

基于Java规则引擎的足球机器人系统决策研究

陈建伟,唐平

(广东工业大学 计算机学院,广东 广州 510090)

摘要:Java规则引擎是一门新兴的技术,本文探讨了利用Java规则引擎进行足球机器人系统策略部分的开发与维护,在此基础上,通过JNI(Java Native Interface)技术实现Java规则引擎与用VC开发的足球机器人系统无缝连接的软件接口。

关键词:规则引擎;瓜哇本地接口;机器人足球;策略;接口

中图分类号:TP242.6

文献标识码:A

文章编号:1007-7162(2003)04-0058-05

在足球机器人系统中,规则库与源代码分离是很有必要的,否则,整个系统会显得很杂乱,要读懂整个系统需要花较长的时间,修改和维护不方便,也不方便后继开发人员继续开发和研究.由于足球机器人系统本身是一个实时的系统,如果过多的规则和链接不合理,会使系统显得庞大,推理时间较多,机器人的反应不够灵敏,系统的效率不高.因此,本文提出了将决策系统作为一个子系统从源代码分离出来,用专门的开发工具——Java规则引擎技术[ILOG JRules,中鹰科技(广州)有限公司]对其进行开发与管理,以提高整个系统的效率.

1 Mirosot 仿真系统的简介及其传统的人工智能实现方法分析

Mirosot(Micro - Robot World Cup Soccer Tournament)仿真系统由比赛服务器和客户端程序组成(洪炳熔,薄喜柱.Mirosot 策略仿真系统.哈尔滨工业大学智能机器人研究室,1999.),如图1所示.

参赛双方通过各自的客户端程序和服务器相连,就可以参加比赛.在比赛过程当中,参赛双方的客户端程序根据从服务器得到的当前双方机器人的

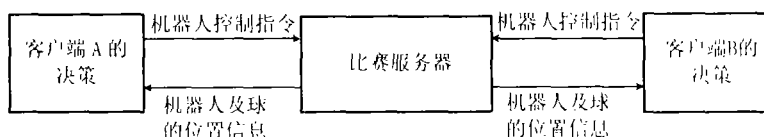


图1 Mirosot 仿真系统结构示意图

位置和球的位置信息,按事先制定好的策略运算得出控制指令,然后将指令信息返回服务器以控制机器人的所有动作.

Mirosot 比赛的服务器端是由大会发布的公版模拟器.而客户端则是已经构建好框架的VC代码,各个参赛队根据自己的策略,在客户端编写代码.而在传统的编程方式中存在以下主要不足:

(1)控制策略以源代码的方式存在,造成规则无法灵活改变,规则的任何修改都需要程序员

收稿日期:2002-12-31

作者简介:陈建伟(1968-),男,研究生,主要研究方向为人工智能与软计算.

的参与,而策略一旦需要快速更新时,相对来说,就难以实现了.例如,如果需要根据前一天其他机器人足球队的比赛情况,进行规则修改,或加入新的策略,往往不那么容易做到.

(2)缺乏一个有效的规则语言造成无法完全表达实际情况.策略的表达不规范,没有形成统一的风格.这就给以后策略的增加和修改造成了极大的麻烦.

(3)没有好的规则编辑器和规则调试器导致规则无法有效地跟踪和调试.如果需要对部分的机器人策略进行调试和评估是难以做到的.例如,要方便地屏蔽一部分的策略,在源代码嵌套复杂的情况下,难以做到.

(4)规则难以读懂,无法实现对规则的汉化问题.如果由别的开发人员接手过来继续开发,需要花很长时间才可以读懂已经写好的策略.

由于以上的不足,本文通过引入一种新技术——Java 规则引擎,将用 VC 编写的系统和规则分开来,以解决上述所存在的问题.

2 ILOG 规则引擎技术

ILOG 规则引擎技术简介(ILOG JRules,中鹰科技(广州)有限公司)

ILOG 规则引擎技术是一种快捷、方便、弹性的软件组件.它主要由三部分组成(1)规则引擎的核心部分.(2)规范化的规则语言.(3)规则语言开发工具.它可以在一秒钟内处理数千条规则,同时可以容纳数万条规则.规则引擎使得开发人员可以把规则从源代码中分离出来,这样,开发人员可以十分容易地开发规则,并使得后期的维护变得十分方便,规则作为源代码的外部扩展而存在,可以随时修改而不必重新编译.

我们用 ILOG 公司提供的专门开发工具 Rule Builder 来开发机器人足球策略,Rule Builder 包括 RuleEditor、Debugger、Profiler.机器人足球系统策略的开发人员可在 RuleEditor 环境下开发自己的策略系统.用 Debugger 来测试开发出来的规则.而在 Profiler 环境下,可以观察到哪些规则触发的次数再多,而哪些不常用,从而优化那些触发次数多的规则.

ILOG 规则引擎工作原理

规则引擎是基于规则建立的,规则分别由 IF 和 THEN 引导的条件和动作组成,规则语言可以表达为如下条件语句:

● IF……THEN……

规则引擎的运行包括储存在 Agenda 里面的规则和储存在 WorkingMemory 里面的对象. Working Memory 是一个控制对象的容器,Agenda 则是规则实例,一旦条件符合,规则就会被触发.

3 利用规则引擎实现足球机器人行为与策略的推理

足球机器人的比赛一般是由 5 个基本动作和 3 个基本行为所组成的,基本行为包括:速控,移动和旋转;基本动作则是:踢球,射门,拦截,守门,避障.而这些动作和行为是由策略库所控制的,它们的关系如图 2 所示.

3.1 基于产生式系统的策略

产生式系统由 3 部分组成:一个总数据库,一个对数据库进行操作运算的规则,一个控制策略.下面是基于产生式系统的足球机器人策略的划分.

(1)比赛场地的划分(洪炳熔,薄喜柱. Mirobot 策略仿真系统.哈尔滨工业大学智能机器人研

究室,1999.):考虑到信息量的恰当性,在总体上可以把场地划为四块,分别是我方左半场 FL,我方右半场 FR,对方左半场 AL,对方右半场 AR.然后在每小块中再分四块(如图3所示)这种划分根据球的位置来确定整个球队应采取的策略,根据机器人的位置来确定机器人的角色分配(进攻,协攻,防守,协守,守门).

(2)策略库标识约定:

- IN(0,A):如果 0 在 A 区内,返回 TRUE.
- READYTOKICK(T):如果 T 队某队员处在射门位置,返回 TRUE.
- LENGTH(01,02):返回 01 与 02 的距离.
- NEAREST(0,T):返回 T 队距 0 最近的队员编号.
- DEFENCE, ATTACK, DANGEROUS:状态变量.

(3)防守策略规则:当球在我方半场时可以初步判断进入防守状态.此时的对策为:我方离球近的队员先拦截球,后与对方顶牛;若距离近则直接顶牛以破坏对方进攻路线.若对方队员仍在对方半场院,且离球较远时,则判断进入进攻状态,我方队员可采取避障射门等进攻策略,具体规则定义如下:

● IF((IN(B,FL1)OR IN(B,FL3)OR IN(B,FR)OR IN(B,FR3))THEN DEFENCE

● IF((DEFENCE)(LENGTH(B,NEAREST(B,OPP))>>LENGTH(B,NEAREST(B,MATE))))THEN ATTACK

● IF((DEFENCE)(LENGTH(B,NEAREST(B,OPP))<<LENGTH(B,NEAREST(B,MATE))))THEN ATTACK

●

(4)进攻策略规则:当球在对方半场时可初步判断进入进攻状态,当我方队员距球很近时,可考虑对方队员的位置而确定进入进攻状态,与球有一定距离时则根据双方与球的距离来判断,与球距离小于或等于对方与球的距离时则坚持进攻,但是当我方离球很远,且对方离球很近时,应转入防守或争球状态,具体规则定义如下:

● IF((IN(B,AR)OR IN(B,FR1)OR IN(B,M1)OR IN(B,AR3))THEN ATTACK

● IF((ATTACK)(LENGTH(B,NEAREST(B,MATE))>L1)(LENGTH(B,NEAREST(B,MATE))>>LENGTH(B,NEAREST(B,OPP))) THEN ROBBALL

● IF((ROBBALL)(ONT IN(B,A))THEN DEFENCE

●

(5)争球策略规则:当场上双方都不占优势时应进入争球状态,争取把形势打乱,再寻找机会,当球进入对方半场且双方距球大致相等时,可采取争球策略.具体策略定义如下:

● IF((LENGTH(B,NEAREST(B,MATE)) $\cong$ LENGTH(B,OPP)))THEN ROBBALL

● IF((ROBBALL)(READYTOKICK(MATE))THEN ATTACK

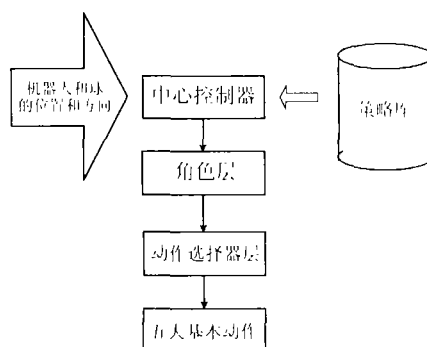


图2 策略软件的层次结构

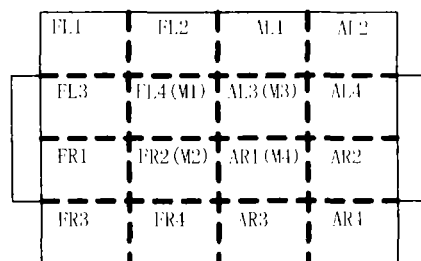


图3 足球场地的划分

- IF((ROBBALL)(READYTOKICK(OPP)))THEN DEFENCE
-

3.2 在Java方法中调用规则引擎

通过用 Rule Builder 提供给我们的开发工具建立好规则库之后,要使规则起作用,我们必须在Java应用程序中调用规则引擎,下面是调用的代码:

```
{.....
IlrRulest ruleset = new IlrRuleset(); //创建一个 ruleset 对象,用来接收规则文件
ruleset.parseFileName("../strategy.ilr"); //strategy 对应的策略文件的名称
IlrContext context = new IlrContext(ruleset); //创建一个规则引擎对象
RobotBall = new RobtBall(); //创建足球对象
RobotMate player0 = new RobotMate(); //创建守门员
RobotMate player1 = new RobotMate(); //其他队员 1—2 号
RobotMate player2 = new RobotMate();

RobotEnemy enemy0 = new RobotEnemy(); //创建对方守门员
RobotEnemy enemy1 = new RobotEnemy(); //对方队员 1—2 号
RobotEnemy enemy2 = new RobotEnemy();

context.assert(ball); //把足球对象插入规则//引擎
context.assert(player0); //把我方队员对象插入规则引擎
context.assert(player1);
context.assert(player2);
context.assert(enemy0); //把对方队员对象插入规则引擎
context.assert(enemy1);
context.assert(enemy2);

context.fireAllRules0; //启动规则引擎
.....
}
```

上面代码通过创建一个规则对象来接收规则库里的规则文件;然后把足球对象,我方队员的对象及对方队员的对象插入规则引擎;最后,触发规则引擎,使之生效。

3.3 在C++系统中调用Java方法

我们在编写好包含规则库的Java应用程序之后,必须要把它嵌入到我们的机器人足球系统中才能达到目的.在此我们利用JNI技术实现本地程序对Java方法的调用。

JNI是Java方法与本地程序的接口^[1],JNI技术使本地程序可以调用Java对象,就如同Java代码使用这些对象一样.图4显示了C++程序对Java方法的调用的接口结构。

在C++编写的足球机器人系统中,利用JNI,就可以把Java规则引擎应用到系统中来.在原有的系统中加入以下代码调用:

```
JNIEXPORT void JNICALL
```

```
RobotBall __nativeMethod(JNIEnv * env,
jobject obj, jint depth)
```

```
{
    jclass cls = env -> GetObjectClass( env,
obj);
```

```
jmethodID mid = env -> GetMethodID
(env, cls, "context", "(I)V"); //context 是一个规则引擎对象
```

```
if( mid == 0)
```

```
{
```

```
    return;
```

```
}
```

```
env -> CallVoidMethod( env, obj, mid, depth);
```

```
.....
```

```
}
```

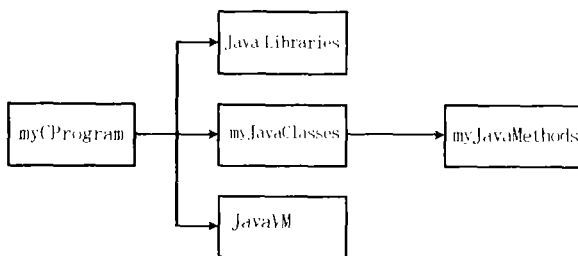


图 4 C++ 程序调用 Java 方法的接口图

上面代码是通过 Java 提供给本地方法(VC)的接口来实现对 Java 方法进行调用。

至此,我们已通过 Java 方法把规则引擎嵌入到用 C++ 开发的足球机器人系统中来,当系统运行时,会自动启动规则引擎,然后规则引擎会自动根据策略,运算出相应的 RobotCommand 来控制足球机器人的动作。

4 结束语

本文提供了一种全新的方法开发足球机器人策略系统,与原有的开发方法相比,利用 ILOG 规则引擎技术开发的系统有以下优点:

- (1)把规则从程序代码中分离出来,易于查找,编辑和修改;
- (2)提供高级自然语言编写,把规则用最自然的方法表示出来;
- (3)提供可视化编辑环境给编程人员,同时提供规则调试和查看规则的运行机制;
- (4)规则可以汉化,给后继开发人员提供方便。

参考文献:

- [1][美]Bratt Spell. Java 专业编程指南[M]. 北京:电子工业出版社,2001.903-958.

The Strategy Research of Robot Soccer System on the Basis of Java Rule Engine

CHEN Jiang-Wei, TANG Ping

(Faculty of Computer, Guangdong University of Technology, Guangzhou 510090, China)

Abstract: Java Rule Engine is a new technology. This paper discusses how to develop and maintain the strategy of soccer game by using Java Rule Engine. A software interface is designed between the strategy of soccer game developed by Java Rule Engine and the system of soccer game developed by VC through JNI methodology.

Key words: Rule Engine; Java Native Interface; robot soccer system; strategy; interface