计算机科学1998 Vol.25№.1

分布式系统的LOTOS规范及其实现

LOTOS Specification and Implementation for Distributed System

谢 冰 张晨东 陈火旺

(国防科技大学计算机系 长沙410073)

摘 要 LOTOS is a standard specification language designed for open system interconnection by ISO. This paper discusses about its use in the area of distributed system design, and the methodlogy for its implementation.

关键词 Distributed system, Specification language, LOTOS, Software engineering

1. 引言

分布式系统的主要特点是靠各独立运转的组成部分协同工作来完成系统的功能。对于电信交换、指挥调度、实时控制以及模拟与仿真等分布交互系统，系统设计的关键在于各组成部分是否可以通过正确的通讯来协调彼此工作。这种协同工作的关系是一种工作协议。这种协议能否保证系统功能的实现、是否安全、有效等问题，都需要使用形式化方法来验证、确保。

形式化描述技术(FDTs)是为了解决大型软件系统设计中由于对复杂性和正确性难于控制的软件危机而采用的一种方法。其目的是采用有坚实理论基础支持的形式化技术以确保系统的正确性、规范性。

一种FDT必然基于一种或几种数学模型，如FSM、Petri网、TL和CCS等，这就为充分地表述协议的各种特性(并发性、不确定性、时序性、递归性等)以便使协议验证、实现、测试和协议转换过程系统化和自动化提供了良好基础。

八十年代以来，计算机界提出了许多种FDT。对于协议工程来说，ISO组织颁布了ESTELLE语言和LOTOS(Language Of Temporal Ordering Specification)语言[1]两种标准FDT，而CCITT组织也颁布了它的SDL(Specificatiion Description Language)语言。ESTELLE与SDL是基于自动机模型，面向协议实现的；而LOTOS基于进程代数模型，是面向协议验证的一种抽象级别较高的FDT。LOTOS语言标准文本中，给出比较完整的协议验证规则，为别的FDT所不及。在应用上，SDL语言由于CCITT的支持，在电信领域使用广泛。LOTOS则因其抽象级别高，不便于直接进行系统设计而受到阻碍。但经过近年的研究与推广，伴随着相应的开发思想与开发环境、工具的研究，LOTOS正以其特有的将进程代数与抽象数据类型(ADT)理论相结合而带来的优越性日益受到重视。

对于不同的系统设计问题，需要选择合适的形式化方法，才可体现出形式化方法的优点。在分布式交互系统中，由于对正确的协调工作更为依赖，系统描述所用的方法在于描述系统与外部环境交互的事件及事件顺序，因而我们选用基于进程代数的形式化方法。

对于交互控制系统，随使用期的延长常会增加新的系统需求，对于维护的要求是相当高的。开发一个高效、可靠易于维护的软件系统需要一个比较好的开发方法，以及一套相应的开发环境(工具)。开发方法应能支持系统开发的整个过程。LOTOS作为一种标准的协议工程描述语言，已有了一个比较完整的设计、开发环境，可以支持快速原型的开发方法。对于以分布交互为设计关键的大型软件系统设计可以提供较好的支持。

2. LOTOS简介

LOTOS[1,2]是ISO组织为了描述开放分布式系统而制定的一种规范语言，是一种适应协议工程、分布处理和并行处理技术的要求而产生的规范语言。其基本点在于一个系统可通过在外部环境中可观察到的交互操作的时序关系来定义。它是基于进程代数和抽象数据类型代数规范的一种FDT。LOTOS规范由两部分构成，一是系统的行为规范，另一个是

系统中的数据规范。

在 LOTOS 中，一个系统被看作多个相互通讯的进程，这些进程又由多个子进程构成。这样，一个系统的 LOTOS 规范是一个进程定义的层次结构。进程是一个黑盒子，由它与其它进程（外界环境）的通讯时序关系定义。LOTOS 中的进程通讯为同步通讯。一个进程可以执行其内部的行为，也可以通过门径（gate）与其外部进程交互。进程是通过在其各门径上的交互行为以及内部行为的时序关系来定义的。

在 LOTOS 中，规范定义的格式为：

```
specification definition;
specification typical_spec[gate list](parameters list):
functionality:=〈type definitions〉
behaviour
    〈behaviour expression〉
where
    〈type definitions〉
    〈process definitions〉
endspec
process definition;
process process_id[gate_list](parameter_list):
  noexit/exit:=
    〈behavior_expression〉
endproc
```

其中 noexit 表示该进程为非终止进程，exit 表示该进程为终止进程，〈behavior-expression〉是描述进程行为的行为表达式。

2.1 LOTOS 的门径概念

LOTOS 引入了门径的概念。门径就是协同事件发生点，或两个进程同步通讯点。门径上的行为由门径名、门径上的操作以及传送的数据组成。

两个进程通过同一门径同步时，可进行数值匹配（纯同步作用）、数值传送和数值生成三种交互作用，如下表所示：

交互方式	进程 A	进程 B	同步条件	结果
数值匹配	g!e1	g!e2	数值相等(e1=e2)	同步
数值传递	g!e	g?x:t	e 为 t 类型数值	x=e
数值生成	g?x:t	g?y:u	类型一致(t=u)	x=y=…

2.2 行为表达式定义

进程的行为用行为表达式描述。行为表达式描述了进程的外部行为、内部行为以及时序关系。一次同步通讯是一个事件。事件是原子操作。同步通讯发生在门径上。

多个协议事件和行为表达式可以通过定义的各类行为算子组合为新的行为表达式，下面我们给出几个基本算子：

①行动前缀 ';'：说明进程顺序执行事件的行为。如："a;B"表示执行事件 a 之后执行行为表达式 B。

②选择 '[]'：说明进程从多个行为表达式中选择一个去执行的行为。如："B1[]B2"表示选择 B1或 B2执行。

③并行 '|[]|'，'||'，'|||'：说明进程并行执行多个行为表达式的行为。如："B1|||B2"表示 B1和 B2独立并行，"B1||B2"表示 B1和 B2通过它们的所有事件并行，"B1|[a_1,…,a_n]|B2"表示 B1和 B2通过门径 a_1,…,a_n 并行。

例如下面为一个信匣的进程定义。信匣有两个门径，投入口（in）和取出口（out），信匣的行为要求必须首先接收一个信件，然后等待该信件被取走，如此周而复始地运转。信匣的行为表达式如下：

```
Process proc_LetterBox[in,out] noexit:=
        in?letter,out!letter
endproc
```

2.3 抽象数据类型

LOTOS 中对于数据结构和数值表达式的描述采用了基于抽象数据类型的代数规范理论。它采用的是 ACT-ONE 抽象数据类型描述语言[3]，使用代数公理化方法来规范数据类型。抽象数据类型仅定义了实现系统时，数据和操作的必要性质。下面给出了一个信匣的简化数据类型描述例子：

```
type letterbox_type is letter_type,
                       letterset_type,boolean_type
    sorts letterbox
    opns creat:→letterbox
         put:letterbox,letter→letterbox
         get:letterbox→letter
         Is_full:→booleam
         Is_empty:→boolean
    eqns forall letterb:letterbox,m,n:letter
         ofsort letter
         get(put(put(letterb,m),n))=n
endtype
```

作为一种代数规范语言，ACT-ONE 还包括一些构造子，如组合、更名、实施、模块等，可以从已有的简单类型构造出结构化、良构的复杂类型。ACT-ONE 是一种核心语言，它只使用了代数规范语言中的基本操作符号集，并且仅有基本的几种预定义类型，这就使得 LOTOS 的数据定义既长又复杂，已有了一些方法与建议来简化其设计[4]。

有关 LOTOS 语言的详细语法和语义可见文[1][2]，限于篇幅，不再详述。

2.4 LOTOS 的设计支持工具

作为协议的形式化描述语言，LOTOS 已有了许多验证、测试、实现的工具与方法，可以支持协议开发的全过程。在 ESPRIT II 的 LOTOSPHERE 项目中开发的 LITE（LOTOS Intergrated Tool Environment）环境中有下列工具：

·crie：LOTOS 文本和结构编辑器，包括支持保

正确性的表达式转换能力，以及语法和静态语义的检验。

·glow：图形化的 LOTOS 浏览器，可将 LOTOS 规范用图形化的方法表达出来。

·smile：LOTOS 规范的仿真分析器，用于分析 LOTOS 规范的行为和抽象数据类型的动态模型。支持单步或指定步数的自动执行规范。

·topo：LOTOS 的实现工具，半自动地根据 LOTOS 规范生成 C 语言代码或 Ada 语言。

·cooper：LOTOS 规范的测试产生器，用于目标系统与规范的一致性测试。

此外，加拿大的 Ottawa 大学也开发了类似的环境。

LOTOS 是可执行的规范，在上述环境（工具）的支持下，我们可以使用快速原型的开发方法。

3. LOTOS 规范开发方法

3.1 规范类型

在系统开发的不同阶段中，形式化规范所表达的设计需求也是不同的。最初的系统需求规范与最终的系统实现规范不仅在规范的抽象、精化程度上有所不同，在规范的书写方式以及规范的结构上也是有所不同的。根据规范所表达的设计特性及其结构特性，可将规范划分为不同的规范类型[5]。

(1)面向约束的类型　在这种类型的规范中，由并行算符将分离的行为约束联结起来。这种类型在系统的需求获取阶段特别合适。它清楚地反映了系统中各个相互独立的行为约束，对于修改或增加新的行为约束非常方便。

(2)单调类型　这种类型的特点是规范中没有使用并行算符。常用于没有或者不考虑系统的分布情况时。

(3)面向资源的类型　这种类型中，规范的结构反映了系统实现时的实际组成单元的情况。规范中的进程与数据都与实际的组件有着较为直接的对应关系。这种类型的规范是属于实现阶段的。

(4)面向状态的类型　这种类型的规范中，单个资源的状态空间已被规范的结构明确地表达出来了。这种类型用在系统实现的最终阶段中，系统的各个部分都已经由状态机表达了。

3.2 规范设计方法

使用面向约束的规范设计方法是较为传统的规范方法[6]，但它是一种典型的面向过程的分析、设计方法。在面向对象方法的优越性以及工程人员对它的广为接受的趋势下，针对 LOTOS 语言设计时未融入面向对象的特性，人们一方面在标准 LOTOS 的基础上提出了基于对象的 LOTOS 规范分析、设计方法[7]；另一方面也在研究对它进行面向对象的新定义。本文仍基于标准的 LOTOS 语言来讨论系统设计问题，是为了能够使用现有的 LOTOS 开发环境与工具。

3.3 LOTOS 规范描述例子

在分布式控制系统中，从用户方面得来的需求常常是系统与外界的交互要求。这是因为用户关心的是系统的外部性质，而对于其内部结构等不太关注。比较明显的例子是在电信系统中的信号系统问题。信号系统是交换机之间协同工作的协议。如下图所示的中国一号信号与七号信号系统的部分信号转换图，其中关键的是标明了各个信号的交换顺序。

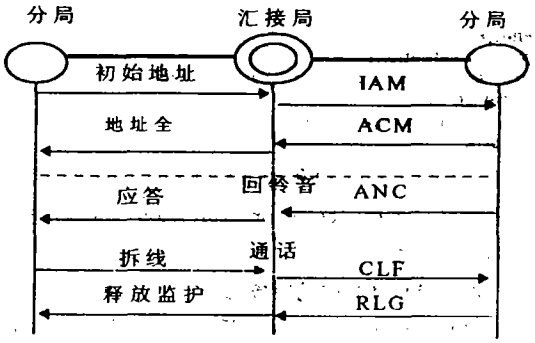

图1　汇接局转发1号信号和7号信号

根据这个信号转换图，我们可以较直接地写出如下的信号接续规范。（为了简化，本文仅写出基本的 LOTOS 规范部分，即行为部分）。其中，汇接局与一号信号接口为：NO1；汇接局与七号信号接口为：NO7。

```
process proc.exchanger[NO1,NO7]:noexit:=
   NO1!Calling;
   NO7!IAM;NO7!ACM;
   NO1!Address.OK;
   NO7!ANC;
   NO1!Disconnect;(NO1!Release;
            |||NO7!CLF;NO7!RLG)
   endproc
```

4. LOTOS 规范设计过程

4.1 设计过程

使用 FDTs 需要有一套相应的方法论作为指导。使用 LOTOS 语言设计分布式系统可以分为图2所示的四个主要步骤。

(1)需求获取阶段。不考虑系统的实际组成与处理过程，关注于用户的需求分析，本阶段的结果是无

歧义的用户需求规范，称为初始规范。在3.2节中介绍了设计方法。3.3节给出了一个简单的例子。由该例子可以看到LOTOS用于同用户交互所具备的精确、简洁性。

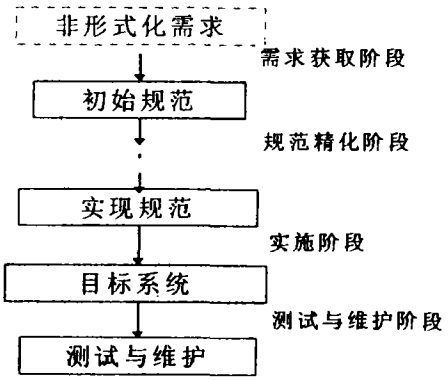

图2 系统的设计过程

(2)规范精化阶段。LOTOS语言支持从抽象级别很高的初始规范到具有大量实现细节的实现规范的描述。它支持逐步精化的开发方法，包括两个方面的精化：一是加入新的细节；二是使系统规范更加符合系统的实际实现环境。

这一阶段的目的在于使用LOTOS的形式化理论基础，从初始规范开始，不断精化，推导出与系统实际组成和处理结构相匹配的实现规范。这一阶段的推导是使用已证明具有正确性保持性质的规范变换算法来进行的，从而保证了实现规范对于初始规范的一致性。在4.2节中将较详细地讨论变换算法。

(3)规范实施阶段。实现规范仍是具有相当抽象程度的LOTOS语言描述，在分布式系统的目标系统中，需要采用执行效率更高的常规语言。这一阶段中，将LOTOS规范所描述的高层控制结构与具体系统运行的实施细节结合起来，形成目标程序设计语言程序。由于C++语言的面向对象特性及其较高的执行效率，以及在工业系统的广泛使用，我们认为采用C++语言实现系统细节更为有利。

(4)测试与维护阶段。由于使用了形式化与面向对象的开发方法，在设计上对于系统的正确性可得到较好的保证。LOTOS语言还具有相应的测试产生工具，可帮助目标软件的测试。同时，由于支持快速原型的开发方法，软件重用、软件维护等方面亦可得到较好的效果。

4.2 结构匹配的精化设计过程

正如上面所述的LOTOS有多种规范类型，一个系统的规范可以写为不同类型，应用在系统开发的不同阶段。这些不同类型的规范都定义的是同一个系统，显然它们之间存在一种等价性。在精化规范过程中，保持一定的等价性是必须的。

在实际系统设计中，一个系统常需要分解为多个组成部分。将原系统规范分解为多个子系统规范也需保证所有子系统协同工作于原系统的规范是等价的。

观察等价性的非形式化定义是：两个系统规范对外界的可观察行为是一致的。在从初始规范开始的结构精化过程中，精化的结果规范与初始规范保持观察等价性，而在规范所体现的结构上则更加符合实际系统的组成结构。为了保证系统设计的正确性，可以使用这种有精确形式化定义的等价关系检验实现规范是否与初始规范等价。

由于观察等价是传递的，另一种更直接的方法则是在第一次精化步骤中，只使用已经证明保等价性的转换方法。这种精化过程即是正确性保持转换(Correctness Preserving Transformation：CPT)。使用CPT，可以自动地在系统分解的划分集基础上，生成结果规范。在[8][9]文中讨论了相应的CPT算法。

本文中我们给出上面汇接局交换机的规范分解作为例子。若该交换机中一号信号处理机与七号信号处理机是两个分离的部分，两者之间有一个通讯门径：sync，使用[8]中的分解算法，可得到精化的规范如下：

```
process pro_exchange[NO1,NO7]:noexit:=
    hide sync in (proc_No1[NO1,sync]|[sync]|
                  proc_No7[sync,NO7])
    where
      process proc_No1[OUT1,IN]:noexit:=
        OUT1!Calling;IN!Calling;
        IN!ACM;OUT1!Address_OK;
        IN!ANC;OUT1!Answer;
        OUT1! Disconnect;(IN!Disconnect
                ||OUT1!Release)
      endproc
      process proc_No7[OUT7,IN]:noexit:=
        IN!Calling;OUT7!IAM;
        OUT7!ACM,IN!ACM;
        OUT7!ANC;IN!ANC;
        IN!Disconnect;OUT7!CLF;
        OUT7!RLG;
      endproc
endproc
```

4.3 分而治之的实施方法

系统规范的实施问题研究的是从系统实施规范生成目标语言源程序的过程，这是一个源-源(source-to-source)的转化过程。可以分为三个相互独立的部分来进行：一是系统的行为部分转化为系统的动态部分；二是系统的数据部分可以从规范所用的抽象数据类型定义较为直接地转换为对象的属

性部分；三是与实现环境密切相关的实现细节，如系统通讯方式、时钟工作方式等需要从实际环境中来考虑。

这种“分而治之”的方法如图3所示。其中，part1基于较为直接的转换方法，形成系统的运行框架；结合part2形成的系统数据对象，以及part3形成的实现细节，即可有机地形成目标源语言程序。采用“分而治之”的方法，使得设计人员可以将注意力放在单一的部分上，既提高了生产效率，降低了开发时间和费用，而且系统的修改不会引起全局的变更，对于维护也无疑是有好处的。part1基本上可以通过开发合适的工具来自动转换，既减少了工作量，又有助于生成系统的规范化及正确性。part2和part3需要部分的手工编程，在面向对象方法的支持下，其生成部分对于固定的系统环境具有较大的重用性，从而更好地支持了快速原型的开发方法。

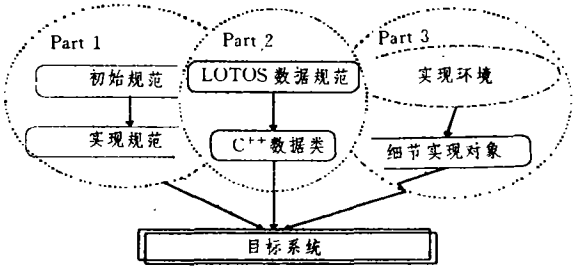

图3 LOTOS规范的实施方法

结束语 LOTOS规范语言因其具有精确定义的语法和形式语义，可以在坚实的数学基础上进行形式化推导，使之在检验、分析、测试精化、综合以及验证等方面可以自动化进行。使用适当的工具与方法论，将其用于分布式交互系统的设计，可以提高系统开发的效率、可靠性以及正确性。

LOTOS语言缺乏支持实时规范的特征，如对于时钟这一实时特征，LOTOS中以内部事件表达，不足以显式地刻画。LOTOS语言也不支持动态配置；采用的间隙(interleaved)并发的形式语义；以及ACT-ONE数据规范过于繁杂等不足，尤其是LOTOS语言没有使用面向对象的主流技术。近年来的研究工作一方面集中在上述缺陷上，另一主要研究领域则是集中在LOTOS语言的应用上。例如使用LOTOS语言进行软、硬件系统设计一体化，进行安全机制研究，以及面向数据库问题的应用等。

参考文献

[1] International Organisation for Standardization, IS 8807: LOTOS: A Formal Description Technique Based on the Temporal Ordering of Observational Behavior, 1989

[2] Introduction to the ISO Specification Language LOTOS, Computer Network and ISDN Systems, 14, 1987

[3] Jan de Meer, Rudolf Roth, Introduction to algebraic specifications based on the language ACT ONE, Computer Network and ISDN Systems, 23, 1992

[4] VLib: infinite virtual libraries for LOTOS, Protocol Specification, Testing and Verfication, XII (C-16) 1993 IFIP

[5] Transformations and Semantics for LOTOS, PhD. thesis Rom Langerak university of twente, Netherlands, 1992

[6] Mohammed Faci et al., Formal specification of telephone systems in LOTOS: the constraint-oriented style approach, Computer Networks and ISDN systems, 21, 1991

[7] ROOA: Rigious Object-oriented Analysis, PhD. thesis Ana Maria university of stirling, UK 1994

[8] 谢冰、王兵山、陈火旺. 基本LOTOS规范的进程分解方法，计算机学报(待发表)

[9] Maria Hultstrom, Structural Decomposition, Protocol Specification, Testing and Verfication, XV 1995 IFIP

(上接第97页)

值[5]或对参数进行变换得到。参数的变换有不同的方法(旋转、比例缩放、…)，这些变换的指定主要取决于所要产生的运动。

五、生成

生成技术可以分为：消隐、Shading、纹理映射、反走样等。消隐是最常用的生成方法，但是要生成具有真实感的人体运动则需要使用Shading和纹理映射。然而情况也不完全是这样，例如用Shading生成的人脸就看起来过于光滑(尤其是对女性的脸部)。头发的生成通常使用的是纹理映射的方法，但是用这种方法产生的头发真实感不强，用粒子系统是一种可能的解决方法。

结论 通过对现有人体动画技术的总结，我们得出如下结论：人体的造型趋向于多层次模型，尽可能地与真实人体一致。然而有些问题亟待解决，例如：如何对具有形变的皮肤进行造型。在运动控制方面趋向于多种模型的混合使用以弥补各自的缺点。目前的许多研究趋向于将动态分析和人工智能的技术运用到计算机动画中，使其产生的运动更为自然。表情运动仍然是一个活跃的研究领域，因为在这方面还有许多问题有待解决。(参考文献共10篇略)