第35卷 第23期 计 算 机 工 程 2009年12月
Vol.35 No.23 Computer Engineering December 2009
·软件技术与数据库· 文章编号：1000—3428(2009)23—0041—03 文献标识码：A 中图分类号：TP311

基于RSL的协议形式化描述与验证方法

顾 翔，邱建林，邵浩然
(南通大学计算机科学与技术学院，南通 226019)

摘 要：讨论使用RAISE规范语言(RSL)描述6种协议元素的方法。在RSL描述的基础上，借助操作符的运算规则、并行扩展规则和同步会合事件隐藏规则，对协议的相关性质进行验证，以一个简化的停止等待协议规范的描述和验证实例证明，与其他形式化方法相比，RSL表现出较强的描述能力。

关键词：协议工程；形式化描述；RAISE规范语言；协议验证

Protocol Formal Description and Verification Method Based on RSL

GU Xiang, QIU Jian-lin, SHAO Hao-ran
(School of Computer Science and Technology, Nantong University, Nantong 226019)

【Abstract】This paper discusses the way to use RAISE Specification Language(RSL) to describe six kinds of protocol elements. Based on RSL description, it uses operation rules of RSL operators, expansion rule of parallel operation and hide rule of synchronous-communicating events to verify related properties of the protocol. Description and verification of a stopping-waiting protocol are presented as an example, which prove that compared with other formal methods, RSL has better description ability.

【Key words】protocol engineering; formal description; RAISE Specification Language(RSL); protocol verification

协议形式化描述技术(Formal Description Technique, FDT)作为协议工程的一个组成部分，贯穿于整个协议的开发过程，是协议工程的基础。本文尝试将RAISE规范语言(RAISE Specification Language, RSL)引入协议的形式化领域，为协议的形式化描述提供一个新的有力手段，并在此基础上对协议验证方法进行探讨。

1 协议的形式化描述

目前，主要的协议形式化描述方法可以归纳为基于状态和基于序列2大类。

使用状态的优点是简单明了，直观易懂，最重要的协议元素-状态和事件的触发变迁关系清晰明了。它的缺点是：抽象级别比较高，难以面向实现；不能完整地描述协议的所有元素和内容，比如研究得最成熟的FSM(Finite State Machine)只能描述协议中30%的内容；当系统“稳定”在一个状态后，其协议行为的描述比较困难(例如RIP协议中路由算法的描述)[1]；较复杂的协议容易遇到“状态爆炸”问题。

使用序列的方法能够很好地描述协议的行为，借助进程代数的演算手段，还可以进行一些协议性质的证明，但是整个协议的状态变迁不如状态方法清晰。

另一方面，在ISO所定义的开放系统互联参考模型(OSI模型)中，协议的设计遵循层次结构，这就要求一个完整的处于某一具体层次上的协议规范由3个部分组成，即服务规范、功能规范和工作环境规范[2]。服务规范指明了本层协议向上层用户所提供的服务；工作环境规范指明了本层协议要求其下层协议提供的服务；功能规范指明了本层协议在接收到各种内部、外部事件(报文)时的反应和行动。

相比之下，基于状态的方法更适合用来描述协议的功能规范；基于序列的方法则更适合用来描述协议的服务规范[3]。一个好的协议形式化描述语言应能综合运用这2种方法，取长补短，从而最充分地表达协议性质。

RSL作为工业软件工程的严格方法规范语言，是一种广谱的语言，不仅可以用来描述高级别、较为抽象的规范，还可以用来描述低级别的具体设计实现。它本质上是一种数学模型，具有数学推理的严密性，可以借助数学的方法对其推理论证。但由于在设计时RSL主要是面向软件工程的，因此时间机制在RSL中没有相应的支持。为了使用RSL进行协议描述，需要对其进行最小化扩充，具体方法见文献[4]。扩充后的RSL可以充分描述上述3部分规范，使得对它们的描述处于同一语义框架内，这就为协议的验证和测试工作带来了便利。这一点是其他许多形式化技术不能达到的。

2 协议元素的RSL描述

协议一般由6种协议元素组成[5]。为了充分全面地描述一个协议，一个好的形式化描述语言应能描述所有这6种协议元素。本文设计的6种协议元素的一般描述方法如下。

(1)服务原语和服务原语时序

n层服务原语和原语参数描述了n层协议和它的用户之间的接口SAP(Service Access Point)。在RSL中，服务原语可以借助RSL中的操作来描述。下例描述了一条服务原语Data-Request：

基金项目：国家自然科学基金资助项目(60773041)；江苏省高校自然科学重大基础研究基金资助项目(07KJA51007)；江苏省高校自然科学基础研究基金资助项目(08KJB520009)；南通市应用研究计划基金资助项目(K2008005)；南通大学自然科学基金资助项目(06Z048)
作者简介：顾 翔(1973－)，男，副教授、博士，主研方向：协议工程；邱建林，教授；邵浩然，副教授、硕士
收稿日期：2009-03-30 **E-mail：**nicolcn@126.com

type DATAREQ
channel ChanOut
value DataRequest : DATAREQ→out ChanOut Unit
axiom DataRequest(DataReq:DATAREQ)≡ChanOut!DataReq

服务原语之间的时序可以由多条相应操作的执行顺序体现。

(2)协议数据单元(Protocol Data Unit, PDU)及其交换时序

PDU 定义了 *n* 层协议实体之间交换的信息。下例描述了一个名字为 A 的 PDU 的格式，它有 2 个域，各域长度分别为 1 Byte 和 2 Byte。

type BYTE, A==_(Field1:BYTE*,Field2:BYTE*)
axiom forall a ∈ A.
len(Field1(a))=1,
len(Field2(a))=2

其中，Field*i*(本例中 *i*=1,2)是解析器函数(Field*i*: A →BYTE*)。在实际的协议描述中，可以使用域名作为相应解析器函数的函数名。对于一个 A 类型的变量 a，直接使用 Field*i*(a)或 a.Field*i* 来获得相应的域，可以对这个域进行读写操作，并认为对该域的写操作直接作用于此变量的相应域。

对于含有变长域的 PDU，假定 A 有 2 个域：第 1 个域为 1 Byte，它给出了第 2 个变长域的长度；第 2 个域含有若干个项，每个项是一个二元组(length, value)，其中，length 为 1 Byte，给出了这个项的长度；value 是这个项的具体值。可以用如下方法描述其格式：

type BYTE
SUB_A==_(SUB_Field1:BYTE*,SUB_Field2:BYTE*)
A==_(Field1:BYTE*,Field2:SUB_A*)
value BYTE_List_Value: BYTE* → Int
axiom forall sub_a ∈ SUB_A, a ∈ A.
len(SUB_Field1(sub_a))=1,
len(SUB_Field2(sub_a))=BYTE_List_Value(SUB_Field1(sub_a)),
len(Field1(a))=1,
len(Field2(a))=BYTE_List_Value(Field1(a))

其中，辅助函数 BYTE_List_Value 用于计算一个字节表相应的整数值。

PDU 各个域的具体值所代表的意义和 PDU 交换时序隐式的定义在整个协议规范的描述中。

(3)协议状态

在 RSL 中，可以将一个协议状态直接看作是一个抽象数据，并在此基础上定义协议状态类型。下例是 BGP4 协议的协议状态类型定义：

type STATE == IDLE | CONNECT | ACTIVE
| OPENSEND | OPENCONFIRM | ESTABLISHED

(4)协议事件

在 RSL 中，可以将一个协议事件直接看作是一个抽象数据，并在此基础上定义协议事件类型；必要时可分类定义多个协议事件类型(例如按输入事件和输出事件分类)，不同协议类型中的协议事件不可以有交叉。下例是 BGP4 协议的协议事件类型定义：

Type BGP_EVENT==BGP_START | BGP_STOP |
BGP_TRANSPORT_CONNECTION_OPEN | …

(5)协议变量

协议变量用以存储协议运行的历史数据、参数以及协议机制本身的设置。在 RSL 的命令式规范中，可以进行变量说明。对于归入协议变量一类的常值元素(例如重发报文间隔时间、同一报文最大重发次数)可以在 value 部分进行说明。

(6)协议操作和谓词

协议功能最终是通过一系列协议操作(行动)来实现的。操作是由输入事件驱动的，同时又受到条件的约束，这些条件是由谓词来表达的。

在 RSL 中，协议操作可以通过一系列的函数和操作来描述，谓词可以用布尔表达式、前置条件 pre、后置条件 post 来描述。下例表示接收到一个路由，当路由表中不含此路由时，将该路由加入路由表表尾；否则不改变路由表。

type ROUTERITEM，
ROUTERLIST=ROUTERITEM*
value Route:ROUTERITEM×ROUTERLIST→ROUTERLIST
axiom
Route (routeritem:ROUTERITEM,routerlist:ROUTERLIST) ≡ routerlist
pre routeritem ∈ elems(routerlist),
Route (routeritem:ROUTERITEM,routerlist:ROUTERLIST) ≡ routerlist~<routeritem>
pre routeritem ∉ elems(routerlist)

3 基于RSL描述的协议验证

协议验证是对协议本身的逻辑正确性进行校验的过程。验证能力的强弱很大程度上取决于描述时所使用的数学模型的优劣[6]。在 RSL 形式化描述的基础上，借鉴通信进程演算(CCS)的方法，运用 RSL 操作符进行运算，可以对协议性质进行验证。

3.1 RSL操作符的运算规则

为便于书写，在协议性质验证演算中将协议进程(操作)P()简写为 P。下文假设 a, b 为协议事件，P, Q, R 为简写的协议操作。

(1)内部选择操作符“∏”具有如下性质：

(a; P) ∏(b; Q)≡(b; Q) ∏(a; P)

(2)并行操作符“‖”具有如下性质：

P ‖ Q ≡ Q ‖ P
P ‖ (Q ‖ R)≡(P ‖ Q) ‖ R

3.2 同步会合事件及其运算

当 2 个协议实体进行同步通信时，一个协议实体的“收”和另一协议实体的“发”事件，或一个协议实体的“读”和另一协议实体的“写”事件就构成一对同步会合事件(CCS 中也称为协同事件)。

(1)对于同步会合事件，并行操作符有如下扩展规则：

假设

A≡(a1; A1) ∏(a2; A2)
B≡(b1; B1) ∏(b2; B2)

则：

A ‖ B≡(a1; A1 ‖ B)∏(a2; A2 ‖ B)∏(b1; A ‖ B1) ∏(b2; A ‖ B2) ∏(I; A1 ‖ B1) ∏(I; A2 ‖ B2)

其中，a*i* 和 b*i* 是同步会合事件；I= (a*i*,b*i*)为同步会合事件对。

进一步地，若

A≡(a1; A1)
B≡(b1; B1)

则上式蜕化为

A ‖ B≡(a1; A1 ‖ B) ∏ (b1; A ‖ B1) ∏(I; A1 ‖ B1)

(2)同步会合事件隐藏

并行操作扩展规则的使用会使公式演算冗长，为运算简便：1)在演算过程中，可以对同步会合事件 a*i* 进行限制隐藏，并约定在式尾以符号标记/a*i* 进行标注。2)如果对同步会合事

件 a*i* 进行了隐藏，则同时需对其对应的事件 b*i* 进行隐藏。但由它们形成的同步会合事件对 I 应当在式中予以保留。隐藏不会对协议性质的验证产生影响，其原因在于若将信道抽象为一个小的子系统，则同步事件对为该子系统的内部事件，成对隐藏不会对外部系统产生影响。

如果把事件 a*i* 或 b*i* 列入隐藏范围，那么前述扩展式中的(a*i*; A*i* ‖ B)或(b*i*; A ‖ B*i*)项可以消去，例如：

(a1; A1 ‖ B) ∏ (b1; A ‖ B1) ∏(I; A1 ‖ B1)

可以写为

(I; A1 ‖ B1) /a1,b1

对于同步会合事件对 I，有如下隐藏规则：

I; P≡P

a; I; P≡a; P

3.3 实例：停等协议的验证

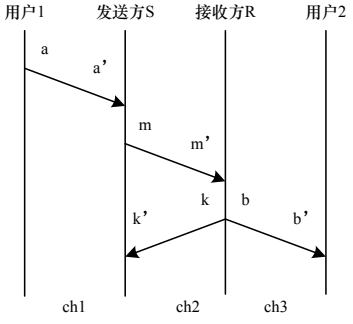
下面以简化的停止等待协议(未使用报文编号)为例，先使用 RSL 对其进行描述，在此基础上对协议的活动性和安全性进行验证。验证过程部分参考了文献[5]。

为了清晰起见，在协议描述中：

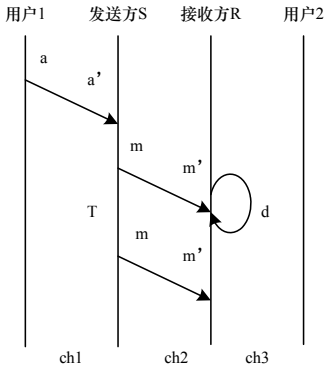
(1)从信道收/发数据抽象为单一事件。

(2)使用符号 a(发/写)和 a’(收/读)表示同步会合事件。

如图 1 所示，简化的停止等待协议由发送进程 S 和接收进程 R 并发而成。



(a)报文正常传输



(b)报文传输出错

图 1　简化的停止等待协议

对于发送协议实体 S：S 从用户 1 接收一个报文(事件 a’)之后，向 R 发送该报文(事件 m)并启动定时器。如果在给定时间内，S 未收到 R 的确认报文(事件 k’)，在超时事件 T 驱动下，S 重发该报文(事件 m)。此过程反复多次，直到 S 收到确认报文(事件 k’)为止。假设确认报文无差错不丢失，那么上述协议实体 S 的行为的 RSL 表达式为

S( )≡a’;m;S1

S1( )≡(T;m;S1) ∏ (k’;S)

对于接收协议实体 R：R 在接收到报文(事件 m’)之后，如果报文有错，R 丢弃报文(事件 d)；如果报文无错，R 首先向 S 发出确认报文(事件 k)，然后将报文交付给用户 2(事件 b)。在报文有错的情况下，协议实体 R 的行为的 RSL 表达式为

R( )≡m’;(d;R) ∏ (k ; b ; R)

整个的停止等待协议可以描述为协议进程 P：

P( )≡S ‖ R≡(a’;m;S1) ‖ (m’;(d;R) ∏ (k ; b ; R))≡a’;Q　　(1)

其中，

Q≡S1 ‖ ((d;R) ∏(k;b;R))≡

(T;d;W) ∏(d;T;W) ∏(I;b;P) ∏(I;a’;b;W)　　(2)

W≡(m;S1) ‖ R≡ (m;S1) ‖ (m’;(d;R) ∏ (k ; b ; R)) /m,m’ ≡ Q　　(3)

式(3)代入式(2)并隐藏 I，得

Q≡(T;d;Q) ∏(d;T;Q) ∏(b;P) ∏ (a’;b;Q)　　(4)

式(1)和式(4)表明，协议系统是活动的，不会发生死锁。式(4)的第 1 项、第 2 项描述了报文丢失重发的递归特性；第 3 项说明 Q 成功将报文交付给用户 2 之后返回到进程 P 的过程；第 4 项表明了协议的并发性，即在进程 R 未将报文交付给用户 2 之前，S 提前从用户 1 取得下一个待发送报文。

4　结束语

除了停止等待协议，还使用 RSL 描述了 TFTP 协议、RIP 协议、BGP 协议等多个协议。研究表明，扩充后的 RSL 比较适合用来进行协议的形式化描述。其描述能力和描述手段的灵活性在一定程度上强于另 2 种协议形式化描述语言 Lotos 和 Estelle。

在 RSL 描述的基础上，灵活运用操作符的运算规则、并行操作扩展规则和同步会合事件隐藏规则，可以对协议的一些性质，如活动性、安全性进行验证。这一步的工作目前还只能借助手工推演，验证工作的自动化完成将是下一步的目标。

参考文献

[1] 孙宇霖. 基于构造类别代数协议测试理论的研究[D]. 合肥: 中国科学技术大学, 2002.

[2] 孙　踊, 杨宏戟. 基于通信顺序进程的计算机网络通信协议形式化描述[J]. 北京大学学报: 自然科学版, 1997, 33(1): 110-121.

[3] Liu Nien-Chen, Liu Ming. Conformity Analysis for Communication Protocols[C]//Proc. of the ACM SIGCOMM Conference on Communications Architecture & Protocols. New York, USA: ACM Press, 1986: 216-226.

[4] 顾　翔, 邱建林, 蒋峥峥. RSL 在协议形式化描述中的应用研究[J]. 计算机应用, 2007, 27(9): 2236-2238.

[5] 龚正虎. 计算机网络协议工程[M]. 长沙: 国防科技大学出版社, 1993.

[6] 毛中全, 刘　楠, 顾纯祥, 等. 基于状态转移系统的安全协议形式模型[J]. 计算机工程, 2008, 34(13): 149-151.

编辑　张　帆