维普资讯 http://www.cqvip.com

多智能体的角色与结构设计分析

余 斌[1,2],李龙澍[1],李学俊[1]
(1.安徽大学 计算机科学与技术学院,安徽 合肥 230039;
2.安庆师范学院 计算机与信息学院,安徽 安庆 246011)

摘 要:角色是面向Agent软件开发方法研究中的一个重要抽象概念。文中认为角色是连接多Agent系统(MAS)微观模型与宏观模型的桥梁,角色与Agent之间的动态性有利于刻画多Agent系统的结构和行为模型。RoboCup是多Agent系统研究的一个很好的平台。建立一支成功的机器人足球队需要很多领域的知识,合理的模型结构和Agent之间的协调与协作是RoboCup比赛中赢球的关键所在。协调与协作是多Agent系统研究的重要课题。

关键词:角色;结构;协作
中图分类号:TP18 **文献标识码**:A **文章编号**:1673-629X(2006)07-0073-03

Analysis of Multi-Agent Role and Structure Design

YU Bin[1,2],LI Long-shu[1],LI Xue-jun[1]
(1.School of Computer Science & Technology,Anhui University,Hefei 230039,China;
2.School of Computer & Information, Anqing Normal College, Anqing 246011,China)

Abstract:Role is an important abstract concept on develop method of Agent-oriented software. In this paper,role is regarded as the bridge between micro and macro model of multi-Agent system(MAS),and relationship between role and Agent,such as the dynamic relation facilitates the depiction of structure and behavior model of multi-Agent system. RoboCup is an excellent platform for multi-Agent. Various fields knowledges make up of a success RoboCup team. Reasonable model-structure and the correspondency and collaboration are the key to prevail in the RoboCup matches,so they are the important problem in the multi-Agent researches. Aiming at the problem of the multi-Agent collaboration in RoboCup emulational matches.

Key words:role;structure;collaboration

0 引 言

无论是现实世界中的智能机器人或机器人团队还是网络空间中的软件自主体,都可以抽象为具有自主性、社会性、反应性和能动性的"Agent"(智能体)。由这些自主体以及相关的人构成的多Agent系统MAS(Multi-Agent Systems),是未来物理和信息世界的一个缩影[1]。多Agent系统的元概念模型以Agent为基础,它认为系统是由多个相互合作的自主Agent所构成,这是面向Agent软件与面向对象软件的根本区别。作为解决分布式复杂问题的一种重要手段,多Agent系统开始得到越来越广泛的重视。其基本问题是自主体(包括人)之间的协调与发展,主要研究内容包括Agent设计、多Agent系统体系结构、Agent协商与合作、自动推理、规划、机器学习与知识获取、认知建模、系统生态和进化等一系列专题。

收稿日期:2005-10-10
基金项目:国家自然科学基金资助项目(60273043)
作者简介:余 斌(1981-),男,安徽安庆人,硕士研究生,研究方向为智能软件开发、多智能体研究等;李龙澍,教授,博士生导师,研究方向为智能软件开发。

在面向Agent方法研究中,软件工程专家引入了角色概念。角色视为系统分析设计中的一个独特概念,作为面向Agent设计的基础,揭示出角色具备的一些重要特性,比如依赖性、动态性等。角色在分析到实现整个开发过程中都具有重要的地位和作用,系统的目标由各个角色承担,Agent间的合作交互则通过绑定在其上的角色决定,使得Agent角色的动态变化可自然实现。RoboCup是一个典型的Multi-Agent系统,文中的一些理论知识通过RoboCup领域里的球员的角色及其架构等具体实例来表现,自然、直观。

1 角色概念

角色被理解为接受信息、加工信息和发送信息的抽象对象。角色概念曾用于管理信息系统的自动生成工具的研究,其中角色指管理信息系统的基本单元。角色理论认为,角色是责任和权利的统一体。其中责任规定了角色的行为规范和约束,也就是说,角色是某一类对象结构、性质、行为、职能等方面所共有的特征集合。它具有目标、能力、责任(obligation)许可、约束和协议等对象多方面本质特征的综合反映,能够作为事物分类的合理标准[2]。

维普资讯 http://www.cqvip.com

在对多 Agent 系统的研究中,作为连接多 Agent 系统宏观模型与微观模型的桥梁,一些研究人员引入了角色概念进行系统分析和设计。在多 Agent 系统中,角色定义如下:从概念角度讲,角色是一种约束,在这种约束下,Agent 参与某些交互和以某种方式进行演变。从执行角度上看,角色是 Agent 的某些属性和行为的封装并且绑定在 Agent 上。目前的研究认为角色与 Agent 之间的关系具有如下一些典型性质:

(1)动态性:在 Agent 生命周期内,可以为其添加新的角色,也可以撤消原有角色。

(2)依赖性:角色不能独立于 Agent 存在,它必须依托于某个 Agent。

(3)多重性:同一 Agent 可能在同一时期承担多个不同的角色;同一 Agent 也可能在不同时期承担不同的角色。由于 Agent 可以承担不同的角色,带给其它 Agent 不同的视点,从而体现出更加全面细致的认知。

在分析阶段,对球队进攻和防守职责(即球员角色)的分析是围绕球队总体目标进行的。在基于角色的仿真足球队设计中,球员不具备具体的角色是没有意义的,因此每个球员至少应该具备一个角色;同时,从设计的合理性考虑,一个角色应当只从属于一个球员,否则,一旦出现某个角色绑定到多个球员的情况,每个球员的职责就会不明确,这种情况下,该角色应该分解细化。在仿真足球队中,球员的角色是比较鲜明的,球员的角色按照具体的阵型可以分为多种基本角色类,例如在 442 阵型中,相应的角色类包括:左右前锋、左右前卫、前后腰、左右后卫、盯人中卫、拖后中卫、守门员。

2 角色转换与合作

在获取了球员角色类的基础上,利用 UML 方法,分析球员比赛过程中的用例情况,对各角色之间的合作和转换关系进行刻画。例如球员通过场上队长的指挥协调进行传递,球员之间的合作可以通过其角色合作关系图描述;球员之间进行战术配合时,根据形势可能会交换场上的位置,这种情况可以通过角色转换关系图进行描述。442 阵型的角色合作和转换关系实例如图 1 所示[3]。

图中的椭圆框代表角色类,矩形框代表球员,Agent 角色之间的连线代表角色的合作与转换关系。带箭头的连线以及上方的文字分别表示角色之间传递的消息和传递的方向。

大体上,进攻单元有两个角色:积极进攻和消极进攻;防守单元也有两个角色:消极防守和积极防守;中间单元中的两个角色为:辅助进攻和辅助防守。很显然,角色主要有两种状态:进攻状态和防守状态。球员处于进攻状态时,如果是控球队员,则根据当前的情况处理球。当在射门范围而且有比较好的进球机会时,则立即射门;否则,把球传给有更好机会的队友;如果没有队友接应,则自己带球作转移,一般是向守方队员密度小的方向转移,继续寻找机会。在防守状态中,很重要的一个概念是防守反攻,所以每时每刻都应该留意球,判断是否能把球抢断下来。如果断球成功,则要分析当前的情况进行分别处理;如果断球失败,则要进行盯人防守策略。比赛时攻防状态的变换主要是由球被哪方所控制来进行的。当球在己方控制时,则进入进攻状态。

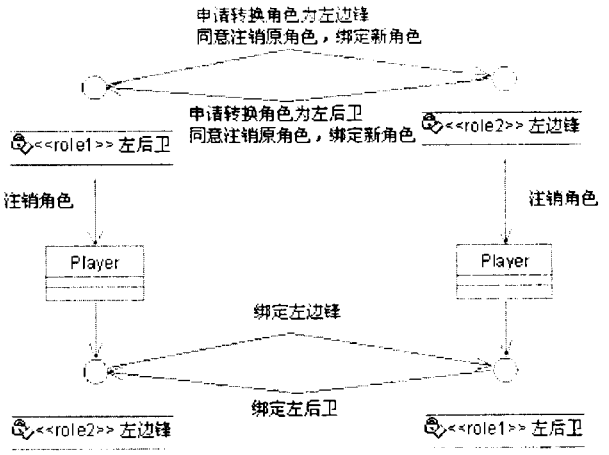

图 1 用 UML 描述角色转换

● 进攻状态可由 4 种情况进入:

1)己方队员开任意球、门球、界外球、角球等;

2)队员在后防区域内截对方的传球或抢断下对方的带球,如果在此区域内的被对方队员再截回去的可能性大,或传球的成功率低,则将球踢出界外;

3)队员在中场范围内得球;

4)在对方的后防区域断它们的失误球和回传球。一旦球进入对方控制,立即切换到防守状态。

● 防守状态可由 3 种情况进入:

1)队员在对方后场丢球,中间单元的球员和进攻单元的球员进行防守状态,防守单元的球员要进行盯人防守;

2)队员在中场丢球,前锋回防中场,辅助防守负责前去抢球;积极防守队员回防;

3)队员在我方后场丢球,辅助防守队员与积极防守队员全力拼抢。

3 运用于 Agent 结构中的角色关系

在各种足球比赛中,团队协作是必须的。对于想设计一个 RoboCup 球队来说,根本问题是设计多 Agent 系统中 Agent 之间的协作技术,并且能够进行实时的反应,表现出目标制导的理性行为。那么设计一种好的 Agent 结构就显得至关重要。

(1)基于 BDI 模型的慎思结构。

在一个动态和不确定的域中,如 RoboCup 比赛,建立一个统一的、稳定的环境模型是必要的。通过一个固定的环境模型,短期的错误信息需要被更正,不精确的信息需要重新进行估算,通过推理获得错过的信息。为了能实现最终的目标,每个 Agent 需要在其内部维护一个稳定的环境模型,含有一个稳定的信念(belief) Hans - Dieter-

维普资讯 http://www.cqvip.com

burkhardetal。1998 年有多支球队使用 BDI 模型,其中最著名的 ATHumboldtTeam,曾获得 1998 年比赛的亚军。ATHumboldtTeam 的结构特点是按照 BDI 模型的结构建立全队的规划过程。

① Belief:环境模型;

② Desires:从固定目标库中选择的目标;

③ Intention:表现为两个阶段的规划过程。

(2)反应结构。

从传感器进入的数据同时传给多个行为模块,如果某些模块的条件满足,则这些行为就不会有输出。但究竟哪一个被执行,或全部执行,需要由 Arbiter 模块进行判决。一般常用的算法有:静态优先级算法、动态优先级算法和混合算法。需要由具体应用环境而定。

(3)层结构。

层结构也可称为混合结构,一般可分为两层或三层:通讯层、控制层和决策层。Sohota[4] 发展了一个决策技术,称之为:reactivedeliberation,它在多个事先定义好的行为之间进行选择。也可以采取全局层、局部层、个体层这样的三层设计方式。其中全局层主要负责对手建模和确定阵型;局部层进行角色变换、策略选取和智能体之间的通讯;个体层负责智能体的行为方式。

由于 RoboCup 中 Agent 之间的通信是受到带宽限制的,而且协作都是实时和突然出现的,因此需要在通信和协作之间找到一个平衡点。在一个实时的多 Agent 系统(MAS)中,Agent 为了各自的目标运作着,每个 Agent 都必须不断地分析它所处的环境。对于一个没有充当任何角色的 Agent 在任意时刻,都有大量不同的外部刺激和行为需要它去决策,可供选择的情况越多,做决策就越困难,而且需要花费大量的时间和资源。无疑,这种 Agent 是相当复杂的,而且不能保证它所做的决策与其它 Agent 所做的决策没有冲突。最坏的情况就是在一个 RoboCup 球队中,所有的 Agent 都在追着球跑,这将是混乱和缺乏效率的场面。在这种情况下,一种好的解决方法就是通过 Agent 之间的通信。然而,通信是很花费时间的,而且会增加复杂度,可靠性因此降低。

如果能给 Agent 确定某种角色,那么它的行为选择的空间将会减少很多。理想的方法是对于任何一个状态,只提供给 Agent 一个或两个可能的行为选择,这样 Agent 将很容易进行决策,而且这种 Agent 是反应型(reactive agent)的。如果 Agent 充当了一定的角色,那么就能排除某些特殊的状态。例如,一个充当守门员角色的 Agent 永远也不会进入对方的禁区,这样就不需要把这个状态所对应的行为考虑在内。在 RoboCup 这种实时系统中,反映型证明是比较有效的,然而慎思型(deliberative agent)作用也同样重要,如两个中后卫在拦截对方中锋带球冲入本方禁区时,就需要通过推理计算,让最靠近对方中锋的中后卫负责拦截,而由另外一名中后卫负责保护。基于按角色分配任务,确保 Agent 在相同与不同角色之间的协调与协作,那么把反映型和慎思型相结合(即混合型),充分发挥两种结构的优点,应成为当前研究的主要方向。

在 RoboCup 中,为了在每个 Agent 之间在不进行通讯或少量通讯的情况下,得到最优动作,还可以使用基于角色的协作图算法。

基于角色的协作图是一个有向图,如图 2 所示,它的节点是一组 Agent 所扮演的角色,即 $M = \{m_1, \cdots, m_k\}$,每个节点维护若干值规则;如果角色 m_i 与 m_j 之间存在协作关系,且角色 m_i 的动作影响 m_j 的决策,则存在边 $m_i \rightarrow m_j$,与 m_i, m_j 相关的规则由 m_j 来维护。

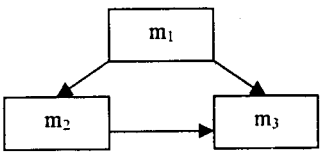

图 2　基于角色的协作图

4　球员设计实例

在 RoboCup 中,利用面向对象的方法来构造球员,这样有助于理解和今后的扩充。在上层中构造了公共角色类,而在下层中构造了特殊角色类。只有下层中的子类才拥有实例,上层中的父类都是抽象类,子类继承了父类的技术,并且加进了新的技术,拥有自身的特殊行为,如图 3 所示[5]。

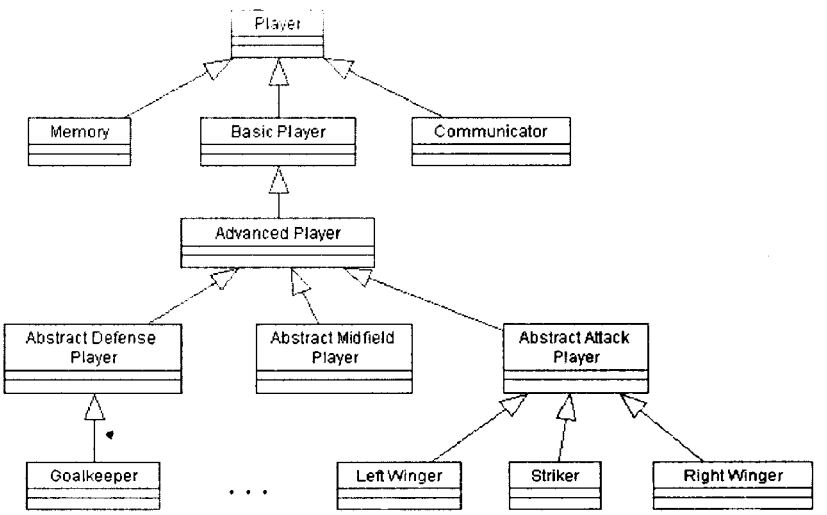

图 3　用 UML 语言描述球员设计类

1)Player。这是最上层的超类,用于处理球员 Agent 与 Soccer Server 的通信。它能发送一些基本的命令,如 turn,dash 和 kick 给 Server,并能从 Server 处接收视觉和听觉信息,储存在 Memory 类中。由 Communicator 类提供

(下转第 78 页)

维普资讯 http://www.cqvip.com

号,所有等待此信号的线程将会接收到通知并竞争互斥量。取得互斥量的线程(称活动线程)执行该次任务。任务完成后,该线程又会成为空闲状态,再次将它放到线程链表中,以备处理新的任务。

线程链表的数据结构如下所示:

```
struct free_threads {
    pthread_mutex_t mutex; /* 互斥量 */
    pthread_cond_t t_cond; /* 条件变量 */
    pthread_t *next; /* 指向下一个线程 */
    };
```

3.2 线程池大小动态调整算法思想

(1)初始化[4]:系统初始化时,创建一定数目的线程放到线程池中。当有任务到来时,从池中取出一个线程处理这一任务,处理完后还放回线程池中,线程池中的线程只是空闲线程。

(2)调整大小[5]:线程池有一个空闲线程个数的上限与下限,当池中的空闲线程个数低于下限时,需要创建一定数目的线程放到池中,当池中的线程个数高于上限时,需要销毁一定数目的线程。

(3)寻找最大值:线程池不能无限制增大,线程个数超过一定的数目,将会降低系统性能。这里用一个结构体变量存储系统吞吐量及所对应的线程的个数,在增加线程池中线程个数后,比较此时的线程个数和结构体变量中线程的个数的大小,如果比纪录的值大,则继续比较此时的吞吐量同结构体变量中存储的吞吐量的大小,若比存储的值大,则更新结构体变量,若比存储的值小很多,则固定结构体变量的值,这个值就是所要找的最大线程数。以后系统中的线程个数不能超过此值。

4 总结和展望

测试结果显示,基于线程池机制 DHCP 服务器的响应速度明显加快,效率有了显著的改进。

文中给出了基于线程池机制 DHCP 的设计思想,提高了系统的效率。下一步工作要解决的问题主要是:如何实现对客户分类,并优先处理特殊客户的服务请求。

参考文献:

[1] Droms R. Dynamic Host Configuration Protocol[S]. RFC 2131,1997.

[2] 李冠一,郑 辉,韩维桓. 基于 WinSock 实现 Internet 协议多线程连接的关键技术[J]. 计算机应用,2001,21(12):50-52.

[3] 翟征得,李大兴. 基于线程池技术 WWW 代理服务器的设计与实现[J]. 计算机应用研究 2004(5):87-88.

[4] 水 超,李 慧. 对象池模式的扩展与设计[J]. 计算机工程,2004(9):26-27.

[5] Xu Dongping. Performance Study and Dynamic Optimization Design for Thread Pool Systems[DB/OL]. http://www.scl.ameslab.gov/Publications/Brett/CCCTFinal-color.pdf,2004-12-01.

(上接第 75 页)

接口来使球员与 Soccer Server 进行通信。

2)Basic Player。Basic Player 类继承了 Player 类,拥有基本的行为代码。该类实现的技术包括:观察物体、跑向球的位置、预测球的位置、控球和截球等。

3)Advanced Player。该类在基本动作代码的基础上,包括了一些特殊技术、动作的代码。

4)Abstract Defense Player, Abstract Midfield Player, Abstract Attack Player。这些类分别是后卫、中场、前锋队员所对应的类的父类,要实现具体角色对应的类只需要扩充相应的代码即可。

5)Goalkeeper, Left Winger, Striker, Right Winger。这些类分别是守门员、左边锋、中场,右边锋队员所对应的类。这是具体的角色,是具体队员的实现。

5 结束语

从目前的研究成果来看,多 Agent 系统的构造方法,主要借用了两方面的研究成果:一个是面向对象的方法;另一个是知识工程的方法。其中,面向对象的方法得到了更广泛的重视,这是因为:一方面,面向对象技术发展已比较成熟,有很多经过实践检验的方法和工具;另一方面,对象和 Agent 是两种比较一致的观察客观世界的工具,因此从对象过渡到 Agent 是直观而且自然的。

文中从面向角色的多 Agent 结构设计的视点出发,介绍了角色的概念、角色之间的转换和合作关系,并且揭示了角色与 Agent 的结构设计相结合的优势所在,最后用一种球员类的设计方法作为理论在实践中的运用。基于角色的流程设计,从理论上来讲还没有趋于完善,在实际工作中还有不少未知的方法,对于从事智能方面研究的软件工作者,尤其对从事多 Agent 研究的行业人士来说,还有巨大的发展空间。

参考文献:

[1] 程显毅. Agent 计算[M]. 哈尔滨:黑龙江科学技术出版社,2003.

[2] 赵卫东,黄丽华. 面向角色的多 agent 工作流模型研究[J]. 管理科学学报,2004,7(2):55-60.

[3] 马 军,闫 琪,毛新军,等. 基于角色的多 Agent 系统软件设计方法[J]. 计算机工程与应用,2004,40(6):118-120.

[4] Reis L P, Lau N, Oliveira E C. Situation Based Strategic Positioning for Coordinating Team of Homogeneous Agents[EB/OL]. Http://www.ieeta.pt/robocup/archive.htm.,2002-05.

[5] 潘凌寒,楚 威,程显毅. 基于角色的 RoboCup 足球策略[J]. 计算机工程与应用,2004,40(26):66-67.