分布实现完全LOTOS规范的转换方法*

谢 冰 陈火旺 王兵山

(长沙工学院计算机科学系 长沙 410073)

摘要 基于LOTOS规范语言,文章从系统功能规范出发,结合实际系统的分布特性,推导出符合实际系统结构的模块化规范的转换方法.用标注的完全LOTOS语言规范表达复杂的系统分布特性,研究了使用广播通信方式进行协同的、直接处理多模块划分的规范分解算法.

关键词 分布式系统,进程代数,LOTOS规范语言,规范分解.

中图法分类号 TP393

系统功能规范是与实现细节无关的.在分布式系统中实现时,设计人员需要设计出各个模块的功能规范.这些模块执行其上所分担的系统功能,通过模块间的通信,协同实现系统功能规范.本文研究了一种设计算法,使得设计人员在确定目标系统的分布特性后,即可直接推导出模块化规范.这种规范转换方法可表示为"系统功能规范+实际系统分布特性⇒系统模块化规范",被称为规范分解方法.相应于分布系统的层次性,在子模块规范基础上可以分解出更下一层的分布模块规范.重复使用规范分解方法,最终可得到完全符合目标系统结构的模块规范,从而完成系统的结构设计过程.

LOTOS[1,2]是ISO组织为了描述开放的分布式系统而制定的一种规范语言.其基本点在于,一个系统可通过从外部环境观察到的交互操作时序关系来定义.对LOTOS规范的分布实现问题已有不少的研究结果[3,4],但都针对不包含数据部分的基本LOTOS规范、研究行为表达式的分解方法,并且基本都是研究一分为二的分解方法.本文则针对完全LOTOS语言规范,研究了复杂的系统分布特性及其抽象表征,并给出了使用广播通信方式进行协同的多模块直接分解转换算法.

1 系统的分布实现模型

分布系统由分布的处理模块组成.每个模块都能独立处理其自身行为,模块间通过一定的通信结构进行协同.本文的研究基于如下分布系统抽象模型:(1) 系统支持广播通信方式;(2) 需要协同的模块之间都有直达的通信信道;(3) 通信信道是可靠的,即具有保持消息顺序、无丢失、无拥塞等性质.

定义 1.1. 分布系统DS是一个二元组,$DS=(M,BG)$,其中

- $M=\{m_1,\ldots,m_n\}$是系统的分布模块集;
- BG是模块间通信门径集,对任意的$g_{ij\ldots k}\in BG$,定义为$g=\{m_i,m_j,\ldots,m_k\}$.

本文使用$sync_{ij\ldots k}$作为模块$m_i,m_j,\ldots,m_k$间协同门径的特定命名.设模块m_i上的行为用进程P_i表示,则可用模块进程集合$P=\{P_1,\ldots,P_n\}$表达模块集合M.对定义1.1中的门径集合BG,若$S_1,S_2\subseteq P$,则$SG_{S_1,S_2}=\{sync_{ij\ldots k}|$有$t_1,t_2\in\{i,j,\ldots,k\}$,满足$P_{t_1}\in S_1$且$P_{t_2}\in S_2\}$表示$S_1$中模块与$S_2$中模块间的所有通信门径集合.

定义 1.2. 分布系统规范是所有组件通过内部通信协同的规范,其中的内部通信行为对外界隐藏.一个分布系统DS的模块化规范$SPEC_{DS}$定义如下

---

* 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**谢冰**,1970年生,博士,主要研究领域为分布式系统,软件工程.**陈火旺**,1936年生,教授,博士生导师,中国工程院院士,主要研究领域为软件生产自动化,计算机科学理论.**王兵山**,1938年生,教授,主要研究领域为计算机科学理论,软件.

本文通讯联系人:谢冰,北京 100871,北京大学计算机科学与技术系CASE室

本文1998-01-16收到原稿,1998-07-08收到修改稿

$SPEC_{DS}$=hide BG in (P_1|[$SG_{\{P_1,\{P_2,\ldots,P_n\}\}}$]|($P_2$|[$SG_{\{P_2,\{P_3,\ldots,P_n\}\}}$]|($P_3$|...|[$SG_{\{P_{n-2},\{P_{n-1},P_n\}\}}$]|$P_n$)...)).

2　系统的分布特性及标注规范

完全 LOTOS 语言描述的系统中包含行为部分和数据部分.对应于系统的分布实现,3 类基本的规范元素(即数据、数据操作和行为)都可能分布在不同模块上.我们在这些规范元素上均标注相应的分布模块,形成了标注规范,以此表达系统的实际分布特性.

本文以一个简化的银行自动取款机(ATM 机)作为例子.ATM 机有下述门径:In-card 是卡片插入口,从卡上读出用户帐号;In 是用于接收用户口令和取款数目的键盘;Out 向用户付款;Display 是显示器输出;Com 是通信口,用于同银行的服务器通信.规范中的 money-box 是 ATM 机中储存的货币数.ATM 机分为 3 个模块:用户接口部分(user-part)为 P_1,通信机(communicator)为 P_2,钱匣(box)为 P_3.反映实际系统分布特性的标注规范如下:

```
Proc ATM[In-card(P1),In(P1),Out(P3),Display(P1),Com(P2)](money-box(P3):Money-type):noexit:=
  In card(P1)? Account-no;In(P1)! GET_PASSWORD? password;Com(P2)! Account-no! password;
   (Com(P2)! PASS-OK;
     (In(P1)! GET_MONEY? m:Money;Com(P2)! GET_MONEY! m;
       (Com(P2)! OK;Out(P3)! m;(Display(P1)! SUCCESS|||Com(P2)! SUCCESS )>>>
         ATM[In-card(P1),In(P1),Out(P3),Display(P1),Com(P2)](new-money(P3)=Dec_Money(P3)(money-box,m))
       []Com(P2)! NOT_ENOUGH;Display(P1)! ERROR;
          ATM [In-card(P1),In(P1),Out(P3),Display(P1),Com(P2)](money-box(P3)))
     []Com(P2)! PASS-INVALID;ATM[In-card(P1),In(P1),Out(P3),Display(P1),Com(P2)](money-box(P3))
)
endproc
```

系统分布特性的另一方面是数据变元的相关模块集,可以从标注规范中分析得到,本文用 $ref(x)$ 表示.ATM 机中数据变元的相关模块集合如下:$ref(money\text{-}box)=\{P_3\}$,$ref(m)=\{P_1,P_2,P_3\}$,$ref(money\text{-}box)=\{P_3\}$,$ref(Account\text{-}no)=\{P_1,P_2\}$,$ref(password)=\{P_1,P_2\}$.

3　规范分解转换算法

分布模块是自主运行的,模块间的行为是并发的.因此,分布的模块间必须通过通信协调来实现系统.对系统行为间的顺序关系,若前后行为分布在不同模块上,需要加入同步事件以保持其间的顺序关系.另一方面,分布的模块间可能具有数据相关性.需要通过模块间的传值通信协同来保证共享数据变元的全局实时一致性.可见,系统内部有纯同步通信和传值通信两种协同.分解转换算法研究使用通信进行协同的方法.

3.1　系统内部协同事件

本文研究包含并行行为表达式的规范分解,需要区分可能并发出现的协同事件,可以在协同事件中加入规范结构信息来命名不同的同步事件.

结构信息是规范结构的表达.有如下几种形式:(1) 对于原规范进程 P,分布在模块 P_k 中的相应进程中使用的内部协同事件均应带有"! P"形式的结构信息;(2) 对选择行为 $B_1[]\ldots[]B_n$,其第 i 个分支 B_i 所对应的各个模块上的相应行为 T_kB_i 中的内部协同事件均应带有"! c_i"形式的结构信息;(3) 对并行行为 $B_1|[\ldots]|B_n$,其第 i 个分支 B_i 所对应的各个模块上的相应行为 T_kB_i 中的内部协同事件均应带有"! p_i"形式的结构信息.

按 LOTOS 语言的语法、语义定义,我们将系统内部协同事件定义为 $sync_{i,\ldots,k}d_1\ldots d_n$ 形式的行为,其中 $sync_{i,\ldots,k}$ 是系统中的内部通信信道,$d_i(1\leqslant i\leqslant n)$ 为结构信息或传值门径操作.

3.2　分解转换算法

规范分解转换算法根据规范元素的分布划分,研究模块规范的产生.我们按操作分解、原子行为分解、行为表达式分解和进程引用、定义分解 4 个层次分别研究规范分解转换算法.在每个层次中,针对所有可能的分布情况均给出转换规则.限于篇幅,本文主要讨论数据操作分解与行为表达式分解.

对任意模块 P_i，T_i："标注规范⇒P_i 模块规范"为其转换函数. 由于分解中产生的内部协同事件需要使用规范结构信息定义，顺序行为分解又与后继行为的分布模块有关，所以可将 T_i 函数定义为"标注规范×结构信息×后继模块信息⇒P_i 模块规范". 在后文的转换算法中明确定义了结构信息及后继模块信息的产生与使用.

在本文的算法中，为表达方便，设对任给的 i，均有 $sync_i d=\varepsilon$，ε 为空串.

3.2.1 数据操作分解算法

在原规范中，一个数据操作的自变元可能是一个分布在其他模块上执行的操作结果，需要首先计算该值，并传送结果. 另一方面，一个操作的值变元可能是多个模块的共享变元，该操作执行完毕后，应该扩散这个值变元的实时值. 由此，一个操作 t 的分布实现包括 3 个顺序执行部分：前向行为 $T_i\_f(t)$ 产生 t 操作执行前的协同；行为变换 $T_i\_m(t)$ 转换出 t 操作分解后的形式定义；后向行为 $T_i\_b(t)$ 则产生 t 操作执行后的相应协同行为. 在数据操作分解中，通过模块间的传值通信协同，确保共享数据变元的实时一致性.

保持多模块间共享的数据变元在系统中全局映像一致的机制可以简述为：在每个共享变元 x 的相关模块上均有一个 x 的局部映像. 模块上引用 x 的局部映像，而在 x 值更改时，使所有相关模块同步更新其上的映像值. 通过广播方式进行同步传送，若 $ref(x)$ 为 $\{P_i,P_j,\ldots,P_k\}$，则广播门径设定为 $sync_{ij\ldots k}$. 下文中使用 $sync_{ref(x)}$ 表示该门径.

不妨设 t 项分布在模块 P_i 中，规范分解的结构信息为 d_x，则对任意的模块 P_k，有

$$T_k(t,i,d_x):=T_k\_f(t,i,d_x);T_k\_m(t,i,d_x);T_k\_b(t,i,d_x),$$

其中 i 为 t 项分布的模块号，";"为分割符，表达行为间的顺序关系.

区分 t 项的不同情况，可以给出转换算法如下：

(1) 操作本身的变换

$$T_k\_m(x,i,d_x):=\begin{cases}"x" & \text{if } k=i\\ \varepsilon & \text{else}\end{cases}$$

$$T_k\_m(x=opn(P_i)(t_1,\ldots,t_n),i,d_x):=\begin{cases}x=opn(T_i\_m(t_1,i,d_x),\ldots,T_i\_m(t_n,i,d_x)) & \text{if}=i\\ \varepsilon & \text{else}\end{cases}$$

$$T_k\_m(x=opn(P_j)(t_1,\ldots,t_n),i,d_x):=\begin{cases}"x" & \text{if } k=i\\ \varepsilon & \text{else}\end{cases}\quad(j\neq i)$$

(2) 操作执行前的协同行为

$$T_k\_f(x,i,d_x):=\varepsilon,$$

$$T_k\_f(x=opn(P_i)(t_1,\ldots,t_n),i,d_x):=T_k\_f(t_1,i,d_x);\ldots;T_k\_f(t_n,i,d_x)$$

$$T_k\_f(x=opn(P_j)(t_1,\ldots,t_n),i,d_x):=\begin{cases}T_k\_f(t_1,j,d_x);\ldots;T_k\_f(t_n,j,d_x);sync_{ref(x)}d_x?\ x; & \text{if } k\neq j,P_k\in ref(x)\\ T_k\_f(t_1,j,d_x);\ldots;T_k\_f(t_n,j,d_x);sync_{ref(x)}d_x?\ x; & \text{if } k=j\\ T_k\_f(t_1,j,d_x);\ldots;T_k\_f(t_n,j,d_x); & \text{else}\end{cases}\quad(j\neq i)$$

(3) 操作执行后的协同行为

$$T_k\_b(x=opn(P_i)(t_1,\ldots,t_n),i,d_x):=\begin{cases}sync_{ref(x)}d_x!\ x; & \text{if } k=i\\ sync_{ref(x)}d_x?\ x; & \text{if } k\neq i,P_k\in ref(x)\\ \varepsilon & \text{else}\end{cases}$$

$$T_k\_b(x=opn(P_j)(t_1,\ldots,t_n),i,d_x):=\varepsilon\quad(j\neq i)$$

3.2.2 原子行为分解算法

原子行为分解中，行为根据其模块划分进行投影，行为中的数据操作可能需要使用操作分解算法确定同其他模块的协同. 对于行为中的变元赋值"?x"，还需考虑 x 的共享性，以决定是否进行 x 变元实时值的传送. 我们针对完全 LOTOS 语言规范中原子行为的不同组成形式，以及其中涉及的共享数据和数据操作的分布位置，分别给出相应的转换规则. 例如，对 $g(P_j)?\ x$ 的分解算法如下：

$$T_k(g(P_j)?\ x,d_x):=T_k\_f(g(P_j)?\ x,d_x);T_k\_m(g(P_j)?\ x,d_x);T_k\_b(g(P_j)?\ x,d_x)$$

$$T_k\_f(g(P_j)?\ x,d_x):=\varepsilon$$

$$T_k\_m(g(P_j)?\ x,d_x):=\begin{cases}g?\ x & \text{if } k=i\\ \varepsilon & \text{else}\end{cases}$$

$$T_k\_b(g(P_j)?\ x,d_x):=\begin{cases}sync_{ref(x)}d_x!\ x; & \text{if } k=i\\ sync_{ref(x)}d_x?\ x; & \text{if } k\neq i,P_k\in ref(x)\\ \varepsilon & \text{else}\end{cases}$$

3.2.3　行为表达式分解算法

行为表达式分解算法解决行为间时序关系的分布实现问题.我们对几种基本行为表达式分别进行讨论,其他行为算子的表达式分解可以依此类推.

(1) 行为前缀表达式分解方法.对于行为的顺序关系,可根据该行为以及其后可能执行的所有行为所属的模块,确定所需要的同步通信.

定义 3.1. 对一个行为表达式 B,它的可能行为集 $PosAct(B)$递归定义如下:

(1) 对行为前缀行为表达式 $B=a;B'$,$PosAct(B):=\{a\}$;

(2) 对选择行为表达式 $B=B_1[]B_2[]\ldots[]B_n$,$PosAct(B):=\bigcup_i^n(PosAct(B_i))$;

(3) 对并行行为表达式 $B=B_1|[G]|B_2$,$PosAct(B):=\bigcup_i^2(PosAct(B_i))$;

(4) 对进程实例 $proc$,且 $proc:=B'$,则 $PosAct(B):=PosAct(proc):=PosAct(B')$.

定义 3.2. 若 $Mod(a)$为行为 a 所属的划分模块,则行为 B 的可能行为模块集定义为

$$PAMS(B)=\{Mod(a)\mid a\in PosAct(B)\}.$$

根据上述定义,对行为前缀表达式,$B=a_{tag(a)};B'_{tag}$,任意模块 P_j 的分解转换算法为

$$T_j(B,d_x,pams)=T_j(a_{tag(a)},d_x);Tsync_j(P_i,d_x,PAMS(B'_{tag}));T_j(B'_{tag},d_x,pams).$$

其中 $Tsync(P_i,d_x,PAMS(B'))$为所需的内部协同行为,定义如下:

$$Tsync_i(P_i,d_x,PAMS(B')):=\text{“}sync_{ij\ldots k}d_x\text{”}(\{P_i,P_j,\ldots,P_k\}=PAMS(B')\cup\{P_i\}),$$

$$Tsync_j(P_i,d_x,PAMS(B')):=\begin{cases}\text{“}sync_{ij\ldots k}d_x\text{”} & \text{if } P_j\in PAMS(B')\\ \varepsilon & \text{else}\end{cases}\quad(j\neq i).$$

在上式中,$Tsync_j(P_i,d_x,PAMS(B'_{tag}))$是需要的模块间纯同步协同.而 a 行为的后向协同行为以及 $PosAct(B')$中行为的前向协同行为也起到了系统行为间的同步关系.因而,最终生成的纯同步事件可以由此化简.

(2) 顺序行为表达式分解方法.对顺序行为表达式 $B=B_1>>B_2$,分解算法为

$$T_i(B,d_x,pams)=T_i(B_1,d_x,PAMS(B_2))>>T_i(B_2,d_x,pams).$$

(3) 选择行为表达式分解方法.本文对于选择行为分解,限制为所有选项均为互斥出现.在这个前提下,若选择行为表达式为 $B=a_1;B_1[]a_2;B_2[]\ldots[]a_m;B_m$,则分解转换方法为

$$\begin{aligned}T_i(B,d_x,pams):=&(T_i\_f(a_1,d_x)\|\|T_i\_f(a_2,d_x)\|\|\ldots\|\|T_i\_f(a_m,d_x))>>(\\&T_i\_m(a_1,d_x);T_i\_b(a_1,d_x!\ c_1);Tsync_i(tag(a_1),d_x!\ c_1,PAMS(B_1));T_i(B_1,d_x!\ c_1,pams)\\&[]T_i\_m(a_2,d_x);T_i\_b(a_2,d_x!\ c_2);Tsync_i(tag(a_2),d_x!\ c_2,PAMS(B_2));T_i(B_2,d_x!\ c_2,pams)\\&\ldots[]T_i\_m(a_m,d_x);T_i\_b(a_m,d_x!\ c_m);Tsync_i(tag(a_m),d_x!\ c_m,PAMS(B_m));T_i(B_m,d_x!\ c_m,pams))\end{aligned}$$

(4) 并行行为表达式分解方法.设 $B=B_1|[g_{1tag(g_1)},\ldots,g_{ntag(g_n)}]|B_2$,则分解算法为

$$T_i(B,d_x,pams)=T_i(B_1,d_x!\ p_1,pams)|[PJ_i(g_{1tag(g_1)},\ldots,g_{ntag(g_n)})]|T_i(B_2,d_x!\ p_2,pams),$$

其中 PJ_i 是门径在模块 P_i 上的投影.

3.3　分解实例

对本文中 ATM 机的标注规范,分解的结果规范如下.显见,模块间通过协同通信、分布实现了原功能规范.

```
Proc ATM [In-card,In,Out,Display,Com] (money-box:Money-type):noexit:=
    hide sync12,sync123,sync23 in USER-PART[In-card,In,Display]( )|[sync12,sync123]|
        (COMMUNICATOR[Com]()|[sync23,sync123]| BOX [Out](money-box))
where
Proc USER-PART[In-card,In,Display]():noexit:=
  In-card? Account-no;sync12! Account-no;In! GET_PASSWORD? password;sync12! password;
    (sync12! c1;
    (In! GET_MONEY? m:Money;sync123! m;
    (sync123! c1! c1;(Display! SUCCESS||sync12! c1! c1! p2)>>USER-PART[In-card,In,Display]()
      []sync12! c1! c2;Display! ERROR;USER-PART[In-card,In,Display]())
    []sync12! c2;USER-PART[In-card,In,Display]())
    )
endproc
Proc COMMUNICATOR[Com]():noexit:=
sync12? Account-no;sync12? password;Com! Account-no! password;
  (Com! PASS-OK;sync12! c1;
    (sync123? m;Com! GET_MONEY! m;
```

```
    (Com! OK;sync23! c1! c1;sync123! c1! c1;(Com! SUCCESS;sync12! c1! c1! p2;)>>>
        COMMUNICATOR[Com]()
      []Com! NOT ENOUGH;sync12! c1! c2;COMMUNICATOR[Com]()))
    []Com! PASS-INVALID;sync12! c2;COMMUNICATOR[Com]()
    )
endproc
Proc BOX [Out](money-box:Money-type);noexit;=
  ((sync123? m;
    (sync23! c1! c1;Out! m;sync123! c1! c1;BOX[Out](new-money=Dec_Money(money-box,m))
      []BOX [Out](money-box)))
    []BOX [Out](money-box))
  endproc
endproc
```

4 结束语

本文提出了标注规范的方法以表达复杂的系统分布特性.在此基础上,研究了将完全LOTOS语言规范按实现环境的分布要求分解成多个协同组件的规范转换方法.该方法可能将一个原规范行为分解为一个内部协同的行为序列,因而分解是受一定条件限制的.在条件满足情况下,可以基于进程代数理论形式化地证明:通过内部通信协同,分解的结果规范与原始规范保持观察等价性,由此保证了系统设计的正确性.

与已有的工作相比,本文针对完全LOTOS语言规范,研究了数据分布的抽象表征以及分布实现问题.本文提出的方法直接处理多划分集的情况,使这种功能分解的方法更适用于实际问题.今后的工作主要是,针对数据项本身的分解进行研究.需从两个方面研究,一方面是基于代数规范语言,讨论数据项的划分子模块表达形式;另一方面则是在子模块规范中操作与在原始规范中操作的相关性,研究行为表达式与操作的分解转换算法.

参考文献

1 ISO. LOTOS, a formal description technique based on the temporal ordering of observational behavior. ISO IS8807, 1988

2 Bolognesi T, Brinksma E. Introduction to the ISO specification language LOTOS. Computer Network and ISDN Systems, 1987,14:25~59

3 Rom Langerak. Transformation and semantics for LOTOS [Ph.D. Thesis]. University of Twente, Netherlands, 1992

4 Maria Hultstrom. Structural decomposition. Protocol Specification, Testing and Verification XV, 1995. 201~216

Decomposition Transformation for the Distributed Implementation of Full LOTOS Specification

XIE Bing CHEN Huo-wang WANG Bing-shan

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

Abstract In this paper, based on the LOTOS specification language, the authors present the transformation which starts from the functional specification and the distributed properties of target system to the corresponding modular specification. The authors present the tagged specification to represent the complex distributed properties of the full LOTOS specifications and study the decomposition transformation of full LOTOS specification which using the broadcasting communication gates in the system's internal coordinating communications and directly decomposing the multiple subsets.

Key words Distributed system, process algebra, LOTOS specification, specification decomposition.