AGoal\}\cup\{\pm q\mid q\in QGoal\}$；

14）永真规划 $TrP=\{\text{True}:\text{True}\leftarrow\text{True}\}$；

15）规划集 $PS=\{\eta:\chi\leftarrow\beta\mid\eta\in TES,\chi\in GPS^*$，其中 $\beta\in(AS\cup LAS\cup AGoal\cup QGoal\cup TP)^*\}$；

16）有限有意动作集 $IntA=\{\phi\prec a(t\cdots)\succ l\cdots\mid l\in IAS\cup\{\text{true}\}\}$，即一个 Agent 在运行一个动作之前，必须要知道该动作的前提条件 ϕ 和效果 l，其中 true 表示没有效果；

17）无限有意动作序列集 $\Sigma=\{\alpha_0,\alpha_1,\cdots\mid\alpha\in IntA\}$，其中 $\Sigma\in IntA^*$。有意动作序列集在人工智能中被称作规划，在本文中，为与 AgentSpeak 中的规划区分开来，仍将其称为有意动作序列集。

在给出了 AgentSpeak 中使用的字符串集合之后，则可在此基础上给出 3 种规划：①原始规划；②任务规划；③反应式规划。以下是 3 类规划的形式化语义。

（1）原始规划：

$PP=\{\eta:x\leftarrow\beta\mid\eta\in\{\pm q\in AGoal\cup QGoal\},x\in GPS^*,\beta\in IntA^*\}$；

（2）任务规划：

$TP=\{\eta:x\leftarrow\beta\mid\eta\in\{\pm q\in AGoal\cup QGoal\},x\in GPS^*,\beta\in(IntA^*\cup AGoal)^*\}$；

（3）反应式规划：

$RP=\{\eta:x\leftarrow\beta\mid\eta\in IAS,x\in GPS^*,\beta\in(IntA^*\cup AGoal)^*\}$。

其中原始规划的触发条件是一个目标的增加或删除（$\pm g$），规划主体是有意动作的集合，不包括任何子目标的增加或删除；任务规划的触发条件和前提条件与原始规划完全相同，不同的是规划主体包括一个或多个子目标的增加或删除，也就是规划主体的执行过程中还需其他规划来处理这个子目标的增加或删除；反应式规划的触发条件是信念改变而不是一个目标的增加或删除。

18）$fulfill$ 语句 $=\{fulfill(p,\sigma,bs)\mid p\in PP\cup TP\cup RP,$

$\sigma\in\Sigma\}$，Agent 在信念集 bs 为真的情况下开始执行有意动作序列 σ，可以实现规划 p 的目标。

3 AgentSpeak 的模型论语义

在证明 AgentSpeak 中意图执行的可靠性与完备性之前，需给 AgentSpeak 的语法构件赋予语义。在本节中，使用模型论语义[18]给上节所构造的语法构件赋予语义。

3.1 AgentSpeak 解释系统

第 2 节所给出的语法结构可通过结构为三元组的模型 $\mathcal{M}=\langle\mathcal{S},\mathcal{A},\mathcal{I}\rangle$ 来表示，其中 $\mathcal{S}$ 为信念库状态，$\mathcal{A}$ 为有意动作，$\mathcal{I}$ 为意图执行。以下对这些符号进行具体描述：

1）$\mathcal{S}=2^{GPS}$，Agent 所有可能信念的集合，其中每个状态 $s\in\mathcal{S}$ 都是一个集合，由当前状态为真的所有信念组成。

2）$A:IntA\times C^*\times\mathcal{S}\to\mathcal{S}$，是一个有意动作的解析。给定 $IntA$ 中的一个有意动作符号、C 中的（与这个有意动作符号相对应的）落实参数和当前的信念状态，$\mathcal{A}$ 将会给出 Agent 在执行这个动作之后所对应的信念状态。简单地说，$\mathcal{A}$ 定义了有意动作的执行效果。

3）$\mathcal{I}:E\times PS\to 2^{PS}$，将一个事件和匹配该事件的规划 $p\in PS$ 映射为原始规划集。因为如果 p 是非原始规划，那么可能存在多个规划对应一个事件，故结果是原始规划的集合。其中函数 $\mathcal{T}(e,p)$ 的转变规则如下定义：

设 $p\in PP$ 是原始规划，且形式为 $p=\eta:\chi\leftarrow\alpha_0,\cdots,\alpha_k$，则 $\mathcal{T}(e,p)=\{(\text{true}:\text{true}\leftarrow\alpha_0',\alpha_1',\cdots,\alpha_k')\}$，其中 $\alpha_0'.\phi=\alpha_0.\phi\wedge\chi,\alpha_i'=\alpha_0\cdot\theta\circ\theta',i\neq 0$。

即原始规划的扩展可通过将规划的触发条件和前提条件分别合成到有意动作的内部，而将规划的触发条件和前提条件设置为 true 来完成。其中合一变量 θ 和 θ' 分别定义为 $\theta=mgu(e,p.\eta)$ 和 $\theta'=mgu(\alpha_i,p.\chi)$。

如果 $p\in TP\cup RP$，即 p 为任务规划或反应式规划。不失一般性，设 $p=\eta:\chi\leftarrow\alpha_0,\cdots,\alpha_j,\cdots,\alpha_k$，其中 $\alpha_j=!\ g_j\in AGoal,j\in[0,k]$，那么 $\mathcal{T}(e,p)=\{\alpha_0',\cdots,\alpha'_{j-1},\mathcal{T}(\eta_j,p_{j_m})\cdot\theta\circ\theta',\alpha'_{j+1},\cdots,\alpha_k'\}$，其中 $p_{j_m}\in p_j$ 是一个满足事件 $!\ g_j$ 的触发规划。

即非原始规划的扩展可通过将规划体中达成型目标替换成与之相对应的子规划来实现。$\alpha_i'=\alpha_i\cdot\theta\circ\theta'$ 的定义和上面所定义的相同，p_j 是对应于 α_j 的规划集。由于可能同时存在多个规划对应于该子目标，因此结果是一个原始规划集。

3.2 AgentSpeak 满足关系

在定义了本文 2.2 小节中所提出的语法构件的解释系统之后，还需寻找到能够满足这个解释系统的模型。因为 Agent 程序 ag 是由信念集 bs 和规划集 ps 构成的，所以 ag 满足关系可分为信念集 bs 的满足关系和规划集 ps 的满足关系。

Agent 的世界状态是通过信念原子集合来表示的。如果信念的表达式包含在信念集中，并且这个信念集是可满足的，那么这个信念就是可满足的。

定义 1（信念集满足关系） 给定信念集 $bs=\{b_0,\cdots,b_n\}$ 和满足这个信念集的模型 $\mathcal{M}=\langle\mathcal{S},\mathcal{A},\mathcal{I}\rangle$，如果 $bs\in\mathcal{S}$，那么 $\mathcal{M}\models bs$。

自治性是 Agent 区别于其他程序个体的显著特性，而体现自治性的重要因素就是 Agent 具有理性行为。基于此，可假设 Agent 有能力知道 Agent 将执行的动作效果。

假设 1 Agent 知道将要执行的动作效果，即 Agent 的动作是有意动作。

在定义了假设 1 之后，则可定义有意动作的满足关系。

定义 2（有意动作的满足关系） 给定有意动作 $at=[\phi,\alpha(t\cdots),l\cdots]$ 和模型 $\mathcal{M}=\langle\mathcal{S},\mathcal{A},\mathcal{I}\rangle$，模型 $\mathcal{M}$ 满足有意动作 at，当且仅当，对于任一落实的代换 θ 和任一信念状态 s，模型 $\mathcal{M}$ 中的有意动作集 $\mathcal{A}$，即 $\mathcal{A}_{\mathcal{M}}$，满足如下属性：

1）如果 $l_1\cdot\theta,\cdots,l_m\cdot\theta$ 在 s 中为真，那么 $\mathcal{A}_{\mathcal{M}}(\alpha,v_1\cdot\theta,\cdots,v_k\cdot\theta,s)=s'$；

2）否则，$\mathcal{A}_{\mathcal{M}}(\alpha,v_1\cdot\theta,\cdots,v_k\cdot\theta,s)$ 无定义。

其中 $s,s'\in BS,s\vdash\phi\cdot\theta,s'=s\cup l^+\cdot\theta/l^-\cdot\theta,l^+\cdot\theta$ 为 $(\tau.l)\cdot\theta$ 的正向文字，$l^-\cdot\theta$ 为 $(\tau.l)\cdot\theta$ 的负向文字。

下面定义满足 $fulfill(p,\sigma,bs)$ 的模型 $\mathcal{M}$，即对于模型 $\mathcal{M}$，以信念集 bs 开始，通过执行动作序列 σ 来完成规划 p 中目标所应满足的条件。

在考虑动作序列 σ 与信念集 bs 以及规划 p 的关系之前，有必要分析触发以及执行动作实例所需的条件。根据规划的定义，可得出下面结论：

1）如果当前事件可满足一个规划的触发事件，那么这个规划就是可触发的；

2）如果当前信念集可满足一个规划的上下文条件，那么这个规划是可执行的。

如果规划 p 所得的规划实例 $p'\in\mathcal{T}(e,p)$ 是可执行的，那么这个规划的触发事件和上下文条件一定是可满足的。因此，可得到规划实例的表达形式为 $p'=\text{true}:\text{true}\leftarrow p.body\cdot\theta\circ\theta'$，其中 $\theta=mgu(e,p.\eta),\theta'=mgu(bs,p.\chi)$，$e$ 是当前事件，$mgu(X,Y)$ 表示 X 与 Y 的合一代换。

根据本文 2.3 小节所述，规划可分成 3 种：①原始规划；②任务规划；③反应式规划。对于 $fulfill$ 语句的满足关系，需根据规划 p 的结构考虑满足 $fulfill$ 语句所需的不同条件。

首先考虑当规划 p 是原始规划的情形。假设模型 $\mathcal{M}$ 满足语句 $fulfill(p,\sigma,bs)$，如果有意动作序列 σ 能达成原始规划 $p\in PP$ 的目标，那么 σ 中的任一动作都必须与根据规划 p 得出的规划实例 p' 中的有意动作具有同样的顺序和同样的项。下面是该描述的形式化表示。

给定根据规划 p 所得到的原始规划实例 $p'=\text{true}:\text{true}\leftarrow\alpha_0(t_{00},\cdots,t_{0m_0}),\cdots,\alpha_i(t_{i0},\cdots,t_{im_i}),\cdots,\alpha_k(t_{k0},\cdots,t_{km})$，以及有意动作序列 $\sigma=\alpha'_0(t'_{00},\cdots,t'_{0m'_0}),\cdots,\alpha'_i(t'_{i0},\cdots,t'_{im'_i}),\cdots,\alpha'_k(t'_{k'0},\cdots,t'_{k'm'_k})$，如果存在模型 $\mathcal{M}$ 满足 $fulfill(p,\sigma,bs)$，则有意动作序列 σ 满足下列条件：

1）$k'=k,\alpha'_i=\alpha_i(0\leqslant i\leqslant k)$，也就是 σ 中所包含的动作与规划实例 p' 中所包含的动作是一样的；

2）$m'=m,t_{ij}=t'_{ij}$，也就是 σ 中所包含动作的项与规划实例 p' 中所包含动作的项是一样的。

然后，考虑 p 是非原始规划的情形。如果 p 是非原始规划，则规划的主体内部一定包含子目标，因此主规划 p 的满足条件取决于内部子目标的满足情况。假设 p' 是对该子目标进行扩展之后所得到的规划实例，可得出满足规划 p 的模型一定满足规划实例 p'。该情况的形式化语义描述如下：假设模型 $\mathcal{M}$ 满足非原始规划 p 的 $fulfill$ 语句 $fulfill(p,\sigma,$

bs)，如果存在规划实例 $p' \in \mathcal{T}_{\mathcal{M}}(p)$，那么模型 $\mathcal{M}$ 也满足 $fulfill(p', \sigma, bs)$。

以下给出规划库的满足关系。模型 $\mathcal{M}$ 满足规划库 PL，当且仅当 PL 中的规划都被满足。模型 $\mathcal{M}$ 满足规划 p 不仅需模型 $\mathcal{M}$ 满足规划 p 本身，还需当该计划作为实现步骤插入到别的计划动作序列后，得到的新动作序列能够实现被插入计划的目标。

定义 3(Agent 满足关系)　如果模型 $\mathcal{M}$ 满足 Agent 的信念集 bs 和规划集 ps 的所有构件，那么模型 $\mathcal{M}$ 满足这个 Agent。

4　AgentSpeak 的操作语义

操作语义(Operational Semantics)指的是给程序或程序系统行为精确描述的语义。换句话说，操作语义是用来描述程序和系统是怎样执行或者运行的。在过去的 20 年时间里，在 Plotkin[19]，Milner[20]，Hoare[21]，Serbanuta[22] 和其他人的贡献之下，针对语法"结构化"框架的研究取得了非常大的进展。用 Plotkin 的话来说，操作语义的研究是"能够明确说明程序语言语义的简单而直接的方法"，操作语义几乎不需要任何数学背景知识，却提供了"简洁而易于理解的语义定义"。最具函数语言特性的 Standard ML[23,24]，在整体上就是一个操作语义框架。

4.1　AgentSpeak 的格局

Moreira 等在文献[25]中，给出了 AgentSpeak 语言的操作语义。为了本文证明的需要，以文献[25]所给出的 AgentSpeak 语言的执行流程为基础，本文对该操作语义进行了修改。

第 2 节中关于 AgentSpeak 的推理循环可总结为图 3 所示的结构，图中所述的步骤如下所示：

1)ProcMsg 为消息处理步骤，用于处理 Agent 接收到的消息；

2)SelEv 为事件选择步骤，用于选择一个事件进行处理；

3)RelPl 为计算相关规划步骤，即根据规划的触发事件以及当前选择事件计算出相关规划集；

4)ApplPl 为计算可用规划步骤，即根据规划的上下文条件计算出可用规划集；

5)SelAppl 为选择可用规划步骤，即从可用规划集中选择一个规划执行；

6)AddIM 为意图添加步骤，该步骤将上一步选择的规划进行实例化，然后添加到意图库中；

7)SelInt 为意图选择步骤，即在意图库中选择一个合适的意图，以备下面执行；

8)ExecInt 为意图执行步骤，即执行选定的意图；

9)ClrInt 为意图清理步骤，即清除规划库中已执行过的意图。

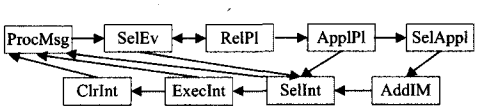

图 3　AgentSpeak 推理循环

为了给出 AgentSpeak 的操作语义，需定义 Agent 的格局，而操作语义就是定义在这个格局之上的转换规则。

定义 4(Agent 的格局)　Agent 的格局是形为 $\langle ag, C, T, st\rangle$ 的四元组，其中：

1)ag 是 Agent 程序，包括一个信念集和规划集。在 AgentSpeak 中，信念集为落实的原子集合。

2)Agent 的环境 C 是一个形为 $\langle I, E, A\rangle$ 的三元组，其中，

(1)I 是形为 $intentions\{i, i', \cdots\}$ 的意图集合，而每个意图 i 是一个部分初始化的规划实例；

(2)E 是形为 $events\{(te, i), (te', i'), \cdots\}$ 的事件集，而每个事件是一个触发事件与意图的元组对 (te, i)，te 是一个触发事件，i 是一个意图；

(3)A 是 Agent 选择用来执行的动作集。

3)T 是具有临时信息的元组 $\langle R, Ap, \iota, \varepsilon, \rho\rangle$。元组中的各组件定义如下：(i)$R$ 是相关规划集；(ii)Ap 是可用规划的集；(iii)ι，ε 和 ρ 分别为特定的意图、事件和可用规划。

4)一个 Agent 推理周期中的当前步骤 st 通过 $st \in$ {SelEv, PelPl, ApplPl, SelAppl, AddIM, SelInt, ExecInt, ClrInt}来表示。

在一般情况下，Agent 的初始结构定义为 $\langle ag, C, T,$ SelEv$\rangle$，程序 ag 通过 Agent 程序给出，C 和 T 的所有组件为空，Agent 的初始步骤为 SelEv。为保持语义规则的简约，在下文中使用如下的规定：如果 C 是一个 AgentSpeak 的运行环境，那么用 C_E 表示 C 的 E 组件。与此类似，转换系统结构中的其他组件也如此表示。另外，用结构 $i[p]$ 将规划 p 放入意图 i 的顶部形成新意图。

4.2　AgentSpeak 的操作语义

AgentSpeak 的操作语义可表示为 Agent 格局序列之间的转移规则。这个转移规则可如下表示：

$$\frac{\text{转换条件}}{\text{转换前的格局}\rightarrow\text{转换后的格局}} \tag{1}$$

基于此，寻找相关规划的工作可用下面的转移规则进行形式化：

$$\frac{T_\varepsilon = \langle te, i\rangle \quad \text{RelPlans}(ag_{ps}, te) \neq \{\}}{\langle ag, C, T, \text{RelPl}\rangle \rightarrow \langle ag, C, T', \text{ApplPl}\rangle} \tag{2}$$

式中，$T_R' = \text{RelPlans}(ag_{ps}, te)$；

$$\frac{T_\varepsilon = \langle te, i\rangle \quad \text{RelPlans}(ag_{ps}, te) = \{\}}{\langle ag, C, T, \text{RelPl}\rangle \rightarrow \langle ag, C, T, \text{SelInt}\rangle} \tag{3}$$

上面的转移规则根据相关规划集的结果是否为空分别进行定义。式(2)表示定义相关规划集不为空的情况。如果相关规划集不为空，则将 T 中的 R 组件初始化为由附属函数 RelPlans 所确定的相关规划集，然后将推理循环进行到下一步(ApplPl)，以确定 T 中可用的规划；式(3)表示如果事件没有相关规划，那么就丢掉这个事件和它的意图，然后从事件集中选择另一个事件进行处理。如果没有事件需要处理，那么跳至意图执行步骤。

其他转移规则与此类似，因为篇幅关系，此处不再赘述，详情可参见文献[25]。

在定义了 AgentSpeak 的操作语义之后，可根据该操作语义得出 Agent 程序 ag 针对特定规划 p 所产生的有意动作序列 $recipe(p, ag)$。

1 $i=0$；

2 对于规划主体 $p.body$ 中的每一个元素 b(有意动作，或者达成类目标)：

2.1 如果 b 是有意动作，那么将动作添加到意图中 $i=i\oplus b$；

2.2 如果 b 是达成类目标，那么 $i=i\oplus recipe(+!\ b, ag)$；

2.2.1 如果 p 是一个原始规划，那么只会执行语句 2.1，也就是说只会把 $p.body$ 中的意图放入到 i 中。

2.2.2 如果 p 是一个任务规划或者反应式规划，假设 b 是一个达成类目标，意图可以递归定义为 $i=i\oplus recipe(+!\ b, ag)$，使用类似于 2.2 节所定义的 $fulfill$ 语句所使用的方法，可得出类似的结果。

3 返回 i。

5 等价定理及意图生成与执行的可靠性与完成性

Agent 意图执行问题可描述为：给定事件$\langle e,i\rangle$和 Agent 程序 $ag=\langle bs,ps\rangle$，通过执行程序 ag 所产生的意图，Agent 将会达到目标状态。在这个证明问题中，前提条件是信念集 bs 和触发事件 e；公理集是有意动作 $IntA$ 和规划库 ps；要证明的公式是目标状态 s_g。从形式上来说，触发事件 e 可以是达成型目标 $\pm!\ \eta\in AGoal$ 或是信念更新 $\pm b|b\in BS$。

如果触发事件 e 的形式为 $\pm b$，那么可通过函数 $RelPlans$ 将触发事件转换为动作和目标公式的序列集；如果触发事件 e 的形式为 $\pm!\ \eta$，则可使用 ps 分解目标，以得到目标状态。所以，证明过程 Δ 就是有意动作和规划的序列。有意动作序列 σ，可看作是从证明过程中过滤得到的有意动作。

定理 1（Agent 意图等价定理） 给定一个 Agent 程序 $ag=\langle bs,ps\rangle$、规划 $p\in ps$ 以及有意动作序列 $\sigma,\sigma\in recipe(p,ag)$ 成立，当且仅当 $ag|=fulfill(p,\sigma,bs)$。

证明：证充分性。

因为 $\sigma\in recipe(p,ag)$，故存在满足 ag 的模型 $\mathcal{M}$ 也满足 $fulfill(p,\sigma,bs)$。由于 $recipe(p,ag)$ 是以完成和归约的方式定义的，因此可通过分解 $recipe(p,ag)$ 的结构来证明该命题。

①假设规划 p 是一个原始规划，也就是 $p\in PP$，需要证明如果 $\sigma\in recipe(p,ag)$，那么满足 ag 的模型也满足 $fulfill(p,\sigma,bs)$。

因为 p 是原始规划，根据 2.2 节所定义的 $recipe$ 的函数可得出，此时 $recipe(p,ag)$ 的定义与满足 $fulfill(p,\sigma,bs)$ 的模型对 σ 的定义完全相同，都等于 $\mathcal{T}(g)$，因此可得出如果 $p\in PP,\sigma\in recipe(p,ag)$，当且仅当 $\mathcal{M}|=fulfill(p,\sigma,bs)$。

②假设规划 p 是非原始规划，也就是 $p\in TP\cup RP$，需证明如果 $\sigma\in recipe(p,ag)$，那么满足 ag 的模型也满足 $fulfill(p,\sigma,bs)$。

因为规划 p 是非原始规划，也就是 $p\in TP\cup RP$，那么可得出存在另一个规划 p'，和 p 中的一个节点 n，满足 $p''=recipe(p',recipe(p,ag))$。设 $\mathcal{M}$ 是满足 ag 的模型，并且也满足 $recipe(p,ag)$，因为函数 $recipe$ 是递归定义的，故可得出该情况最后也是①的情形，因此可得出如果 $p\in TP\cup RP,\sigma\in recipe(p,ag)$，当且仅当 $\mathcal{M}|=fulfill(p,\sigma,bs)$。

从①和②可得出如果模型满足 $\sigma\in recipe(p,ag)$，那么该模型也满足 $fulfill(p,\sigma,bs)$。

证必要性。用反证法证明。

只需证明并不存在一个 $\sigma'\notin recipe(p,ag)$，并且模型 $\mathcal{M}$ 满足 $fulfill(p,\sigma',bs)$。

下面根据条件来构造模型 $\mathcal{M}=\langle\mathcal{S},\mathcal{A},\mathcal{T}\rangle$。设 $\mathcal{S}=2^{\{\text{落实原子集合}\}}$。

$\mathcal{A}(f,c_1,\cdots,c_k,s)=(s-N\cdot\theta)\cup(P\cdot\theta)$，其中操作符 f 的形式是 $(f(v_1,\cdots,v_k),l_1,\cdots,l_k)$，$\theta$ 是形如$\{c_i/v_i\mid 1\leqslant i\leqslant k\}$的代换，$N,P$ 为$\{l_1,\cdots,l_k\}$中的负、正文字。

$\mathcal{T}(p)=\{p'\mid p'$是从 p 经过有限次的约简之后所取得规划的落实实例$\}$，其中 p 是任意非原始规划，即任务规划或者反应式规划。

根据 $\mathcal{A}$ 的定义方式，可得出所有 ps 的操作都能得到满足。根据 $\mathcal{T}$ 的定义方式，可得出只要 $p'\in recipe(p,ag)$，就可得到 $\mathcal{T}(p')\subseteq\mathcal{T}(p)$。因此，可得出 $\mathcal{M}$ 满足 ps 中的所有规划，从而可得出 $\mathcal{M}|=ag$。

给定一个原始规划 p，并且 $\sigma\notin recipe(p,ag)$。那么 $\sigma\notin fulfill(p,\sigma,bs)$也成立。在充分性证明中，已经得到对于满足运算符的任意一个模型，$fulfill(p,\sigma,bs)$就是解决原始规划 p 的动作集。因此可得出模型 $\mathcal{M}$ 不满足 $fulfill(p,\sigma,bs)$。

考虑另一种情况，p 是满足 $\sigma\notin recipe(p,ag)$的非原始规划。假设 $\mathcal{M}$ 满足 $fulfill(p,\sigma,bs)$，那么存在一个原始规划 $p'\in\mathcal{T}(p)$，使得 σ 是 p'的一个有序动作集。从充分性证明可得出，σ 必须为包含在 $recipe(p',ag)$中的一个有序动作集。但是因为 $\mathcal{T}(p)$只包含可以从 p 中经过有限次约简所得到的原始规划，所以可得出 $\sigma\in recipe(p,ag)$。产生矛盾。证毕。

推论 1（意图完成目标的可靠性定理） 给定事件 e 和 Agent 程序 ag，如果存在模型 $\mathcal{M}$ 满足 ag，也就是满足 Agent 的信念库和计划库，那么该模型 $\mathcal{M}$ 所产生的意图能够满足对事件 e 的规划目标。

推论 2（意图完成目标的完备性定理） 给定事件 e 和 Agent 程序 ag，如果不存在模型 $\mathcal{M}$ 能够同时满足 ag 以及针对事件 e 所产生的意图，那么 Agent 不可能完成事件 e 的规划目标。

结束语 综上所述，本文提出了一种 Agent 形式化分析方法，用于分析软件 Agent 的意图能否完成其目标。本方法首先根据 Agent 的语法结构构造出软件 Agent 的形式化系统，进而通过该形式化系统的模型论语义给出 Agent 执行意图完成目标的可满足条件；然后分析软件 Agent 的操作语义同其模型论语义在表示意图时的关系，以得到软件 Agent 在自治情况下通过执行意图完成目标的性质。本文以当前最流行的 BDI 结构 Agent 编程语言 AgentSpeak 为例对 Agent 的理性行为进行了形式化分析，得出了 Agent 中意图生成与执行的可靠性与完整性定理，验证了 AgentSpeak 中意图完成目标的正确性。

参 考 文 献

[1] Wooldridge M. An introduction to multiagent systems[M]. Wiley, 2009

[2] Shoham Y. Agent-oriented programming [J]. Artificial intelligence, 1993, 60(1): 51-92

[3] Bernon C, Cossentino M, Pavón J. Agent-oriented software engineering[J]. The Knowledge Engineering Review, 2006, 20(2): 99-116

[4] Rao A S, Georgeff M P. BDI agents: From theory to practice[C]// AAAI Press/The MIT Press, 1995: 312-319

[5] 吴朝晖，忻栋，潘云鹤. 基于对策论的 MAS—BDI 主体模型[J]. 计算机科学，2001，28(009)：5-8

[6] Zambonelli F, Jennings N R, Wooldridge M. Developing multiagent systems: The Gaia methodology[J]. ACM Trans Softw Eng Methodol, 2003, 12(3): 317-370

[7] DeLoach S, Wood M, Sparkman C. Multiagent systems engineering[J]. International Journal of Software Engineering and

Knowledge Engineering,2001,11(3):231-258
[8] 李超明,苏开乐.一个基于智能的 MAS 模型及其方法论[J].计算机研究与发展,2007,44(6):980-989
[9] 张新良,石纯一.多 Agent 联盟结构动态生成算法[J].软件学报,2007,18(3):574-581
[10] Jo C,Chen G,Choi J. A new approach to the BDI agent-based modeling[C]// ACM. 2004:1545
[11] 毛新军,屈婷婷,王戟.自适应多 Agent 系统的面向 Agent 软件开发方法学 ODAM[J].计算机研究与发展,2008,45(011):1892-1901
[12] Bordini R, Hübner J. BDI Agent Programming in AgentSpeak Using Jason(Tutorial Paper)[M]. Berlin, German: Springer, 2006
[13] Bellifemine F,Caire G,Poggi A,et al. JADE:A software framework for developing multi-agent applications. Lessons learned [J]. Information and Software Technology, 2008, 50(1/2): 10-21
[14] Rao A. AgentSpeak(L):BDI agents speak out in a logical computable language[C]// Eindhoven. The Netherlands: Springer, 1996:42-55
[15] Bratman M E. Intention, Plans and Practical Reason[M]. Cambridge: Harvard University Press, 1987
[16] 路军,王亚东,王晓龙.BDI Agent 解释器的研究和改进[J].软件学报,2000,11(8):1118-1125
[17] Bordini R, Moreira Á. Proving BDI Properties of Agent-oriented Programming Languages: The asymmetry thesis principles in AgentSpeak(L)[J]. Annals of Mathematics and Artificial Intelligence, 2004, 42(1): 197-226
[18] Blackburn P, De Rijke M, Venema Y. Modal logic[M]. Cambridge Univ Pr, 2001
[19] Plotkin G. A structural approach to operational semantics[R]. Department of Computer Science, Aarhus University, 1981
[20] Milner R. Operational and algebraic semantics of concurrent processes[R]. Edinburgh: Department of Computer Science, University of Edinburgh, 1990
[21] Moore J. Inductive assertions and operational semantics[J]. International Journal on Software Tools for Technology Transfer (STTT), 2006, 8(4): 359-371
[22] Serbanuta T F, Rosu G, Meseguer J. A rewriting logic approach to operational semantics [J]. Information and Computation, 2009, 207(2): 305-340
[23] Milner R, Tofte M, Macqueen D, et al. The definition of standard ML: revised[M]. The MIT Press, 1997
[24] Wikstrom A. Functional programming using standard ML[M]. Prentice Hall International(UK) Ltd., 1987
[25] Moreira A, Bordini R. An operational semantics for a BDI agent-oriented programming language[C]// Toulouse. France, 2002: 45-59

(上接第 221 页)

计算中同样也存在着参数的选择问题,参数 a 指定最小阈值的分界点,一般可取 0.5,而参数 b 决定最大阈值的起始点,控制着 $S(r;a,b)$ 函数的斜率,可以想象,若 ε_{max} 越大,b 越小,边界对象将越多,ε_{max},b 共同控制着边界,对算法提供了一种灵活的调整机制,图 8 是针对 data1,不同 ε_{max},b 值时,边界元素的个数情况。从图 8 中可以看到,随着 ε_{max} 的增大,b 相应增大依然可以达到较好的效果。

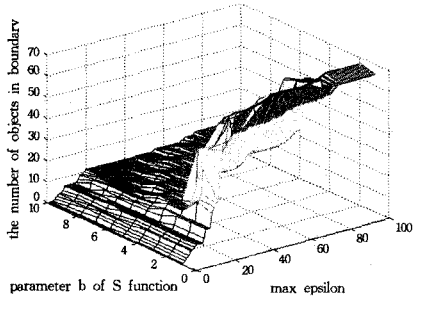

图 8 边界大小 VS 参数

结束语 聚类在数据挖掘、模式识别、图像分割等领域有着重要的应用。聚类算法中,C 均值算法是一类最经典的算法,粗糙 C 均值算法通过引进上下近似,改善了经典 C 均值算法分类的绝对性,承认了类间边界的存在。但粗糙 C 均值算法引入的参数带了算法的使用的困难,参数的设置对粗糙 C 均值算法的应用具有关键性的作用,本文提出了动态调整阈值 ε 的方法,通过最大阈值和最小阈值的设定从一定程度上降低了选择阈值的困难,实验结果也表明算法具有较好的效果,下一步的方向应该集中在如何在一个统一的框架下动态调整算法的所有参数。

参 考 文 献

[1] Pawlak Z. Rough Sets [J]. International Journal of Computer and Information science, 1982(11): 241-356
[2] Bezdek J C. Pattern recognition with fuzzy objective function algorithms[J]. New York: Plenum, 1981
[3] Lingras P, West C. Interval set clustering of Web users with rough k-means[R]. 2002-002. Department of Mathematics and Computer Science, St. Mary's University, Halifax, Canada, 2002
[4] Peters G. Some refinements of Rough K-means clustering[J]. Pattern Recognition, 2006(39): 1481-1491
[5] Mitra S. An evolutionary rough partitive clustering[J]. Pattern Recognition Letters, 2004(25): 1439-1449
[6] Malyszko D, Stepaniuk J. Rough Entropy Based k-Means Clustering[C]// RSFDGrc. 2009: 406-413
[7] Mitra S, Banka H, Pedrycz W. Rough-Fuzzy Collaborative[J]. IEEE transactions on systems, man, and cybernetics-part B: cybernetics, 2006, 4(36): 795-805
[8] 孙吉贵,刘杰,赵连宇.聚类算法研究[J].软件学报,2008(1):48-61
[9] 王丹,吴孟达.粗糙模糊 c 均值算法及其在图像聚类中的应用[J].国防科技大学学报,2007(2):76-80
[10] 郑超,苗夺谦,王睿智.基于密度加权的粗糙 K-均值聚类改进算法[J].计算机科学,2009,36(3):220-222
[11] 邵锐,巫兆聪,钟世明.基于粗糙 K 均值算法在图像分割中的应用[J].测绘信息与工程,2005(5):1-2
[12] Herawan T, Deris M M, Abawajy J H. A rough set approach for selecting clustering attribute [J]. Knowledge-Based Systems, 2010, 23: 220-231
[13] Miao Duo-qian, Duan Qi-guo, Zhang Hong-yun, et al. Rough set based hybrid algorithm for text classification[J]. Expert Systems with Applications, 2009, 36: 9168-9174
[14] Sarkar M. Fuzzy-rough nearest neighbor algorithms in classification[J]. Fuzzy sets and systems, 2007, 158: 2134-2152