基于时序逻辑的Object－Z类切片的扩展

佟长英，沈云付
（上海大学 计算机工程与科学学院，上海 200072）

摘 要：文中对程序切片进行了研究，报告了程序切片的现状，进行了Object－Z类切片方面的调查。为使Object－Z规格易于验证，对Object－Z作切片处理是必须的。以Object－Z的通用堆栈类为实例，对类的描述进行扩展，给出了相应的Kripke结构。通过面向对象程序依赖图对Object－Z进行切片，实现了Object－Z的抽象，给出了对于一个类或具有继承关系的多个类的刻划，即证明了在状态和事件上的公式在原模型与抽象模型中保持，扩展了Object－Z类切片方法。

关键词：程序切片；Kripke结构；Object－Z；时序逻辑；间隔逻辑；通用堆栈
中图分类号：TP311.5 **文献标识码**：A **文章编号**：1673－629X(2009)06－0013－04

Extending Slices of Object－Z Classes Based on Temporal Logic

TONG Chang-ying, SHEN Yun-fu
(School of Computer Engineering and Science, Shanghai Univ., Shanghai 200072, China)

Abstract: In this paper, program slicing is studied and present situation of program slicing is reported, then Object－Z class slicing is investigated. In order to verify Object－Z easily, slicing is must be done. The description of class is extended and the corresponding Kripke structure is presented by taking general stack of Object－Z as a case. Then the Object－Z is sliced by object－oriented program dependence graph and the abstraction of Object－Z is realized. A result about system properties by one class or many classes with inherited relations is proved. It is proved that formula on states and events preserves under origin model and abstract model. Finally, the slicing way of Object－Z class is extended.

Key words: program slicing; Kripke construct; Object－Z; temporal logic; interval logic; general stack

0 引 言

软件的正确性和可靠性是软件开发面临的一个主要问题，因此必须通过有效地验证才能保证软件的顺利使用。而Z[1]是一个基于集合论和逻辑的面向状态的形式规格语言，Object－Z[2]是对Z的一种面向对象扩展。Object－Z是编写软件需求规格的语言，通过对Object－Z进行验证能有效地减少软件需求规格中出现的错误，提高软件的质量。

切片是一种重要的程序分析和理解技术，自提出以来已经广泛应用于软件工程的各个领域，吴方君等[3]用Z语言来形式化描述程序切片，考虑了程序切片中诸如程序依赖图和程序切片算法等常用的方面，使得理解程序切片更为容易。程序切片分为前向切片和后向切片，易彤等[4]分析了两者的关系。最近，切片技术已经进入模型检测[5]领域，不仅变量值作为切片准则，时序逻辑公式也可以作为切片准则。同一般程序切片类似，在全部规格上具有的时序逻辑公式特定的属性，在切片后仍然保证存在该属性。

吴方君等[6]给出了对Z规格的切片，首先引入逻辑依赖，并发展了模式依赖图和规格依赖图，形成了规格依赖图，在此基础上对Z进行切片。Brückner等[7]提出了利用时序逻辑的方式对Object－Z类切片，在状态和事件之上的基于时序逻辑的间隔公式切片Object－Z规格。通过数据和控制依赖图构建规格依赖图，从公式中原子命题开始作后向遍历，并确定潜在影响这些原子命题的规格部分，由此得到Object－Z规格的切片。

但是对于状态和事件在一个操作模式中的Object－Z类以及具有继承关系的多个Object－Z类，却无法直接使用文献[7]中的方法进行切片。因此，给出了Object－Z类的操作模式中状态和事件的拆分方法，以扩展的通用堆栈类为实例，并给出了具有继承关系的两个Object－Z类的面向对象程序依赖图，依此作为切片的依据，该依赖图可以推广到多个类的情况，在作后向切片时，Object－Z类的操作模式需要满足拆分后

收稿日期：2008－10－08；**修回日期**：2009－02－11
基金项目：国家863项目(2007AA01Z144)
作者简介：佟长英(1980－)，男(满族)，吉林吉林人，硕士研究生，研究方向为模型检查；沈云付，博士，副教授，研究方向为模型检查。

的模式。文志诚等[8]给出了关于 Object－Z 多态的推理,文中给出了具有继承关系的多个 Object－Z 类的切片。

1 Object－Z 类实例分析

关于时序逻辑上的 Object－Z 映射关系,请参阅文献[7]。以下给出 Object－Z 类操作模式的拆分方法。

对一个 Object－Z 类,其中的操作模式按 enable 和 effect 方式拆分的方法为:该操作模式执行的判断条件谓词作为 enable 模式中的谓词,而声明部分和其它谓词分别作为 effect 模式中的声明和谓词。

现在对文献[1]给出的通用堆栈类 Stack[T]进行扩充,按上述方法,则原有的 pop 和 push 模式拆分为 enable_pop, effect_pop 模式和 enable_push, effect_push 模式,使得 pop 和 push 中的事件和状态分离。这里假定 MyStack[T]为 Stack[T]的子类,并且 MyStack[T]具有 Init, enable_remove, effect_remove 等模式,简单起见,MyStack[T]表示如图 1 所示。

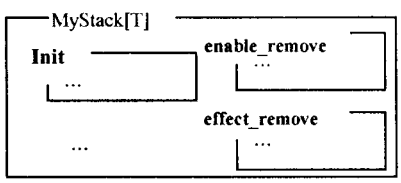

图 1 MyStack[T]

扩展的通用堆栈类 Stack[T]表示如图 2 所示。

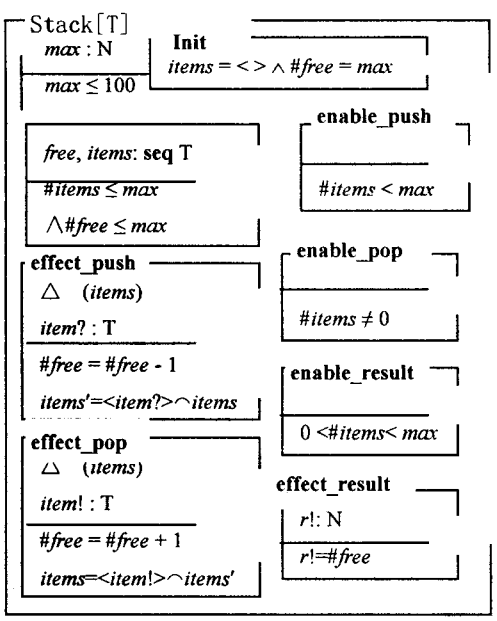

图 2 通用堆栈类 Stack[T]

对于通用堆栈类,感兴趣的是以下两个属性:

$\varphi_1 :=\square\lceil \#\text{free}=100-\#\text{items}'\rceil$

$\varphi_2 :=\neg\Diamond(\square\lceil \text{true}\rceil\wedge \text{push})\wedge\neg\Diamond(\square\lceil \text{true}\rceil\wedge \text{pop})$

φ_1 表示空闲块总是等于总块数与已使用块数的差,它显示的是与原始规格的差别,通过移除多余的谓词得到与模式 result 一致的最终结果。φ_2 表示的是并非会出现总是压栈或出栈的情况。在某个阶段可能一直会压栈或出栈,但并不一定出现这种情况,因此该属性可以作为一个反例的公式。

2 Object－Z 类的切片

对规格进行切片必须考虑切片准则,切片准则发生变化,则计算的规格切片也会不同[4]。程序切片中的切片准则考虑的是变量值和谓词,而模型检测中使用时序逻辑公式,因而就变得更为复杂。对于一般的切片,关于时序逻辑公式的切片的技术是一致的:切片需要关于不同程序/规格之间的依赖性精确的信息,这样的依赖性是在程序依赖图/系统依赖图(PDG/SDG)中表示。考虑到 Object－Z 面向对象的特点,下面构建 Object－Z 类的面向对象程序依赖图[4](OPDG)和关于 SE－IL 公式[7]的切片。

2.1 面向对象程序依赖图

假定 V 是类变量的集合,E 是其方法/事件,p 是 V 上的谓词。

对位于某个模式中的变量集合 vars(p) 上的谓词 p,mod(p) 定义为那些发生在主要形式中的变量,在模式的 Δ 列表中,ref(p) 定义为发生在次要形式中的变量。对于输入输出变量,这里设定:谓词 p 的输出变量在 mod(p) 中,输入变量在 ref(p) 中。

构建面向对象程序依赖图(OPDG)以构建类层次图(CHG)开始,相对于 PDG 图,OPDG 图不仅要考虑类内部的控制关系,还要考虑类之间的控制关系。如图 3 所示,对具有继承关系的类之间,标记从子类到父类的继承关系。

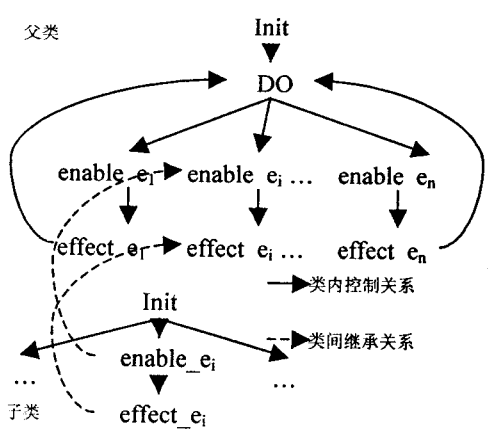

图 3 类层次图 CHG

OPDG 的构建过程分三步:规格的标准化,构建 PDG 图,构建 OPDG 图。前两步请参阅文献[7]中

PDG 图的构建。

图 4 为通用堆栈及子类的 OPDG。

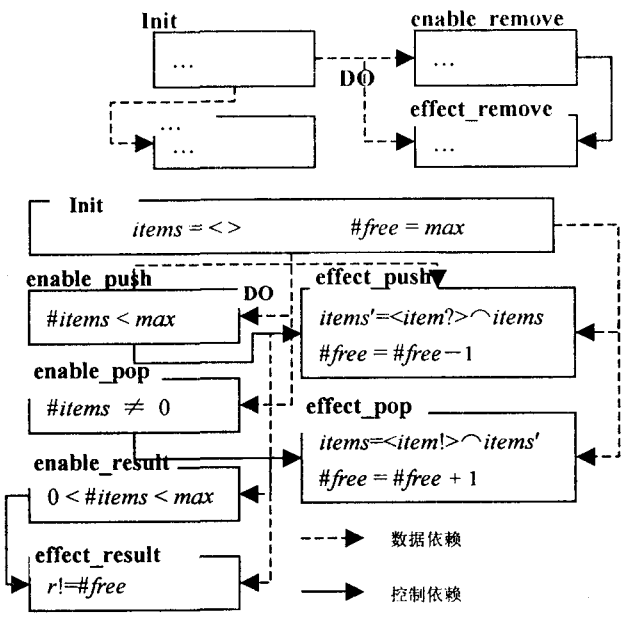

图 4　通用堆栈及子类的 OPDG

2.2　动态后向切片

下面针对具有继承关系的 Object－Z 类的后向切片方法进行分析:

对于具有继承关系的类,这里考虑从超类到子类逐层对每个类进行切片,并把每一层的上一层所有父类的切片加到当前层的子类中作为该子类最终的切片。

作后向切片时,首先找出切片的"开始节点",即表示切片准则的节点。根据公式 φ 可以得出感兴趣的事件 E_φ 集合和变量 V_φ 集合。从中能确定 PDG 中节点 N_φ 直接操纵这些变量或影响这些事件的执行:

$$E_\varphi = E(\varphi)$$

$$V_\varphi = V(\varphi) \cup \{v \mid \exists e \in E_\varphi, \exists p_{en\_e}: v \in vars(p)\}$$

$$N_\varphi = \{p_x \mid \exists v \in V_\varphi: v \in mod(p)\} \cup \{p_y \mid \exists e \in E_\varphi: y = en\_e\}$$

N_φ 中的节点是那些来自随后开始切片的节点。N_φ 中的后向切片的所有节点可能潜在的影响 E_φ 中事件的执行或 V_φ 中的值。

$$bs(N_\varphi) = \{n' \in p \mid \exists n \in N_\varphi: n'(\rightarrow \cup \Rightarrow)^* n\}$$

后向切片包含的节点影响了 φ 的真值,因此事件、谓词和变量仍然要在抽象后的类规格中。

$$N' = bs(N_\varphi)$$

$$V' = \bigcup_{p_x \in N'} vars(p)$$

$$E' = \{e \mid \exists p: p_{en\_e} \in N' \vee p_{eff\_e} \in N\}$$

但 V' 中的某些变量值不会被引用,所以它们不影响公式的成立。因此这里定义另一个变量集合:

$$V'' = V_\varphi \cup \{v \in V' \mid \exists p_x \in N': v \in ref(p)\}$$

V'' 表示那些确实被引用的变量集合。在 $V' \setminus V''$ 外的变量仍然需要在抽象的规格中,因为可能有谓词引用这些变量的主要形式。用 AP'' 表示在 V'' 上的原子命题的集合。

2.3　规格抽象

规格抽象就是利用后向切片算法给出的集合 N', V' 和 E',排除无关的规格,进而抽象出切片来构建抽象规格。这里只对扩展的通用堆栈类例子利用公式切片,省略了子类的切片。对于子类的切片,包括两部分:子类本身的切片和从父类继承部分的切片。

φ_1: = □⌈# free = 100 － # items′⌉

其结果如下:

$$N' = N \setminus \{(r! = \#free)_{effect\_result}\}$$

$$V' = V = V''$$

$$E' = E$$

当通用堆栈类关于下面公式的切片时,

φ_2: = ¬◇(□⌈true⌉∧push)∧¬◇(□⌈true⌉∧pop)

其结果如下:

$$N' = N \setminus \{(items = < >)_{Init}, (\#free = max)_{Init}, (\#free = \#free - 1)_{effect\_push}, (items' = < item? > ⌢ items)_{effect\_push}, (\#free = \#free + 1)_{effect\_pop}, (items = < item! > ⌢ items')_{effect\_pop}, (r! = \#free)_{effect\_result}\}$$

$$V' = V \setminus \{\#free\} = V''$$

$$E' = E$$

由前面的分析,对于扩展的通用堆栈可以得出如图 5 所示的切片规格。

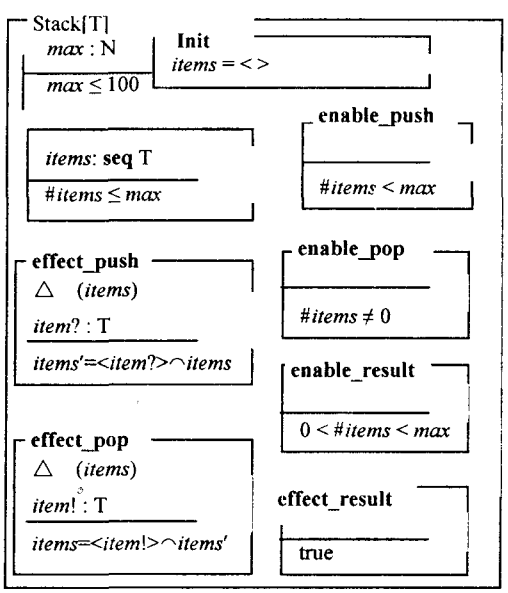

图 5　切片规格

3 Object－Z 切片规格的刻划

这里给出了具有继承关系的 Object－Z 类切片后的刻划：

引理 1[7]：对于 $e \in E'$，p 是模式 enable_ e 外的谓词，那么 $p_{en\_e} \in N'$。

引理 2[7]：对于 $e \notin E'$，所有出现在 effect_ e 中的谓词，有 $\mathrm{mod}(p) \cap V'' = \varnothing$。

定理：C 是一个类或具有继承关系的多个类的规格，φ 为 SE－IL 公式，C' 是 C 关于 φ 的切片。E' 和 AP'' 分别是公平事件和原子命题，K 和 K' 分别对应 Kripke 结构，那么有以下公式成立：

$K' \in Pr_{AP'',E'}(K)$

证明：

1) C 为一个不含子类的类时，仿照文献[7] 中定理 2 的证明；

2) C 为具有继承关系的多个类的规格时，φ 为 SE－IL 公式，C' 是 C 关于 φ 的切片，V' 和 V'' 分别是 C' 上的变量集合和确实被引用的变量集合。这里假定有 m 层继承关系和 n 个具有继承关系的类（$n > 1, 1 < m \leqslant n$），继承关系是第 $i+1$ 层类继承自第 i 层类（$1 \leqslant i < m$）。以下分两种情况：

不存在多重继承时，假定 V'_i 和 V''_i 分别是第 i 个类本身的切片 C'_i 上的变量集合和确实被引用的变量集合。

当存在多重继承时，假定第 $i+1$ 层类继承自第 i 层的 k 个父类（$1 < k < n$）分别记为 $i_1, \cdots, i_k$，并假定 V'_{ij} 和 V''_{ij} 分别是第 ij 个类本身的切片 C'_{ij} 上的变量集合和确实被引用的变量集合（$1 \leqslant j \leqslant k$），于是第 i 层的这 k 个类在 C'_i 上的变量集合和确实被引用的变量集合分别记为：

$$V'_i = \bigcup_{j=1}^{k} V'_{ij} \quad V''_i = \bigcup_{j=1}^{k} V''_{ij}$$

由以上两种情况，所以第 $i+1$ 层类在 C' 上的变量集合和确实被引用的变量集合分别记为：

$$V' = \bigcup_{t=1}^{i} V'_t \quad V'' = \bigcup_{t=1}^{i} V''_t$$

假定 $\pi = s_0 e_1 s_2 e_3 \cdots$ 为 K 的公平事件 E' 的路径，其中 π 由子类本身路径和从父类中继承部分的路径两部分构成。构建序列 $\pi'' = s'_0 e'_1 s'_2 e'_3 \cdots$ 同样由这两部分路径构成，即：

子类本身路径：

s'_i：$s_i \mid V'_s$

e'_i：e_i if $e_i \in E'$，else no operation

从父类中继承部分的路径：

s'_i：$s_i \mid V'_f$

e'_i：e_i if $e_i \in E'$，else no operation

其中，V'_s 和 V'_f 分别表示子类本身切片的变量集合和从父类中继承部分切片的变量集合。由前面分析 V'_s 和 V'_f 都包含在子类切片的变量集合 V' 中。同样设定 V''_s 和 V''_f 分别表示子类本身切片确实被引用的变量集合和继承自父类部分切片确实被引用的变量集合，且 V''_s 和 V''_f 都包含在子类切片确实被引用的变量集合 V'' 中。在 π'' 外，通过消除无操作的 s'_i 形式的所有子序列构建一个序列 π'。

要证明 π' 是 K' 的公平事件 E' 的路径。

公平性：π' 包含来自 E' 的无穷多事件，因为 π 包含这些事件并且因为构建而保存。

路径：因为模式中 K' 比 K 包含更少的谓词，因此也就有更少的变量约束。考虑特殊情况（其它情况可以得到相同的结论），这里假定状态 s_0, s_i 属于子类本身路径，s_{i+2} 和事件 e_{i+1} 属于从父类中继承部分的路径，于是有：

$\mathrm{Init}(s_0) \Rightarrow \mathrm{Init}'(s_0 \mid V'_s) \Rightarrow \mathrm{Init}'(s_0 \mid V')$

$\mathrm{enable\_}\ e(s_i) \Rightarrow \mathrm{enable\_}\ e'(s_i \mid V'_s)$

$\Rightarrow \mathrm{enable\_}\ e'(s_i \mid V')$

$\mathrm{effect\_}\ e(s_i, s_{i+2}) \Rightarrow \mathrm{effect\_}\ e'(s_i \mid V'_s, s_{i+2} \mid V'_f)$

$\Rightarrow \mathrm{effect\_}\ e'(s_i \mid V', s_{i+2} \mid V')$

进一步，由引理 2 可知 $e_{i+1} \in E'$ 暗示 $s_i \mid V''_s = s_{i+2} \mid V''_f \Rightarrow s_i \mid V'' = s_{i+2} \mid V''$。当 $e_{i+1} \in E'$，如果 K 中有迁移 $R(s_i, e_{i+1}, s_{i+2})$，那么当 $0 \leqslant j \leqslant i$，$\forall k: j < k < i: e_k \notin E'$ 且 $e_{j-1} \in E'$ 时，K' 中迁移 $R(s'_j, e_i, s'_{j+1})$ 是可能的。

因为：

(1) $s'_j \mid V''_s = s_j \mid V''_s = s_i \mid V''_s$

$\Rightarrow s'_j \mid V'' = s_j \mid V'' = s_i \mid V''$ 且

$s'_{i+2} \mid V''_f = s_{i+2} \mid V''_f$

$\Rightarrow s'_{i+2} \mid V'' = s_{i+2} \mid V''$

(2) $\mathrm{enable\_}\ e'_i$ 仅引用 V'' 中的变量；

(3) $\mathrm{effect\_}\ e'_i$ 仅引用 V'' 中非主要变量，而且对 V' 上的主要变量不作限制或者与 $\mathrm{effect\_}\ e_i$ 作相同的限制。

因此，π' 为 K' 的路径，π' 是到 AP'' 和 E' 上 π 的映射。

反过来，设 $\pi' = s'_0 e'_1 s'_2 e'_3 \cdots$ 为 K' 的公平事件 E' 的路径，根据上述证明方法并仿照文献[7] 中相关的证明，可以构造 K 中的路径 $\pi = s_0 e_1 s_2 e_3 \cdots$，即 π 为 K 的路径，π' 是到 AP'' 和 E' 上 π 的映射。

3) 综合以上两点，定理证明完毕。

（下转第 21 页）

一个预选创建的线程池处理连接请求。其中一个线程叫“接受线程”,它等待新的连接并接受它们。然后,接受线程将这个连接排队,并通知工作线程(工作线程有多个)有新完成的连接。于是,其中一个工作进程将一个连接从队列中摘下,接着在此连接读取请求并处理,处理完后再接受下一个请求(如这个连接是持续连接的话)或者再从队列中取下新的连接。线程池技术能有效地提高系统节点(特别是服务器)的并发处理能力,减少处理连接的延时。

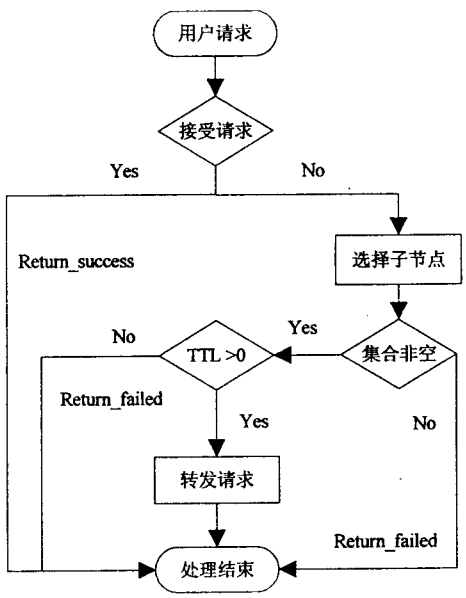

图 6　节点处理流程

4　结束语

针对 P2P 视频点播系统存在的可扩展性以及可靠性等问题,提出了一个视频点播系统的具体设计与实现方案,解决了部分现有 P2PVOD 系统中存在的问题。并对系统在设计与实现中所采用的相关技术与方法作了详细的探讨。随着 IPTV 的提出,基于 P2P 网络模型的流媒体系统将得到更为广泛的应用。

参考文献:

[1] 叶保留,李春洪,姚　键,等.应用层组播研究进展[J].计算机科学,2005,32(6):6-10.

[2] Gao L, Towsley D. Threshold-based multicast for continuous media delivery[J]. IEEE Trans on Multimedia, 2001, 3(4): 405-414.

[3] 刘亚杰,窦文华.P2P 流媒体:一种新型的流媒体服务体系[J].计算机科学,2004,31(4):1-3.

[4] 廖小飞,殷江培,程　斌.基于 P2P 的 VOD 系统中数据缓存策略研究[J].华中科技大学学报,2006,35(8):67-71.

[5] Guo Y, Suh K, Kurose J, et al. P2Cast: P2P patching scheme for VoD service[C]//In: Proc. of the WWW 2003. New York: ACM Press, 2003: 301-309.

[6] Do T, Hua K, Tantaoui M. P2VoD: Providing fault tolerant video-on-demand streaming in peer-to-peer environment [C]//In: Proc. of the IEEE ICC 2004. Paris: IEEE Communications Society, 2004: 1467-1472.

[7] Guo Y, Suh K, Kurose J, et al. A peer-to-peer on-demand streaming service and its performance evaluation [C]//In: Proc of the IEEE ICME 2003. Maryland: IEEE Computer Society, 2003: 649-652.

[8] Hefeeda M, Bhargava B. On-demand Media Stream Over the Internet[C]//Proc of 9th IEEE Workshop on Future Trends of Distributed Computing System (FTDCS'03). San Juan, Puerto Rico: [s.n.], 2003.

[9] 刘亚杰,窦文华.一种 P2P 环境下的 VoD 流媒体服务体系[J].软件学报,2006,17(4):876-884.

[10] 郑常熠,王　新,赵　进,等.P2P 视频点播内容分发策略[J].软件学报,2007, 18(11):2942-2954.

(上接第 16 页)

4　结束语

给出了 Object-Z 类中操作模式拆分为 enable 和 effect 模式的方法和具有继承关系的 Object-Z 类的面向对象程序依赖图,并依此进行切片,最后给出对于具有一个类或具有继承关系的多个类的规格的刻划,通过切片前后 Kripke 结构 K 和 K' 满足 K' 是在公平事件和原子命题之上的 K 的映射来保持验证性质。把具有继承关系的 Object-Z 类整体作为一个模块单独切片,该方法可以扩展到整个 Object-Z 项目,以实现对软件需求规格的验证。

参考文献:

[1] 缪淮扣.软件工程语言-Z[M].上海:上海大学出版社,1999.

[2] Smith G. The Object-Z Specification Language[M]. Hingham, Massachusetts: Kluwer Academic Publisher, 2000.

[3] 吴方君,徐升华.用 Z 形式化描述程序切片[J].小型微型计算机系统,2007,28(8):1444-1447.

[4] 易　彤.前向切片与后向切片之间关系的研究[J].计算机工程与应用,2008,44(12):42-44.

[5] 单卓为,鱼　滨.基于 SPIN 的 CSCW 系统的验证[J].计算机技术与发展,2008,18(4):9-15.

[6] Wu Fangjun, Yi Tong. Slicing Z Specifications[J]. ACM SIGPLAN, 2004, 39(8): 39-48.

[7] Brückner I, Wehrheim H. Slicing Object-Z specifications for verification[J]. LNCS, 2005, 3455: 414-433.

[8] 文志诚,缪淮扣,孙军梅.基于 Object-Z 多态推理[J].计算机科学,2006,33(7):230-233.