UML顺序图形式化语义的研究综述

郭艳燕 张 楠 童向荣
(烟台大学计算机与控制工程学院 烟台 264005)

摘 要 为UML顺序图构建形式化语义,不仅有利于精确描述软件系统的动态交互过程,而且有利于进行基于UML模型的分析和验证,是有效提高软件系统可靠性的重要保障。结合近年来国内外对UML顺序图形式化语义的研究工作,分类阐述了各种方法,综合分析和比较了不同方法的工作机制和优缺点,指出了定义UML顺序图语义时需重点关注的问题。最后,对未来的研究工作与研究思路进行了梳理与展望。

关键词 统一建模语言UML,形式化方法,顺序图,组合交互片段,指称语义,操作语义

中图法分类号 TP311.5 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2017.02.002

Survey on Formal Semantics of UML Sequence Diagram

GUO Yan-yan ZHANG Nan TONG Xiang-rong
(School of Computer and Control Engineering, Yantai University, Yantai 264005, China)

Abstract Formal semantics of UML sequence diagrams is critical to express the dynamic interaction of software system accurately. Therefore, a well-formed sequence diagram is a prerequisite for the analysis and verification of UML model and an important guarantee to improve the reliability of software systems. In this paper, different methods used in UML sequence diagram's semantics were summarized and compared based on the working mechanisms and pros&cons, respectively. Meanwhile, the special issues with respect to how to define the semantics of UML sequence diagram were discussed as well. Finally, some specific research topics and directions in this area were suggested and proposed.

Keywords Unified modeling language, Formal methods, Sequence diagram, Combined interaction fragments, Denotational semantics, Operational semantics

1 引言

随着模型驱动软件开发方法的发展和形式化方法在软件开发中的应用,已经成为软件行业标准的统一建模语言UML(unified modeling language)的精确语义问题引起了业界的关注。现有的UML是一种半形式化语言,诸多元素不具有形式化的语义,因此缺乏精确语义的UML模型不仅会带来理解上的模糊性,而且会增加对模型进行形式化分析和验证的难度。因此,对UML的语义进行形式化定义,对增强该建模语言的准确性、等价性、一致性和扩展性有一定的帮助,并为实现基于UML的模型检验、模型间的一致性检测、模型驱动开发等奠定坚实的理论基础。

研究人员已经在UML的形式化研究方面开展了很多工作并取得了一些成绩,例如在OMG(Object Management Government)官方UML描述文档(superstructure specification)[1-2]中已经包含了大量的形式化研究成果,UML模型的语法和部分模型的语义已经采用了形式化方法来进行描述。研究人员对UML静态模型的语义研究相对成熟,例如类图和对象图,但对UML动态模型的语义研究还存在许多尚未解决的问题,例如顺序图、状态机图和活动图等。其中,UML顺序图的形式化语义一直是研究的热点和重点。

UML顺序图是UML动态交互图中描述能力最强的图,它着重体现对象间消息传递的时序关系,是描述情景语义最有力的工具,并且与其它类型的UML图有着密切的关联[3]。具有形式化语义的UML顺序图无论是在理论层面还是在应用层面都具有积极意义和重要的作用,主要体现在如下方面:1)语义形式化后的UML顺序图,有利于不同开发人员对模型的准确理解和交流。2)语义形式化后的UML顺序图有利于进行语义精化(Refinement),有利于在相应自动工具的支持下进行分析、推理和验证,有利于及时发现和消除模型中存在的错误和不同模型间的不一致性,从而提高模型的可靠性。3)语义形式化后的UML顺序图有利于实现模型到代码的自动转换,为建模工具实现正向工程提供语义支持。4)语义形式化后的UML顺序图有利于实现测试用例的自动生成。5)语义形式化后的UML顺序图有利于语义扩展从而满足不同领域的软件需求。

到稿日期:2016-01-13 返修日期:2016-04-26 本文受国家自然科学基金项目(61403329,61502410,61572418),山东省自然科学基金项目(ZR2015PF010,ZR2013FQ020,ZR2014FL009,ZR2014FQ016),山东省高等学校科技计划项目(J15LN09,J14LN23)资助。

郭艳燕(1980-),女,硕士,讲师,主要研究方向为软件工程、人工智能,E-mail:smallgyy@sina.com;张 楠(1979-),男,博士,讲师,主要研究方向为粗糙集、人工智能;童向荣(1975-),男,博士,教授,主要研究方向为多Agent系统、人工智能。

根据 UML 顺序图的使用者、使用目的和建模领域的不同，可以采用不同的形式化方法对 UML 顺序图的语义进行定义。由于国内至今尚缺少对 UML 顺序图形式化语义定义方法的综述，本文针对 UML 顺序图的官方描述以及存在的问题，结合近年来的国内外研究现状，对多种方法进行了阐述和分类，综合分析和比较了不同方法的工作机制和优缺点，总结出了定义 UML 顺序图语义时需重点关注的问题，并对将来的研究内容和思路进行了展望。

2 OMG 官方描述

OMG 通过事件发生序列来定义 UML 顺序图的交互语义，称为路径语义。交互中不仅可以建模可能路径，还可以建模非法路径，OMG 将路径分为有效路径(valid traces)和无效路径(invalid traces)。OMG 通过偏序关系集来描述顺序图中事件发生的前后关系，满足偏序关系集的路径称为有效路径。UML 顺序图的交互语义采用一对路径集合〈有效路径集，无效路径集〉来表示。有效和无效路径集合的并集不一定是路径全集，没有在交互中描述的路径称为非确定性路径(contingent traces)。OMG 对并发交互采用交错语义(interleaving semantics)的方式进行定义。交错语义表示在同一时间内不可能有两个事件同时发生，与真实并发语义(true concurrency semantics)相对应。OMG 对具有决策性的分支交互采用线性时间语义(linear time semantics)的方式进行定义，与分支时间语义(branching time semantics)相对应。

OMG 将情景描述引入到 UML 顺序图中，并借鉴了其他描述情景的消息顺序图 MSC(message sequence chart)和活性顺序图 LSC(live sequence chart)中的元素和思想，增加了组合交互片段(combinded fragments)来描述更为复杂的动态交互场景[4]。UML 顺序图通过组合操作符来描述不同类型的组合交互片段，以达到描述多种消息控制流的目的，如表 1 所列。通过包含多个组合交互片段的顺序图可以描述多条包含复杂行为的路径聚合，实现多个复杂的交互情景集成到一个单独的顺序图中，如图 1 所示。OMG 通过弱时序 seq 连接产生含有多个组合交互片段顺序图的路径语义。

OMG 官方问题网站 http://www.omg.org/issures 上列举了一些已解决和尚未解决的 UML 语法和语义问题。OMG 虽然完成了对组合交互片段语法的形式化定义，但对组合交互片段语义的定义并不精确和完整。因此，组合交互片段的引入在增强 UML 顺序图表达能力的同时，也增加了理解和分析顺序图的难度，尤其是组合交互片段允许进行嵌套表示更为复杂的交互情况，导致很难从顺序图中抽取其路径语义。另外，OMG 对无效路径产生的说明也存在矛盾，例如“只有用 neg 操作符会产生无效路径”的官方描述并不准确，因为 OMG 在 assert 操作符、状态不变量(stateInvariant)、持续约束(durationConstraint)和时间约束(timeConstraint)的描述中也有对无效路径产生的描述。

UML 顺序图语义研究大都基于 OMG 描述文档，重点关注或强调在描述文档中未定义或定义模糊的部分，重新进行精确的语义定义。OMG 对 UML 顺序图语义的定义存在很多待解决的问题，而现有的研究只解决了这些问题中的子集，并且 UML 的版本在不断更新，因此对 UML 顺序图的语义研究和应用是一个长期的过程。

表 1 UML2.4 顺序图组合交互片段操作符表

操作符类型	类型说明	操作符	操作域个数
与选择和迭代执行相关	选择	opt(单分支选择) alt(多分支选择)	单操作域 多操作域
	迭代	loop(循环)	单操作域
	交互中断控制	break(中断)	单操作域
与顺序和并发执行相关	顺序	seq(弱时序) strict(强时序)	多操作域
	并发	par(并发)	多操作域
	事件执行控制	critical(临界)	单操作域
与一致性关系相关	与无效路径相关	neg(禁止执行) assert(强制执行)	单操作域
	事件控制	consider(考虑事件集) ignore(忽略事件集)	单操作域

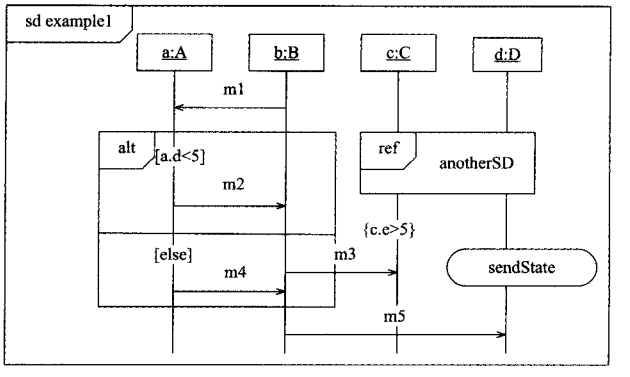

图 1 包含组合交互片段的顺序图实例

3 UML 顺序图语义定义的形式化方法

UML 顺序图主要关注对象交互发生时消息发送的时间的先后次序，因此定义顺序图的语义要从此角度出发。根据语义表达方式的不同，可以将 UML 顺序图的语义分为指称语义(denotational semantics)和操作语义(operational semantics)两大类[5]。

指称语义是将语言成分映射为指称物，指称物均为数学对象，如整数、集合和映射等，指称物的集合称为论域。语言的指称语义就是确定该语言的相关论域，并给出语法成分到论域上的语义映射，映射要求满足以下条件：1)每个成分都对应有指称物；2)复合成分的指称只依赖于它的子成分的指称，用定义在子成分指称物上的运算表达出复合成分的语义。UML 顺序图指称语义的定义也要符合上述条件。

操作语义是将语言中的语法成分用一系列有序的可计算步骤来进行语义表达。开发人员不仅需要准确理解 UML 模型的精确语义，而且需要理解模型实现的精确步骤。UML 顺序图的操作语义是构建该建模语言的解释器，通过输入顺序图的抽象语法，输出顺序图的交互执行路径。

以下从指称语义和操作语义两方面对已有的 UML 顺序图语义定义方法进行简要说明。

3.1 指称语义定义方法

3.1.1 Störrle 方法

交互的路径语义是使用事件序列来描述交互过程的语义方法。UML 顺序图基于路径的语义最早是由 Harald Störrle

提出[6]。该方法中的路径全集为有效路径集、无效路径集和非确定性路径集。Harald Störrle 主要关注有效路径的定义，采用偏序路径和交错语义方式进行定义，并进行了后续的语义精化，将非确定性路径转换为有效路径或无效路径，如图 2 所示。2014 年 Alexander Knapp 和 Harald Störrle 使用了非对称的事件结构(asymmetric event structures)对具有时间约束的 UML 顺序图进行了有效的语义描述[7]，完成了对 UML 顺序图语义的时间扩展。

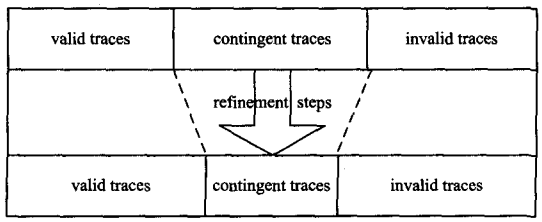

图 2　Störrle 方法中的路径分类和路径精化[6]

在语法定义上，Störrle 仅考虑交互是基本交互或组合交互片段的情况，而未考虑它们二者的组合。UML 标准中将 alt，seq，par 和 strict 交互操作符定义为多元操作符，而 Störrle 却将它们看作二元操作符。在语义定义上，Störrle 方法遵循并接近 UML2 规范，重点关注有效路径集的定义。Störrle 方法虽然对 par，opt，alt，strict，seq，loop，break，critical，neg，assert，ignore 和 consider 交互操作符都进行了交互定义，但在对 seq，par 和 strict 进行语义定义时，采用了一些复杂的函数与操作，影响了语义的易理解性，并且缺少对监护条件(guard)的定义，导致在出现监护条件为 false 时路径未定义的情况。虽然 Störrle 也对交互的无效路径进行了描述，但对 neg 和 assert 操作符语义的描述存在多种语义可能性和不确定性，尤其是 neg 交互片段与其它交互片段的嵌套。

3.1.2　STAIRS 指称语义方法

STAIRS(steps to analyze interactions with refinement semantics)[8]是由 Øystein Haugen 和 Ketil Stølen 提出的基于路径的语义。该方法有利于进行后期的语义精化，有利于实现基于 UML 顺序图的一致性检测和模型检验，有利于实现基于 UML 顺序图的增量式迭代开发。

STAIRS 指称语义方法将路径分为有效路径、无效路径和非确定性路径。顺序图的语义用必须被满足的职责集合来进行定义，每个职责是一对有效路径集和无效路径集，这种新的语义使定义路径集属性成为可能。STAIRS 指称语义方法还区分了可能行为和强制行为，可能行为和强制行为产生有效路径，无效路径由 neg 操作符来构建。为了表示强制选择行为，STAIRS 指称语义方法定义了一个新操作符 xalt，区分于表示可能选择行为的 alt 操作符，这种语义特性有利于实现交互的增量式开发。

语义精化是 STAIRS 指称语义方法关注的焦点。通过 supplementing，narrowing 和 detailing 3 种精化策略实现交互职责的精化，最终达到语义精化的目的。如图 3 所示，STAIRS 指称语义方法通过 narrowing 将有效路径变为无效路径，通过 supplementing 将非确定路径变为有效路径或无效路径。STAIRS 指称语义方法与 Störrle 方法、Cengarle & Knapp 指称语义方法相比的最大优势在于可以为顺序图在增量式软件开发系统中的应用提供语义支持。

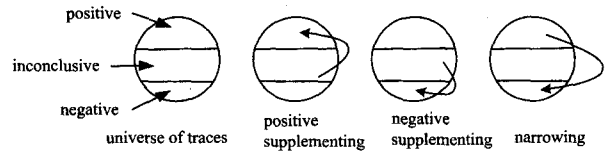

图 3　STAIRS 方法中的路径分类和路径精化[8]

STAIRS 指称语义方法只对 seq，alt，par，neg 交互操作符和新定义的 xalt 操作符进行了语义定义，也是将 alt，xalt，seq 和 par 当作二元操作符来进行处理，并且语义描述过于复杂不利于直观的理解。

建立在 STAIRS 指称语义方法上的 UML 顺序图语义研究工作一直没有停止[9-13]，很多研究在 STAIRS 指称语义方法的基础上不断进行改进。Mass Soldal Lund 和 Ketil Stolen 针对于 STAIRS 指称语义提出对应的操作语义[9]，并进行了顺序图测试产生算法的研究[10]。Timed STAIRS[11] 是对 STAIRS 进行的时间约束扩展。Probabilistic STAIRS[12] 是 Atle Refsdal 对 STAIRS 进行的概率或随机扩展。Ragnhild Kobro Runde 在 STAIRS 指称语义方法的基础上，针对描述不足(underspecification)的顺序图和具有内在不确定性(inherent nondeterminism)的顺序图，提出检验计算机系统的执行是否满足顺序图描述的方法，并对语义上的精化规则进行了传递性和单调性证明[13]。

3.1.3　Cengarle & Knapp 指称语义方法

Cengarle & Knapp 指称语义方法是一种基于路径语义的方法，在路径分类上分为有效路径、无效路径和非确定性路径。在有效路径定义的方式上，Cengarle & Knapp 指称语义方法、STAIRS 指称语义方法和 Störrle 方法都很相似。但 Störrle 方法主要关注 UML 顺序图中有效路径的定义，而 Cengarle & Knapp 指称语义方法则更加关注对无效路径的定义。该方法中的路径语义是基于事件标签偏序多重集 pomset(partially ordered multiset)[14-15]，通过对给出的 pomset 进行线性化，获取可能的路径集。Cengarle & Knapp 指称语义方法通过 pomset 上的并联(concurrence)、串联(concatenation)操作以及标签上的二元对称关系，给出了基本交互和 strict，seq，par，alt，loop，neg，assert，ignore 交互片段的路径语义，但缺少 opt，break，critical 和 consider 交互片段的定义。为了能够进行形式化验证，Cengarle 还给出了交互的实现过程、交互的等价形式以及交互的精化方法。在精化关系上，Cengarle & Knapp 指称语义方法与 STAIRS 指称语义方法的不同在于进行精化后非确定性路径集可能会增加，这是一个有关复合运算符单调性的问题。

虽然 Cengarle & Knapp 指称语义方法完成了对各种组合交互片段无效路径集的定义，但由于这些定义过多地参考有效路径集的定义，因此定义相对比较牵强，与交互真正要表达的含义不太相符。

3.1.4　条件路径方法

Lu Lunjin 提出的条件路径方法[16-18]也是一种基于路径语义的方法，但与 Störrle 方法、STAIRS 指称语义方法和

Cengarle & Knapp 指称语义方法中的路径语义有很大不同。后 3 种方法用所有可能的路径集来描述软件系统可能执行的行为，用无效路径集表示软件系统不允许执行的行为，因为此类路径中的元素只有事件，所以其非常适合描述系统的安全属性，但不适合描述一致性规则。而 Lu Lunjin 提出此方法的初衷是为了设计模式和设计方面中顺序图重用后的精化描述、精化推理和一致性检测。该方法中的路径语义主要关注系统必须执行的行为描述，在路径中不仅有事件元素也有监护条件，此类路径语义可以对顺序图的需求行为进行准确的描述，有利于描述顺序图的一致性规则。Lu Lunjin 还给出了语义精化原则并进行了推理来验证精化结果的正确性，并在此基础上给出了设计模式中的顺序图与实际应用中的顺序图之间的行为一致性检测方法。

Lu Lunjin 还以该条件路径语义为基础完成了一系列的相关研究：语义精化、语义精化验证、顺序图间的一致性检测，并取得了比较好的效果。此方法对将 UML 顺序图应用到设计模式中进行一致性检测起到了很大的推动作用。

Lu Lunjin 只关注了系统中必须执行的交互描述，而缺乏对系统禁止行为的交互描述。该方法对 UML 顺序图中的大部分操作符都进行了语义定义，例如 alt，opt，par，loop，seq，strict 和 critical，但还缺少一致性关系操作符的语义定义。

3.1.5 集合论方法

中山大学李师贤教授带领的研究组利用集合论以组合定义的方式进行了 UML 顺序图的语义定义[19]。该方法将交互的语义定义为事件发生集和事件发生的偏序关系集，并且证明了形式化后的语义为拟序集，此拟序集的线性化恰好就是 OMG 规范所定义的路径集。因此这种基于集合论的方法与 UML 顺序图的基于事件发生轨迹的语义规范更为贴近，比其它适用于某种特殊应用背景的形式化技术更具有通用性，可以作为 UML2 交互规范的很好补充。

该方法首先定义基本元素的语义，然后定义复合结构的语义，复合结构的交互语义由其直接子成分的语义定义，符合指称语义的定义规则。该方法实现了对 alt，opt，par，seq，strict，assert，loop，ignore 和 consider 交互片段的语义定义，但缺少对 break 和 critical 交互片段的语义定义。由于 break 和 critical 交互片段的语义涉及到与其外层环境的交互，因此不能由其直接子成分的语义来进行定义。此外，由于描述不合法情景的 neg 交互片段的语义在该方法中并未定义，因此该方法仅能对交互的有效路径集进行描述，不能对无效路径集进行描述。

3.1.6 LTL 框架方法

线性时态逻辑 LTL(linear temporal logic)框架方法[20]是 Shen Hui 提出的一种描述 UML 顺序图路径语义的方法。

LTL 框架方法可以为 UML 顺序图和各种组合交互片段分别进行语义定义，并且各个部分的语义定义相互正交、相互独立、互不包含。采用该方法可以降低将带有组合交互片段的顺序图作为整体进行形式化的复杂性。在该方法中，12 种组合交互片段的通用语义部分表示成 LTL 模版，每种组合交互片段的特性通过对模版附加相应的约束等信息形成完整的语义。嵌套的组合交互片段也可以通过 LTL 定义的连接来进行描述。UML 预留了许多语义变化点让用户进行重新定义，LTL 框架方法提出了一种可以进行可定制语义的重用框架。在逻辑架构语义中，不同方面的语义用独立简单的 LTL 公式进行表达，整个顺序图的语义则是不同方面的语义组合。

LTL 框架方法的最大益处是可以通过自动分析工具进行顺序图语义基础上的分析、推理和验证工作，例如顺序图的一致性检测、安全性检验和无死锁检测等。使用 LTL 模版可以实现将表示强制行为的 assert 交互片段转换为用 LTL 描述的一致性属性，将表示禁止行为的 neg 交互片段转换为用 LTL 描述的系统安全属性。模型检验工具可以检验顺序图的可能路径是否满足系统的需求属性。该方法的出现可以取代用户直接去描述系统的 LTL 属性。但由于 LTL 表达不直观，当顺序图达到一定规模时，转换的表达式难以阅读和理解。

Störrle 方法、STAIRS 指称语义方法、Cengarle & Knapp 指称语义方法、条件路径方法、集合论方法和 LTL 框架方法都是定义 UML 顺序图的路径语义。

3.1.7 NuSMV 方法

NuSMV 方法[21]是由 LTL 框架方法的提出者 Shen Hui 提出的另外一种描述 UML 顺序图语义的方法，是使用 NuSMV 模型检验工具的输入语言并按照指称语义定义方式来对 UML 顺序图进行语义定义的方法。

NuSMV 方法中，首先根据最大独立集(maximal independent set)将 UML 顺序图进行片段分割，然后映射交互片段到它所覆盖的每条生命线来获取一个组合执行单元 CEU (compositional execution unit)，映射交互片段内的每一个交互操作域到它所覆盖的每条生命线来获取一个基本执行单元 EU(execution unit)；最后，整个顺序图被描述为 NuSMV 模型的主模块，每一条生命线映射成独立的 NuSMV 模块在主模块中被声明和实例化，同时将得到的 CEU 和 EU 表示成 NuSMV 模块，最终实现 UML 顺序图到 NuSMV 模型的等价转换。该方法规定在进出交互片段时所覆盖的所有生命线要进行同步化(synchronization)，规定相邻的事件或片段使用强时序 strict 进行连接，嵌套在相同交互片段内的事件或交互片段使用弱时序进行连接。

NuSMV 方法实现了 UML2 版本中所有组合操作符的语义定义，并对部分嵌套交互片段、同步和异步消息以及交互约束的语义进行了定义。NuSMV 方法不仅实现了 UML 顺序图的形式化语义定义，而且实现了基于 UML 顺序图的模型检验。

Shen Hui 提出的 NuSMV 方法和 LTL 框架方法常结合使用[22]。对于用多个顺序图来展示的软件系统，可以将表示可能行为的顺序图转换为 NuSVM 模型，将表示禁止行为和强制行为的顺序图转换为 LTL 公式，然后通过模型检验工具来完成一致性和安全性的验证。

3.1.8 共代数方法

Sun Meng 和 Barbosa L. S 为了能够对形式化后的 UML 顺序图进行重构、精化和推理，采用共代数(coalgebra)的方法对 UML 顺序图进行语义定义[23-24]，其属于指称语义方法。共代数作为研究面向对象技术理论基础的数学工具，不仅可

以从范畴论及观察的角度研究面向对象的形式语义及行为关系，而且可以对其性质进行描述与验证。UML 顺序图作为描述对象交互行为的模型，用共代数来定义其语义是适用的。

Sun Meng 和 Barbosa L. S 首先对基本交互，alt，opt，par，seq，strict 和 loop 交互片段进行语义定义，然后在共代数的语义基础上进行顺序图的模型转换。在共代数理论下，通过互模拟(bisimulation)可以保持顺序图的转换结构，通过对转换后的模型进行行为检验，达到检验与其等价的顺序图行为的目的。在共代数基础上，使用态射(morphisms)很容易实现模型间的互模拟和精化。

Sun Meng 使用共代数完成了对多种类型 UML 图的形式化定义，包括类图、对象图、用例图、状态机图和顺序图，形成了可以应用到不同模型之上的统一语义架构，并在其基础上对 UML 模型图进行重构来实现精化。

共代数语义方法有利于实现 UML 模型的行为推理和 UML 转换模型行为保持的正确性检测。Sun Meng 和 Barbosa L. S 研究中的不足在于缺少一致性关系操作符的语义定义。

3.1.9　标签事件结构方法

Küster-Filipe 提出基于标签事件结构(labelled event structures)的真实并发语义[25-26]，给出从 UML 顺序图到标签事件结构的转换过程。事件结构能够用来描述分布式计算，体现事件发生之间的联系。根据事件发生之间的联系，可以将事件关系用因果关系(causality)、非确定性的冲突关系(conflict)和并发关系(concurrent)来进行表达。因果关系描述事件间的偏序关系，冲突关系描述哪些事件不能同时发生，因果关系和冲突关系之外的关系是并发关系。

Küster-Filipe 首先提取 UML 顺序图中的交互事件来构建出对应的事件结构模型，然后将 UML 顺序图上的非形式化信息通过标签形式附加到转换模型上以构建出标签事件结构。通过标签事件结构使用一个真正并发的两层逻辑公式集合来描述顺序图语义，上层逻辑为通信逻辑(communication lo-gic)，用来实现对象间的描述，例如信息发送或接收、事件的并发执行等；下层逻辑为内逻辑(home logic)，用来实现对象内的描述，例如状态不变量、交互约束等。使用这种并发的分布式的时态逻辑，既可以展示交互又可以描述各种约束。通过捕获交互属性(例如禁止行为、活性属性、状态和不变量)来验证对象间的行为模型是否满足这些属性。将顺序图的整个交互转换成公式集合后，可以通过模型检验工具直接验证顺序图是否符合基于状态的行为模型。为了实现交互片段的嵌套定义，Küster-Filipe 给出每一个事件的前任事件来表达嵌套层次关系。

该方法对 UML 顺序图组合交互片段的定义并不完整，只完成了对 seq，alt，par，neg 和 assert 操作符的语义定义。

3.1.10　Hammal 方法

Youcef Hammal 采用分支时间语义[27]，结合事件偏序关系，按照指称语义的定义规则对 UML 顺序图进行语义描述。Youcef Hammal 首先提出一种类似于格状图(lattice-like graph)的数学模型以及可应用在该模型上的代数操作(包括选择操作、参数化的笛卡儿积和星操作)，并在此基础上采用组合定义的方式对 UML 顺序图的交互片段语义进行描述。在进行数学模型定义时，需要给出有效路径上事件的偏序关系。通过该模型，不仅可以记录交互的路径还能记录分支情况，并且能够保存事件的偏序关系。由于可以直接将该模型转换成捕获预期行为的转换系统，因此该语义比基于路径的语义能够更好地描述预期的行为。

Youcef Hammal 还在该模型的基础上提出了抽取交互中的时间属性来形成时间约束从而附加到转换模型上的方法。通过逻辑时钟和时间公式来扩展建立的形式体系，通过时钟的值来表达复杂系统的定时约束，实现了 UML 顺序图的时间语义扩展，并且有利于实现将模型转换为时间自动机，利用时间自动机的模型检测来完成对模型的时效性和性能的分析。

Hammal 方法还有利于进行 UML 顺序图与其它类型 UML 图的一致性检验，尤其是与 UML 状态机图在语义和时间上的一致性检测[28]。但不足之处在于 Youcef Hammal 仅对 UML 顺序图中的 alt，break，opt，par，strick，seq，loop 操作符进行了定义，缺少对其它操作符的定义。

3.2　操作语义定义方法

3.2.1　ASM 方法

Cavarra 和 Küster-Filipe 借鉴活性顺序图 LSC 中可以体现可能行为(possible behavior)、强制行为(mandatory behavior)和禁止行为(forbidden behavior)的活性属性描述思想，来扩展 UML2.0 顺序图的活性属性，按照操作语义定义方式来对 UML 顺序图进行描述。

LSC 中的位置元素(location)代表事件的发生，LSC 中的温度元素(temperature)用 cold 和 hot 来区分可能执行的元素和必须执行的元素。元素包括位置、消息和子图(交互片段)。如果位置是 cold/hot，则表示该位置可能/必须到达；如果消息是 cold/hot，则表示该消息在发送后可能/必须被接收；如果子图是 cold/hot，则表示该子图所体现的交互可能/必须被执行。

Cavarra 和 Küster-Filipe 在文献[29]中使用对象约束语言 OCL(object constraint language)来扩展 UML 顺序图交互中的活性属性，在文献[30]中采用抽象状态机 ASM(abstract state machines)来丰富 UML 顺序图的活性语义。将 UML 顺序图转换为对应的 ASM 模型来定义顺序图的交互语义，通过两个阶段来完成。第一阶段，对顺序图添加位置元素，并假设所有位置的 temperature 值都是 cold。采用 ASM 规则来定义交互实例的转换过程，此阶段得到顺序图的基本 ASM 模型。第二阶段，对基本的 ASM 模型进行精化，增加 temperature 值为 hot 的元素，从而完成对 UML 顺序图的活性语义扩展。

ASM 方法只完成了对 alt，par，seq，strict，neg 和 assert 操作符的操作语义定义。使用 ASM 建立简单顺序图模型时，其逻辑语义清晰，但对于包含复杂交互的 UML 顺序图，由于缺乏精确的定义，将会给验证工作带来难度。

3.2.2　MSD 方法

Harel 和 Maoz 借鉴活性顺序图 LSC 中的多模态思想[31]，使用模态顺序图 MSD(Modal Sequence Diagrams)[32]对 UML2.0 顺序图进行模态语义扩展来表达可能场景和强

制场景。模态和模型的活性、安全性相关。与 OMG 标准中 assert 和 neg 作为交互片段操作符不同的是，Harel 和 Maoz 将 assert 和 neg 作为模态而不是操作符来看待。在语法定义上，首先定义一个如图 4 所示的只包含模态属性 interactionMode 的构造型《modal》，interactionMode 的值只能设置为 hot(必须执行)或 cold(可能执行)，然后增加一个构造型是《modal》的新交互片段——模态交互片段(modal interaction fragment)，如图 5 所示。interactionMode 的值为 cold 时表示可能执行或有效的交互片段，此时不考虑 assert 和 neg 交互片段；interactionMode 的值为 hot 时表示必须执行的交互片段，此时需要考虑 assert 和 neg 交互片段。最后，使用交替的弱字自动机 AWW(alternating weak word automation)作为必须执行的 MSD 的路径语义。AWW 根据 UML 顺序图中的事件偏序关系来产生状态以及状态间的转换。如果所有的系统运行都能被自动机所接受，则表示这个系统模型满足必须执行的 MSD。如果至少有一个系统运行能被自动机所接受，则表示这个系统模型满足可能执行的 MSD。

《enumeration》 InteractionMode	《stereotype》 modal
hot cold	interactionMode:InteractionMode

图 4 模态构造型

《modal》 InteractionFragment
interactionMode:InteractionMode=cold

图 5 模态交互片段元模型

MSD 与 UML 顺序图的标准语义在没有 assert 和 neg 交互片段的条件下，对可能片段(cold fragment)的路径语义定义是一致的。但引入 hot 交互片段后，MSD 与 UML 顺序图的标准语义会在路径定义上有所不同。通过 MSD 增强了 assert 和 neg 交互片段的语义，便于实现模型的活性和安全性描述。使用 MSD 对 UML2.0 顺序图进行描述不仅有利于 UML 顺序图的精确语义定义，还有利于将 LSC 上的模型形式化验证、合成和基于情景的执行技术应用到 UML 顺序图上，扩展 UML 顺序图的应用范围。

MSD 方法除了重点关注 assert 和 neg 操作符的语义表达外，还进行了 break，loop，alt，consider 和 ingore 操作符的语义定义。但 MSD 方法只关注了同步消息和强时序，并且在语法使用上进行了相应的限制，例如不允许 hot 交互片段的嵌套。

3.2.3 uMSD 方法

河海大学王志坚教授带领的研究组借鉴活性顺序图 LSC 中的模态语义来扩展 UML2.0 顺序图，形成带有强制场景的消息顺序图 uMSD(universal modal sequence diagram)，用弱交换 Büchi 自动机 WABA(weak alternating büchi automaton)来解释 uMSD 的基本语义规则，根据基本规则给出 seq，par，loop，alt，negate，critical，consider 和 ignore 操作符的执行算法[33]。并在 uMSD 语法和语义的基础上，提出一种面向 Web 服务组合的模型检验方法[34]，实现顺序图形式化语义的可靠性应用。

uMSD 方法之所以采用 WABA 而未采用 BA(büchi automaton)，首先是因为 WABA 比 BA 更简洁，且具有可能(existential)和强制(universal)结构，适用于软件规约；其次，WABA 的状态空间被划分为偏序集合，可以从某个状态集合迁移到更小的状态集合，因此将 uMSD 转换为 WABA 比转换为 BA 更容易实现；此外，WABA 能够在可能结构和强制结构间进行转换，利于 WABA 补的求解和验证算法的设计[33]。

uMSD 方法中用 neg 操作符来描述不期望发生的场景，隐含了禁止的事件发生后不会再发生其它事件。但在该方法中对 neg 操作符的使用进行了限制，不允许 neg 操作符自嵌套，也不允许其它操作符嵌套 neg 操作符，通过语法上的使用限制来减少出现模糊语义的机会。

uMSD 方法中将两个 WABA 的连接分为顺序串联组合(merge)和并联组合(compose)。在 alt 交互片段中，由于监护条件的存在会产生操作域的选择，因此在进行 compose 操作时，必须考虑条件，而在 uMSD 方法中却没有考虑，导致将操作符 par 和 alt 视为完全一样的交互。因此，可以对该方法进行改进，将并联组合分为无条件并联(conditionless compose)和有条件并联(condition compose)，以对不同类型的并联产生的状态转移加以区分。

ASM 方法、MSD 方法和 uMSD 方法都将活性顺序图 LSC 中的模态思想引入到 UML 顺序图中，通过可能场景和强制场景的表达来扩展 UML 顺序图的活性和安全性语义。

3.2.4 西安电子科技大学系列方法

西安电子科技大学段振华教授带领的研究组于 2010—2012 年对 UML 顺序图进行了一系列的形式化研究[35-38]，包括 UML 顺序图的形式化语义描述、基于 UML 顺序图的模型验证以及基于 UML 顺序图的测试用例产生等。

为了准确表达顺序图中各对象在消息交互过程中的状态迁移以及顺序图组合交互片段对迁移关系产生的影响，文献[35-36]采用事件确定有限自动机 ETDFA(event deterministic finite automata)对能够体现复杂交互的 UML 顺序图进行形式化描述，具体给出了包含 loop，opt，alt，break 和 par 操作符的顺序图到 ETDFA 的构造算法，采用命题投影时序逻辑规约系统来进行模型验证，使用验证正确且与顺序图等价的自动机模型，结合用例描述来生成测试用例，并保证该用例满足事件与全路径覆盖准则[37]；但不足之处在于缺少对 neg，critical，assert，ignore，strick，seq 和 ref 操作符的形式化定义。

文献[38]从模型的安全性角度出发，对 UML 顺序图进行语法和语义定义，使用事件迁移图 ETD(event transition diagram)进行顺序图语义描述。首先给出顺序图到 ETD 的转换算法，建立顺序图与测试用例的中间表达形式，再给出 ETD 到测试用例的生成算法。ETD 可以表达在顺序图中全序关系与偏序关系共同作用下事件的发生顺序以及执行路径的分支结构。由于该方法仅关注模型的安全性属性，因此仅对会使执行路径产生分支的 loop，opt，alt，break，neg 和 ref 操作符进行了定义，缺少其他交互操作符的定义。

3.2.5 具体语法代数图转换方法

Roy Grønmo 采用代数图转换(algebraic graph transfor-

mation)[39]的方法来研究顺序图的语义定义问题。与多数研究者采用基于 UML 顺序图抽象语法的研究方法不同，Roy Grønmo 提出基于 UML 顺序图具体语法的研究方法，给出从 UML 顺序图到状态机图的转换规则，并提出新的操作符来完成 UML 顺序图中组合交互片段的匹配和转换。该方法给出的转换规则能够映射成传统的图转换规则，然后通过代数图转换工具 AGG(attributed graph grammar)来完成具体的 UML 顺序图到状态机的转换，从而实现通过状态机的形式化语义来定义 UML 顺序图的语义。

由于 UML 顺序图是对系统交互的部分描述，因此在顺序图中可能存在非确定性路径，而状态机体现的是完整交互，即在状态机中不存在非确定性路径，因此 Roy Grønmo 在进行 UML 顺序图到状态机的转换时，是将顺序图中的非确定性路径精化成无效路径，而这种精化方法会使转换后的状态机所表达的语义相比转换前的顺序图所表达的语义有所缺失。另外，Roy Grønmo 只实现了基本交互，alt，loop，par，opt 和 neg 交互片段的模型转换，未实现其它交互片段的模型转换，并且 Roy Grønmo 只关注顺序图中异步消息的处理，缺少对同步消息的处理。虽然 Roy Grønmo 提出的方法存在一些问题，但 Roy Grønmo 在 UML 顺序图上的形式化研究还在继续[40-41]。

3.2.6　抽象语法代数图转换方法

Nabil Messaoudi 通过将 UML 顺序图转换为等价 Büchi 自动机来构建顺序图的操作语义[42]。构造自动机是为了建立系统建模与验证之间的桥梁。该方法首先给出 UML 顺序图的抽象语法，将每一个基本交互转换为一个 Büchi 自动机；然后将得到的 Büchi 自动机作为操作域，应用对应的交互操作符(seq，strict，alt，neg 和 loop)语义来产生一个完整的 Büchi 自动机。该方法也是基于代数图转换和代数图转换工具 AGG。为了实现转换，首先给出 UML 顺序图和 Büchi 自动机的元模型，然后给出图语法来实现将 UML 顺序图转换为 Büchi 自动机元模型的一个实例。该方法得到的结果可以通过模型检验工具来验证某些系统属性。

Nabil Messaoudi 和 Roy Grønmo 都采用代数图转换和 AGG 工具来完成 UML 顺序图的语义定义。但一个重要的不同之处在于，Nabil Messaoudi 给出的是基于 UML 顺序图抽象语法的转换方法，而 Roy Grønmo 给出的是基于 UML 顺序图具体语法的转换方法。并且为了能够精确定义反应式系统的交互，Nabil Messaoudi 提出的方法采用可以接受无穷域的 Büchi 自动机。

3.2.7　模版语义方法

UML 顺序图的模版语义(template semantics)方法[43-44]是 Niu Jianwei 和 Shen Hui 提出的一种符合操作语义定义规则的方法。模版语义是用来构造表达符号操作语义的形式体系。在模版语义中，最基本的可计算模块是一个层次转换系统 HTS(hierarchical transition system)，即一种被扩展的状态机。多个 HTS 通过组合操作符形成更加复杂的行为描述，称为组合的层次转换系统 CHTS(composed hierarchical transition system)。一个 HTS 的模版语义被定义成有序的操作步骤，组合操作符用来表达多个 HTS 执行的逻辑约束。组合操作符的模版语义描绘了多个 HTS 如何通过交换事件、数据和转换控制实现并发、通信和同步。HTS 的模版语义中已对 7 个组合操作符进行了形式化定义，它们分别是 interleaving，parallel，sequence，choice，interrupt，synchronization 和 rendezvous。

Niu Jianwei 和 Shen Hui 为了描述 UML 顺序图的语义，将 UML 顺序图进行语法分解后映射成 HTS。首先将一个含有多条生命线的 UML 顺序图映射为多个 CHTS 的组合，每条生命线对应一个 CHTS。然后将每个 CHTS 分割成最大块(maximal block)集合，每一个最大块对应一个 HTS。这些 HTS 通过 UML 组合交互操作符合并成 CHTS，而 UML 组合交互操作符映射成 HTS 模版语义中的组合操作符。为了能够精确描述 UML 顺序图组合交互操作符的控制约束，对 HTS 模版语义中的组合操作符进行扩展。

模版语义不仅有利于描述交互的顺序和并发特性，而且有利于模型检测工具的研发。但该方法仅实现了对 alt，opt，break，par，critical 和 seq 操作符的语义定义，还缺少一致性关系操作符的语义定义。

3.2.8　STAIRS 操作语义方法

Mass Soldal Lund 针对 UML 顺序图的 STAIRS 指称语义进行了操作语义的定义。STAIRS 操作语义建立在执行系统(execution system)和投影系统(projection system)两种转换系统的结合上。在执行的每一个阶段，这两种系统联合完成工作，执行系统通过传递通信媒介的现行状态来更新投影系统，投影系统通过选择要执行的事件以及返回执行事件后的状态来更新执行系统。Lund 还证明了该操作语义在关于 STAIRS 指称语义上是可靠且完整的，并且用该操作语义表达的交互执行可终止。

使用该方法是为了尽可能地接近 UML 标准的语法和语义，并尽量减少应用该语义到不同顺序图上的扩展或变化。Lund 对 seq，alt，par，neg 和 loop 组合片段进行了操作语义定义，并引入了 xalt 操作符来解决 alt 语义上的模糊性，alt 表达可能的选择，而 xalt 表达强制的选择。但不足之处在于对 seq，par，alt 和 xalt 的处理上，其被当作二元操作符看待，与 OMG 标准中它是多元运算符的描述不一致。

Lund 对 UML 顺序图形式化的研究并没有仅仅停留在操作语义本身上。Lund[45]提出了一种从 UML 顺序图抽取测试顺序图的方法，并允许被抽取的顺序图中包含 neg 和 assert 操作符。Lund 还开发了工具 Escaltor[46]来实现对 UML2.0 顺序图的精化验证和精化测试。

3.2.9　安全-活性语义方法

Grosu 和 Smolka 通过结合自动机理论(automata theoretic)来解决反应式系统(reactive system)设计中 UML 顺序图的语义问题，提出使用安全-活性语义(safety liveness semantics)[47]来对顺序图语义进行定义。该方法使用模型检测中的活性和安全性去解释交互中的有效和无效部分。首先将顺序图根据事件偏序关系转换成高层非确定性自动机(high-level nondeterministic finite automata)。由于活性和安全性属性的正交性，可以将顺序图中的有效部分和无效部分加以分离，从高层非确定性自动机中抽取得到有效的高层非确定性

自动机和无效的高层非确定性自动机，然后完成对应的转换以构建出描述有效路径的活性 büchi 自动机(liveness büchi automata)和描述无效路径的安全 büchi 自动机(safety büchi automata)；最后基于活性和安全自动机能够接受的语言，对顺序图的有效和无效路径进行定义。

安全-活性语义方法还使用强时序思想来对 seq，alt，par，loop 和 neg 组合操作进行语义精化。

3.2.10 OHST 方法

Li Xiaoshan 等人为了实现 UML 顺序图与类图、对象图和状态图的一致性检测，在提出的有序分层结构树 OHST (ordered hierarchical structure tree)[48-50]抽象语法的基础上，对 UML 顺序图从静态和动态两个方面进行了语义定义。

有序层次结构树展示了交互中对象消息传递间的层次关系，而有序性体现为消息的执行顺序必须遵循有序层次树的遍历规则。带有组合交互片段的 UML 顺序图通过使用类似编程语言中的控制结构来连接基本交互。一个体现复杂交互的 UML 顺序图对应的有序分层结构树如图 6 所示。

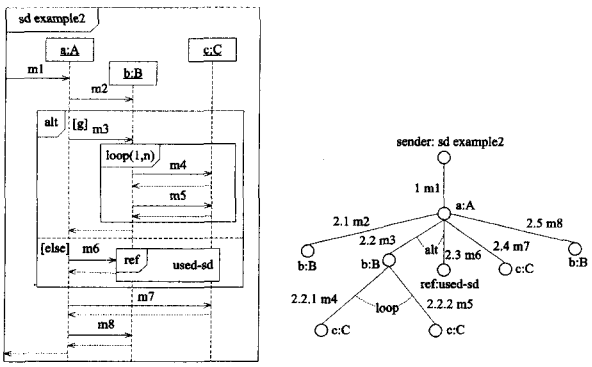

图 6 带有组合交互的顺序图实例及对应的有序分层结构树[65]

顺序图的静态语义被定义为有序分层结构树中所有消息的集合，用来检验是否和类图一致。顺序图的动态语义被定义为所有可能的事件的发生路径，路径上的事件的发生顺序必须符合有序分层结构树的遍历规则。通过顺序图的语义形式可以从中获取对象的状态转换，从而实现与 UML 对象图、状态图的一致性检测。

Li Xiaoshan 提出的语义方法不仅有利于 UML 建模工具中模型一致性检验的实现，还可以应用到设计模型与需求模型之间的正确性推理。该语义的优势还在于能够帮助系统分析者和设计者在理解面向对象方法论和面向对象编程的基础上进行交互信息的抽取，易于理解，并且有利于将 UML 模型和面向对象编程的语言建立映射关系。由于 Li Xiaoshan 只关注了顺序图中的同步消息，因此仅实现了 alt，opt 和 loop 操作符的语义定义，没有实现能够体现并发执行相关的 par，seq 和 strict 操作符以及一致性关系操作符的语义定义。

3.2.11 进程代数方法

进程代数(process algebra)[51-52]是一个描述进程行为的代数理论，基于标签迁移系统 LTS(labelled transition system)，用一组算子作为基本成分，通过算子间的组合刻画复杂系统。顺序图是由事件以及组合算子按照生成规则组合而成，因此 UML 顺序图的许多特征都与进程代数中的知识相关，如状态算子和组合算子，所以进程代数在顺序图形式化语义定义方面是一个有用的工具。

进程代数方法将顺序图中的事件动作及执行序列映射为进程代数中的进程表达式。首先给出基本规则，然后给出组合算子的演绎规则，最后利用进程代数语义框架以操作语义方式构建顺序图的形式化语义。由于进程代数中的组合算子只能反映进程间的弱顺序组合关系，不能直接用来定义顺序图之间的组合关系，因此需要在已有进程代数的语义框架下，给出新的组合算子来满足 UML 顺序图的实际语义需求，例如引入次序约束条件来定义顺序图进程代数的新组合算子[53]。

除了建立顺序图和进程代数之间的直接映射外，还可以通过 LTS 中间结构来建立间接映射[54]。首先将 UML 顺序图抽象为一个 LTS，然后赋予 LTS 操作语义及转换规则，而非直接将其转换为某种进程代数规范。由于进程代数是基于 LTS 结构的，因此采用这种方式的优势在于如果给定 LTS 操作语义，就可以根据不同的语法将其转换为任意一种进程代数形式规范(例如通信顺序进程 CSP 和 pi 演算等)，而非某一种进程代数形式规范。

采用进程代数易于实现对描述简单交互的操作符进行语义定义，例如 alt，loop，par，opt 和 break 操作符，但难以实现对描述复杂交互的操作符的语义定义。

3.2.12 Promela 语言描述方法

Promela(protocol meta language)是模型检测工具 SPIN 的模型输入语言，用于对有限状态系统进行形式化建模，具有类似 C 语言的语法，支持选择、循环和跳转等常见控制结构，并允许动态创建并行的进程，进程之间异步交错执行。

Promela 描述方法的提出是为了能够方便地实现基于 UML 顺序图的模型检验与一致性检测[55-56]。Promela 能够为 UML2.0 顺序图中的大部分组合交互片段提供精确语义，通过 SPIN 模型检验器，结合软件属性的线性时态逻辑 LTL 描述，可以验证构建的模型是否符合需求行为或验证模型间的一致性。

从 UML 模型到 Promela 模型的转换最重要的是要保证转换的一致性，即 UML 顺序图的路径语义与其对应的 Promela 代码执行路径要相同，这样才能实现通过 Promela 模型来模拟 UML 顺序图的目的。转换过程通过图转换(grahph transformation)来实现，根据图语法(graph grammar)给出的构图形式化规则来进行等价图的重构。转换前需要给出转换规则，即顺序图与 Promela 之间的元素映射关系。现有的多种研究方法使用已有的自动化工具来实现从 UML 顺序图到 Promela 代码的转换，例如 Eclipse 插件工具、AToM3 工具[57]。除了对 UML 顺序图与 Promela 代码建立直接映射关系外，还可以通过中间第三方结构来建立转换桥梁，例如通过顺序图的 XMI 文档解析来实现转换[58]。

由于 Promela 语言中控制结构的局限性，不利于实现除 seq，opt，alt，par，loop 和 break 以外的交互片段的语义定义。此外，生成的与 UML 顺序图等价的 Promela 程序代码也不利于后期测试用例的自动生成。

3.2.13 Cengarle & Knapp 操作语义方法

Cengarle 和 Knapp 针对顺序图的指称语义给出了与其

对应的 UML 顺序图的操作语义[59]，并且证明了该操作语义的正确性和完整性。

Cengarle 和 Knapp 提出的指称语义和操作语义都是以偏序标签多重集 pomset 为基础。顺序图交互的操作语义通过两个交互以及一个事件间的三元关系来描述，分别给出了了空交互，基本交互，alt，loop，seq，strict，par，ignore，assert 和 neg 组合交互片段的操作语义。

3.2.14　Petri 网方法

Petri 网是一种基于状态的形式化建模方法[60]，具有直观的图形化表示及相对成熟的语义和可执行性，同时具有丰富的结构描述能力，包括顺序、并发、冲突和混惑结构，并且提供了抽象和精化的机制。此外，已经有大量工具可用来设计、模拟和分析基于 Petri 网的系统。

结合 Petri 网来描述 UML 顺序图的语义是一种重要的方法。为了满足不同领域的需求，针对不同的建模系统，可将 UML 顺序图转换为 Petri 网及其扩展形式。文献[61]将 UML 顺序图转换为基本 Petri 网。文献[62]提出了将 UML 顺序图的组合交互片段转换为着色 Petri 网的方法。文献[63]使用标签广义随机 Petri 网(Labeled Generalized Stochastic Petri Net，LGSPN)来进行顺序图语义定义。针对实时、嵌入式系统，文献[64]提出一种将 UML 顺序图转换为定时 Petri 网和着色 Petri 网相结合的扩展 Petri 网的方法。为了进行 UML 顺序图的一致性检测，João Pascoal Faria[65] 提出将 UML 顺序图转换为事件驱动 Petri 网(event-driven Petri net)和着色 Petri 网相结合的扩展 Petri 网的方法。为了进行 UML 顺序图和状态机图的一致性检测，文献[66]提出将顺序图和状态机图转换为标签 Petri 网(labeled Petri net)的方法。除了进行 UML 顺序图到 Perti 网在具体层上的转换，文献[67]还提出一种建立在 UML 顺序图元模型和 Petri 网元模型之间的转换方法，该方法有利于进行系统模型的验证和需求确认。

Petri 网方法的优点在于可以描述系统交互的真正并发语义，并且在 UML 顺序图转换为 Perti 网及其扩展形式后，可以基于现成的一些算法和工具利用 Petri 的性质进行模型检验和一致性检测。Petri 网方法的缺点在于难以描述除了 alt，opt，break，loop 和 par 交互操作符以外的其它操作符的语义。另外，Petri 网方法侧重描述系统内部的交互行为，难以表示系统与参与者的复杂交互。

4　方法研究

4.1　语义选择

UML 顺序图语义的定义规则是在 UML 标准的基础上先定义顺序图语法，再定义顺序图语义。结合 UML 和形式化方法的优点，将非形式化的 UML 顺序图转换为具有精确语义的形式化规范，在非形式化的图形表示与形式化定义之间建立映射关系。本文只介绍了多种语义定义方法，没有指出哪种方法最好，因为不同的方法有不同的应用范围和使用目的，并且实现方式也不相同。本文也没有给出各种方法被工具支持的程度以及能够被实际应用的程度。

表 2 和表 3 分别从指称语义和操作语义两方面对第 3 节中介绍的方法进行了说明。

表 2　使用指称语义对 UML 顺序图进行形式化定义的方法列表

方法名	指称语义说明
Störrle	路径语义
STAIRS 指称语义	利于迭代开发的路径语义
Cengarle&Knapp 指称语义	基于偏序多重集的路径语义
条件路径	路径语义-路径中不仅有事件而且有条件
集合论	基于集合论的路径语义
LTL 框架方法	基于线性时态逻辑 LTL 的路径语义
NuSMV	基于 NuSMV 模型检验工具的输入语言
共代数	基于共代数的语义
标签事件结构	基于标签事件结构的真实并发语义
Hammal	基于偏序的分支时间语义

表 3　使用操作语义对 UML 顺序图进行形式化定义的方法列表

方法名	操作语义说明
ASM	基于抽象状态机
MSD	使用模态顺序图并结合弱字自动机
uMSD	基于弱交换 Büchi 自动机
西安电子科技大学	基于事件确定有限自动机
具体语法代数图转换	基于 UML 顺序图具体语法的代数图转换
抽象语法代数图转换	基于 UML 顺序图抽象语法的代数图转换
模版语义	基于层次转换系统的模版语义
STAIRS 操作语义	基于执行系统和投影系统两种转换系统
安全-活性语义	基于不确定性自动机的活性安全语义
OHST	基于建立在有序分层结构树上的转换系统
进程代数	基于进程代数的语义
Promela 描述	基于模型检测工具 SPIN 的模型输入语言
Cengarle & Knapp 操作语义	与 Cengarle & Knapp 指称语义对应的操作语义
Petri 网	基于 Petri 网及其扩展结构

结合具体定义方法，得到以下分析结果。

(1)如表 2 所列，构建 UML 顺序图所表达的交互的路径语义是进行顺序图指称语义定义的主要方法，与“顺序图语义的中心概念是路径”的 OMG 描述是一致的。路径语义的基本单元是事件，不同的语义方法中对事件的定义方法也不同，事件信息中不仅可以包含消息的名称、发送和接收者，还可以包含消息的类型(同步或异步)和消息序号等。定义路径语义的前提是进行路径分类，现有方法中的主要路径分类如表 4 所列，其中三分法是最常采用的路径分类，此类方法一般在给出语义定义后还给出语义精化规则，最典型的就是 STAIRS 指称语义方法。Lu Lunjin 提出的条件路径方法还对给出的精化规则进行了精化推理，以验证精化结果的正确性。以模型检验或一致性检测为目的形式化定义常将路径分为有效和其它或无效和其它。使用有效和其它分类时，主要目的是检测模型的活性；使用无效和其它分类时，主要目的是验证模型的安全性，例如 Grosu 和 Smolka 提出的具体语法代数图转换方法。Cengarle & Knapp 指称语义方法中采用有效和无效路径分类两分法，将有效路径和无效路径的并作为路径全集，因此有效路径集的补集就是无效路径集。

表 4　路径分类方法表

方法	路径分类	特殊目的
三分法	有效，无效，非确定性	精化；迭代建模
两分法	有效，其它	模型检验；活性
	无效，其它	模型检测；安全性
	有效，无效	OMG 对路径的分类

路径的有效性和无效性不仅与路径分类相关，而且与有效路径是用全序路径(complete traces)还是用偏序路径(partical traces)的计算和表达有关。采用不同方法进行路径有效

性判断的结果是不同的，例如一个顺序图中的同一条路径在某种方法下是有效的，而在另一种方法下就是无效的，还可能出现既有效又无效的情况。

(2)如表 3 所列，UML 顺序图的操作语义常采用两种方式进行语义定义。一种方式是将顺序图转换为与其行为等价的已经成熟的图结构或扩展形式来实现语义定义，其最关键的是解决顺序图的等价图转换问题，例如抽象状态机、Büchi 自动机、层次转换系统、标签事件结构、有限事件状态机、petri 网等，其中多种方法采用 Büchi 自动机来实现顺序图语义的等价转换，例如 uMSD 方法、抽象语法代数图转换方法和 MSD 方法。构建等价的转换系统时，可以针对整个顺序图直接构建转换系统，例如 MSD 方法；或先对各个生命线上的实例构建各自的转换系统，然后根据消息的连接合并生成整个图的转换系统，例如标签事件结构方法。此方式通过已有的转换系统中已经建立好的语义和结论来分析和定义顺序图语义，但往往需要对要使用的图结构或转换系统进行操作符扩展才能准确地描述 UML2 顺序图中的各种交互片段。另一种方式是利用其它对系统行为建模的形式化表述方法来描述顺序图的操作语义，例如进程代数、Promela 语言等。但此方式常受到形式化表述方法本身描述能力的限制，不能实现 UML 顺序图中所有组合交互操作符的语义定义，尤其是一致性关系操作符。

使用指称语义和操作语义进行 UML 顺序图语义定义各有优缺点。指称语义的优点是能更简单地对系统进行抽象描述，适合数学推理和对属性进行形式化分析，但缺点是不利于对工具开发者进行指导，对使用用户来说太过复杂且不易理解，很难表达状态和操作。操作语义的优点是利于形式化的实现，对开发者有很好的指导作用，并且很容易表达状态和操作；缺点是定义太过具体，因此抽取其中的形式化证明很困难，并且依赖抽象计算机的底层语义。

(3)大部分方法对并发交互中事件序列的描述采用的是交错语义，小部分方法采用真正的并发语义，例如 petri 网方法、标签事件结构方法等。大部分方法采用线性时间语义对决策性的分支交互进行描述，极少的方法采用分支时间语义，例如 Hammal 方法。

(4)早期的语义定义方法仅仅关注语义本身的解释，而后期的语义定义方法不仅关注语义本身的表达能力还关注该语义基础上的后续工作，例如以顺序图间的一致性检测为目的的条件路径方法，以顺序图与其它类型 UML 图之间的一致性检测为目的的 Youcef Hammal 方法，Petri 网方法，以模型检验为目的的西安电子科技大学的系列方法，Promela 方法、抽象语法代数图转换方法，NuSMV 方法和 LTL 框架方法，以自动产生测试用例的西安电子科技大学系列方法，便于实现时间语义扩展的 STAIRS 指称语义方法，Hammal 方法，Petri 网方法。

4.2 组合交互片段

根据顺序图中是否包含组合交互操作符，可以将交互分为基本交互和复杂交互。组合交互片段的引入在增强 UML 顺序图表达能力的同时，也带来了理解和分析上的模糊性。因此，如何对 UML 顺序图中的组合交互片段进行更加精确的形式化描述，是对能够体现复杂交互的顺序图进行准确而无二义性理解和诠释的关键，也是进行 UML 顺序图语义研究要解决的关键问题。

现有的方法都完成了对基本交互的定义，但除了 Shen Hui 提出的 NuSMV 方法和 LTL 框架方法实现了对 UML2 顺序图中所有组合交互操作符的语义定义外，其余方法都只实现了部分组合交互操作符的语义定义，如表 5 和表 6 所列。

表 5 多种 UML 顺序图指称语义形式化方法中定义的元素列表

操作符 / 方法名	alt	opt	loop	break	par	seq	strict	critical	assert	neg	consider	ignore
Störrle	√	√	√	√	√	√	√	√	√	√	√	√
STAIRS 指称语义	√	√	√		√	√			√	√		
Cengarle & Knapp 指称语义	√		√		√	√	√		√	√		√
条件路径	√	√	√		√	√	√	√				
集合论	√	√	√		√	√	√		√		√	√
LTL 框架	√	√	√	√	√	√	√	√	√	√	√	√
NuSUM	√	√	√	√	√	√	√	√	√	√	√	√
共代数	√	√	√		√	√	√					
标签事件结构	√				√	√			√	√		
Hammal	√	√	√	√	√	√	√					

表 6 多种 UML 顺序图操作语义形式化方法中定义的元素列表

操作符 / 方法名	alt	opt	loop	break	par	seq	strict	critical	assert	neg	consider	ignore
ASM	√				√	√	√		√	√		
MSD	√		√	√					√	√	√	√
uMSD	√		√		√	√		√		√	√	√
西安电子科技大学	√	√	√	√	√					√		
具体语法代数图转换	√	√	√		√					√		
抽象语法代数图转换	√		√			√	√			√		
模版语义	√	√		√	√	√	√					
STAIRS 操作语义	√		√		√	√				√		
安全-活性语义	√		√		√	√				√		
OHST	√	√	√	√	√	√		√		√		
进程代数	√	√	√	√	√							
Promela 描述	√	√	√	√	√	√						
Cengarle & Knapp 操作语义	√		√		√	√	√			√		√
Petri 网	√	√	√	√	√							

通过分析得到以下结论。

(1)在选择与迭代执行相关的交互操作符中，alt，opt 和 loop 操作符在绝大部分方法中进行了语义定义，但只有半数方法完成了 break 操作符的语义定义。break 交互片段用于中断并跳出它所在的内层交互。从已经实现了对 break 操作符语义定义的方法中可以看出，采用操作语义定义方式相对指称语义更利于实现对 break 操作符的语义定义。

(2)在顺序与并发执行相关的交互操作符中，par，seq 和 strict 操作符在绝大部分方法中进行了语义定义，但大部分方法缺少 critical 操作符的语义定义。critical 交互片段相当于对其内部交互进行约束或限定，常称为临界区，该区域中的事件是原子的且不能与其它事件交错执行。在顺序场景中，类似于数据库中的事务操作；在并发场景中，类似于操作系统临界区的访问控制。从已经实现了对 critical 操作符语义定义的方法中可以看出，采用指称语义定义方式相对操作语义更利于实现对 critical 操作符的语义定义。

由于 break 和 critical 操作符的语义都涉及到其所在的外层交互片段，因此对它们进行完整的语义定义存在一定的难度。

(3)UML 顺序图使用 assert 和 neg 交互片段来增强对必须发生的行为和禁止发生的行为的表达，ignore 和 consider 操作符可以更深层地操作在路径中出现的事件集。这些与路径的有效性和无效性相关的操作符称为一致性关系操作符。OMG 对 UML2 顺序图的语义描述建立在一对有效和无效路径上，无效路径的产生受 assert 和 neg 交互片段的影响。但 OMG 对一致性关系操作符的语义定义并不精确和充分，它们的使用可能会产生很严重的语义模糊问题。此外，对 UML 顺序图进行活性和安全性测试和验证的前提条件是使用 assert 和 neg 操作符来构建系统的需求行为和禁止行为。因此，一致性关系操作符的语义问题是顺序图语义研究的关键问题。

多种方法尚未解决一致性关系操作符的语义定义，尤其缺乏与 assert 和 neg 联合使用的 consider 和 ignore 操作符的定义。在已有方法中，对 assert 操作符的定义采用指称语义更容易实现，对 neg 操作符的定义采用操作语义更容易实现。

当一个事件 m 既出现在一致性关系操作符所在的交互域内又出现在一致性关系操作符所在交互域外(见图 7)，一个给定的路径?m. !m 可能会产生既有效又无效的情况。不同的方法对此类问题的解决方案也是不同的，该路径在 STAIRS 指称语义方法中是有效的，在 Cengarle & Knapp 指称语义方法中仍具有二义性。有些方法也对一致性关系操作符提出了特殊的使用限制来避免语义模糊的问题，如 uMSD 方法。因此，如何解决此类路径语义问题也是进行 UML 顺序图语义研究的内容。

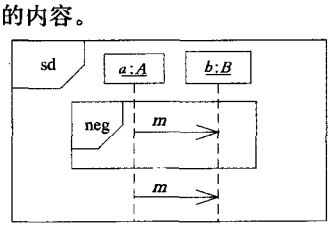

图 7　出现既有效又无效路径的图

4.3　嵌套交互的定义

进行完整的 UML 顺序图语义定义不仅要解决基本交互和组合交互片段的语义问题，还要解决由组合交互片段间的嵌套所产生的复杂语义问题。嵌套语义问题，尤其是一致性关系操作符的嵌套语义问题，是大部分现有方法中尚未解决的问题。

标签事件结构方法是现有方法中比较成功地定义了交互片段嵌套语义的方法。Küster-Filipe 通过给出每一个事件的前任来表达嵌套层次关系，这种通过交互的最基本单元进行交互嵌套定义的方式是值得借鉴的。NuSMV 方法通过将包含嵌套交互的片段映射成一个分层的执行单元 HEU(hierarchical execution unit)和将不包含嵌套交互的片段映射成基本执行单元来实现交互片段嵌套语义的定义。在 LTL 框架方法中，由于各个组合交互片段定义的独立性，导致嵌套的组合交互片段可以通过 LTL 定义的连接来实现。

但在嵌套语义明确前，一些方法建议避免使用它或限制使用它。

4.4　交互片段连接符和同步化机制的采用

在对 UML 顺序图进行语义计算前，常采用两种方法进行图的预处理工作。第一种是根据 UML 顺序图的语法将整个图分解或划分为基本交互和组合交互片段，然后使用顺序操作符进行语义连接，例如 Hammal 方法、petri 网方法、模版语义方法等。首先通过对图的具体语法或抽象语法进行分析形成中间表达或描述，然后通过递归展开交互片段，并按照每一个交互操作符的定义规则将它们粘合在一起进行语义定义，例如对 loop 交互片段的定义。OMG 将弱时序 seq 作为顺序图中交互片段连接的默认操作符。但此类方法会产生路径集合过多且不唯一的问题，因为不同生命线上发生的事件顺序可以是任意的，例如使用弱时序来定义 loop 交互片段的语义会形成与直观理解上的差异。因此有些方法使用强时序 strict 来进行交互的连接。采用此种方法需要考虑两个重要问题：图的划分算法的设计以及分块图的语义连接操作符的选取(seq 或 strict)。第二种方法是将整个图作为一个整体，在图中进行位置(location)标记，根据位置相对序列的约束进行语义计算，在此种方法中如何进行位置标记是关键。此时，不仅需要为生命线上的发送和接收消息的时间位置点设置标记，而且需要考虑是否对交互片段所覆盖的所有生命线采用同步化机制。交互对象生命线上的标记位置信息与要构建的转换系统(状态机、自动机等)中的状态集的获取有关。在 UML 顺序图上，同步化可以体现在两方面：进出交互片段的同步化和监护条件(guard)的同步化。进出交互片段上的同步化关系到交互边框与所覆盖的生命线相交位置点的状态获取；监护条件上的同步化关系到监护条件 guard 所在水平时间位置和各生命线相交位置点的状态获取。guard 同步化是在 guard 位置对应时间上的交互片段所覆盖的所有生命线上标记位置点，guard 不同步表示只在 guard 所在的生命线上标记位置点。

图 8－图 10 分别展示了对同一个顺序图采用 ASM 方法、标签事件结构方法和 MSD 方法得到的不同位置标记图。这 3 种方法都采用了进出交互片段的同步化机制，但 guard

的同步化方案是不同的。ASM 方法中没有将 guard 作为位置点;标签事件结构方法中,只将 guard 作为位置点,但没有进行 guard 同步化;MSD 方法中,不仅将 guard 作为位置点,而且进行了 guard 同步化。显而易见,这 3 种方法得到的各个交互对象的状态集是不同的。

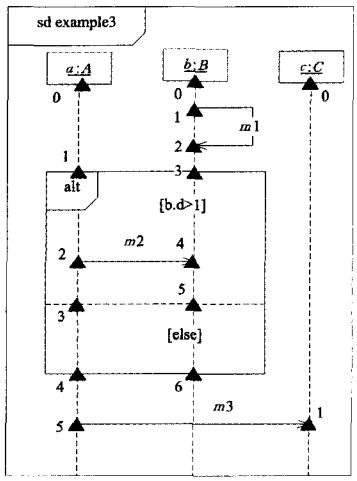

图 8 ASM 方法对应的位置标记图

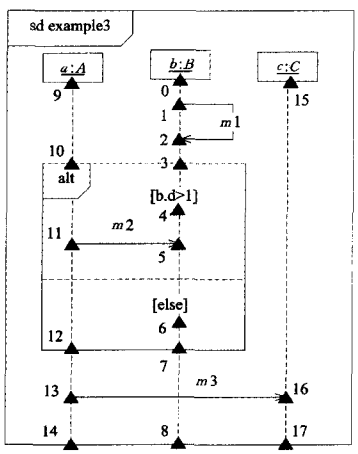

图 9 标签事件结构方法对应的位置标记图

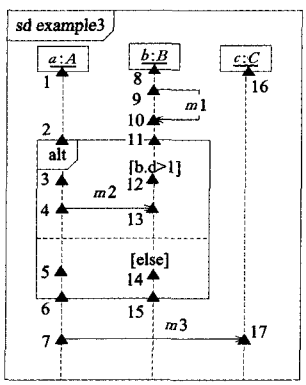

图 10 MSD 方法对应的位置标记图

在 OMG 中并没有同步化点的规定,即并不将穿过交互片段的同步化点看作是事件,因此不会出现在路径中,而 OMG 给出的事件排序约束仅仅是对事件排序进行规约。如果不使用同步化而仅仅用弱时序进行交互片段连接,则会引发很多违反直觉的事件排序,例如 alt,opt,loop,break 等交互片段产生的有效路径会与直觉上的有效路径违背,会影响对图模型的理解。而使用同步化后,不仅在视觉上会使消息位置和事件前后关系更接近一致,而且可以对多个事件偏序集进行缩减,使建立的模型更适合检测。如果要实现同步化,就必须使用强时序而不是弱时序对交互片段进行连接。如果在没有同步化的前提下最终要实现同步化,就需要给出因果规则,在进行位置状态合并时遵循给出的规则。

4.5 事件顺序的计算

事件顺序的计算与偏序或全序关系的选取相关,因此进行多条路径合并时,事件发生顺序产生的规则是不同的。计算时要遵循或满足 OMG 给出的事件发生时序规则,例如 OMG 给出的发送和接收事件序列规则、弱时序 seq 和强时序 strict 的事件发生规则等。弱时序和强时序都是事件发生时序关系上的约束,但强时序的约束更强。弱时序只对同一生命线的事件顺序进行约束,而强时序对操作域的时序也有约束。

在事件的顺序计算上也与进出交互片段所覆盖的生命线上是否进行同步化有关。在存在具有监护条件 guard 的 opt 和 alt 交互片段内,事件的顺序计算也与 guard 的同步化有关。另外,在事件的顺序计算上,确定一个事件的前一个事件有时是很重要的,尤其是对于含有分支交互、并发交互以及嵌套组合的交互来说。对于分支交互来说,事件的确定与 guard 监护条件有关;对于并发交互来说,由于并发事件执行快慢的存在,因此事件顺序的确定存在多种可能情况。由于既要考虑事件的顺序计算又要将路径分为有效和无效路径,语义合成是比较困难的,因为很难从组合交互片段和语法组合操作符形成的图中抽取所要表达的含义。

5 展望

在 UML 顺序图形式化语义的研究上,还有许多未解决的问题,今后可以在已有研究的基础上借鉴已有的研究方法开展进一步的研究工作,主要的工作内容和研究思路如下:

(1)无效路径的表达不仅与 assert 和 neg 交互片段有关,而且与状态不变量、持续约束、时间约束相关。现有的方法中对无效路径的研究仅关注 assert 和 neg 交互片段,而缺少对状态不变量、持续约束、时间约束的考虑。因此在今后的研究中可以关注这些约束违背引起的无效路径的定义。

(2)使用 ref 交互操作符表示的引用交互(interaction-Use)和用来连接交互片段内外同一消息的 gate 元素在已有的很多方法中也缺少语义定义。引用交互实现对交互模型的复用,有着与程序设计中的文件包含类似的效果。而 gate 的引入对于组合交互片段会引发很多路径问题,例如使用 gate 来连接 loop 交互片段内外的同一消息会产生不合法路径或违反官方约束问题。因此,有许多方法建议在 gate 语义未定义明确前进行使用限制,仅限于与引用交互的联用。因此,在今后的研究中可以关注 ref 和 gate 的语义的定义,并在 UML 顺序图语义描述前进行图形替换等图预处理工作,类似程序设计中的预处理命令和内联(inline)函数的实现。

(3)在对 ignore 和 consider 的语义定义上,可以考虑被 ignore 忽略的消息应该是 consider 所要考虑的消息,因此 ignore 和 consider 有相应的等价关系。因此在今后的语义定义上可以利用此类等价关系。

(4)嵌套语义问题虽然是 UML 顺序图语义研究中的难点问题，但也是对复杂交互进行精确描述必须解决的问题。在今后的工作中可以借鉴 UML 状态图中的深层历史状态来记录嵌套层次，或利用层次状态机思想结合顺序图来实现交互的嵌套定义。

(5)形式化方法的选择与顺序图的使用目的有直接关系。早期的研究仅仅关注语义的精确表达，而近期的研究不仅关注语义的精确表达，同时还关注在此语义基础上的进一步研究，例如代码的自动生成、一致性检测、模型检验、测试用例的自动生成以及时间语义的扩展等。因此为了特殊目的所进行的语义定义，要保证能够为后期的进一步研究提供语义支持。如果需要对系统属性进行形式化证明，就需要用具有精确语义的数学或逻辑来进行语义描述；如果要确保模型能够进行自动代码生成或自动模型检测，就需要语义系统能够被计算机所处理。例如，以模型检验为目的的语义定义方法的选择可以根据后期要使用的模型检验工具的输入语言来确定。

同时，对 UML 顺序图进行活性、安全性测试和验证的前提条件是对系统需求行为和禁止行为的描述，在 UML 顺序图中这两种行为由 assert 和 neg 交互片段来体现。因此 assert 和 neg 交互片段的语义定义对顺序图交互的准确表达及后期建立在语义基础上的模型检验等研究都是至关重要的。

(6)形式化方法的选择还与被建模的软件系统的应用类型有关，例如一般的软件系统、实时系统、嵌入式系统等。在建模时除了需要获取系统中的重要属性和特性，针对不同的系统还需要捕获功能属性、非功能属性、时间约束、行为约束等。因此如何对特殊需求的软件系统进行模型语义扩展是今后研究的一个重要分支，例如为实时系统进行 UML 顺序图的时间语义扩展。

(7)结合本人前期对 UML 状态图语义的形式化研究成果[68]，通过对 UML 顺序图语义的形式化定义来实现顺序图和状态图的一致性检测。

结束语　为 UML 顺序图构建形式化语义有利于对 UML 模型的准确理解，可以为基于 UML 模型的可靠性分析和验证提供语义支持，可以为进行基于 UML 模型的模型驱动开发奠定基础，并且对 UML 建模工具功能的增加和完善有一定的推动作用。

本文针对 UML 顺序图形式化语义的定义方法进行了综述，对多种方法进行了比较研究，并从指称语义和操作语义两种方式上进行方法分类，总结出各自的优缺点。文中还从语义选择、组合交互片段的定义、嵌套交互的定义、同步机制的引入和事件顺序的计算几方面指出在进行顺序图语义定义时需要关注的重要问题，并提出了一些建议。文中提到的方法几乎覆盖了现有研究中的大多数方法，可以为该领域的进一步研究提供参考。

参 考 文 献

[1] Object Management Group. UML 2. 4. 1 Superstructure Specification[R/OL]. [2011-08-06] (2015-10-01). http://www. omg. org/spec/UML/ 2. 4. 1/ Superstructure/PDF.

[2] Object Management Group. UML 2. 5[R/OL]. [2015-03-01]. http:// www. omg. org /spec/UML/2. 5/PDF.

[3] EGYED A. Automatically Detecting and Tracking Inconsistencies in Software Design Models[J]. IEEE transactions on software engineering, 2011, 37(2): 188-203.

[4] MICSKEI Z, WAESELYNCK H. The many meanings of UML 2 Sequence Diagrams a survey [J]. Software & Systems Modeling, 2011, 10(4): 489-514.

[5] LUND M S, REFSDAL A, STØLEN K. Semantics of UML Models for Dynamic Behavior-A Survey of Different Approaches [C] // Model-Based Engineering of Embedded Real-Time Systems, 2010: 77-103.

[6] STÖRRLE H. Semantics of interactions in UML 2. 0[C] // IEEE Symposium on Human Centric Computing Languages and Environments. Washington: IEEE Computer Society. 2003: 129-136.

[7] KNAPP A, STÖRRLE H. Efficient Representation of Timed UML2 Interactions [M] // System Analysis and Modeling: Models and Reusability. 2014, 8769: 110-125.

[8] ØYSTEIN H, KNUT EILIF H, RAGNHILD KOBRO R, et al. STAIRS towards formal design with sequence diagrams [J]. Software & Systems Modeling, 2005, 4(4): 355-357.

[9] MASS SOLDAL L, KETIL S. A Fully General Operational Semantics for UML 2. 0 Sequence Diagrams with Potential and Mandatory Choice [C] // International Conference on Formel Methods. Springer-Verlag. 2006: 199-204.

[10] MASS SOLDAL L, STØLEN K. Deriving Tests from UML 2. 0 Sequence Diagrams with neg and assert [C] // Proceedings of the 2006 International Workshop on Automation of Software Test, 2006. New York: ACM, 2006: 22-28.

[11] REFSDAL A, RUNDE R K, STØLEN K. Stepwise refinement of sequence diagrams with soft real-time constraints [J]. Journal of Computer and System Sciences, 2015, 81(7): 1221-1251.

[12] REFSDAL A. Specifying computer systems with probabilistic sequence diagrams [D]. Oslo: University of Oslo, 2008.

[13] RUNDE R K, REFSDAL A, STØLEN K. Relating computer systems to sequence diagrams: the impact of under-specification and inherent nondeterminism [J]. Formal Aspects of Computing, 2013, 25(2): 159-187.

[14] CENGARLE M V. UML 2. 0 Interactions: semantics and refinement: TUM-I0415 [R]. Institut für Informatik: Technische Universität München, 2004.

[15] CENGARLE M V, KNAPP A. An Institution for UML 2. 0 interactions, TUM-I0808 [R]. Institut für Informatik: Technische Universität München, 2008.

[16] LU L J, KIM D K. Required Behavior of UML Sequence Diagrams: Semantics and Refinement[C] // IEEE International Conference of Engineering of Complex Computer Systems. Las Vegas: IEEE. 2011: 127-136.

[17] LU L J, KIM D K. Refinement Inference for Sequence Diagrams [M] // SDFSEM 2013: Theory and Practice of Computer Science. Berlin Heidelberg, 2013: 432-444.

[18] LU L J, KIM D K. Required Behavior of Sequence Diagrams: Semantics and Conformance [J]. ACM Transactions on Software Engineering and Methodology, 2014, 23(2): 1-16.

[19] GU S S, CAI S B, LI S X. A formal semantics for interactions in UML2 [J]. Journal of Frontiers of Computer S-cience and Technology, 2012, 6(7): 631-643. (in Chinese)

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.

[20] SHEN H,ROBINSON M,NIU J W. A Logical Framework for Sequence Diagram with Combined Fragments,CS-TR-2011-015 [R]. Austin:University of Texas at San Antonio,2011.

[21] SHEN H,ROBINSON M,NIN J W. Formal analysis of sequence diagram with combined fragments[C] // International Conference on Software Paradigm Trends. Nature Publishing Group. 2012:44-54.

[22] SHEN H,KRISHNAN R,SLAVIN R,et al. Sequence Diagrams Aided Privacy Policy Specification[J]. IEEE Transactions on Dependable and Secure Computing,2014(99):1-14.

[23] SUN M,BARBOSA L S. A coalgebraic semantic framework for reasoning about UML sequence diagram[C]// The 8th International Conference on Quality Software. Oxford:IEEE,2008:17-26.

[24] BARBOSA L S,SUN M. UML Model Refactoring as Refinement:A Coalgebraic Perspective[C] // Symbolic and Numeric Algorithms for Scientific Computing,2008. Washington:IEEE Computer Society,2008:340-347.

[25] FILIPE J K. Modelling concurrent interactions [J]. Theoretical Computer Science,2014,351(351):203-220.

[26] FILIPE J K. Decomposing interactions [J]. Algebraic Methodology and Software Technology,2006,4019:189-203.

[27] HAMMAL Y. Branching Time Semantics for UML 2.0 Sequence Diagrams[C] // Formal Techniques for Networked and Distributed Systems(FORTE 2006). 2006:259-274.

[28] HAMMAL Y. A Formal Methodology for Semantics and Time Consistency Checking of UML Dynamic Diagrams[J]. Journal of the Chinese Institute of Engineers,2011,34(2):197-211.

[29] CAVARRA A,KÜSTER-FILIPE J. Combining Sequence Diagrams and OCL for liveness[J]. Electronic Notes in Theoretical Computer Science,2005,115:19-38.

[30] CAVARRA A,KÜSTER-FILIPE J. Formalizing Liveness-Enriched Sequence Diagrams Using ASMs[M] // Abstract State Machines 2004 Advance in Theory and Practice. Springer Berlin Heidelberg,2004:62-77.

[31] HAREL D,KANTOR A. Modal Scenarios as Automata [M]// Language,Culture,Computation. Computing-Theory and Technology. Springer Berlin Heidelberg,2014:156-167.

[32] HAREL D,MAOZ S. Assert and negate revisited:modal semantics for UML Sequence Diagrams [J]. Software & Systems Modeling,2008,7(2):237-253.

[33] LI W R,WANG Z J,ZHANG P C. Formal Semantics of Universal Modal Sequence Diagram [J]. Journal of Software,2011,22(4):659-675. (in Chinese)
李雯睿,王志坚,张鹏程. 模态顺序图 uMSD 的形式语义[J]. 软件学报,2011,22(4):659-675.

[34] WANG Z J,LI W R,YANG G X,et al. Research on Verification of Web Service Composition Based on uMSD[J]. Computer Science,2011,38(9):119-125. (in Chinese)
王志坚,李雯睿,杨种学,等. 基于 uMSD 的 Web 服务组合验证方法研究[J]. 计算机科学,2011,38(9):119-125.

[35] ZHANG C,DUAN Z H,TIAN C. Specification and verification of UML2.0 sequence diagram based on event deterministic finite automata [J]. Journal of Software,2011,22(11):2625-2638. (in Chinese)
张琛,段振华,田聪. 基于事件确定有限自动机的 UML2.0 序列图描述与验证[J]. 软件学报,2011,22(11):2625-2638.

[36] ZHANG C. Testing and Vefification Methods Based on UML2.0 models [D]. Xi'an:Xidian University,2012. (in Chinese)
张琛. 基于 UML2.0 模型的测试与验证方法 [D]. 西安:西安电子科技大学,2012.

[37] ZHANG C,DUAN Z H. Test Case Generation Based on UML 2.0 Models [J]. Journal of Xi'an Jiaotong University,2011,45(8):18-23. (in Chinese)
张琛,段振华. 应用 UML2.0 模型的测试用例生成方法[J]. 西安交通大学学报,2011,45(8):18-23.

[38] ZHANG C,DUAN Z H. A UML2.0-Based Software Safety Testing Method [J]. Journal of Wuhan University (Natural Science Edition),2010,56(2):165-169. (in Chinese)
张琛,段振华. 基于 UML2.0 的软件安全测试方法[J]. 武汉大学学报(理学版),2010,56(2):165-169.

[39] GRØNMO R,MØLLER-PEDERSEN B. From UML 2 Sequence Diagrams to State Machines by Graph Transformation [J]. Journal of Object Technology,2011,10(8):1-22.

[40] GRØNMO R,KROGDAHL S,MØLLER-PEDERSEN B. A collection operator for graph transformation [J]. Software & Systems Modeling,2013,12(1):121-144.

[41] GRØNMO R,RUNDE R K,MØLLER-PEDERSEN B. Confluence of aspects for sequence diagrams [J]. Software & System Modeling,2013,12(4):789-824.

[42] MESSAOUDI N,CHAOUI A,BETTAZ M. An Operational Semantics for UML 2 Sequence Diagrams Supported by Model Transformations [J]. Procedia Computer Science,2015,56(1):604-611.

[43] NIU J W,ATLEE J M,DAY N A. Template semantics for model-based notations [J]. IEEE Transactions on Software Engineering,2003,29(10):866-882.

[44] SHEN H,VIRANI A,NIU J W. Formalize UML 2 Sequence Diagrams[C] // High Assurance Systems Engineering Symposium. IEEE. 2008:437-440.

[45] LUND M S,STØLEN K. Deriving tests from UML 2.0 sequence diagrams with neg and assert[C] // Automation of Software Test,2006. New York:ACM,2006:22-28.

[46] MASS SOLDAL L. Escalator Tool,v. 1.5.5 [J/OL]. [2015-10]. http://heim.ifi.uio.no/~massl/escalator/Escalator-Tool_v1.5.5.pdf.

[47] GROSU R,SMOLKA S A. Safety-liveness semantics for UML 2.0 Sequence Diagrams[C] // Application of Concurrency to System Design,2005. Washington:IEEE Computer Society,2005:6-14.

[48] LI X S,LIU Z M,HE J F. A formal semantics of UML sequence diagram[C] // Software Engineering Conference,2004. Australian:IEEE,2004:168-177.

[49] LI X S A. Characterization of UML Diagrams and their Consistency[C] // Engineering of Complex Computer Systems,2006. Washington:IEEE Computer Society,2006:67-76.

[50] LI D,LI X S,LIU Z M,et al. Automated transformations from UML behavior models to contracts[J]. Science China Information Sciences,2014,57(12):1-17.

(下转第 64 页)

[59] GAVER W, DUNNE A. Projected Realities: Conceptual Design for Cultural Effect[C]// Proc of the CHI 99 Conference on Human Factors in Computing Systems: the CHI is the limit. 1999: 600-607.

[60] GAVER W W, DUNNE A, PACENTI E. Cultural Probes[J]. Interactions, 1999, 6(1): 21-29.

[61] VISSER F S, STAPPERS P J. Contextmapping: experiences from practice[J]. Codesign, 2005, 1(2): 119-149.

[62] VAN LEEUWEN J P, KARNIK M, KEANE K. Discovering Madeira: a case study of cultural probes[C]// Procedings of the Second Conference on Creativity and Innovation in Design. ACM, 2011: 439-447.

[63] KOSKINEN I, KUUSELA K, BATTARBEE K, et al. Morphome: a constructive field study of proactive information technology in the home[C]// Conference on Designing Interactive Systems. 2006: 179-188.

[64] SANDERS B N, STAPPERS P J. Probes, toolkits and prototypes: three approaches to making in codesigning[J]. Codesign International Journal of Cocreation in Design & the Arts, 2014, 10(1): 5-14.

[65] BROTMAN R, BURLESON W, FORLIZZI J, et al. Building Change: Constructive Design of Smart Domestic Environments for Goal Achievement[C]// CHI. 2015: 3083-3092.

[66] SANDERS B N. Generative tools for co-designing[M]. Springer London, 2000.

[67] VISSER F S, VISSER V. Re-using users: co-create and co-evaluate[J]. Personal & Ubiquitous Computing, 2006, 10(2/3): 148-152.

[68] SVENSSON J, ERIKSSON C I, EBBESSONE. User Contribution in Innovation Processes-Reflections from a Living Lab Perspective[C]// 46th Hawaii International Conference on System Sciences. IEEE Computer Society, 2010: 1-10.

(上接第 30 页)

[51] TRIBASTONE M, GILMORE S. Automatic translation of UML sequence diagrams into PEPA models[C]// Quantitative Evaluation of Systems, 2008. Washington: IEEE, 2008: 205-214.

[52] BOWLES J, KLOUL L. A Strongly Consistent Transformation from UML Interactions to PEPA Nets[C]// Computational Science and Its Applications, 2014. Berlin Heidelberg: Springer, 2014: 90-105.

[53] LI Q S, CHU H, CHEN P. The Formal Semantics of UML Sequence Diagram Based on Process Algebra [J]. Computer Science, 2004, 31(4): 173-175, 183. (in Chinese)
李青山, 褚华, 陈平. 基于进程代数的 UML 序列图的形式语义[J]. 计算机科学, 2004, 31(4): 173-175, 183.

[54] ZHAO Y F. The study on formal semantics of dynamic UML digrams [D]. Shanghai: East China Normal University, 2010. (in Chinese)
赵也非. 动态 UML 子图的形式语义研究[D]. 上海: 华东师范大学, 2010.

[55] QIAN C, YAN X F, ZHOU Y, et al. Model checking consistency of sequence diagram and state machine based on state reduction [J]. Application Research of Computers, 2014, 31(5): 1452-1455. (in Chinese)
钱成, 燕雪峰, 周勇, 等. 基于状态约简的顺序图和状态图一致性检测[J]. 计算机应用研究, 2014, 31(5): 1452-1455.

[56] LIMAV, TALHI C, MOUHEB D, et al. Formal verification and validation of UML 2.0 sequence diagrams using source and destination of messages [J]. Electronic Notes in Theoretical Computer Science, 2009, 254: 143-160.

[57] YOUNSI N, AMIRAT A, MENASRIA A. From UML 2.0 Sequence Diagrams to PROMELA code by Graph Transformation Using AToM3 [J/OL]. [2015-10-1]. http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.231.9003.

[58] ZHANG T. Research on formal verification methods of model of complicated information system [D]. Harbin: Harbin Engineering University, 2012. (in Chinese)
张涛. 复杂信息系统模型的形式化验证方法研究[D]. 哈尔滨: 哈尔滨工程大学, 2012.

[59] CENGARLE M, KNAPP A. Operational semantics of UML 2.0 interactions, TUM-I0505 [R]. Institut für Informatik: Technische Universitat München, 2005.

[60] GIRAULT C. 系统工程 Petri 网[M]. 王生原, 余鹏, 霍金健, 译. 北京: 电子工业出版社, 2005.

[61] STAINES T S. Transforming UML Sequence Diagrams into Petri Nets [J]. Journal of Communication and Computer, 2013 (10): 72-81.

[62] BOUABANA-TEBIBEL T, RUBIN S H. An interleaving semantics for UML 2 interactions using Petri nets[J]. Information Sciences. 2013, 232(5): 276-293.

[63] BOUARIOUA M, CHAOUI A, ELMANSOURI R. From UML Sequence Diagrams to Labeled Generalized Stochastic Petri Net Models Using Graph Transformation[M]// e-Technologies and Networks for Development. Springer Berlin Heidelberg, 2011: 318-328.

[64] YANG N H, YU H Q, SUN H, et al. Modeling UML sequence diagrams using extended Petri nets [J]. Telecommunication Systems, 2012, 51(2/3): 147-158.

[65] FARIA J P, PAIVA A C R. A toolset for conformance testing against UML sequence diagrams based on event-driven colored Petri nets [J]. International Journal on Software Tools for Technology Transfer, 2016, 18(3): 1-20.

[66] TAN H B, YAO S Z, XU J J. Behavioral Consistency Analysis of the UML Parallel Structures[J]. Natural Biotechnology, 2004, 22(9): 1146-1149.

[67] OUARDANI A, ESTEBAN P, PALUDETTO M, et al. A Metamodeling Approach for Sequence Diagrams to Petri Nets Transformation within the requirements validation process[J/OL]. 2006 [2015-10-1]. http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.460.2653.

[68] GUO Y Y, LIU J L. Formalization for model Element of UML Statechart in RSL [J]. Computer Science, 2013, 40(5): 177-183, 205. (in Chinese)
郭艳燕, 刘惊雷. UML 状态机模型元素的 RSL 形式化定义[J]. 计算机科学, 2013, 40(5): 177-183, 205.