ISSN 1000-9825, CODEN RUXUEW
Journal of Software, Vol.18, No.1, January 2007, pp.94−104
DOI: 10.1360/jos180094

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel/Fax: +86-10-62562563

一种基于模型的特征交互检测方法*

左继红+, 王千祥, 梅 宏

(北京大学 信息科学技术学院 软件研究所,北京 100871)

A Model-Based Approach to Detecting Feature Interactions

ZUO Ji-Hong+, WANG Qian-Xiang, MEI Hong

(Institute of Software, School of Electronics Engineering and Computer Science, Peking University, Beijing 100871, China)

+ Corresponding author: Phn: +86-13810096800, E-mail: zuojihong@gmail.com

Zuo JH, Wang QX, Mei H. A model-based approach to detecting feature interactions. *Journal of Software*, 2007,18(1):94−104. http://www.jos.org.cn/1000-9825/18/94.htm

Abstract: To adapt to continually changing businesses, many software systems have to undergo evolutions through plugging new extensions into common base subsystems. Although it can facilitate concurrent development and deployment, this evolution strategy faces the problem of unexpected feature interactions between extensions. So far, formal method is still one of the most effective methods to detect feature interaction problems. The method has been proved to be successful by some small scale experiments. However, it also faces some challenges, e.g., the non-monotonicity of extension, the fast increase of extension combinations and the lack of extension details due to market competitions. Actually, many feature interactions are caused by the inappropriate modifications to the base subsystem and the existing extensions by new extensions. Based on this observation, the paper proposes a new approach for analyzing the causes of feature interactions and for formulating the corresponding constraints such that the conflicts with the same reason can be avoided. The approach has been applied to the analysis of the feature interactions of telecom systems. The experimental results show that most of the conflict-prone behaviors can be discovered quickly. In addition, the approach can help preserve the stability of the original base model and extension models, bypass the problem of extension combinations, and eliminate the requirement to publish the details of all extension models.

Key words: feature interaction; model checking; model; constraint; non-monotonicity; formal method

摘 要: 为了适应业务的不断更新,许多软件系统通过向公共的基础系统插入新的扩展来实现演化.这种演化策略虽然有利于并行开发和部署,但也面临着扩展间可能发生非预期特征交互的问题.目前,形式化方法在检测特征交互问题方面仍然是最有效的方法之一.这类方法着眼于检测扩展之间是否会发生冲突.虽然在小规模实验上较为成功,但是它们也面临着一些挑战.例如:扩展的非单调性、扩展组合的激增以及扩展模型可能无法获知的问题.实际上,许多特征交互都源于新扩展对基系统和已有扩展造成的不恰当影响.基于这种认识,集中关注由于扩展的不恰当影响所导致的交互冲突问题,提出了如何从已知的特征交互实例来分析产生冲突的原因的

* Supported by the National Natural Science Foundation of China under Grant Nos.60233010, 90412011 (国家自然科学基金); the National Grand Fundamental Research 973 Program of China under Grant No.2005CB321805 (国家重点基础研究发展规划(973))

Received 2005-11-28; Accepted 2006-05-24

具体方法,并说明了如何制定约束以限制扩展中易导致冲突的行为,从而预防同一类行为可能导致的各种冲突.该方法被应用到电信系统特征交互的分析上,实验结果表明,大部分特征交互中导致冲突的行为都可以被检测出来.该方法不仅能够保证原有基系统或扩展模型的稳定、有效,避免扩展组合带来的问题,而且它无须公布扩展的模型细节.

关键词: 特征交互;模型检测;模型;约束;非单调性;形式化方法
中图法分类号: TP311 文献标识码: A

为了适应不断变化或新增的需求,许多软件系统采用基于扩展的演化机制:先设计出一个稳定的基础子系统(简称基系统),然后通过插入新的功能模块(简称扩展,在其他文献中也被称为服务、特征等)来增强系统的功能.一个典型的例子是电信软件,该软件的基系统负责连接的建立和管理.此外,系统还提供一系列附加的特征,如:呼叫等待、呼叫转移等等[1].基于扩展的演化策略支持不同扩展的并行开发和部署,因此有助于提高服务的开发效率.然而,它也会导致特征交互问题(feature interaction problem),即当不同的扩展应用到基系统中时,可能会发生非预期的交互,从而使系统陷入混乱、矛盾的运行状态.这个问题最先在电信领域被广泛报道和研究[2].作为例子,我们来看一下电信系统中呼叫等待(call waiting,简称 CW)和无回答时呼叫转移(call forward-don't answer,简称 CFDA)这两个特征的交互.交互的根源在于它们对信号 idle_signal 的使用方式不同.idle_signal 由基系统定义,表示信号发送方的线路空闲.假设用户 *A* 同时订购了特征 CW 和 CFDA.当用户 *A* 在和用户 *B* 通话时,接到来自用户 *C* 的呼叫,则 CW 被激活,它在用户 *A* 的线路并非空闲的情况下向用户 *C* 发出 idle_signal*,以期掩盖用户 *A* 的线路繁忙的事实,因为一般用户听到电话忙音会立刻挂断,而听到空闲音则会多等待一会儿.CW 期望以这种方式让用户 *C* 短暂等待.然而,CW 发出 idle_signal 的系统环境与原先的假设不再一致.CFDA 的作用是:如果持续一段时间接到 idle_signal,则认为无人接听,随即将呼叫转移到另一条线路.因此,当 CFDA 持续接到 CW 发出的 idle_signal 后,会将呼叫转移,这就违背了用户 *A* 希望短暂等待之后接听用户 *C* 的意图[3].

特征交互问题广泛存在于基于扩展的演化系统中.除了电信软件之外,E-mail 系统[4]、IP 电话软件[5]、移动手机软件[6]都存在着类似现象.例如:在 E-mail 系统中,特征 RemailMessage 会修改邮件发送者,导致 VerifyMessage 的验证工作失败;在 VoIP 系统中,特征 Call Forward on Busy 会导致特征 Camp-on 失效.特征交互不仅会干扰系统的正常运行,而且会严重降低服务的开发效率.以电信软件为例,目前的系统部署有成百上千的特征,检测这些特征之间是否存在干扰非常费时、费力.新的市场和技术进展也给这个问题的解决带来了新的挑战:首先,传统 PSTN 网络、移动通信网络、Internet 网络正趋于融合,这可能导致网络上的 3 种服务发生复杂的交互[7];其次,像 IP 电话、电信软件等系统,服务的生产和消费形成了开放的服务市场,由于商业利益的不同,不同服务的提供商不愿公布服务的细节.因此,特征交互的解决方案必须考虑服务细节可能无法获知的约束[7].

针对特征交互问题,已经提出了很多技术方法.目前,形式化检测技术是研究最多、也是最有效的方法之一[1,8].已有的形式化方法都是着眼于直接检测特征之间的冲突.为此,首先要编写基系统和扩展的形式模型,然后将其组合成复合模型,再使用适当的工具检测复合模型是否违反某些属性.这些属性或者是系统特定的,或者是一般化的,如死锁、不确定性等等.为了检测冲突,复合模型一般要包含至少两个扩展的模型[9−16].小规模的实验(通常是选定 10 个左右的扩展)表明,这类方法是较为有效的.但是,这类方法面临一些棘手的问题:首先,大多数形式化检测方法都假设模型不会发生改变[8].而实际上,新的扩展可能以某种方式影响原有的基系统和某些既存的扩展,从而可能造成对原有基模型或扩展模型的假设失效,这一性质被称为扩展的非单调性(non-monotonicity)[17].非单调性被看作是对形式化检测的一大障碍.其次,待检测的扩展组合数量会随着扩展的数量迅速增长.假设有 n 个扩展,即使只考虑成对检测,扩展组合的数量将为 $O(n^2)$.因此,对于包含较多扩展的系统而言,现有检测方法的伸缩性(scalability)较差.最后,已有的检测方法假定所有扩展的模型细节都可以得到.而如上所述,如果扩展的生产和消费形成了一个开放的市场,要获得扩展的模型是不现实的.

* 考虑到这个问题,现在的电信系统中 CW 已作修改,不再发送 idle_signal.

对电信系统特征交互的分析表明,非单调扩展所导致的假设失效是特征交互的主要原因.这个观察与 Velthuijsen 在文献[17]中的观点相吻合.本文提出:可以在基模型中定义公共约束,通过检测单个扩展可能导致冲突的行为来限制非单调扩展.假设基系统 B 有两个扩展 E_1,E_2,其模型记为 M_B, M_{E_1} 和 M_{E_2}.如果(M_B, M_{E_1}, M_{E_2})上的交互存在冲突,且冲突是因为新扩展模型(比如说 M_{E_2})不恰当影响基模型 M_B 或已有扩展模型(比如说 M_{E_1})造成的,则可以设法分析出造成冲突的行为,从而制定相应的约束,将约束置于基模型 M_B.利用模型检测技术即可检测出 M_{E_2} 中易导致冲突的行为.而且,对任意的扩展模型 M_x,都可以通过对(M_B,M_x)的验证来检测 M_x 中是否包含类似的易导致冲突的行为.这样就能预防同一类行为可能引发的各种冲突.由于检测模型只包含基模型和单个扩展模型,故称其为单扩展检测方法.这种方法的价值在于:

- 有利于保证扩展的单调性.通过添加约束到基模型,能够防止新扩展对基系统和其他扩展的任意修改,这有利于保证原有基模型和扩展模型的稳定和有效;
- 方法的伸缩性更好.对于本文所提方法,复合模型只包含单个扩展模型,因此,不会出现扩展组合所带来的问题.图 1 显示了传统检测方法与本文方法的区别;
- 能够预防相同原因可能引发的各种冲突.本文方法不是着眼于冲突本身,而是导致冲突的行为原因.通过检测和限制易导致冲突的行为,能够预防这种行为可能导致的多种冲突;
- 无须公开扩展的设计细节.假设扩展来自不同的提供商,由于单扩展检测只要求带有约束的基模型被公开,检测完全可以在每个扩展的提供商本地完成,因而可避免公布扩展的模型细节.

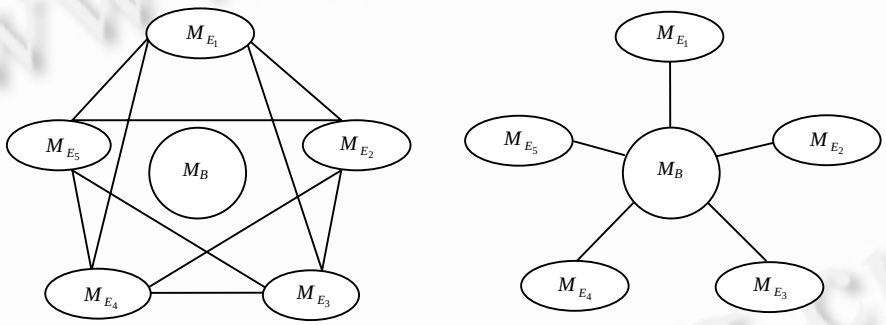

Fig.1 Configurations for traditional checking approach (left) and for our approach (right)

图 1 传统检测方法的配置(左)和本文方法的配置(右)

本文的主要贡献在于:提出了一种新的可行的检测方法.与以往的形式化方法不同,它的思路是通过向基模型中增加必要的约束,检测和限制可能导致冲突的行为,这不仅能够预防特征交互,而且能够保证原有基模型和扩展模型的稳定、有效;其次,根据经验归纳出 3 类易导致冲突的行为,并揭示了它们可能导致的后果,这有利于指导设计者为基模型和扩展模型定义约束.此外,本文还提出了如何根据已知的交互场景分析导致冲突的行为的可行方法.

利用这种方法,本文分析了文献[2]中的 18 例特征交互,发现其中大多数冲突的确源于新扩展对基系统或已有扩展的不恰当影响.在实际实验中,以 Promela 语言来描述这个系统的基模型和扩展模型,以断言的形式描述了对应的约束,然后使用 SPIN 工具即可很快检测出扩展模型中易导致冲突的行为.

本文第 1 节讨论本方法的基本原理,并给出实例.第 2 节讨论如何扩增基模型以增强检测方法的能力.第 3 节提供实验细节.第 4 节介绍相关工作.第 5 节总结全文.

1 方 法

1.1 基本原理

以电信基础呼叫模型(basic call model,简称 BCM)及其上的两个电信特征——呼叫等待和无回答时呼叫转

移的交互为例,冲突的根源在于 idle_signal 的发送条件发生改变.idle_signal 由 BCM 定义,发送 idle_signal 有一个潜在的条件,即信号发送方的线路空闲.而且,这个潜在条件也体现在 CFDA 的假设之中.可是,CW 在线路非空闲时发送 idle_signal 会违反这个潜在条件,它不仅会使 BCM 和 CFDA 的设计失效,而且也会导致冲突的发生.因此,应该把基系统的潜在条件显式地表达出来,例如,可以用谓词逻辑的形式表示如下:

$$f\text{: send(idle\_signal},x,i)\rightarrow \text{line\_state}(i,\text{idle}).$$

f 表示若线路 i 处的进程 x 发出 idle_signal,则线路 i 的状态必须为 idle. f 可以作为约束加入 BCM 的模型,再结合 BCM 模型与 CW 模型进行检测,即可发现 CW 违反了 f.为了满足约束 f,必须调整 CW 的设计,去掉可能导致冲突的行为,例如,为 CW 引入新的信号 wait_signal.利用这种方法就能防止该冲突的出现.由于本文的重点是检测易导致冲突的行为,因此不过多论述如何调整扩展模型.值得指出的是:约束 f 限制的不仅是 CW 发送 idle_signal 的行为,而且也是所有扩展发送 idle_signal 的行为.如果任意一个扩展 E 的模型都满足 f,就能防止所有因发送 idle_signal 可能导致的冲突.

通过对已有交互实例的分析,可归纳出 3 类易导致冲突的行为.在介绍它们之前,我们首先引入一些记号.设有基系统 B 和扩展 E_1,E_2,它们各自的模型记为 M_B, M_{E_1} 和 M_{E_2}.对于 $x,y,z\in\{B,E_1,E_2\}$,令(M_x,M_y,M_z)表示 M_x,M_y,M_z 之间的交互,令(M_x,M_y)表示 M_x,M_y 之间的子交互(sub-interaction);且记号(M_x,M_y)中模型的前后顺序是有意义的,它表示 M_y 的行为影响了 M_x.令 Msg_x 表示定义在 M_x 上的消息集合,V_x 表示定义在 M_x 上的公共变量集合,对于变量 $v\in V_x$,S_v 表示 v 的所有可能取值构成的集合.

设 M_x 包含消息集合 Msg_x 和公共变量集合 V_x,变量 $v_1,v_2\in V_x$,消息 $m\in Msg_x$,关系 $R\subseteq S_{v1}\times S_{v2}$,则 M_y 中易导致冲突的行为包括如下 3 类:

(1) 消息发送类型(SEND):M_y 发送消息 m:
可能后果 1:行为违反发送 m 的前置条件;
可能后果 2:行为改变 m 的内容.

(2) 消息截取类型(RECEIVE):M_y 截取到并非显式发送给它的消息 m:
可能后果 1:行为抑制其他以 m 为输入的构件;
可能后果 2:行为违反接收 m 的后置条件.

(3) 关系修改类型(UPDATE):M_y 修改关系 R:
可能后果:行为违反 R 的属性.

设 M_y 包含行为β,记β的类型为 $Class_\beta$,若 $Class_\beta\in\{SEND,RECEIVE,UPDATE\}$,且至少有一个$\beta$的可能后果成立,则称 M_x 受β影响,称 m 或 R 为受β影响的对象.下面将以实例演示这 3 类行为的可能后果,并说明它们之所以会构成某些冲突的必要原因.

- 对于消息发送类型的可能后果 1 的说明:考虑电信基系统 BCM 和特征呼叫等待(CW)之间的交互.CW 发送 idle_signal,它违反了发送 idle_signal 的前置条件.由此可能造成其他扩展对发送条件产生误判,参见 CFDA[3].
- 对于消息发送类型的可能后果 2 的说明:考虑 E-mail 基系统和特征 Remail 之间的交互.Remail 能够修改用户邮件的 sender 域.对邮件内容的改变可能造成其他特征误判,例如 VerifyMessage 特征对邮件的验证就会失败[4].
- 对于消息截取类型的可能后果 1 的说明:考虑特征三方呼叫(three way calling,简称 TWC)和呼叫等待之间的交互.如果用户 A 同时使用这两个特征,当 A 与 B 通话时,用户 C 呼叫 A,用户 A 发出 flash_hook 信号,本意是希望 C 暂时处于等待状态,但是,flash_hook 被 TWC 截取,TWC 产生三方通话,使得 CW 被抑制[2].
- 对于消息截取类型的可能后果 2 的说明:考虑特征三方呼叫和特征 911 紧急呼叫(911 emergency call,简称 EC)之间的交互.用户 A 在与 B 通话时,如果想让用户 C 进入,建立三方通话,则必须先向 B 发出 hold_signal,接收 hold_signal 的后置条件是接收方必须进入 onhold 状态.但是,如果用户 B 是 911 的接

线员,则特征 EC 截取到 hold_signal 不进入 onhold 状态[2].

- 对于关系修改类型的可能后果的说明:考虑电信基系统 BCM 和特征个人通信服务(personal communication service,简称 PCS)之间的交互.设 $R=\{\langle uid,lid\rangle|uid\in S_{userID}\wedge lid\in S_{lineID}\wedge userID\in V_{BCM}\wedge lineID\in V_{BCM}\}$,它表示 BCM 上的电话线路与用户的绑定关系.$R$ 有一个潜在的属性:一条电话线路只关联一个用户.可是,PCS 能够修改 R,使一条线路关联多个用户.这就违反了 R 的属性,由此可能导致冲突.例如:呼叫等待 CW 原本由用户 A 订购,可是,用户 B 与用户 A 都使用同一条 PCS 线路,这样,当用户 B 在通话时,接到一个呼叫请求,CW 照样会发生作用[2].

归纳出 3 类易导致冲突的行为有利于指导设计者为基模型和扩展模型建立必要的约束.例如:对基模型 M_B 中定义的任意消息 m,可以询问 m 的发送是否关联前置条件、m 的接收是否关联后置条件、m 在传播过程中是否允许被修改.另一方面,这也有助于分析冲突的行为原因.定义 1 描述了我们如何根据上述 3 类行为来分析冲突的原因.

定义 **1**. 设交互(M_B, M_{E_1}, M_{E_2})是有冲突的,若存在 $x,y\in\{B,E_1,E_2\}$,$x\neq y$,使得 M_y 包含行为β,$Class_\beta\in\{SEND, RECEIVE, UPDATE\}$,且 M_x 受β影响,β出现在(M_B, M_{E_1}, M_{E_2})之中,则称冲突源于β.

若(M_B, M_{E_1}, M_{E_2})的冲突源于 M_y 的行为β,且 M_x 受β影响,则(M_B, M_{E_1}, M_{E_2})上的检测可归约为(M_x,M_y)上的检测,并称(M_B, M_{E_1}, M_{E_2})上的检测是可归约的.由于本文的目标是防止非单调扩展所造成的冲突,因此,我们集中考虑可归约的交互检测.(M_B, M_{E_1}, M_{E_2})上的检测可以归约为如下 4 种形式(如图 2 所示).注意:(M_B, M_{E_1}, M_{E_2})上的检测不可能被归约为(M_{E_1}, M_B)或(M_{E_2}, M_B)上的检测,因为我们假设基系统不会对扩展造成不恰当影响.

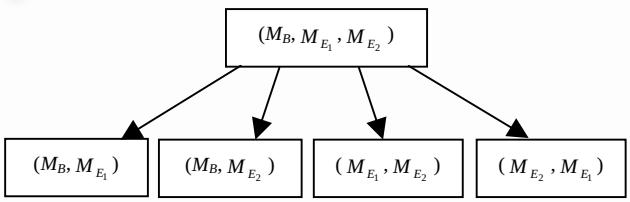

Fig.2 Four possible reductions of interaction verifications

图 2 交互检测的 4 种可能归约形式

在根据定义 1 定位了导致冲突的行为之后,下一步就需要制定约束.约束的目的是限制扩展模型中易导致冲突的行为,以保持基模型的稳定.设行为 B 的类型为β,它所影响的对象是 d,则可定义约束 P:$B(\beta,d)\rightarrow C$.P 声明:如果发生行为 B,则 B 应满足条件 C.虽然约束 P 来自对已有交互(M_B, M_{E_1}, M_{E_2})的分析,但是,该约束并不局限于用来检测 M_{E_1} 或 M_{E_2} 中易导致冲突的行为,它还可以用来限制所有扩展模型中同一类易导致冲突的行为.这种能力正是单扩展检测方法有用性的基础.

约束一般使用断言或者时序逻辑公式表达.在我们的研究中使用了 SPIN 工具的 Promela 语言建模,基模型由一组并发的进程组成.约束以断言的形式被加入到一个独立的监视进程中.扩展模型是以独立进程的形式与基模型相结合的.为了检测扩展的行为,扩展进程必须通过一个特殊的监视信道向监视进程汇报其外部行为.例如:当线路 i 上的扩展进程 p 发出消息 idle_signal 时,应将 send(idle_signal,p,i)通知监视进程,由监视进程判断此行为是否满足给定的约束.独立的监视进程和监视信道保证了新添加的约束对原有基模型不会产生影响.

值得注意的是:如果基模型中包含多条约束,怎样保证它们彼此是无矛盾的?本文的策略是:对于同一个类型且作用在同一个对象上的行为,保证它们所满足的条件是唯一的.形式地,设有约束 $B_1(\beta_1,d_1)\rightarrow C_1$ 和约束 $B_2(\beta_2,d_2)\rightarrow C_2$,若$\beta_1=\beta_2$,$d_1=d_2$,则 $C_1\equiv C_2$.

1.2 实 例

本节将演示如何分析导致冲突的行为,进而解释如何分解交互、如何制定约束.交互$(M_{\mathrm{BCM}},M_{\mathrm{CW}},M_{\mathrm{CFDA}})$出现了两种易导致冲突的行为:

- β_1:CW 在线路非空闲的条件下发送 idle_signal.此行为属于发送消息类型,即 $Class_{\beta_1}$=*SEND*.
- β_2:CFDA 截取并非显式发送给它的 idle_signal.此行为属于截取消息类型,即 $Class_{\beta_2}$=*RECEIVE*.

由第 1.1 节所述可知,β_1 的可能后果 1 是违反发送消息的前置条件.如前所述,发送 idle_signal 的前置条件是发送方线路为空闲状态.但是,CW 在发送 idle_signal 时,线路为非空闲状态.显然,β_1 的可能后果 1 成立,因此,冲突源于 β_1,M_{BCM} 受 β_1 的影响,对(M_{BCM},M_{CW},M_{CFDA})的检测可归约为对(M_{BCM},M_{CW})的检测.

再看 β_2,它的可能后果 1 是抑制其他以 idle_signal 为输入的构件,这里不存在此种情况,因此,可能后果 1 不成立.它的可能后果 2 违反了接收 idle_signal 的后置条件.在基系统中,idle_signal 只起着提示作用,并未规定信号接收方应采取什么行为,即接收 idle_signal 没有后置条件,故可能后果 2 不成立.因此,冲突并非源于 β_2,M_{BCM} 不受 β_2 的影响.

接下来是制定约束.这里使用 SPIN 工具的 Promela 语言作为形式化描述.如图 3 所示,约束被表示为监视进程 Monitor 中的一项断言,它声称某条线路上的特征在发送 idle_signal 时,该线路的当前状态是 s_ringing,前一时刻状态为 s_idle.这项约束显式地声明了发送 idle_signal 的前置条件.

```
proctype monitor() {
  mon_chan ? actor, action, signal, arg1, arg2→
    if
    :: (action==SEND) && (signal==idle_signal)→
       assert((state[arg1]==s_ringing) && (state_prev[arg1]==s_idle));
    …
```

Fig.3 The constraint on sending idle_signal in the BCM model

图 3 BCM 模型中对发送 idle_signal 的约束

此外,为了检测出 M_{CW} 中的不恰当行为,需将 M_{CW} 与 M_{BCM} 结合.如图 4 所示,在复合模型中,M_{CW} 对应于独立的进程 cw.进程 cw 发出 idle_signal 前的瞬间,通过监视信道 *mon_chan*,向监视进程 Monitor 主动汇报自己的外部行为.利用 SPIN 工具,可以很快检测出 CW 中所包含的 β_1 行为违反约束.显然,无论扩展模型是什么,只要它遵循此方法,都可以检测其是否包含发送 idle_signal 的行为以及该行为是否满足约束.

```
proctype cw(byte id) {
  …
  (state[id]==s_connected) && sig_chan[id]?[src, eval(id), call_request]→
    mon_chan ! CW, SEND, idle_signal, id, src;
    sig_chan[id] ! id, src, idle_signal;
    …
```

Fig.4 CW reporting its behavior of sending idle_signal to monitor

图 4 CW 模型将发送 idle_signal 的行为向监视进程汇报

2 扩增基模型

在为基模型制定约束时会遇到另外一个问题:无论表达形式如何,约束的参数都会包含对受影响对象的引用和其他一些相关变量或常量的引用.例如约束 *f*:send(idle_signal,*x*,*i*)→line_state(*i*,idle),其中会包含常量 idle_signal,idle 和变量 *x*,*i* 的引用.由于约束置于基模型 M_B 中,因此,要求约束引用的变量与常量的定义都包含在 M_B 中.但是,如果(M_B, M_{E_1}, M_{E_2})的检测被归约为(M_{E_1}, M_{E_2})或(M_{E_2}, M_{E_1})的检测,受影响对象的定义以及一些其他变量或常量定义是处于 M_{E_1} 或 M_{E_2},而非 M_B.这样就无法制定 M_B 中的约束.受此限制,仅能检测扩展模型中影响基模型的不恰当行为,而不能检测新扩展模型中影响已有扩展模型的不恰当行为.为了能够支持后一种不恰当行为的检测,需要扩增基模型.

设冲突是由 M_{E_1} 对 M_{E_2} 的不恰当影响所造成的,扩增的方法是在基模型 M_B 中添加 M_{E_2} 所包含的某些变量或常量定义、必要的辅助性代码片段以及相应的约束.称这是 M_B 相对于 M_{E_2} 的扩增.其中,辅助性代码片段的作用是在 M_B 中引入 M_{E_2} 的某些状态转换和外部行为.为了避免对基模型 M_B 造成影响,辅助性代码片段被置于一

个新的独立进程中.令 M_{B*}表示扩增后的基模型.M_B 相对于 M_{E_2} 的扩增实际上是 M_{E_2} 将保持模型稳定的任务委托给 M_{B*}.这样,在(M_{B*}, M_{E_1})上即可检测出 M_{E_1} 影响 M_{E_2} 的不恰当行为.类似地,对于任意扩展模型 M_x,可在(M_{B*}, M_x)上检测出 M_x 是否包含影响 M_{E_2} 的不恰当行为.

一个重要的问题是:如果两个扩展模型 M_{E_1}, M_{E_2} 相互影响,究竟应该由哪个扩展模型来承担冲突的责任?例如:假设分析表明,冲突既源于 M_{E_1} 的行为β_1, M_{E_2} 受β_1 影响;同时,冲突还源于 M_{E_2} 的行为β_2, M_{E_1} 受β_2 影响,且受β_1, β_2 影响的对象为 d,那么,M_B 应该相对于 M_{E_1} 扩增还是相对于 M_{E_2} 扩增?对此问题可能有许多种考虑.本文的原则是:依据应用的广泛程度与修改的难易程度,选择风险代价更小的扩展作为责任方.在上例中,假设修改 M_{E_1} 的代价更大,则判定 M_{E_2} 是责任方,因此,M_B 应该相对于 M_{E_1} 扩增.

作为例子,我们来看一下电信基础系统 BCM 上的两个特征三方呼叫和 911 紧急呼叫之间的交互.TWC 定义了 hold_signal.假设用户 *A* 正由于紧急情况和 911 接线员通话,同时,*A* 想利用 TWC 让用户 *B* 进入三方通话,为此,TWC 向 911 发出 hold_signal,期望对方接收到此信号进入 onhold 状态.而 EC 截取到 hold_signal 后并不进入 onhold 状态,而是维持原状态不变,于是造成建立三方通话的尝试失败.分析如下:此冲突只出现一个易导致冲突的行为——EC 截取 TWC 定义的 hold_signal 信号,此行为属于 RECEIVE 类型,它的可能后果 2(违反接收 hold_signal 的后置条件)成立,根据定义 1,冲突源于 EC 截取 hold_signal.

然而,由于约束所引用的变量或常量的定义位于 M_{TWC} 中,所以需要将基模型 M_{BCM} 相对于 M_{TWC} 扩增,使之包含 hold_signal,onhold 等常量的定义.此外,还需要在 M_{BCM} 中增加辅助性代码片段,如图 5 所示,这个代码片段被置于新的进程 bcm_aug 之中.它的作用是:如果在 s_connected 状态下从 *act_chan* 信道接受到用户的 a_flashhook 动作,即通过 *sig_chan* 信道发出 hold_signal.相应的约束置于 Monitor 进程中,如图 6 所示,该断言声明:如果扩展接收到 hold_signal 之后,线路必须处于 s_onhold 状态.最后,扩展模型 M_{EC} 以独立进程 ec 的形式被结合进模型 $M_{\mathrm{BCM}*}$.图 7 显示出:当 ec 进程从 *sig_chan* 信道接收到 hold_signal 时,即通过 *mon_chan* 信道向 Monitor 进程汇报此行为.利用 SPIN 就能检测出 M_{EC} 中易导致冲突的行为.

```
proctype bcm_aug(byte id) {
  do
  :: (state[id]==s_connected) && (act_chan[id] ? [eval(id), _, a_flashhook])→
       act_chan[id] ? eval(id), _, a_flashhook;
       sig_chan[id] ! id, partner[id], hold_signal;
  …}
```

Fig.5 The assisting code fragment for sending hold_signal

图 5 添加辅助性代码片段以发送 hold_signal

```
proctype monitor() {
     mon_chan ? actor, action, signal, arg1, arg2→
     if
     :: (action==POST_RECEIVE) && (signal==hold_signal)→
          assert((state[arg1]==s_onhold));
     …
```

Fig.6 The constraint on receiving hold_signal in the BCM model

图 6 向 BCM 模型添加接收 hold_signal 的约束

```
proctype ec(byte id) {
    (state[id]==s_connected) && (sig_chan[id] ? [src, eval(id), hold_signal])→
    sig_chan[id] ? src, eval(id), hold_signal;
    /* doing nothing when receiving hold signal */
    mon_chan ! EC, POST_RECEIVE, hold_signal, id, src;
    …
```

Fig.7 EC reporting its behavior of receiving hold_signal to monitor

图 7 EC 模型将接收 hold_signal 的行为向监视进程汇报

3 实例研究

本文选用电信领域的特征交互问题进行了实例研究,它包含了文献[2]中列举的18例特征交互.分析结果见表1,其中:M_{BCM}表示基础系统模型;M_{BCM^*}表示扩增后的基础系统模型.

表1列出了所有18例交互冲突,每例交互有两个特征,共包含19个特征.对于每例交互冲突,解释了是什么行为导致冲突,并简要说明了行为的后果.此外,表中还表明两个特征模型加基模型上的检测是否可以被归约,如果可以,则列出了归约后的检测所包含的模型.从表中可见:只有标记为 S5,S18 的冲突不是源于不恰当影响(S5 是由公共数据配额的限制所导致的特征间相互排斥;S18 是由于特征同步访问、修改公共变量所导致的活锁),其余16例冲突均源于某种不恰当影响,其中:11例是源于扩展影响基系统的行为;5例是源于新扩展影响已有扩展的行为.

Table 1 The analysis results for the feature interactions in telecom

表 **1** 电信软件中特征交互的分析结果

S1: Call waiting vs. Answer call	Reduced verification: (M_{BCM},M_{CW}), (M_{BCM},M_{AC})
Conflict-Prone behaviors: Both features intercept the call request signal defined by BCM, thus inhibiting each other.	
S2: Call waiting vs. Three-Way calling (SUSC)	Reduced verification: (M_{BCM^*},M_{TWC})
Conflict-Prone behaviors: TWC intercepts the flash hook signal defined by CW, thus inhibiting CW.	
S3: 911 Emergency call vs. Three-Way calling	Reduced verification: (M_{BCM^*},M_{EC})
Conflict-Prone behaviors: EC intercepts the hold signal and does not produce the expected response.	
S4: Terminating call screening vs. Automatic ReCall	Reduced verification: (M_{BCM},M_{ARC})
Conflict-Prone behaviors: ARC intercepts the call request signal, thus inhibiting TCS.	
S5: Originating call screening vs. Area number calling	Reduced verification: None
Explanation: It is not caused by some modification but is caused by the restriction on the quota of some public variable.	
S6: Operator services vs. Originating call screening	Reduced verification: (M_{BCM},M_{OS})
Conflict-Prone behaviors: OS intercepts the call request signal, thus inhibiting OCS.	
S7: Credit-Card calling vs. Voice-Mail service	Reduced verification: (M_{BCM^*},M_{CCC})
Conflict-Prone behaviors: CCC intercepts the “#” signal defined by VM, thus inhibiting VM.	
S8: MBS-ED vs. CENTREX	Reduced verification:(M_{BCM^*},$M_{MBS\text{-}ED}$), (M_{BCM^*},$M_{CENTREX}$)
Conflict-Prone behaviors: Both features can intercept a 4-digit number, thus inhibiting each other.	
S9: Call Forwarding vs. Originating call screening (MUSC)	Reduced verification: (M_{BCM},M_{CF})
Conflict-Prone behaviors: CF intercepts and sends the call request signal, thus inhibiting OCS.	
S10: Call waiting vs. Personal communication services	Reduced verification: (M_{BCM},M_{PCS})
Conflict-Prone behaviors: PCS changes the one-line-one-user relationship, thus invalidating the supposition of CW.	
S11: Originating call screening vs. MDNL	Reduced verification: (M_{BCM},M_{MDNL})
Conflict-Prone behaviors: MDNL changes the one-line-one-number relationship, thus invalidating the supposition of OCS.	
S12: Call forwarding vs. Originating call screening (MUMC)	Reduced verification: (M_{BCM},M_{CF})
Conflict-Prone behaviors: CF intercepts and sends the call request signal, thus inhibiting OCS.	
S13: Call waiting vs. Automatic CallBack	Reduced verification: (M_{BCM},M_{CW})
Conflict-Prone behaviors: CW sends the idle_signal, thus changing the precondition on sending the signal.	
S14: Call waiting vs. Call waiting	Reduced verification: (M_{BCM},M_{CW})
Conflict-Prone behaviors: One CW instance intercepts the onhook signal, thus inhibiting another CW instanc.	
S15: Call waiting vs. Three-Way calling (MUMC)	Reduced verification: (M_{BCM},M_{CW}), (M_{BCM},M_{TWC})
Conflict-Prone behaviors: Both features can intercept the onhook signal, thus inhibiting each other.	
S16: Calling number delivery vs. Unlisted number	Reduced verification: (M_{BCM^*},M_{UN})
Conflict-Prone behaviors: UN intercepts the deliver number signal defined by CND, thus inhibiting CND.	
S17: Call forwarding vs. Call forwarding	Reduced verification: (M_{BCM},M_{CF})
Conflict-Prone behaviors: CF intercepts and sends the call request signal, thus leading to infinite loops.	
S18: Automatic CallBack vs. Automatic ReCall	Reduced verification: None
Explanation: It is not caused by some modification but is caused by the synchronous access and update of some variable.	

本文的实验使用了模型检测工具 SPIN,它常被用于检测通信协议是否安全、可靠[18].SPIN 支持的进程间交互方式包括同步通信、异步通信、读写公共变量.该工具使用 Promela 作为描述语言,它的语法结构类似 C 语言.Promela 要求显式写出消息发送和接收,其语法形式借鉴了 CSP 的记号.例如:*chan_name* ! *msg* 表示通过 *chan_name* 的信道发送消息 *msg*;*chan_name* ? *msg* 表示通过 *chan_name* 的信道接收消息 *msg*.一个 SPIN 模型除了要定义一组进程之外,还需要定义一组变量和信道.

实验的模型包括一个电信基础系统模型 M_{BMC} 以及一组特征模型.M_{BCM} 的体系结构如图 8 所示,它包括如下进程:

- 一或多个 bcm 进程,它们代表用户端进程,其任务是响应用户动作和其他进程的外部信号.每个 bcm 进程都带有一个变量,表示它的线路 id.
- 一个 Network 进程,它是对网络的抽象,其主要任务是在不同信道之间转发 bcm 进程间的信号.
- 一个 Usr 进程,其任务是对所有的用户动作建模.
- 一个 Monitor 进程,其任务是监测进程的外部动作,判断约束是否被违反.

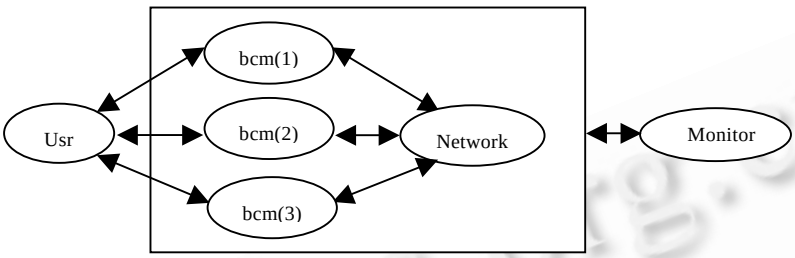

Fig.8 The architecture of BCM model

图 8 BCM 模型的体系结构

在一个典型的交互场景中,会有 3 个 bcm 进程实例,它们代表 3 个用户端进程.bcm(*i*)进程间并不直接通信,消息通过 Network 转发.每个 bcm(*i*)进程通过 *sig_chan*[*i*]信道与 Network 进程通信.例如:假设 bcm(1)要向 bcm(2)发送一条消息 *m*,bcm(1)先通过 *sig_chan*[1]发送 *m*,Network 进程通过 *sig_chan*[1]接收到 *m* 之后,通过 *sig_chan*[2]将 *m* 发送给 bcm(2),bcm(2)再从 *sig_chan*[2]接收 *m*.

此外,每个 bcm(*i*)进程通过 *act_chan*[*i*]信道与 Usr 进程通信,以便接收来自用户的动作.Monitor 是一个监视进程,除 Usr 之外,所有进程的外部行为都通过 *mon_chan* 信道向 Monitor 汇报.所有的约束均以断言的形式置于 Monitor 之内.当 Monitor 接收到一条行为汇报时,即判断该行为是否违反约束.

为了保证模块性,特征一般实现为独立的进程.特征的进程也带有参数,表示它是由哪个线路的用户所使用.例如,feat(1)表示线路 1 的用户所用的特征 feat.特征进程 feat(*i*)与对应的 bcm(*i*)进程不通信,feat(*i*)只是借用 *sig_chan*[*i*]与 Network 进程通信.此外,feat(*i*)也需要通过 *mon_chan* 向 Monitor 汇报自己的外部行为.

实验结果如下:在 16 例源于不恰当影响的冲突中,实验覆盖了其中 13 例.实验无法覆盖的 3 例交互是 S7,S8 和 S10,这是因为对于 S7 中的特征 CCC,S8 中的特征 MBS-ED,CENTREX 以及 S10 中的特征 PCS,文献中缺少必要的介绍,无法获得其模型细节.实验定义了 9 条约束,所有约束均以断言的形式表示.对于不同的约束,SPIN 的搜索深度差异很大,对于较快的检测搜索深度不超过 200,而对于较慢的检测搜索深度超过 15 万.

4 相关工作

根据文献[8],对特征交互问题的已有的形式化检测方法可以分为如下 3 类:

- 只使用属性.基系统和扩展都被描述为某种逻辑属性.交互冲突也被定义为某种逻辑公式,通常为不一致性(inconsistency)或者不可满足性(unsatisfiability).典型的工作参见文献[9].
- 只使用行为.使用某种形式的自动机或转换系统来描述基系统和扩展的行为模型.交互冲突被定义为某种一般化属性,如死锁、不确定性等.典型的工作参见文献[10,11].
- 结合使用属性和行为.除了为基系统和扩展定义行为模型之外,还要定义它们各自的特定属性,交互冲突被定义为某种属性被违反.典型的工作参见文献[12,13].

文献[9]使用一种经过改良的语言结构来包装线性时序逻辑(linear temporal logic,简称 LTL).特征被描述为一组逻辑属性,基系统则被描述为一组公理(axiom).为了形成检测模型,两种特征的逻辑属性将以析取(disjunction)的方式相结合.该方法使用了 COSPAN 工具来检测复合模型是否满足公理.

文献[10]使用类似于动作时序逻辑(temporal logic of action)的逻辑语言来描述特征和基系统的行为模型.复合模型是基系统模型与两种特征的逻辑公式的合取.文献[10]定义了模型的良构性.如果检测模型不是良构

的,则意味着交互冲突.文献[11]使用一种图形化的方式来描述基系统和特征的行为模型,该模型刻画了状态转换和消息交换.交互的检测方法由两个步骤组成:第一步是从图形规约中提取特征的触发信息,例如,特征的状态、触发消息和触发条件;第二步是对特征的触发信息进行两两分析.该方法可以检测两种交互:一种是直接交互,即两种特征具有相同的触发消息;另一种是间接交互,即一种特征的输出消息与另一种特征的触发消息重叠.

文献[12]使用 LOTOS 语言对电信系统建模,系统被描述为一个网络进程和一组用户进程,这些进程的通信方式是在特定动作上同步.每个用户进程由一个基系统子进程和一组特征子进程以逻辑析取方式组合而成.用户进程代表用户使用的特征集合与基系统形成的复合体.此外,文献[12]使用μ演算(μ-calculus)来描述系统特定的属性.作者声称,此方法能够支持多种交互冲突的分析,包括系统属性的不一致性和不可满足性,系统行为模型的死锁、不确定性以及行为模型违反系统属性.所用的检测工具是 CAESAR.文献[13]使用 Promela 语言对 E-mail 系统和特征进行建模,模型由一个网络进程和若干客户进程组成,基系统和特征的属性被表示为 LTL 公式.作者自行开发了 Perl 程序,它可以自动选择一对特征,进行参数配置,然后调用 SPIN 工具进行检测.

第 1 类方法的特点是对基系统和扩展的描述层次较高,因此,该类方法更适宜检测需求层次的特征交互;与之相对,在第 2 类方法中,系统模型的描述层次较低,模型包含了更多的具体实现细节,因此,这类方法更适宜检测设计层次的特征交互;第 3 类方法结合了较高层次的需求属性和较低层次上的行为模型,这类方法一般都是采用模型检测技术.从模型的特点来看,本文的方法更接近第 3 类方法.在本文方法中,要求定义基系统和扩展的行为模型,而约束则是系统特定的属性.

但是,已有的形式化方法与本文方法显著的不同在于:无论哪一类方法,其目标都是检测冲突本身.因此,这些工作都要求将两个以上的扩展的模型组合起来,进行两两检测;而本文方法的目标是检测冲突的原因,即易导致冲突的行为.这导致了如下的区别:从属性的内容来看,本文的属性描述的是扩展的消息发送、消息截取和关系修改 3 类行为应该满足的条件,而在已有方法中,属性描述的是基系统和扩展的高层需求;从属性的效用来看,本文的属性作用在于发现易导致冲突的行为,保持基模型和已有扩展模型的稳定.作为对比,在已有方法中,属性的作用在于发现冲突,它们无助于保证原有模型的稳定;从模型的组合来看,本文的复合模型只包含一个基模型和单个扩展模型,而已有方法的复合模型包含一个基模型和至少两个扩展模型.正因如此,本文方法可以规避扩展组合所造成的问题,也无须公布扩展的模型细节.同时,基模型和已有扩展模型的稳定、有效也能得到保证.

5 结 论

许多特征交互都源于新扩展对基系统和已有扩展的不恰当影响.为此,本文提出,应该通过向基系统模型添加约束来检测并限制这些易导致冲突的行为.本文归纳出 3 类易导致冲突的行为,并解释了它们的可能后果;在此基础上,提出如何分析产生交互冲突的原因,制定相应的约束.这种方法不仅可以保证原有基模型和扩展模型的稳定、有效,而且也能预防同一类行为所可能导致的各种冲突.它的其他优点包括:可规避扩展组合所造成的问题,并允许检测在提供商本地进行,不需要公布扩展的模型细节.实验表明:本文的方法是快速而有效的,大部分特征交互中导致冲突的行为都可以通过这种方法加以检测.现有方法的一个特点是:约束来源于对已有的特征交互的分析,因此,约束制定是在“事后”完成.实际上,由于已经归纳出 3 类易导致冲突的行为,某些约束应该可以在设计期“事前”提取,这对于指导设计和预防特征交互更有积极的意义.我们将在这方面做进一步研究.

References:

[1] Keck DO, Kuehn PJ. The feature and service interaction problem in telecommunications systems: A survey. IEEE Trans. on Software Engineering, 1998,24(10):779−796.

[2] Cameron EJ, Griffeth ND, Lin YJ, Nilson ME, Schnure WK, Velthuijsen H. A feature interaction Benchmark for IN and beyond. In: Bouma LG, Velthuijsen H, eds. Feature Interactions in Telecommunications Systems II. Amsterdam: IOS Press, 1994. 1–23.

[3] Grégoire JC, Ferguson MJ. Neglected topics of feature interactions: Mechanisms, architectures, requirements. In: Dini P, Boutaba R, Logrippo L, eds. Feature Interactions in Telecommunications Networks IV. Amsterdam: IOS Press, 1997. 3–12.

[4] Hall RJ. Feature interactions in electronic mail. In: Calder M, Magill E, eds. Feature Interactions in Telecommunications and Software Systems VI. Amsterdam: IOS Press, 2000. 67−82.

[5] Lennox J, Schulzrinne H. Feature interaction in internet telephony. In: Calder M, Magill E, eds. Feature Interactions in Telecommunications and Software Systems VI. Amsterdam: IOS Press, 2000. 38−50.

[6] Lorentsen L, Tuovinen AP, Xu JL. Modeling feature interactions in mobile phones. Technical Report, No.2001-14, Karlsruhe: ECOOP, 2001. 7−13.

[7] Kolberg M, Kimbler K. Service interaction management for distributed services in a deregulated market environment. In: Calder M, Magill E, eds. Feature Interactions in Telecommunications and Software Systems VI. Amsterdam: IOS Press, 2000. 23−37.

[8] Calder M, Kolberg M, Magill EH, Reiff-Marganiec S. Feature interaction: A critical review and considered forecast. Computer Networks, 2003,41(1):115−141.

[9] Felty A, Namjoshi K. Feature specification and automated conflict detection. ACM Trans. on Software Engineering and Methodology, 2003,12(1):1−24.

[10] Blom J, Bol R, Kempe L. Automatic detection of feature interactions in temporal logic. In: Cheng KE, Ohta T, eds. Feature Interactions in Telecommunications Systems III. Amsterdam: IOS Press, 1995. 1−19.

[11] Jouve H, Gall PL, Coudert S. An automatic off-line feature interaction detection method by static analysis of specifications. In: Reiff-Marganiec S, Ryan MD, eds. Feature Interactions in Telecommunications and Software Systems VIII. Amsterdam: IOS Press, 2005. 131−146.

[12] Thomas M. Modelling and analysing user views of telecommunications services. In: Dini P, Boutaba R, Logrippo L, eds. Feature Interactions in Telecommunications Networks IV. Amsterdam: IOS Press, 1997. 168−182.

[13] Calder M, Miller A. Generalising feature interactions in email. In: Amyot D, Logrippo L, eds. Feature Interactions in Telecommunications and Software Systems VII. Amsterdam: IOS Press, 2003. 187−204.

[14] Kawauchi S, Ohta T. Mechanism for 3-way feature interactions occurrence and a detection system based on the mechanism. In: Amyot D, Logrippo L, eds. Feature Interactions in Telecommunications and Software Systems VII. Amsterdam: IOS Press, 2003. 313−327.

[15] Yoneda T, Kawauchi S, Yoshida J, Ohta T. Formal approaches for detecting feature interactions, their experimental results and application to VoIP. In: Amyot D, Logrippo L, eds. Feature Interactions in Telecommunications and Software Systems VII. Amsterdam: IOS Press, 2003. 205−212.

[16] Bruns G. Foundations for features. In: Reiff-Marganiec S, Ryan MD, eds. Feature Interactions in Telecommunications and Software Systems VIII. Amsterdam: IOS Press, 2005. 3−11.

[17] Velthuijsen H. Issues of non-monotonicity in feature-interaction detection. In: Cheng KE, Ohta T, eds. Feature Interactions in Telecommunications Systems III. Amsterdam: IOS Press, 1995. 31−42.

[18] Holzmann GJ. The model checker SPIN. IEEE Trans. on Software Engineering, 1997,23(5):279−295.

左继红(1974－),男,四川成都人,博士生,主要研究领域为特征交互,软件工程.

梅宏(1963－),男,博士,教授,博士生导师,CCF 高级会员,主要研究领域为软件体系结构,软件工程.

王千祥(1970－),男,博士,副教授,CCF 高级会员,主要研究领域为软件工程,软件中间件,软件自适应技术.