基于状态约简的顺序图和状态图一致性检测*

钱 成[1]，燕雪峰[1]，周 勇[1]，徐海生[2]
(1. 南京航空航天大学 计算机科学与技术学院，南京 210016；2. 中国船舶工业系统工程研究院，北京 100016)

摘 要：为了解决系统设计过程中模型一致性问题，提出了一种对 UML 顺序图和状态图的语义一致性检测方法。该方法对顺序图和状态图一致性进行符号化描述，为一致性检测提供理论基础；提出状态约简规则和状态约简算法，能够减少冗余状态和迁移，证明了状态约简不影响一致性检测；提出改进的 UML 模型到 PROMELA 的转换方法并使用 SPIN 进行验证。实验表明上述方法能够有效地检测顺序图和状态图的一致性，在验证过程中减少冗余状态和迁移，转换后的代码结构简单、执行效率高。

关键词：统一建模语言；顺序图；状态图；状态约简；模型一致性验证
中图分类号：TP311.5 **文献标志码**：A **文章编号**：1001-3695(2014)05-1452-04
doi:10.3969/j.issn.1001-3695.2014.05.040

Model checking consistency of sequence diagram and state machine based on state reduction

QIAN Cheng[1], YAN Xue-feng[1], ZHOU Yong[1], XU Hai-sheng[2]
(1. *School of Computer Science & Technology, Nanjing University of Aeronautics & Astronautics, Nanjing* 210016, *China*; 2. *System Engineering Research Institute, Beijing* 100016, *China*)

Abstract: The proposed a new method for model checking the consistency of the semantics of sequence diagram and state machine in the process of design. The method formalized the description of the consistency of sequence and state machine to establish the theoretical basis of model checking. This paper presented some state reduction rules and a state reduction algorithm which proved not affect the consistency to decrease the redundancy of states and transitions. Moreover, it translated the UML model to PROMELA which was the specification language of the model checker SPIN which was used to verify the consistency. Experiments indicate that, consistency of sequence diagram and state machine diagram is effectively verified, states and transitions are decreased during the model checking process, and the codes generated are simpler and more effective.

Key words: UML; sequence diagram; state machine; state reduction; model consistency checking

UML 作为可视化建模语言被广泛使用，它提供了一种使用不同视图对系统某个特征进行描述的方法。用户可以使用类图、用例图、复合结构图等来描述系统的静态特征，使用活动图、顺序图、状态图等来描述系统的动态特征，然而这些图之间可能是互相冲突、不一致的。模型检测作为一种用来检测给定的系统是否满足某些性质的自动验证技术，正广泛用于 UML 模型的性质验证。

状态图描述了对象通过某些事件触发作出的状态变换，顺序图描述了系统中的对象是如何通过事件和消息来交互的，这两种图用互补的形式描述了系统行为的不同方面，并且应用在系统设计的不同阶段中。当前使用模型检测技术对 UML 模型验证研究现状如下：文献[1]把状态图转换为作者开发的模型检测工具 PAT 并自动验证；文献[2]将安全策略模型的类图和状态图模型及性质转换为 PROMELA 并验证；文献[3]把状态图转换为 PROMELA，把协作图转换为自动机并使用 SPIN 检测两者之间的一致性；文献[4]为了生成系统的测试用例，提出了一种将各对象的状态图合并成系统级别的状态图的方法，并对合并后的状态图作出简化；文献[5]提出了顺序图中交互事件序列与其所有的状态图之间的一致性检查，并给出了一种将层次状态图分解并转换为 PROMELA 的方法；文献[6]对 UML 行为模型的验证提出模型转换为可执行 C 语言，然后验证模型是否满足 LTL 公式表示的属性的方法。

本文针对顺序图和状态图的一致性问题，旨在提供一种更高效的检测方法。

1 顺序图与状态图一致性

1.1 顺序图

顺序图通过突显出互相传递的消息序列和相应的生命线中的事件(event occurrence)来描述交互作用[7]。在 UML 建模过程中，通常使用顺序图来描述对象之间的交互，它将这些对象之间的交互建模成消息传递。为了对顺序图作一致性检测，对顺序图作如下定义：

定义 1 顺序图。一个顺序图是一个三元组 $SD=(L,M,E)$，其中：L 即生命线，是对象的有穷集合；$M=(N,S,R)$，即消息的有穷集合，其中 N 表示消息名称的全集，S 表示消息发送

收稿日期：2013-08-19；**修回日期**：2013-10-14 **基金项目**：国防科工局“十二五”重大基础科研资助项目(c0420110005)

作者简介：钱成(1989-)，男，江苏南通人，硕士，主要研究方向为系统建模与仿真、软件工程(qianchengnuaa@foxmail.com)；燕雪峰(1975-)，男，江苏泰兴人，副教授，硕导，博士，主要研究方向为分布交互仿真和计算机网络；周勇(1975-)，男，江苏南通人，副教授，硕导，博士，主要研究方向为人工智能；徐海生(1976-)，男，山东德州人，高级工程师，硕士，主要研究方向为舰艇电子信息系统.

者生命线集合,R 表示消息接收者生命线集合,定义消息名称函数 name:$M\to N$,定义消息发送者函数 sender:$M\to S$,定义消息接收者函数 receiver:$M\to R$; E 即事件的有穷集合。

UML 为顺序图中的交互定义了一种基于事件发生轨迹的语义,在其语义定义中,轨迹是事件发生的序列,记为$\langle e_1,e_2,\cdots,e_n\rangle$[8]。顺序图中每一条消息都含有两个有序事件,即发送事件和接收事件,因此消息也是顺序传递的。可以定义消息发生的先后顺序关系。

定义 2　$<_M$。它是 M 上的二元关系,设 M 是消息集合,满足$<_M=\{\langle m_1,m_2\rangle \mid m_1,m_2\in M\wedge m_1$ 在 m_2 之前传递$\}$。

顺序图描述的是多个对象之间的消息交互,而一致性检查要检测的是某个对象的生命线的消息序列与这个对象的状态图的一致性。因此,有必要对与单个生命线相关的消息序列语义进行定义。设 $M_l=\langle m_1,m_2,\cdots,m_n\rangle$是生命线 lifeline 上的消息序列,满足如下特征:对任意的 $1<i<j<n$,且 m_i、m_j 的发送者或者接收者为 lifeline,则 $m_i<_M m_j$。那么顺序图中生命线 lifeline 上的消息序列语义可表示为$\langle M_l,<_M\rangle$。

1.2　状态图

状态图着重描述对象和事件产生的状态迁移[9]。通常使用状态图对系统的动态特征进行建模,状态图包含了一个对象在其生命周期内所有状态的序列以及对象接收到事件所产生的反应。利用状态图可以精确地描述对象的行为:从对象的初始状态起,开始响应事件并执行某些动作,事件引起状态的迁移,迁移可能会执行某些动作;对象在新的状态下又响应事件并执行动作,如此循环直到状态终止。

定义 3　状态图。一个状态图是一个六元组 $SC=(S,T,s_0,Ev,Ga,Ac)$。其中:S 是状态图中所有状态的有限集合及复合状态到子状态的扩展函数 sub:$S\to S$;T 是状态图中所有迁移的集合及扩展函数,包括迁移的源状态函数 source:$T\to S$,迁移的目标状态函数 tar:$T\to S$,迁移的触发事件函数 tr:$T\to Ev\cup\phi$,迁移的监护条件函数 ga:$T\to Gv\cup\phi$,迁移的动作函数 act:$T\to Ac\cup\phi$;s_0 是初始状态;Ev 是状态图中事件的集合;Ga 是状态图中监护条件的集合;Ac 是状态图中动作的集合。

设状态图按照给定的触发事件集合 Ev 进行状态迁移的过程是状态图的一次执行,那么一次执行中迁移的执行是有序的,因为迁移的触发事件是按时间顺序执行的。特别地,对于并发状态,假设系统在执行过程中会将当前能触发的迁移按随机顺序执行,因此可以定义在状态图的一次执行中迁移的先后关系。

定义 4　$<_T$。它是 T 上的二元关系,设 T 为迁移集合,满足$<_T=\{\langle t_1,t_2\rangle \mid t_1,t_2\in T\wedge t_1$ 在 t_2 之前执行$\}$。

由此,可以对基于迁移执行序列的状态图的语义进行定义,设 $T=\{t_1,t_2,\cdots,t_n\}$是在给定的触发事件集合 Ev 触发的一个迁移集合,满足对任意 $1\leqslant i<j\leqslant n$,有 $t_i<_T t_j$,那么状态图基于迁移执行序列的语义可以表示为$\langle T,<_T\rangle$。

1.3　顺序图与状态图语义一致性

顺序图和状态图分别表示了对象的外部交互特性和内部状态特性,两者是紧密联系的,当一个对象与其他对象发送或者接收消息时,对象内部则发生状态迁移。如果顺序图生命线上的每一条消息序列,在状态图中都有一个迁移序列与之对应,则顺序图与状态图是一致的。顺序图与状态图的一致性定义为类型一致性和执行踪迹一致性。

定义 5　类型一致性。顺序图与状态图类型一致性是消息与迁移中触发事件或动作的类型一致性。在顺序图和状态图的一次执行中,令 SD 为一个顺序图,M 为 SD 的消息序列;令 SC 为状态图,T 为 SC 的迁移集合。如果对任意 $t\in T$, $\mathrm{tr}(t)\neq\mathrm{null}$ 或者 $\mathrm{act}(t)\neq\mathrm{null}$,都存在唯一 $m\in M$,使得消息 m 触发了迁移 t 或者迁移 t 的动作传递消息 m,那么称顺序图 SD 和状态图 SC 满足类型一致性,同时消息 m 与迁移 t 类型一致。

定义 6　执行踪迹一致性。顺序图与状态图执行踪迹一致性是如果对任意 $m_1\in M,m_2\in M,t_1\in T,t_2\in T$,其中 m_1 和 t_1 类型一致,m_2 和 t_2 类型一致,当且仅当 m_1 在 m_2 之前传递,即 $m_1<_M m_2$,有 t_1 在 t_2 之前执行,即 $t_1<_T t_2$,则顺序图与状态图满足执行踪迹一致性。

图 1 展示了对象 O_1 与 O_2 之间的消息交互和对象 O_2 的状态迁移过程。在顺序图中,对象 O_2 先后接收到一个消息 recv()并发送一个消息 send();在 O_2 的状态图中,状态 S_1 由触发事件 recv()产生迁移,状态变更为 S_2,状态 S_2 有一个不带触发事件的迁移,执行动作 send()并迁移到 S_3。图中两条消息与两个迁移的类型一致并且执行踪迹一致,所以顺序图与状态图是一致的。

2　基于状态图约简的一致性验证方法

2.1　针对无触发迁移的状态图约简算法

定义 7　无触发迁移。如果一个迁移不含有触发或动作,则此迁移为无触发迁移。

状态图的建模过程中,为了直观表示对象的状态及迁移,通常会加入伪状态如初始、终止、合并、分支等,伪状态与其他状态之间的迁移通常是无触发迁移;并且两个一般状态之间的迁移也可能是无触发迁移,用来表示状态的完成。无触发迁移不包含触发事件或者动作,迁移的过程与其他对象没有交互,在后续一致性检测过程中是冗余信息。

与直接提取状态图信息并转换为 PROMELA 的方法不同,该方法首先对状态图遍历,对无触发迁移作出预处理。现提出如下三条规则,用于约简无触发迁移及迁移的目标状态和相关迁移。

规则 1　如果一个状态迁移为无触发迁移,且目标状态不为复合状态,则此迁移可以被约简。

规则 2　如果一个迁移可以被约简,且源状态和目标状态之间没有非无触发迁移,则可以迁移的目标状态可以被约简。

规则 3　如果一个迁移 T 与目标状态 S 可以被约简,S_1 为 T 的源状态,则遍历所有迁移 T_n,如果迁移的目标为 S,则将该迁移的目标设为 S_1;如果迁移的源为 S,则将该迁移的源设为 S_1。

图 1 所示的顺序图描述了对象 O_2 接收一条消息 recv()并发送一条消息 send(),状态图描述状态由 S_1 迁移到 S_2 再到 S_3 的过程,其中初始状态到 S_1 的迁移与 S_3 到终止状态的迁移为无触发迁移,在状态提取时可以被约简。按照规则 1、2、3 约简后的状态图如图 2 所示。

按照上述规则,设计状态图约简算法,算法可以解析状态图元素并对无触发迁移进行约简。算法使用广度优先遍历来对状态图搜索,对搜索的每一个迁移按照定义 1 进行判断。如果是无触发迁移,按照规则 1 ~ 3 操作;如果是有效迁移,则保留并等待验证。状态图的约简算法伪代码描述如下:

输入:原始状态模型的 XMI 文件。

输出:约减后状态图模型的 XMI 文件。

stateList←从 XMI 文件中读出初始状态;

allStateList←从 XMI 文件中读出所有状态；
allTransitionList←从 XMI 文件中读出所有状态迁移；
repeat
state←从 stateList 中读出第一个状态；
如果 state 是复合状态 then
　　statsList←获取 state 的子状态；
　　transitionList←获得 state 的所有状态迁移；
for i = 0 to transitionList. size( )
　　如果是无触发迁移 then
　　按照规则 1、2、3 处理无触发迁移
　　statsList←获取 state 的下一个状态；
end for
从 stateList 中删除 state 状态；
在 allStateList 中将 state 设置为已到达；
until stateList. size( ) = =0
删除 allStateList 中不可到达的状态；
将约减后的状态和状态迁移保存到目标 XMI 中

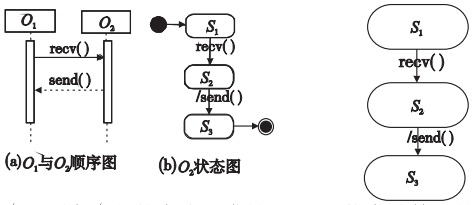

图1 顺序图与状态图一致性　图2 状态约简后的状态图

对状态图的约简操作删除了冗余迁移和状态，简化了要检测的状态图。下面证明使用约简算法不影响顺序图与状态图的一致性。设$\{m_1, m_2, \cdots, m_n\}$是顺序图生命线 l 的语义$\langle M_l, <_M\rangle$上的一个消息序列，$\{t_1, t_2, \cdots, t_m\}$是状态图语义$\langle T, <_T\rangle$上的迁移执行序列，并且两者满足类型的一致性和执行踪迹的一致性。证明对状态图使用状态约简操作不影响其与顺序图的一致性，即证明已知$\langle M_l, <_M\rangle$与$\langle T, <_T\rangle$满足一致性；对$T' = \{t_1, t_2, \cdots, t_k\}$，有$\langle M_l, <_M\rangle$与$\langle T', <_T\rangle$满足一致性，其中$T' \subset T$且$\forall i. 1 \leqslant i \leqslant k$有$t_i$不是无触发迁移。

显然，由于状态约简操作只影响无触发迁移，对任何$t \in T'$，总有$t \in T$且t不是无触发迁移。所以在一个$m \in M_l$，使得m和t类型一致，由类型一致性定义可知$\langle M_c, <_M\rangle$与$\langle T', <_T\rangle$满足类型的一致性。

由上可知，对$\forall t_{x'}, t_{y'} \in T', x' < y'$且$t_{x'}, t_{y'}$不是无触发迁移，则必有$t_{x'}, t_{y'} \in T$。又$\langle M_l, <_M\rangle$与$\langle T, <_T\rangle$满足执行踪迹的一致性，所以对任意$m_{a'}, m_{b'} \in M_l, a' < b'$，如果$t_{x'}$和$m_{a'}$类型一致，$t_{y'}$和$m_{b'}$类型一致，当且仅当$m_{a'} <_M m_{b'}$有$t_{x'} <_T t_{y'}$，即$\langle M_l, <_M\rangle$与$\langle T', <_T\rangle$满足执行踪迹一致性。

综上所述，对状态图使用约简算法不影响其与顺序图的一致性的结论成立。

2.2 UML 模型到 PROMELA 的转换

SPIN 最初是用来验证协议的，其建模语言 PROMELA 建模的方式是以进程为单位，进程之间异步交错执行。执行过程中，SPIN 通过产生的随机数随机选择路径执行，直到到达最大执行步数或者出错[10]。针对文献[5,11]给出的状态图的转换方法程序结构复杂且会产生冗余状态和迁移的问题，本文提出一种改进的 UML 模型到 PROMELA 转换方法，方法将顺序图和状态图分别转换为一个进程，并定义一个信道，两个进程之间使用信道进行通信。各个状态之间不是交错乱序执行，而是通过信道的接收发送消息来顺序执行。

顺序图进程根据发送或接收的消息序列依次向状态图进程发送或者接收消息，状态图同样根据触发和动作来接收或者发送消息。顺序图和状态图中的语句执行不是任意的，而是由两者之间的消息序列来确定的。如果状态图接收的消息与迁移的触发一致，或者顺序图接收的消息与状态图的动作不一致，进程会阻塞，SPIN 会终止执行并给出反例。表 1 给出了顺序图和状态图的基本元素与 PROMELA 之间的映射关系。

表 1　UML 模型与 PROMELA 映射关系

UML 模型	PROMELA 语句
消息	#define message (int) var
事件和动作	#define trigger (int) var; #define action (int) var
状态	#define state (int) var
顺序图和状态图	active proctype sequence( ) { } ; active proctype statemachine( ) { }
消息序列	messagechan! message; 或 messagechan? message;
迁移	if:: messagechan? trigger; skip; messagechan! action; skip; fi;
状态变更	state = (int) var;

根据上述转换方法，提出一种顺序图和状态图到 PROMELA 的自动转换算法，能够不需要手动干预地实现转换过程，转换算法步骤如图 3 所示。以图 1 为例，将图中所示的状态图经状态约简后再将顺序图和状态图模型转换为 PROMELA 的顺序图和状态图，进程代码片段如图 4 所示。

```
输入：messageList(顺序图消息序列)。
输出：顺序图对应PROMELA代码。
createProctForSequence(); // 创建顺序图的进程
Repeat
message←从messageList 中读出一条消息;
if message 是alt 或opt 片段
    输出 if::fi结构，createMessageChan(message );
if message 是par片段
    输出 do ::od 结构，createMessageChan(message ),
加入跳出条件，保证每条消息执行一次;
if message 是 loop 片段
    输出 do::od结构，按初始化值设置循环次数,
createMessageChan (message );
else createMessageChan (message);
从messageList 中删除 message ;
until messageList .size ()== 0
```

```
输入：stateList（状态图所有状态 ）
输出：状态图对应 promela 代码
createProctFor State (); // 创建状态图的进程
Repeat
state←从stateList 中读出一个状态 ;
transitionList ←获取state的所有迁移;
for (each transition in transitionList )
    输出 if::fi结构，createStateChan (transition );
end for
if state 是复合状态
    createProct (state); //创建子状态进程
    stateList ←获取state子状态;
从stateList 中删除 state
until stateList .size ()== 0
```

图 3 顺序图和状态图到PROMELA转换算法

```
active proctype sequence ()
{
do
:: messagechan!message_recv;
:: messagechan?message _send;
:: messagechan!message_send;
od
}
```

```
active proctype statemachine()
{
loop:
 state = S1;
if
:: messagechan?trigger_recv;skip;
fi;
state = S2;
messagechan !action_send;
if
:: messagechan? action_send;skip;
fi;
state =S3;
goto loop ;
}
```

图 4 图1中模型转换为PROMELA代码片段

3 一致性检测方法实例验证

集成设计平台是一套基于开源 UML 建模框架 Papyrus 的 UML 模型和领域模型建模工具。在上述研究及集成设计平台基础上，实现了 UML 顺序图和状态图一致性检测工具的原型。

本章实验使用文献[5]中订票系统为基础进行验证，图 5 是订票系统的部分顺序图，图 6 是订票系统中旅行社的状态图。图 5 左侧是与状态图一致的顺序图，描述了客户给旅行社发送订票消息 book，旅行社收到后向机场发送申请机票消息 requestTicket，在机场返回有余票的消息 hasTicket 后，旅行社给客户回复订票成功的消息 success；图 5 的右侧是与状态图不一致的顺序图，其中旅行社在收到机场消息前，错误地回复客户订票成功。图 6 的状态图描述了服务系统的状态迁移过程：服务系统从请求队列中应答一个请求，接收订票请求，当有余票时，到达成功状态；否则到达失败状态。

状态图中有多个可约简迁移和状态，如初始状态到 service 状态的迁移、stopped 状态到终止状态的迁移、复合状态 service

等。对状态图使用状态约简算法并将顺序图和状态图转换为 PROMELA 代码,使用 SPIN 验证工具对顺序图和状态图语义一致性进行检测,得到检测结果如图 7 所示。图 7 左侧和右侧分别是两个顺序图与状态图一致性检测结果,结果表明,左侧顺序图与状态图是一致的;对右侧顺序图与状态图,工具执行到 18 个状态和 18 个迁移的时候由于阻塞停止运行,说明两者是不一致的。

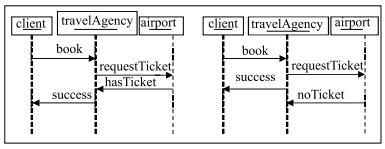

图 5　订票系统顺序图

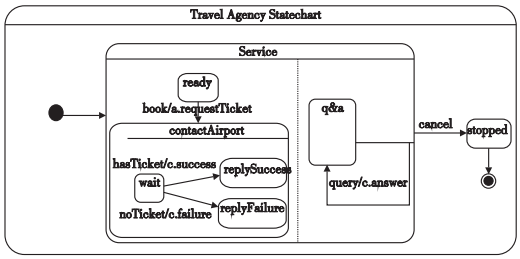

图6　订票系统中旅行社状态图

```
state ¬vector 24 byte, depth reached37,errors: 0
    27 states, stored
     2 states, matched
    29 transitions (= stored +matched )
     0 atomic steps
hash conflicts: 0 (resolved )

stats on memory usage (in Megabytes):
    0.001    equivalent memory usage for states
(stored *(State ¬vector + overhead ))
    0.290    actual memory usage for states
   64.000    memory used for hash table (¬w24 )
    0.343    memory used for DFS stack(¬m10000 )
   64.539    total actual memory usage
```

```
state ¬vector 24 byte, depth reached  24, errors: 1
    18 states, stored
     0 states, matched
    18 transitions (= stored +matched )
     0 atomic steps
hash conflicts: 0 (resolved )

stats on memory usage (in Megabytes ):
    0.001    equivalent memory usage for states
(stored *(State ¬vector + overhead ))
    0.290    actual memory usage for states
   64.000    memory used for hash table (¬w24)
    0.343    memory used for DFS stack(¬m10000 )
   64.539    total actual memory usage
```

图7　订票系统一致性检测结果

另外,为了验证改进的方法对状态数量的优化,本文对文献[3]中的 ATM 系统、文献[5]中的订票系统、文献[6]中的安全系统、文献[11]中的电话呼叫系统和集成设计平台实际应用中的攻防系统的一致性进行验证,并将验证过程中产生的状态数量进行对比。图 8 和 9 给出了验证过程中复杂度情况,可以看出使用文中状态约简和转换方法后检测工具 SPIN 中产生的状态和迁移数量较少,并且随着模型复杂度的提高,检测效率的提高更为明显。

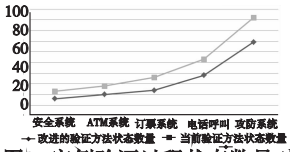

图8　实例验证过程状态数量对比

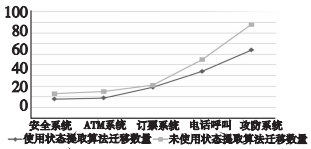

图9　实例验证过程迁移数量对比

4　结束语

本文给出了一种改进的顺序图和状态图一致性检测方法,并实例验了证该方法的可行性。该方法有如下优点:a)给出基于消息序列的顺序图语义和基于迁移序列的状态图语义,并对顺序图和状态图的语义一致性进行精确描述;b)提出了状态约简算法,减少检测过程中冗余状态和迁移;c)给出了改进的顺序图和状态图到 PROMELA 的转换方法,将顺序图和状态图分别转换成一个进程,转换后代码结构简单、执行效率高。后续工作包括对其他 UML 模型如对象图、协作图、活动图之间的一致性研究,给出检测出的一致性反例到 UML 模型的映射方案[12]等。

参考文献:

[1] ZHANG Shao-jie, LIU Yang. An automatic approach to model checking UML state machines[C]//Proc of the 4th International Conference on Secure Software Integration and Reliability Improvement Companion. [S.l.]:IEEE Press, 2010:1-6.

[2] CHENG Liang, ZHANG Yang. Model checking security policy model using both UML static and dynamic diagrams[C]//Proc of the 4th International Conference on Security of Information and Networks. New York:ACM Press, 2011:159-166.

[3] TIMM S, KNAPP A, MERZ S. Model checking UML state machines and collaborations[J]. Electronic Notes in Theoretical Computer Science, 2001, 55(3):357-369.

[4] MMIN H S, CHUNG S M, CHOI J Y. Deriving system behavior from UML state machine diagram: applied to missile project[J]. Journal of Universal Computer Science, 2013, 19(1):53-77.

[5] ZHAO Xiang-peng, LONG Quan, QIU Zong-yan. Model checking dynamic UML consistency[C] // Proc of the 18th International Conference on Formal Methods and Software Engineering. Berlin: Springer, 2006: 440-459.

[6] ORNA G, MELLER Y, YORAV K. Applying software model checking techniques for behavioral UML models[C]// Proc of the 18th International Symposium on Formal Methods. Berlin: Springer, 2012: 277-292.

[7] UML 2.0 superstructure specification[EB/OL]. http://www.omg.org/spec/UML/2.0/.

[8] 古思山,蔡树彬,李师贤. 一种 UML2 的交互的形式化语义[J]. 计算机科学与探索, 2012, 6(7): 631-643.

[9] RUSS M, HAMILTON K. Learning UML 2.0[M]. [S.l.]:O'Reilly Media, 2008.

[10] HOLZMANN G. SPIN model checker: the primer and reference manual[M]. Boston: Addison-Wesley Professional, 2004.

[11] 李兴锋,张新常,杨美红,等. 基于 SPIN 的模块化模型检测方法研究[J]. 电子与信息学报, 2011, 33(4) : 902-907.

[12] ISLAM A, SCHNEIDER S, TREHARNE H. Towards a practical approach to check UML/fUML models consistency using CSP[C]//Proc of the 13th International Conference on Formal Methods and Software Engineering. Berlin :Springer-Verlag, 2011: 33-48.

(上接第 1451 页)

[12] 覃晖,周建中,王光谦. 基于多目标差分进化算法的水库多目标防洪调度研究[J]. 水利学报,2009,40(5): 513-519.

[13] 刘波,王凌,金以慧. 差分进化算法研究进展[J]. 控制与决策, 2007,22(7):721-729.

[14] 王凌,钱斌. 混合差分进化与调度算法[M]. 北京:清华大学出版社, 2012:33-48.

[15] 孟红云,张小华,刘三阳. 用于约束多目标优化问题的双群体差分进化算法[J]. 计算机学报,2008,31(2):228-235.

[16] 肖术骏,朱学峰. 一种改进的快速高效的差分进化算法[J]. 合肥工业大学学报,2009,32(11):1700-1703.

[17] 张雪霞, 陈维荣, 戴朝华. 带局部搜索的动态多群体自适应差分进化算法及函数优化[J]. 电子学报,2008,38(8):1825-1830.

[18] 戈剑武,祁荣宾,钱锋,等. 一种改进的自适应差分进化算法[J]. 华东理工大学学报,2009, 35(4):600-605.

[19] PAN Quan-ke, SUGANTHAN P N, WANG Ling, *et al*. A differential evolution algorithm with self-adapting strategy and control parameters [J]. Computers & Operations Research, 2011, 38(1):394-408.

[20] MALLPEDDI R, SUGANTHAN P N, PAN Quan-ke, *et al*. Differential evolution algorithm with ensemble of parameters and mutation strategies [J]. Applied Soft Computing, 2011, 11(2):1679-1696.

[21] LIAO T W. Two hybrid differential evolution algorithms for engineering design optimization[J]. Applied Soft Computing, 2010, 10(4) : 1188-1199.