协议形式化开发环境的规范语言*

罗铁庚 陈火旺 齐治昌 龚正虎

(长沙工学院计算机系 长沙 410073)

摘要 LOTOS(language of temporal ordering specification)是一种基于进程代数CCS的协议规范语言,面向协议验证,但它不能描述协议的某些性质.本文提出了一种LOTOS的扩充语言ELOTOS(extended LOTOS),它在LOTOS的基础上引入了异步通讯机制、时间描述、事件发生的随机性描述.

关键词 协议规范语言,进程代数,LOTOS,实时系统,概率规范.

中图法分类号 TP311

一个协议提供网络系统中不同部分之间相互作用的规则.每条规则可能很简单,但由于进程是并发的、非终止的,很多规则组合成的协议非常复杂.因而,设计和实现正确的通讯协议是一项富有挑战性的工作.采用形式化技术描述的协议规范精确、无二义性,能避免协议不同实现的不兼容性,并能验证协议的正确性,测试协议实现的一致性(Conformance).目前采用的形式化描述技术一般基于进程代数(Process Algebras)、时态逻辑(Temporal Logic)、自动机或者Petri网.已有3种协议规范语言被采纳为国际标准:SDL,LOTOS,Estelle.

(1) SDL(specification and description language)由CCITT(international consultative committee for telephone and telegraph, 1993-03更名为WTSC(world telecommunication standardization conference))开发于1976年,在1982、1985、1987、1992[5]先后进行了修改,由非形式化版本进化到了形式化版本.它包括两种等价的语法形式:图形形式和语言形式,已有一些工具能将两种语法形式相互转换,并能将SDL自动转换到C,chill,Ada,Coral,Pascal,PLM.SDL是一种接近于实现的规范语言,对于程序员来说比较易读.

(2) Estelle由ISO(international standards organization)于1989年开发.它基于扩充的有穷状态机(EFSM)是一种面向实现、专为协议描述而设计的类Pascal的形式化描述语言.它对并发、不确定性、超时、异步通讯、状态转换有较强的表达能力,但对递归、共享通道(广播通道)、同步通讯、协议性质(如不变性)的表示缺少有力的手段.用Estelle描述的协议易于提取FSM模型和Petri网模型,但不容易转变成时态逻辑和进程代数CCS模型.

* 本文研究得到国家自然科学基金和国家863高科技项目基金资助.作者罗铁庚,1970年生,博士生,主要研究领域为协议验证与测试,CASE.陈火旺,1936年生,教授,博士导师,主要研究领域为计算机科学理论,软件工程.齐治昌,1942年生,教授,博士导师,主要研究领域为软件工程,CASE.龚正虎,1945年生,教授,博士导师,主要研究领域为计算机网络,协议工程.

本文通讯联系人:罗铁庚,长沙 410073,长沙工学院计算机系

本文1996-12-17收到修改稿

(3) LOTOS(language of temporal ordering specification)[2]由 ISO 于 1989 年开发.它的基本想法是通过定义事件的时序关系描述系统.LOTOS 包括两个部分:第 1 部分基于进程代数 CCS,描述进程的行为和相互作用;第 2 部分基于抽象数据类型语言 ACT ONE,描述数据结构和值表达式.LOTOS 是一种适应协议工程、分布处理和并行处理要求而产生的语言.它面向协议验证,抽象级别比较高.用它描述的协议很容易转换成 CCS 模型、时态逻辑模型、FSM 模型、Petri 网模型.

因此,为了能有效地验证协议的正确性,测试协议的一致性,我们选择 LOTOS 作为协议形式化开发环境的协议规范语言.然而,LOTOS 不能直接描述异步通讯,这使得 LOTOS 使用起来不太方便,也不易读;同时 LOTOS 不能描述具体时间、事件发生的随机性,使得协议无法描述事件延迟和系统性能.所以,很有必要在 LOTOS 的基础上扩充其描述能力.本文介绍了 LOTOS 的扩充版 ELOTOS(extended LOTOS),在第 1 节介绍 LOTOS 语言之后,我们在第 2~4 节分别介绍如何描述异步通讯、时间和随机事件发生,最后还给出了一个例子以解释 ELOTOS 的使用.

1 LOTOS

在 LOTOS 中,用进程来描述分布式系统,一个系统被看作多个相互作用的子进程组成的进程,这些子进程又是由多个子进程构成.这样,一个系统的 LOTOS 规范是一个进程定义的层次结构.在 LOTOS 中,进程定义的格式为:

```
Process process-id [gate-list] (parameter-list) : noexit
  :=(behavior-expression)
endproc
```

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

LOTOS 引入了门径(Gates)的概念.门径就是协同事件发生点或两个进程同步通讯点.门径的定义包括多个通讯变量的定义和输入、输出参数的定义.两个进程通过同一门径变量可进行"数值匹配"、"数值传递"、"数值生成"三种交互作用.

多个协议事件和行为表达式可以通过下列行为算子组合为新的行为表达式:

① 行动前缀(Action-prefix) ';',如:"$a;B$"表示执行事件 a 之后执行行为表达式 B;

② 选择(Choice) '[]',如:"$B_1[]B_2$"表示选择 B_1 或 B_2 执行;

③ 并行,如:"$B_1 \|\|\| B_2$"表示 B_1 和 B_2 独立并行,"$B_1 \| B_2$"表示 B_1 和 B_2 通过它们的所有事件并行,"$B_1 |[a_1,\cdots,a_n]| B_2$"表示 B_1 和 B_2 通过门径 $a_1,\cdots,a_n$ 并行;

④ 串行'≫',如:"$B_1 \gg B_2$"表示如果 B_1 成功结束,而不是由于死锁终止,则使能 B_2;

⑤ 作废'[>',如:"$B_1[>B_2$"表示执行 B_1 时如果 B_2 有事件发生则停止 B_1 的执行;

⑥ 隐藏'hide',如:"hide a in B"表示 B 中事件 a 对外不可见,可以没有环境的参与而发生;

⑦ 看守(Guard)'→',如:"[cond]→B"表示条件[cond]满足后方可执行 B;

⑧ 终止(Termination),如:exit, stop;

⑨ 内部事件'I'.

2 描述异步通讯

在LOTOS中，我们如果要描述异步通讯，就必须引入一个队列进程，这给用户带来了极大的不便，也与现实不太相符. 因而，我们在ELOTOS中引入异步通讯机制. 实际上，只要在门径中有一个无穷长队列作为数据缓冲区就可以实现所有的异步通讯. 因此，我们将门径分为同步门径和异步门径，并对LOTOS作如下扩充：

(1) 异步门径和同步门径的定义是一样的，但异步门径有一个无穷长的队列. 我们增加并行符号'|〈…〉|'表示两个进程之间的异步通讯，如："$B_1|\langle a_1,a_2\rangle|B_2$"表示 B_1 和 B_2 通过异步门径 a_1,a_2 进行异步通讯，发送进程不必等到接受进程已准备好，就可以直接将数据放到门径的无穷队列中.

(2) 我们用'??'和'!!'分别表示异步通讯的发送数据和接受数据.

事实上，我们能用同步通讯等价地描述出异步通讯：

If B 是一个并行表达式，B_1 和 B_2 是行为表达式，$g_1,\ldots,g_n$ 是在 B_1 中发送数据的门径名字的列表，$g_{n+1},\ldots,g_{n+m}$ 是在 B_2 中发送数据的门径名字的列表，$g_1\_par,\ldots,g_n\_par$，$g'_{n+1}\_par,\ldots,g'_{n+m}\_par$ 是门径 $g_1,\ldots,g_n$，$g'_{n+1},\ldots,g'_{n+m}$ 的变量和参数.

with $B=B_1|\langle g_1,\ldots,g_n,g_{n+1},\ldots,g_{n+m}\rangle|B_2$,

then 替换 B 用 B_1('??'/'?','!!'/'!')$|[g_1,\ldots,g_n,g_{n+1},\ldots,g_{n+m}]|P(init,\ldots,init)|[g'_1,\ldots,g'_n,g'_{n+1},\ldots,g'_{n+m}]|B_2[g_1/g'_1,\ldots,g_n/g'_n,g_{n+1}/g'_{n+1},\ldots,g_{n+m}/g'_{n+m}]$('??'/'?','!!'/'!')

where $P(g_1\_queue,\ldots,g_n\_queue,g'_{n+1}\_queue,\ldots,g'_{n+m}\_queue):=$

$(g_1;inqueue(g_1\_queue,g'_1\ g_1\_par)\ |||\ first(g_1\_queue);outqueue(g_1\_queue))$

$|||\ldots|||$

$(g_n;inqueue(g_n\_queue,g'_n\ g_n\_par)\ |||\ first(g_n\_queue);outqueue(g_n\_queue))$

$|||$

$(g'_{n+1};inqueue(g'_{n+1}\_queue,\ g_{n+1}g'_{n+1}\_par)\ |||\ first(g'_{n+1}\_queue);outqueue(g'_{n+1}\_queue))\ |||\ldots|||$

$(g'_{n+m};inqueue(g'_{n+m}\_queue,g_{n+m}g'_{n+m}\_par)\ |||\ first(g'_{n+m}\_queue);outqueue(g'_{n+m}\_queue))$

where

(*------events queue------*)

type　*events_queue is octetstring with*

sorts　*eventsqueue_sort*

opns　*init:→eventsqueue_sort*

inqueue:eventsqueue_sort, octetstring→eventsqueue_sort

outqueue:eventsqueue_sort→eventsqueue_sort

first:　eventsqueue_sort→octetstring

eqns　*forall x: eventsqueue_sort, a, b: octetstring*

ofsort eventsqueue_sort

outqueue(init)=init;

outqueue(inqueue(init,x),a),b)=init;

outqueue(inqueue(inqueue(x,a)),b)=inqueue(outqueue(inqueue(x,a)),b);

ofsort octetstring

first(init)=0;　　(*---empty queue---*)

first(inqueue(init,a))=a;

first(inqueue(inqueue(x,a),b)=first(inqueue(x,a));

endtype

3 时间描述

LOTOS只能描述事件的时序关系，而不能描述事件的具体发生时间. 一些已有的工作用事件的时间来描述系统，并不能表达全局时间约束. 因而，我们根据实际需要在ELOTOS

中引入了3种描述事件延迟的语句、两个超时语句和一个全局时间变量.

(1)"*delay_attr* $g\langle t_1,t_2\rangle$ in B"表示在表达式B中,时间区间$\langle t_1,t_2\rangle$赋给了门径g.只有当在g的行动的年龄在区间$\langle t_1,t_2\rangle$内时,B描述的进程才可以在g执行该行动.其中行动的年龄指行动被不间断地使能后的时间.值得注意的是:g可能是几个子进程同步的门径,这时,在g的行动的年龄从在g的最后一个进程准备就绪开始算起.例如:"*delay_attr* $g\langle 3,10\rangle$ in B"表示在表达式B中,门径g上被使能的事件将在时间区间$\langle 3,10\rangle$内发生.

(2)"*p_delay_attr* $g\langle t_1,t_2\rangle$ in B"是一种预同步语句,表示几个进程一旦在门径g达到一致后,马上将在g的同步提交.这样,g的行动一旦被使能后就不会因其它事件发生而退出同步.例如:"*p_delay_attr* $g\langle 3,10\rangle$ in B"表示在表达式B中,门径g上被使能的事件将马上被提交,并在时间区间$\langle 3,10\rangle$内发生.

(3)"*m_delay_attr* $g\langle t_1,t_2\rangle$ in B"是一种存储时间语句,表示在表达式B中,在门径g的交互能在总时间属于区间$\langle t_1,t_2\rangle$时发生.其中在门径g的行动的总年龄指行动被使能的所有时间区段的累计时间,使能时间可以不连续,可能是几段使能时间的总和.如:描述在分时处理系统中被分时服务的进程中的行动的延迟.这个语句可以用来表达公平性.例如:"*m_delay_attr* $g\langle 3,10\rangle$ in B"表示在表达式B中,门径g上被使能的事件在累计使能时间在时间区间$\langle 3,10\rangle$时发生.

(4)"*timeout_attr* $\langle g,t\rangle$"是超时语句,表示g被使能后,如果在小于t的时间内未发生,则超时事件(Timeout)发生.

(5)"*m_timeout_attr* $\langle g,t\rangle$"是超时语句,表示g被使能后,如果在小于t的累计时间内未发生,则超时事件发生.

(6) GT是一个表示全局时间的变量.

4 随机事件发生的概率描述

在第3节描述事件延迟的语句中,只给出了时间的闭区间,不能描述不同延迟时间的概率分布;同时LOTOS也不能描述非确定选择的选择概率(如信道出错的概率).因此,为了能很好地描述系统性能,我们参考已有工作[3]在ELOTOS中扩充了第3节的语句:

(1)"*delay_attr* $g\langle t_1,t_2,pdf(),priority,weight\rangle$ in B",其中

- g是一个门径;
- $[t_1,t_2](t_2>t_1)$描述门径上事件延迟的时间域;
- $pdf():[t_1,t_2]\rightarrow[0,\infty]$是事件延迟时间的概率密度函数,$\int_{t_1}^{t_2} pdf(\ )dx=1$,省缺值为$1/(t_2-t_1)$;
- *priority*表示优先级,用一种确定的方式解决因使能事件具有相同延迟而导致的冲突,省缺值为均值.
- *weight*是权值,表示选择概率,用于解决*priority*仍不能解决的冲突,省缺值为均值.
- B是行为表达式.

(2)"*p_delay_attr* $g\langle t_1,t_2,pdf(),priority,weight\rangle$ in B",各项意义同上.

(3) "*m_delay_attr* $g\langle t_1,t_2,pdf(),priority,weight\rangle$ in B"，各项意义同上.

下面用一个例子说明：

p_delay_attr out⟨3,15,,1,98⟩ in

delay_attr loss⟨0,0,,1,2⟩ in *CH*

where $CH:=inp;$ (*out*; *CH* [] *loss*; *CH*)

表示一个通道接受数据(Inp)后，信道不出错时(98%)，输出(Out)延迟为 3～15s，而出错(Loss)的概率为 2%.

5 例　子

AB 协议[4]是用来评价协议规范语言的著名例子. 为了使 AB 协议更接近于实际，我们假定通讯方法是异步的，发送者的操作系统是分时系统. 我们用 ELOTOS 来描述这个协议. (*…*)内为注释.

```
Specification AB_Protocol [Uap, Nap, Ackap] :
noexit
  library
  (* Data type nat and string from library *)
    boolean, octetstring
  endlib
  (*----------global type definitions---------*)
  type seq_type is
  sorts seq_sort
  opns into:→0
       inc: seq_sort→seq_sort
  eqns  forall x: seq_sort
        ofsort seq_sort
        init(x) = 0;
        inc( inc(x) ) = x;
  endtype
  (*-----Category of package------*)
  type Id_type is
  sort Id_sort
  opns  DATA:→1
        ACK:→2
  endtype
  (*-------Data string-------*)
  type Udata_type is octetstring with
  sorts Udata_sort
  opns empty: 0
  endtype
  type Ndata_type is Id_type, seq_type,
       Udata_type, boolean with
  sorts  Ndata_sort
  opns  encode_data: DATA, seq_sort,
              Udata_sort→Ndata_sort
    encode_ack: ACK, seq_sort→Ndata_sort
    decode:     Ndata_sort→Udata_sort
    Id_is:      Ndata_sort→Id_sort
    seq_is:     Ndata_sort→seq_sort
    recv_ok:    Ndata_sort, seq_sort→boolean
    send_ok:    Ndata_sort, seq_sort→boolean
  eqns forall x: Ndata_sort, y: seq_sort,
              z: Id_sort
  ofsort Ndata_sort
    Id_is(x)=DATA⇒encode_data
      (Id_is(x),seq_is(x),decode(x))=x;
    Id_is(x)=ACK⇒encode_ack (Id_is(x),
      seq_is(x))=x;
  ofsort boolean
    (Id_is(x)=DATA) and (seq_is(x)=z)
      ⇒recv_ok(x,z)=true;
    (Id_is(x)=ACK) and (seq_is(x)=z)
      ⇒send_ok(x,z)=true;
  endtype
  (*------definition of gates------*)
  Uap:      User_data:      Udata_sort
  Nap:      Network_data:   Ndata_sort
  Ackap:    Ack_data:       Ndata_sort
  behaviour
  delay_attr          Uap⟨0,∞,exp(10),,⟩
  (* pdf(x)=exp(10)=1/ln10 e^(-x/ln10), and
  ∫0∞ pdf(x)dx=1, ∫0∞ x·pdf(x)dx=10*)
  m_delay_attr     Nap⟨0,∞,exp(5),,⟩
  p_delay_attr     Ackap⟨0,∞, exp(3),,⟩
  timeout_attr (Nap, 50)
  in
  (     p_sender[Uap, Nap, Ackap]
        (empty(A_buf), init(Send_seq))
    |⟨Nap⟩|  CH [Nap, Ackap])
  ||| (receiver[ Uap, Nap, Ackap]
        (empty(B_buf), init(Recv_seq))
  |⟨Ackap⟩|CH [Nap, Ackap])
  where
  (*------Process definitions------*)
  (* p_sender:sender process *)
  process p_sender [Uap, Nap, Ackap] (A_buf:
    Udata_sort, Send_seq: seq_sort)
```

```
  ;noexit:=
    (* ---- Inquire user---- *)
Uap          ? A_buf              (* data *)
  (*-----Send data----- *)
Nap !!Send_buf = encode_data (DATA, Send_seq,
    A_buf)
q_sender[Uap,Nap,Ackap]( Send_buf )
endproc (* p_sender *)
  (* --q_sender:waiting for acknowledge- *)
process q_sender[Uap, Nap, Ackap]
  ( Send_buf:Ndata_sort ):noexit:=
  (*----- Receive acknowledge----- *)
  ( Ackap ?? Recv_buf: Ndata_sort
    [ Id_is(Recv_buf) = ACK ];
[ send_ok(Recv_buf, Send_seq) ]→
    p_sender[ Uap, Nap]( empty(A_buf),
        inc(send_seq) ) )
[]( I;                          (* --timeout-- *)
    (* ---Send again if time-out--- *)
    Nap          !! Send_buf);
q_sender[Uap, Nap, Ackap] ( Send_buf )
endproc ( * q_sender * )
  (* ---receiver: receiver process--- *)
process receiver [Uap, Nap, Ackap](B_buf:
Udata_sort, Recv_seq: seq_sort): noexit :=
  (*-----receive data----- *)
Nap     ?? Recv_buf: Ndata_sort
      [ Id_is(recv_buf) = DATA ]
  (*-----send acknowledge----- *)
( [ recv_ok(Recv_buf, Recv_seq) ]→
  Ackap !! Ack_buf = encode_data(ACK,
    Recv_seq );
      (*-----tell user----- *)
Uap     ! B_buf = decode(Recv_buf);
    receiver[Uap,Nap,Ackap] ( empty(B_buf),
      inc(Recv_seq) )
[]( * -----receive again----- * )
([not recv_ok(Recv_buf,Recv_seq)]→
            receiver[Uap, Nap, Ackap]
      (empty(B_buf), Recv_seq))
endproc                 (* receiver *)
  (* -----CH: chanel process----- *)
process CH[Nap, Ackap]:noexit:=
  hide ok,err in
      delay_attr ok(0, 0, , ,99)
      delay_attr err(0, 0, , ,1)
  in
    Nap;
          (* ---channel is ok--- *)
    (  ok; CH[Nap, Ackap]
        (* ---chanel is error--- *)
    [] err; CH[Nap, Ackap] )
  []
    Ackap;
        (* ---channel is ok--- *)
    (  ok; CH[Nap, Ackap]
        (* ---chanel is error--- *)
      [] err; CH[Nap, Ackap] )
endproc                 (* --CH-- *)
endspec                 (* --AB_Protocol-- *)
```

在这个例子中,*Uap* 是用户和计算机交互的门径,*Nap* 是发送者向接收者发送数据的门径,*Ackap* 是接收者向发送者发送应答的门径.*Uap* 是同步门径,它的延迟时间在区间〈0,∞〉,均值是 10,$pdf(x)=\frac{1}{\ln 10}e^{\frac{-x}{\ln 10}}$;*Nap* 是异步门径,它的延迟时间在区间〈0,∞〉,均值是 5,$pdf(x)=\frac{1}{\ln 5}e^{\frac{-x}{\ln 5}}$,由于发送方的操作系统是分时系统,我们用"*m_delay_attr Nap*〈0,∞,*exp*(5),,〉"描述 *Nap* 的时间属性;*Ackap* 也是异步门径,它的延迟时间在区间〈0,∞〉,均值是 3,$pdf(x)=\frac{1}{\ln 3}e^{\frac{-x}{\ln 3}}$."*p_delay_attr Ackap*〈0,∞,*exp*(3),,〉"表示一旦接收者收到数据包,马上发出应答;"*delay_attr ok*〈0,0,,,99〉, *delay_attr err*〈0,0,,,1〉"表示信道出错的概率是 1%.

6 小 结

在 ELOTOS 中,我们引入了异步通讯机制、时间描述、事件发生的随机性描述. ELOTOS 能用一种更直接、更方便的方式描述协议. 在后面的工作中,我们将研究确保用 ELOTOS 描述的协议的正确性的形式化方法[5,6](包括验证和测试).

致谢 王兵山教授和王戟博士给本文提出很多有益的建议,在此表示感谢!

参考文献

1 Frgemand O, Olsen A. Introduction to SDL-92. Computer Networks and ISDN Systems, 1994, **26**: 1143～1167.

2 Brinksma. Information processing systems-open systems interconnection-LOTOS-a formal description technique based on the temporal ordering of observational behaviour. International Standard, ISO 8807.

3 罗铁庚，陈火旺，齐治昌. 分布式实时系统的概率规范和证明形式化. 计算机科学，1995，**22**(6).

4 龚正虎. 计算机网络协议工程. 长沙：国防科技大学出版社，1993，12.

5 Alur R, Courcoubetis C, Dill D. Model-checking for probabilistic real-time systems. Proc. 18th Int. Coll. on Automata, Language and Programming, Madrid, Spain, July 1991, LNCS 510.

6 Hansson H A. Modeling real-time and reliability. In: Vytopil J ed. Formal Technique in Real-Time and Fault-Tolerant System, 1993.

A SPECIFICATION LANGUAGE FOR FORMAL DEVELOPMENT ENVIRONMENT OF PRACTICAL PROTOCOLS

LUO Tiegeng　CHEN Huowang　QI Zhichang　GONG Zhenghu

(*Department of Computer Science　Changsha Institute of Technology　Changsha* 410073)

Abstract　LOTOS(language of temporal ordering specification) is a protocol specification language based on process algebra CCS. It is geared to protocol verification, but it is not powerful enough for describing some properties of practical protocols. This paper introduces a language ELOTOS(extended LOTOS), with the power of describing asynchronous communication, time, and stochastic event occurring.

Key words　Protocol specification language, process algebra, LOTOS, real-time system, probabilistic specification.

Class number　TP311