网络通信协议的属性文法规格说明*

房鼎益

(西北大学计算机科学系 西安 710069)

摘要 本文给出一个基于属性文法的网络通信协议的形式说明与自动生成的模型.首先引入了一个扩展的属性文法描述工具,讨论了用其描述协议的并行、同步、延时(时序)等特性的有关问题;然后研究了基于属性文法的网络通信协议自动生成环境及其核心算法——属性计算(即语义分析)算法.

关键词 属性文法,计算机网络,通信协议,规格说明.

中图法分类号 TP393

通信协议(Protocol)是网络技术的关键,它构成了网络操作系统的核心,同时也是分布式应用软件的实现基础.由于其固有的复杂性,网络通信协议系统的研制被公认为是高难度的领域.

多年的研究工作表明,提高网络通信协议开发效率的根本出路在于协议软件工程化,需要借助于合适的协议设计工具,特别是协议自动实现工具.另外,由于协议的实现必须符合一个外部标准,而该标准常疏于严格定义,因此所实现的协议的正确性、可靠性难以保证,使得协议软件工程化更为困难.解决该问题的主要途径就是使用形式说明技术FST(formal specification techniques)和协议自动生成技术(Automatic Protocol Generation).

目前所使用的FST主要有两类:①基于有穷状态机FSM(finite-status machines)模型,包括CSP(communicating sequential processes)[1]、时态逻辑方法[2]和Petri网方法.[3,4]这类模型的特点是协议的语义描述精确,验证方便.但纯粹的FSM模型只能描述简单或理想的协议,因为当协议较复杂或规模较大时,FSM的状态爆炸问题难于解决.另外,利用FSM模型难于完成协议的自动生成;②FST是基于高级语言的模型,如专用于协议描述的语言Estelle[5],LOTOS[6]和通用语言如Pascal,C,C++等.这类模型的特点是容易掌握,自动实现较方便,但对协议的语法和语义的分离不自然,且实现的协议质量(效率等)直接依赖于相应的语言编译程序.

属性文法AG(attributed grammar)[7,8]自1964年被Knuth提出后,已被广泛用于编译程序自动生成、图形编辑器设计和分布式系统形式开发等领域.我们从近年对AG的研究中[9,10]发现,如果将AG方法扩充,则能很方便地进行协议的形式定义.这种方法吸收了AG语义局部化特点,支持协议设计中的信息封装和抽象,允许对复杂协议进行划分和子协议的独立定义以及子协议的并行操作.此外,若在属性文法描述语言中吸收高级语言的控制流定义成分,则可以减化网络协议的AG定义.用AG定义的协议可方便地进行验证和分析,而且借助于AG分析器,能够自动生成高效的协议实现.Anderson采用属性文法描述工具,提出了一个用于协议自动生成的实时异步文法RTAG(real-time asynchronous grammar)模型[11],所生成的协议的效率已基本达到实用要求,但该模型缺乏协议分析和验证功能.

本文首先讨论用于协议定义的属性文法,然后给出一个基于属性文法的协议自动生成环境,最后研究了协议自动生成的关键算法,即属性计算算法.

1 协议的属性文法表示模型

协议形式描述的核心是描述对等实体层间实体间的信息交换.使用属性文法来定义协议,就是用AG的语法部分来描述协议实体之间的控制关系(即控制流),用AG语义来定义实体间的信息流和时序.

1.1 协议的AG定义

我们从3个方面对传统的AG进行扩充,以便于描述协议.

(1) 引入并行描述机制,以刻画进程之间的并行操作(即时间上的重叠);

---

* 本文研究得到国家自然科学基金资助.作者房鼎益,1959年生,博士生,副教授,主要研究领域为计算机网络与分布式系统,用户界面管理系统.

本文通讯联系人:房鼎益,西安 710069,西北大学计算机科学系

本文1996-08-02收到原稿,1997-05-22收到修改稿

(2) 引入特殊的终结符号，以描述协议的层间输入输出接口和时序关系；

(3) 引入条件产生式，以简化 AG 的语法定义，扩大其描述能力.

我们所使用的用于协议定义的属性文法 PSAG(protocol specification based on attributed grammar)是一个七元组：PSAG=(N,T,P,S,A,V,R)，其中 N 为非终结符集合，代表可分解的协议或子协议；T 为终结符集，代表与其他层协议的输入输出接口(事件)和定时器装置等；P 为产生式集，用于刻画协议的分解和子协议之间的控制关系；S 为起始符，表示协议工作的初始状态；A 为属性符号集；V 为属性值域，与每个语法符号相联系的属性值表示相应子协议所传递的信息；R 为属性计值规则集，用于描述子协议之间的信息交换和同步关系.

为了便于表达，以下将使用进程模型及术语对 PSAG 描述的协议动作进行说明. 每个文法符号〈X〉与一个进程定义(简称进程 X)相联系. 如产生式：〈X〉→〈Y〉〈Z〉定义了顺序调用进程 Y 和进程 Z 的进程 X.

每个文法符号〈X〉可以有一组属性值 $a1,a2,\ldots$，记为〈x〉.$a1$，〈x〉.$a2$，…. 由文献[10]的分析得出，符号 X 的属性构成了进程 X 的输入输出参数. 由 PSAG 所定义的协议就是与开始符号 S 相关的进程 S. 文献[10]详细讨论了通过属性分析构造子进程的方法.

1.2 协议的语义描述

(1) 并行性

按以上定义，产生式〈X〉→α 定义了一个由 α 中的子进程顺序执行而形成的复合进程 X，即 α 中每个子进程，当其左邻居完成后才启动执行. 为了支持进程的并行执行，即描述协议动作在执行时间上的重叠，在 PSAG 中引入了并行性描述机制，即用一对"//"括住一个产生式的右部，例如，在一个传输连接上可同时进行正常和加速数据的发送与接收，则用如下产生式表达该并行动作：

〈TRANS〉→//〈SEND_RD〉〈SEND_ED〉〈RECV_RD〉〈RECV_ED〉//

(2) 定时器装置

在 PSAG 中引入了一个代表定时器的特殊终结符〈TIMER〉，以实现协议的定时控制功能. 〈TIMER〉取整型值的属性 timeout，进程 TIMER 被启动后，便睡眠于一个 timeout 的时间间隔. 例如，产生式

〈X〉→〈TIMER〉α

{〈TIMER〉.timeout:=10;
〈other attribute rules〉}

所表达的语义是：当进程 X 启动时(即产生式〈X〉→〈TIMER〉α 被引用时)创建进程 TIMER，且属性〈TIMER〉.timeout 被赋于时间片10，并使进程 TIMER 睡眠10个时间片后完成，然后 α 中的子进程被创建，其余属性规则被执行.

(3) 条件产生式

在一条产生式中可以引入一个"前置条件"来决定该产生式能否被引用. "前置条件"是一个用"IF_THEN"括起来的逻辑表达式，被放在一条产生式右部的最前端. 条件产生式的一般形式记为〈x〉→COND α. 例如，产生式

〈x〉→IF 〈x〉.packet_expected=〈x〉.packet_arriv THEN
〈packarrival〉〈pack_to_upper〉
{attribute rules}

的语义是：当符号〈x〉的属性 packet_expected 与属性 packet_arriv 相等时，该产生式可被引用，语义树可被扩展，即进程〈x〉被激活，并创建 packarrival 和 pack_to_upper 进程，执行相应的属性规则.

为了高效地实现语法语义分析，PSAG 的支撑文法采用 LL(1)文法[12]，其属性依赖关系满足 L 属性化(即 L-AG)条件.[8]前置条件的引入，其一可以扩充 LL(1)的语法描述能力. 例如，包含产生式 $A\to b\alpha$ 和 $A\to b\beta$ 的文法不是 LL(1)的，但是如果加上两个不同时为真的前置条件：A→COND1 $b\alpha$，A→COND2 $b\beta$，则可以进行正确的 LL(1)分析；其二，可使 LL(1)具有描述上下文相关信息的能力，这是因为在前置条件中可以包含上下文相关的语义信息.

(4) 层间接口描述

协议的层间接口反映了该层协议实体(进程)的输入输出关系(称为输入输出事件).

我们分别用输入输出终结符来描述输入输出事件. 输入终结符刻画了信息的接收和缓冲，即用收到的信息对该符号的相应属性域赋值；输出终结符刻画了信息的发送和递交，即以该符号的相应属性值为参数，将其发送出去或递交给上层.

输入符号的语义用内部属性赋值规则(过程)来实现，而输出符号的语义则可通过调用外部事件处理子程序来实现. 这样的子程序与所描述的协议的运行环境(下层所提供的服务和上层的调用)有关，因此，应独立于 PSAG 而

实现.

2 基于PSAG的协议自动生成

如前所述,引入PSAG模型的目的是为了实现协议的自动生成.下面给出一个用于网络通信协议自动生成的集成环境EPAG(an environment for protocol auto-generation).

EPAG由协议的设计环境、生成环境和运行环境3部分构成.前两部分独立于协议的宿主系统,而第3部分与协议的宿主环境密切相关.EPAG的总体结构如图1所示,其主要部分和模块简介如下.

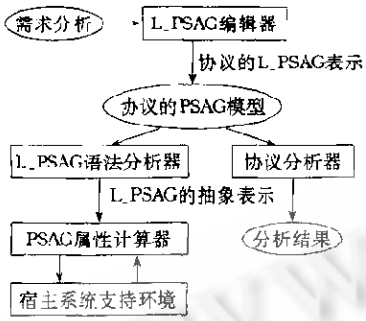

图1 基于属性文法的协议开发环境

·PSAG编辑器 利用我们所设计的PSAG描述语言L_PSAG对协议进行描述,可借助于普通的文本编辑器.

然而,由于协议的固有复杂性,虽然L_PSAG具有一种类似于C语言的编程风格,但是,用L_PSAG描述协议仍然是一件较困难的工作.如果借助于一个可视化的软件设计工具VDtool(visual design tool)[13],对网络通信协议进行描述,则可提高协议设计的直观性,简化设计难度.若再设计一个转换工具(VDtoAG),将协议的图示化描述变为L_PSAG描述,则可与协议的生成环境相配合,提高协议软件的开发效率.目前VDtool和VDtoAG还有待研制.

·L_PSAG编译器 对协议的L_PSAG表示进行语法、语义检查,判定AG类型,并将其翻译成独立于具体宿主系统的抽象表示形式(如C语言代码和文法分析表等).L_PSAG编译器的实现可借助于UNIX工具lex和Yacc等.

·协议分析器 分析协议描述的不一致性(Inconsistency)、不完整性(Incompleteness)、二义性和死锁(Dead-lock)等性质.例如,检查和发现所有不可达终结符、非终结符、无用产生式和外部子程序,保证所有的非终结符都能推导出一个终结符串,每个全局属性被使用前都有初值,等等.

·PSAG属性计算器(Evaluator) 即PSAG的语义分析程序,它以协议的PSAG抽象描述为基础,响应输入事件,产生输出事件,是协议在一个宿主系统上的具体实现.可以用一组接口例程和外部函数对一个宿主系统进行抽象表示.输入例程负责接收来自下层的分组(或帧),检查校验和,并提取有关参数,也负责处理来自上层的传输请求,二者均产生输入事件.输出例程既负责向上层递交数据帧或控制信息,也负责形成要发送的分组(或帧),并利用下层的功能调用将其发送出去.该计算器将在下节进一步讨论.

3 PSAG属性计算器

属性计算器完成AG的语义分析,即属性计算,是所描述的协议的具体实现,它用语法和语义联合制导的方法进行属性计算.由于我们采用L属性化的LL(1)文法,因此,语法和语义分析可同时一遍(自顶向下、从左到右)完成.通过对一棵部分属性化的分析树的非终结符叶结点应用一系列的产生式(扩展分析树),导出有关输入符的方法对输入事件进行响应.

3.1 概念与记号

为了便于描述PSAG属性计算算法,需要给出如下定义和记号.

设PSAG中支撑文法LLCG=(N,T,S,P),将LLCG中有限步左推导记为$\Rightarrow *$.

定义1. 若满足如下条件之一,则称如下产生式是可被应用的:

(1)$\langle X\rangle\rightarrow\langle Y\rangle\alpha$,是以$\langle X\rangle$为左部的唯一产生式,且$\langle y\rangle\in N$;

(2)$\langle X\rangle\rightarrow\alpha$,而$\langle X\rangle\rightarrow *\varepsilon$或$\langle X\rangle\Rightarrow *\langle O-Y\rangle$;

(3)$\langle X\rangle\rightarrow\langle TIMER\rangle\alpha$;

(4)$\langle X\rangle\rightarrow$COND α,而COND=true.

定义2. 设$S\subseteq N$,$\langle x\rangle\in N\cup T$,称$D=\{\langle y\rangle|\langle y\rangle\in S,\langle y\rangle\Rightarrow *\alpha\langle x\rangle\beta\}$为$\langle x\rangle$的可导出集,记为$D(S,\langle x\rangle)$;若$\langle y\rangle\in D(S,\langle x\rangle)$,则从$\langle y\rangle$左推导出$\langle x\rangle$时所引用的产生式序列称为$\langle x\rangle$的可导出产生式序列,记为$DPS(\langle y\rangle,\langle x\rangle)=(p_1)(p_2)\ldots(p_s)$;若存在一个$DPS(\langle y\rangle,\langle x\rangle)$,其中每个产生式都是可被应用的,则称该DPS为可被应用的导出序列,记为$ADPS(\langle y\rangle,\langle x\rangle)$.

定义3. 对分析树PT中的结点(即符号实例)$\langle x\rangle$,当满足下述条件之一时,称其为可扩展的:

(1)$\langle X\rangle$是PT的根结点$\langle S\rangle$;

(2)$\langle X\rangle$为可扩展结点的最左儿子结点;

(3)$\langle X\rangle$为其左兄弟结点已万成(扩展)的结点;

(4)$\langle X\rangle$为可扩展结点的各并行子结点(即并行产生式的右部各符号实例).

需要注意的是,一个可扩展的结点并不一定能立即被扩展,仅当由该结点对应的符号实例所导出的产生式可被应用时,它才能被立即扩展.

结点的可扩展性和产生式的可应用性共同决定了分析树的生成次序和PSAG的属性计算过程:对于所有可扩展结点,引用相应可应用产生式进行扩展,这又会导致新的可扩展结点的加入,直到不存在可扩展的结点时,属性计算终止.

分析树PT中可扩展结点集记为$Expandable(PT)$;PT中非终结符叶结点集记为$LeavN(PT)$,符号$\langle X\rangle$的属性集记为$A(\langle X\rangle)$.

3.2 PSAG属性计算算法

下述算法描述了PSAG属性计算的过程.这里仅给出算法的主要框架,算法由3部分组成,分别用于处理输入事件、超时和可扩展结点.

算法:

(0) 初始化

$PT=\{S\}$;$Expandable(PT)=\varnothing$;$LeavN(PT)=\varnothing$;

(1) 响应上层调用或下层中断事件$\langle I_X\rangle$

(1.1) 计算 $LeavN(PT)$;

(1.2) 计算 $D(LeavN(PT),\langle I\_X\rangle)$;

(1.3) 计算 ADPS$(\langle y_i\rangle,\langle I_X\rangle)=(p_1)(p_2)\ldots(p_i)$;

IF not_exist_this kind_of_y_i THEN report('invalid input event');

(1.4) FOR $j=1$ TO k

$APP(p_j)$;

(1.5) 更新 $Expandable(PT)$;

(2) 响应超时中断

(2.1) IF the_timeout_refers_to_the_timer_in_production P;$\langle x\rangle\rightarrow\langle$TIMER$\rangle\alpha$ and $\langle x\rangle\in Expandable(PT)$ THEN $APP(P)$;

(2.2) 更新 $Expandable(PT)$;

(3) 处理 $Expandable(PT)$

FOR all $\langle x\rangle\in Expandable(PT)$ DO

(3.1) CASE1 $\langle x\rangle\rightarrow$COND $\langle$TIMER$\rangle\ \alpha$ DO start ($\langle$TIMER$\rangle$);

CASE2 $\langle x\rangle\rightarrow$COND α DO IF COND=true THEN $APP(\langle x\rangle\rightarrow\alpha)$;

CASE3 $\langle x\rangle\rightarrow\langle O\_X\rangle$ DO CALL external_routine_$X(A(\langle x\rangle))$;

(3.2) 更新 $Expandable(PT)$;

UNTIL $Expandable(PT)=\varnothing$

其中函数$APP(P)$表示引用产生式P扩展分析树并进行相关属性计算,其结果可能会改变有关产生式的可应用性和有关结点的可扩展性,导致新的可扩展结点的加入;此外,$Expandable(PT)$中的结点被扩展后应将其删掉.因此,在实现时是将"更新$Expandable(T)$"与$APP(P)$合在一起的.

4 结束语

本文所给出的基于属性文法的协议规格说明模型具有以下特点:

(1) 描述模型高度抽象.利用子AG描述子协议的输入输出关系,并将协议的控制流和数据流自然分离,而将其实现屏蔽起来(留给编译器和属性计算器完成),这样做有益于对协议的理解和正确描述以及协议的增量式开发.

(2) 支持复杂协议的划分和子协议的并行操作.复杂协议可逐步分解成一系列子协议,子协议之间通过属性传递信息,协调运行.

(3) 高效的实现算法.现存多种高效的LL(1)分析器和L-AG属性计算器,且其效率与协议的规模和复杂程度无关,因而与其他模型相比,该模型更便于描述和实现复杂协议.

虽然本文给出的模型和设计环境主要用于网络通讯协议的形式描述、分析验证和自动生成,但它也完全可用于其

他普通软件系统设计,特别适用于分布式系统的设计.

参考文献

1 Hoare C A R. Communicating sequential processes. Englewood Cliffs, NJ: Prentice-Hall, 1985

2 Manna Z, Wolper P. Synthesis of communicating processes from temporal logic specification. ACM Transactions Programming Language Systems, 1984,6:137～144

3 Billingto J *et al*. Automated protocol verification. In: Diaz M ed. Protocol Specification, Testing and Verification V. Amsterdam, 1986. 59～70

4 Billington J *et al*. PROTEAN: a high-level Petri-net tool for the specification and verification of communication protocols. IEEE Transactions on Software Engineering, 1988,14(3):301～316

5 Estelle. A formal description technique based on an extended state transition model. Report, ISO/TC97/SC21/WG16-1 DP9074, 1986

6 Lotos. A formal description technique. Report, ISO/TC97/SC21/WG16-1 DP8807, 1986

7 Knuth D E. Semantics of context-free languages. Mathematical System Theory, 1968,2(2):127～145

8 Bochmann G V. Semantic evaluation from left to right. Communications of ACM, 1976,19(2):55～62

9 房鼎益,顾元祥. 二级属性文法的设计. 西北大学学报,1990,20(4):21～28
(Fang Ding-yi, Gu Yuan-xiang. Design of two-level attribute grammar. Journal of Northwest University, 1990,20(4):21～28)

10 房鼎益等. 二级属性文法与二级属性计算器. 计算机应用与软件,1991,8(6):9～17
(Fang Ding-yi *et al*. Two-level attribute grammar and its evaluator. Computer Application and Software, 1991,8(6):9～17)

11 Anderson D P. Automatic protocol implementation with RTAG. IEEE Transactions on Software Engineering, 1988,14(3):291～230

12 Aho A, Ullman J. The theory of parsing, translation and compiling. Parsing: Prentice-Hall, 1975

13 Keng Ng, Jeff Kramer, Jeff Magee. A CASE tool for software architecture design. Journal of Automated Software Engineering, August 1996,3(3/4):261～284

Computer Network Protocol Specification with Attribute Grammar

FANG Ding-yi

(*Department of Computer Science Northwest University Xi'an* 710069)

Abstract A model for computer network protocol specification and automatic generation is described in this paper. An extended attribute grammar to specify a protocol is introduced, on which the representations about parallelism, synchronization and timing in protocols are emphasized. Then, a protocol development environment is illustrated, and some referring issues, especially a semantics evaluation algorithm, are investigated.

Key words Attribute grammar, computer networks, communication protocol, specification.