面向安全需求的测试用例自动生成技术研究

周绍君[1],徐中伟[1],喻 钢[1,2],李弋强[1],吴 剑[1]
ZHOU Shao-jun[1],XU Zhong-wei[1],YU Gang[1,2],LI Yi-qiang[1],WU Jian[1]

1.同济大学 电子与信息工程学院,上海 201804
2.上海大学 悉尼工商学院,上海 201800
1.School of Electronics and Information,Tongji University,Shanghai 201804,China
2.School of Sydney Business,Shanghai University,Shanghai 201800,China
E-mail:zsjaidyj@126.com

ZHOU Shao-jun,XU Zhong-wei,YU Gang,et al.Research of auto-generation techniques for safety requirement oriented test case.Computer Engineering and Applications,2009,45(28):75-78.

Abstract: The traditional technique for test case generation is usually function requirement oriented,however,it isn't applicable for safety testing.A new algorithm for generating test case automatically is proposed.It is safety requirement oriented.By means of defining safety factor and creating the sequence diagram of UML with safety factor,test case is generated based on the least safety factor path coverage criteria.The proposed means has been adopted in simulation testing of train control system.

Key words: sequence diagram of UML;safety factor;least safety factor path;automatic test case generation;safety critical system

摘 要:传统的测试用例生成技术通常都是面向系统功能性需求,并不适用于安全苛求软件系统的安全性测试。面向安全需求,通过定义安全因子,建立带有安全因子的UML顺序图,采用最小安全因子路径,提出了基于最小安全因子路径完全覆盖准则的测试用例自动生成算法,并将其成功应用到高速铁路列车运行控制系统的仿真测试中。

关键词:UML顺序图;安全因子;最小安全因子路径;测试用例自动生成;安全苛求系统
DOI:10.3778/j.issn.1002-8331.2009.28.022 **文章编号**:1002-8331(2009)28-0075-04 **文献标识码**:A **中图分类号**:TP311

安全苛求软件系统(Safety Critical Software System,SCRS)在满足业务功能性需求外,还必须满足特定的安全性需求,因为它一旦失效即可能导致不可预期的灾难性后果,所以确保它的安全性至关重要[1]。软件安全性测试是验证此类系统功能和确认系统具有足够安全性防护能力的重要手段之一,在安全苛求软件测试中测试用例的自动生成是一个重要环节,但是传统的测试用例生成技术通常都是面向系统的功能性需求,并不完全适用于安全性测试。

在传统的基于UML顺序图的测试用例生成算法[2]的基础上,面向安全性需求,提出安全因子的概念,通过建立带有安全因子的UML顺序图,采用最小安全因子路径完全覆盖准则,提出一种新的安全苛求软件测试用例的自动生成算法,并将其成功应用到高速铁路客运专线CTCS-2列控中心的仿真测试系统中。

1 带安全因子的顺序图

1.1 安全因子的定义

安全因子是在安全苛求软件系统的系统风险的基础上提出的,风险是指危害事故发生的可能性和严重性的综合表述。参照国外相关安全性技术标准,可以得到安全苛求软件系统风险矩阵,如表1:

表1 安全苛求系统风险矩阵

可能性等级	严重等级			
	Ⅰ(灾难的)	Ⅱ(严重的)	Ⅲ(轻度的)	Ⅳ(轻微的)
A(频繁)	很高	很高	高	低
B(很可能)	很高	很高	高	低
C(有时)	很高	高	低	很低
D(极少)	高	低	低	很低
E(不可能)	低	低	低	很低

根据安全苛求系统风险矩阵,给出安全因子的定义:

定义1 安全因子(Safety factor),是交互故障或场景失效导致系统风险的定量描述,定义域为{0,1,2,3},其中0表示风险等级为很高,1表示风险等级为高,2表示风险等级为低,3表示风险等级为很低。安全因子越小,系统的风险等级越高。

1.2 带安全因子的UML顺序图定义

定义2 带安全因子β的UML顺序图SDWSF(Sequence

基金项目:国家自然科学基金(the National Natural Science Foundation of China under Grant No. 60674004)。
作者简介:周绍君(1985-),女,硕士研究生,研究方向为系统建模与仿真,安全测试用例自动生成;徐中伟(1964-),男,教授,博士生导师,研究方向为安全软件测试评估;喻钢(1977-),男,讲师,博士研究生,研究方向安全软件形式化;李弋强(1985-),男,硕士研究生,研究方向为安全软件形式化、建模、测试和仿真;吴剑(1983-),男,硕士研究生,研究方向为安全软件评估。
收稿日期:2009-03-31 **修回日期**:2009-05-25

Diagram with Safety Factor)可以表示为一个六元组:$SDWSF=<O,M,E,\rightarrow,msg,obj>$,其中:

$O=\{O_1,O_2,\cdots,O_m\}$是对象的集合,$O_1,O_2,\cdots,O_m$都是顺序图中的对象。

$M\subseteq guard\times message\times name\times parameter\_list$,是消息集合。顺序图中的每一个消息都可表示为:[监护条件]消息名(参数1,参数2,…,参数n)。

$E=M\times\beta\times Os\times Or$,是事件集合。事件是指消息的发送和接收,$Os$表示事件中消息的发送者,$Or$表示事件中消息的接收者。安全因子$\beta$取值范围为{0,1,2,3}。

$\rightarrow$是消息集合M上的一个全序关系,表示顺序图中的消息在纵向时间轴上的先后关系。

msg是从E到M的一个函数关系,$msg(e)\in M$表示事件e所对应的消息。

obj是从E到O的一个映射关系,$obj=\{objs,objr\}$。$objs(e)\in O$表示事件e所对应的消息发送对象,$objr(e)\in O$表示事件e所对应的消息接收对象。对象O_i上所有事件的集合记为E_i,$E_i=\{e|e\in E\wedge objs(e)\wedge objr(e)=O_i\}$。

1.3 带安全因子的UML顺序图创建

根据带安全因子的UML顺序图定义,在传统UML顺序图基础上,对图中每个事件增加安全因子β,用以度量该事件可能导致的系统风险的等级,由此可创建如图1所示的带安全因子的顺序图:

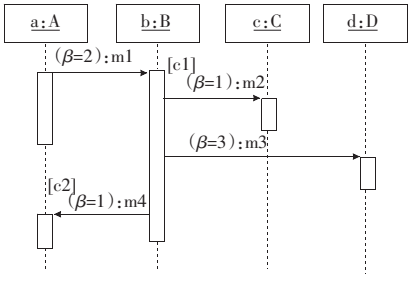

图1 带安全因子的UML顺序图

2 带安全因子的控制流图

为自动生成测试用例,需要将带安全因子的UML顺序图转化为带安全因子的控制流图(Control Flow Graph with Safety Factor,CFGWSF)。

2.1 安全运营场景定义

为了创建CFGWSF,提出安全运营场景(Operation Scenario)的定义,每一个安全运营场景都包含唯一的标识、起始状态、事件的集合和终止状态:$aOpnScn=<ScnID,StartState,EventSet,EndState>$[3]。

图1所示的顺序图包含4个安全运营场景,如表2所示。

2.2 带安全因子的控制流图定义

定义3 CFGWSF可以表示为一个四元组:$CFGWSF=<N_{CFGWSF},\sum_{CFGWSF},S_{CFGWSF},F_{CFGWSF}>$,其中:

N_{CFGWSF}表示所有事件节点的集合;

$\sum_{CFGWSF}$表示从一个事件转移到另一个事件的边的集合;

S_{CFGWSF}表示起始状态;

F_{CFGWSF}表示终止状态集合。

2.3 带安全因子的控制流图创建

所有的安全运营场景集合构成CFGWSF。如图1所示的带有安全因子的顺序图转化成的CFGWSF如图2所示:

表2 图1对应的4个安全运营场景

<Scn1	<Scn2
StatusS	StatusS
e1:(m1,2,a,b)	e1:(m1,2,a,b)
e2:(m2,1,b,c)|c1	e3:(m3,3,b,d)
e3:(m3,3,b,d)	StatusX
StatusX	>
>	
<Scn3	<Scn4
StatusS	StatusS
e1:(m1,2,a,b)	e1:(m1,2,a,b)
e2:(m2,1,b,c)|c1	e3:(m3,3,b,d)
e3:(m3,3,b,d)	e4:(m4,1,b,a)|c2
e4:(m4,1,b,a)|c2	StatusY
StatusY	>
>	

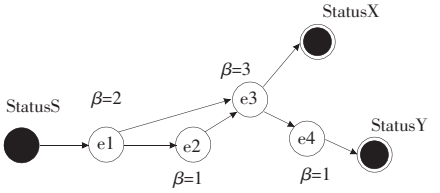

图2 与图1赌赢的带安全因子的CFG

CFGWSF中的每一个节点都代表一个事件(Event),包含了从XMI文档中抽取的生成测试用例的必要信息。

$aEvent=<messageName$(消息名),β(安全因子),$fromObject$(消息的发送者),$toObject$(消息的接收者)[/$guard$(监护条件)]>

3 安全性测试用例自动生成

对安全苛求软件系统进行安全行测试,关键是要验证系统异常情况的响应,检验其实际运行结果与软件规格说明书的要求是否一致。很明显,在一个CFGWSF中覆盖了包括消息序列在内的从起始状态到终止状态的所有路径,包括异常路径,可以对系统的功能性及安全性进行测试,而安全因子用以衡量某事件可能引起的系统风险等级。而测试覆盖准则是判断测试用例集是否充分、是否可以停止测试的重要标准,它在整个测试过程中起指导作用。

3.1 最小安全因子路径完全覆盖准则

传统的完全CFG路径覆盖准则可以保证测试的充分性,然而,对于相对复杂的系统,CFG路径数目往往非常巨大,而由于循环和条件事件的存在,同一条CFG路径中可能存在很多部分相同的路径,为了提高测试效率,节省资源,人们又提出了改进的完全CFG路径覆盖准则,对这部分重复路径进行化简即当再次遇到时随机选择一条路径[4]。然而,这两种覆盖准则都是面向系统功能性需求,并不适用安全苛求软件系统的安全性测试。

因此,在改进的完全CFG路径覆盖准则的基础上,提出了最小安全因子路径完全覆盖准则。

定义4 路径的安全因子,是指该路径上所有事件的安全因子的最小值。

定义5 最小安全因子路径,是指该路径的路径的安全因子最小,如果路径的安全因子相同,则取事件较多的路径为最小安全因子路径。

最小安全因子路径完全覆盖准则:对CFGWSF中每一个节点只扩展一次。即,当节点A与节点B间(事件A发生在B

之前)有条件分支时,第一次遇到节点 A 时,对其所有后继节点进行扩展,当再次遇到覆盖 A—B 路径的路径时,选择 A 到 B 的最小安全因子路径进行覆盖;当有循环分支存在时,作出绕过循环,循环一次,循环多次的类条件分支策略,选择可能导致的系统风险等级高的循环次数。

3.2　安全性测试用例自动生成算法

输入:带有安全因子的控制流图(CFGWSF)。

输出:系统测试用例集 TestSet(T)。

1.基于最小安全因子路径完全覆盖准则,遍历 CFGWSF 中从起点状态到终点状态的全部路径 $P=\{P_1,P_2,\cdots,P_n\}$,每条路径代表一个场景 Scn。

2.For each path $P_i\in P$ do

3. $t_i\leftarrow\emptyset$　//t_i 是场景 scn_i 的测试用例,初始化为空

4. $n_c=n_s$　//n_c 是当前节点,n_s 是当前路径的起始状态

5. $preC_i$ 是场景 scn_i 的前置条件,该信息储存在 n_s 中

6. $n_c=n_b$ //将当前节点移动到场景 scn_i 的第一个事件节点

7. while($n_c\neq n_f$) do //从 scn_i 的第一个事件节点开始,依次对路径中所有节点进行如下操作,直到该路径终止状态 n_f

8. e_c=<m,β,$fromObj$,$toObj$,c> //e_c 是当前节点 n_c 的事件,消息 m 由消息发送者的一系列参数 $a_1,a_2,\cdots,a_p$ 触发,β 是事件 e_c 的安全因子,c 是事件 e_c 的监护条件

9. If ($c=\wedge$) //如果事件 e_c 没有监护条件

10. $t=\{preC,I(a_1,a_2,\cdots,a_p),O(d_1,d_2,\cdots,d_q),postC\}$ //t 是当前节点的测试用例,$preC$ 是消息 m 的前置条件,$I(a_1,a_2,\cdots,a_p)$是消息发送者为触发消息 m 所输入的输入值集合,$O(d_1,d_2,\cdots,d_q)$是消息 m 执行后,消息接收者的期望执行结果值,$postC$ 是消息 m 的后置条件

11. $t_i=t_i\cup t$ //将当前节点测试用例并入场景 scn_i 测试用例中

12. EndIf

13. If($c\neq\wedge$) //如果事件 e_c 具有监护条件

14. $c(\lambda)=(c_1,c_2,\cdots,c_p)$ //$c(\lambda)$为事件 e_c 的监护条件集合

15 $t=\{preC,I(a_1,a_2,\cdots,a_p),O(d_1,d_2,\cdots,d_q),c(\lambda),postC\}$ //含义同上

16. $t_i=t_i\cup t$ //将当前节点测试用例并入场景 scn_i 测试用例中

17. EndIf

18. $n_c=n_k$　//将当前节点移动到路径 P_i 的下一个节点 n_k

19. $T\cup t_i=T$ //将场景 scn_i 的测试用例并入系统测试用例集 T 中

20. EndWhile

21. 确定场景 scn_i 的最终期望输出值 O_i 和后置条件 $postC_i$,此两项信息储存在终止状态 n_f 中,当所有事件节点遍历完毕,$n_c=n_f$

22. $t=\{preC_i,I_i,O_i,postC_i\}$

23. $T\cup t=T$

24. EndFor

25. Return(T)

26.End

4　高速铁路 CTCS–2 级列车运行控制系统仿真测试

随着我国步入高速铁路的发展阶段,作为列车运行控制系统的核心,车站列控中心系统是一种典型的安全苛求系统,它是基于轨道电路和点式应答器传输列车运行许可信息并采用目标–距离制动模式监控列车安全运行的列车运行控制系统。

在高速铁路 CTCS–2 级列控系统中,调度集中需要将各种临时调度命令下发到车站列控中心,而这些信息的正确性会直接影响列车的运行安全,所以它是安全相关信息,必须正确及时发送到车站列控中心。针对此安全需求可构建如图 3 所示带有安全因子的 UML 顺序图,并创建带有安全因子的控制流图,如图 4 所示:

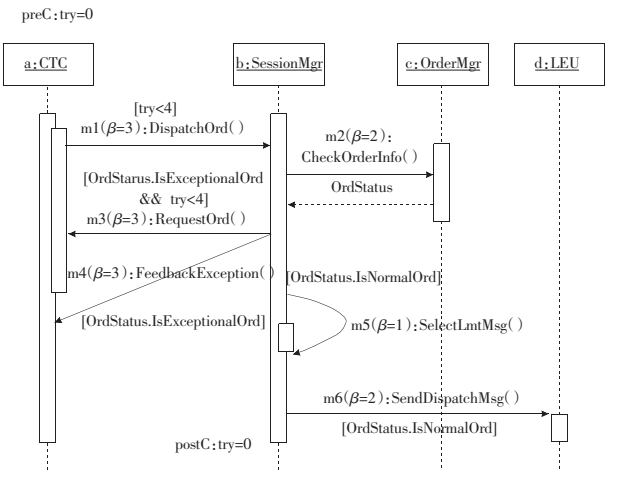

图 3　CTC 向 TCC 发送调度命令用例的 UML 顺序图

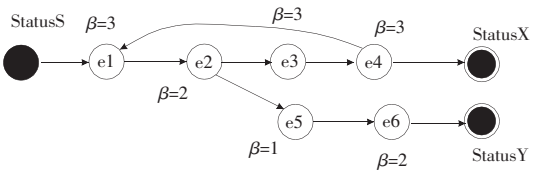

图 4　对应的 CFGWSF

如图 4 所示,该用例有一个惟一的起始状态 StatusS,其终止状态有两个,一是系统按照软件规格说明书的要求正确实现其功能,即 TCC 收到来自 CTC 的正常调度命令,取得预存报文并将其发送给 LEU 的终止状态 StatusY,二是收到异常的调度命令,如限速命令号重复、限速区超长等,TCC 给 CTC 发送异常帧回执并要求 CTC 重发调度命令的终止状态 StatusX。

基于完全路径覆盖准则,可遍历得到如下 7 条路径:

StatusS→e1→e2→e3→e4→StatusX,e1→e4 路径执行 1 次

StatusS→e1→e2→e3→e4→e1→e2→e3→e4→StatusX,e1→e4 路径执行 2 次

StatusS→e1→e2→e3→e4→e1→e2→e3→e4→e1→e2→e3→e4→StatusX,e1→e4 路径执行 3 次

StatusS→e1→e2→e5→e6→StatusY,e1→e4 路径执行 0 次

StatusS→e1→e2→e3→e4→e1→e2→e5→e6→StatusY,e1→e4 路径执行 1 次

StatusS→e1→e2→e3→e4→e1→e2→e3→e4→e1→e2→e5→e6→StatusY,e1→e4 路径执行 2 次

StatusS→e1→e2→e3→e4→e1→e2→e3→e4→e1→e2→e3→e4→e1→e2→e5→e6→StatusY,e1→e4 路径执行 3 次

根据最小安全因子路径完全覆盖准则,对于存在循环分支的情况,第一次经过该循环路径时,对所有循环情况进行遍历,当再次经过该循环路径时,为了使路径经历更多导致系统风险事件,并对完全路径覆盖准则中的部分重复路径进行简化,得到与图 4 对应的如下 5 条完全最小安全因子路径:

(1)StatusS→e1→e2→e3→e4→StatusX,e1→e4 路径执行 1 次

(2)StatusS→e1→e2→e3→e4→e1→e2→e3→e4→StatusX,e1→e4 路径执行 2 次

(3)StatusS→e1→e2→e3→e4→e1→e2→e3→e4→e1→e2→e3→e4→StatusX,e1→e4 路径执行 3 次

(4)StatusS→e1→e2→e5→e6→StatusY,e1→e4 路径执行 0 次

(5)StatusS→e1→e2→e3→e4→e1→e2→e3→e4→e1→e2→

e3→e4→e1→e2→e5→e6→StatusY,e1→e4 路径执行 3 次

测试用例由五部分构成:前置条件 preC,输入序列 I,期望输出序列 O,监护条件 C,后置条件 postC。由安全行测试用例生成算法,可得到每条路径的测试用例如下:

路径 1 preC1,i1,o1,postC1,preC2,i2,o2,postC2,preC3,i3,o3,c3,postC3,preC4,i4,o4,c4,postC4,preCa,ia,oa,postCa

路径 2 preC1,i1,o1,postC1,preC2,i2,o2,postC2,preC3,i3,o3,c3,postC3,preC4,i4,o4,c4,postC4,preC1,i1,o1,postC1,preC2,i2,o2,postC2,preC3,i3,o3,c3,postC3,preC4,i4,o4,c4,postC4,preCb,ib,ob,postCb

路径 3 preC1,i1,o1,postC1,preC2,i2,o2,postC2,preC3,i3,o3,c3,postC3,preC4,i4,o4,c4,postC4,preC1,i1,o1,postC1,preC2,i2,o2,postC2,preC3,i3,o3,c3,postC3,preC4,i4,o4,c4,postC4,preC1,i1,o1,postC1,preC2,i2,o2,postC2,preC3,i3,o3,c3,postC3,preC4,i4,o4,c4,postC4,preCc,ic,oc,postCc

路径 4 preC1,i1,o1,postC1,preC2,i2,o2,postC2,preC5,i5,o5,c5,postC5,preC6,i6,o6,c6,postC6,preCd,id,od,postCd

路径 5 preC1,i1,o1,postC1,preC2,i2,o2,postC2,preC3,i3,o3,c3,postC3,preC4,i4,o4,c4,postC4,preC1,i1,o1,postC1,preC2,i2,o2,postC2,preC3,i3,o3,c3,postC3,preC4,i4,o4,c4,postC4,preC1,i1,o1,postC1,preC2,i2,o2,postC2,preC3,i3,o3,c3,postC3,preC4,i4,o4,c4,postC4,preC1,i1,o1,postC1,preC2,i2,o2,postC2,preC5,i5,o5,c5,postC5,preC6,i6,o6,c6,postC6,preCe,ie,oe,postCe

所有路径的测试用例组合即得到 CTC 向 TCC 发送调度命令的安全性测试用例集:

T =(preC1,i1,o1,postC1,preC2,i2,o2,postC2,preC3,i3,o3,c3,postC3,preC4,i4,o4,c4,postC4,preCa,ia,oa,postCa,preC1,i1,o1,postC1,preC2,i2,o2,postC2,preC3,i3,o3,c3,postC3,preC4,i4,o4,c4,postC4,preC1,i1,o1,postC1,preC2,i2,o2,postC2,preC3,i3,o3,c3,postC3,preC4,i4,o4,c4,postC4,preCb,ib,ob,postCb,preC1,i1,o1,postC1,preC2,i2,o2,postC2,preC3,i3,o3,c3,postC3,preC4,i4,o4,c4,postC4,preC1,i1,o1,postC1,preC2,i2,o2,postC2,preC3,i3,o3,c3,postC3,preC4,i4,o4,c4,postC4,preCc,ic,oc,postCc,preC1,i1,o1,postC1,preC2,i2,o2,postC2,preC5,i5,o5,c5,postC5,preC6,i6,o6,c6,postC6,preCd,id,od,postCd,preC1,i1,o1,postC1,preC2,i2,o2,postC2,preC3,i3,o3,c3,postC3,preC4,i4,o4,c4,postC4,preC1,i1,o1,postC1,preC2,i2,o2,postC2,preC3,i3,o3,c3,postC3,preC4,i4,o4,c4,postC4,preC1,i1,o1,postC1,preC2,i2,o2,postC2,preC3,i3,o3,c3,postC3,preC4,i4,o4,c4,postC4,preC1,i1,o1,postC1,preC2,i2,o2,postC2,preC5,i5,o5,c5,postC5,preC6,i6,o6,c6,postC6,preCe,ie,oe,postCe)

表 3 覆盖准则性能比较

	完全路径覆盖准则	改进的完全路径覆盖准则	最小安全因子路径完全覆盖准则
路径数	7	5	5
时间复杂度 O	$O(n^3)$	$O(n^3)$	$O(n^3)$
最小安全因子为 0 路径数	0	0	0
最小安全因子为 1 路径数	4	2	2
最小安全因子为 2 路径数	3	3	3
最小安全因子为 3 路径数	0	0	0
经历不安全事件 $e2$ 次数	16	8	11

由表 3 可知,三种覆盖准则具有相同数量级的时间复杂度,而对于相对复杂的系统,在有循环路径或条件分支路径的情况下,最小安全因子路径覆盖准则遍历所得路径数的确低于完全路径覆盖准则,不仅可以提高测试效率,也可以节省大量资源;而与改进的完全路径覆盖准则相比,基于最小安全因子路径覆盖准则生成的测试用例序列经历不安全事件 $e2$ 的次数更多,对系统的安全性测试更加完备。

5 结论

传统的测试用例自动生成技术主要面向功能性测试,并不适用于安全苛求软件系统的安全性测试。通过建立带有安全因子的 UML 顺序图,提出了最小安全因子路径完全覆盖准则,构建了安全性测试用例生成算法,并将其应用到高速铁路列控中心仿真测试中。新算法比传统算法得到的安全性测试用例更完备。

参考文献:

[1] 施寅生,邓世伟,谷天阳.软件安全性测试方法与工具[J].计算机工程与设计,2008,29(1):27-30.

[2] 黄陇,陈致明,于洪敏,等.基于 UML 的软件测试用例自动生成技术研究[J].计算机应用与软件,2004,21(11):16-17.

[3] Sarma M,Mall R.Automatic test case generation from UML models[C]//10th International Conference on Information Technology,2007,26:196-201.

[4] 王纪立.基于 UML 顺序图的测试方法的研究与实现[D].南京:南京航空航天大学,2007.

(上接 66 页)

参考文献:

[1] Jennings N R,Dang V D.Generation coalition structures with finite bound from optimal guarantees[C]//Proc of the AAMAS 2004,2004,2:572-579.

[2] Anderson J,Tanner B,Baltes J.Dynamic coalition formation in robotic soccer[C]//Proc of the AAAI 2004,2004:1-11.

[3] Klusch M,Blankenburg B.On safe kernel stable coalition formation among Agents[C]//Proc of the AAMAS 2004,2004,2:580-587.

[4] Milch D K.Probabilistic models for agents' beliefs and decisions[C]//Proc of 16th Conference on Uncertainty in Artificial Intelligence (UAI-00),Stanford,California,2000:389-398.

[5] Konishi H.Coalition formation as a dynamic process[J].Journal of Economic Theory,2003,110:1-41.

[6] Sandholm T,Larson K,Andersson M,et al.Coalition structure generation with worst case guarantees[J].Artificial Intelligence,1999,111(2):209-238.

[7] 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.

[8] 张新良,石纯一.基于描述逻辑的 Agent 组织[J].计算机研究与发展,2005,42(11):1843-1848.

[9] 张双民,石纯一.基于群体 Agent 合作求解的测试床——MAS-Soccer[J].清华大学学报:自然科学版,2005,45(4):467-470.

[10] 张双民.群体 Agent 合作求解方法的研究[D].北京:清华大学,2005.

[11] 吴敏,曹卫华,桂卫华,等.一种新的多智能体系统结构及其在 RoboCup 中的应用[J].自动化学报,2006,32(5):686-694.