计算机支持的协同工作系统的时序逻辑模型*

王国意 史元春 徐光佑

(清华大学计算机科学与技术系 北京 100084)

摘要 为了使群体能够协同完成任务,CSCW(computer supported cooperative work)系统不仅要解决各种分布性、处理应用领域的特殊性,而且要提供面向用户的协作支持,从而使其行为异常复杂.然而对系统行为进行形式化的描述是构造软件系统的必经阶段.为了清晰地描述CSCW系统的行为,使其特定性质的验证成为可能,本文在时序逻辑的基础上,建立了CSCW系统行为的抽象描述模型.在此模型中,CSCW系统由分布运行实体和信息对象组成,系统的主要行为表现为用时序逻辑语言XYZ/E描述的实体间的交互.此模型可较好地对系统的分析和构造进行指导.

关键词 计算机支持的协同工作,时序逻辑,描述模型.

中图法分类号 TP311.1

对系统行为进行形式化的描述是构造软件系统的必经阶段[1],它不仅能方便开发人员之间的交流,而且可使系统各种性质的验证成为可能,保证系统实现的正确性.计算机支持的协同工作CSCW(computer supported cooperative work)的研究适应了信息化社会中人们工作方式的群体性、交互性、分布性和协作性特点,因此被认为是未来社会中广泛采用的技术.[2]为支持在空间上分布、时间上不同步的群体成员协同完成任务,CSCW系统不仅要解决分布性问题,处理应用领域的特殊性,而且要提供面向用户的协作支持.这使得CSCW系统非常复杂,其行为的描述比较困难.

目前,对CSCW系统的描述针对的都是群体成员开展协作的过程[3,4],这种过程并不能完全表示系统本身的行为,它只是系统行为的一种体现.对系统本身行为的描述大多数还是来自软件工程领域,然而现有的方法(如IDEF系列)并不能满足既有功能分解又有复杂交互的CSCW系统的描述要求.现有网络协议的描述方法[1]可处理分布运行实体间的交互,然而它没有涉及以用户为中心的复杂性.为了全面地描述CSCW系统的行为,本文基于时序逻辑建立了描述模型TLDM_CSCW.

逻辑系统作为程序正确性验证及公理语义的基础,已用于软硬件系统的描述和验证.[5~8]Manna和Pnueli基于一阶逻辑提出的线性时序逻辑系统,为描述系统的时序行为提供了一种形式化的途径,此外,它还适合对具有大量并发性和不确定性的系统进行描述和论证.CASE环境XYZ[9]的语义基础就是时序逻辑,其中的核心语言XYX/E允许在统一的框架下对系统行为进行不同层次的抽象描述,使系统的逐步求精设计成为可能,该环境中的验证工具还可以对这些描述进行验证.TLDM_CSCW模型以时序逻辑为基础,对CSCW系统进行了合理的抽象,它将系统表示为分布运行实体和信息对象的集合,并将系统的行为抽象成实体间的各种交互.通过对系统中各实体及对象的定义,并用XYZ/E语言对各种交互过程进行形式化的描述,此模型可以清晰地描述CSCW系统的行为,并使系统性质的验证成为可能,因此可以指导系统的分析与设计.

1 CSCW系统要素

为了建立CSCW系统的描述模型,应首先分析其组成要素,并对这些要素进行合理的抽象.CSCW系统的最终目的是支持群体的协作,下面就以群体利用系统进行协作的情行为背景,来考察系统的行为及所涉及到的要素.

1.1 群体协作与系统行为

用户群体通过CSCW系统进行协同工作的情形如图1所示.系统提供一个虚拟工作空间,消除时间与空间的限

---

* 本文研究得到国家自然科学基金和"211"学科建设项目资金资助.作者**王国意**,1970年生,博士生,主要研究领域为计算机支持的协同工作,多媒体技术.**史元春**,女,1967年生,在职博士生,副教授,主要研究领域为计算机支持的协同工作,多媒体信息模型.**徐光佑**,1940年生,教授,博士导师,主要研究领域为计算机支持的协同工作,计算机视觉,多媒体.

本文通讯联系人:史元春,北京 100084,清华大学计算机科学与技术系

本文1997-01-23收到原稿,1997-04-25收到修改稿

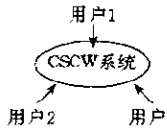

图1 用户通过CSCW系统的协作

制,将各用户联在一起,使他们可以彼此交换信息,并对共享对象进行协同操作.

从支持协作这个角度来看,CSCW 系统是一个不可分割的整体.群体成员的所有行为都只是对系统的操作,而这些操作的结果是使整个系统的状态发生改变.因此 CSCW 系统状态的变化反映了群体的协作过程.群体协作关系的建立与维护、用户间的各种交互以及用户的每个协同动作都是通过系统状态的改变完成的.若用 Σ 表示系统的所有状态,Sb 表示系统的初始状态(即协作开始状态),Se 表示系统的结束状态(即协作结束状态),则群体的一次协作过程表现为 $Sb \xrightarrow{操作1} S1 \xrightarrow{操作2} S2 \rightarrow \cdots \rightarrow Se$.系统处于开始状态 Sb 时,在某个用户执行操作 1 时转到状态 S1,此后用户的每次有效的操作都会导致系统状态的改变,直到出现 Se,系统停止运行,协作结束.

然而 CSCW 系统毕竟是分布的,每个用户都是通过某个节点上的运行部件参与协作的,如图 2 所示.在用户的一次操作下系统各部件要进行复杂的交互,才能使整个系统从一种状态转换到另一种状态.也就是说从状态 Si 到 Si+1 要经过一个交互过程.这种交互过程反映了 CSCW 系统处理问题的原则,它对系统所提供的功能以及系统对协作的支持起着至关重要的作用,是系统设计与构造时所关心的主要问题.系统本身的行为主要表现为在允许的用户操作下,系统各部件经过交互,使系统从一种状态转换到另一种状态.对系统行为的描述应以这些交互为中心.

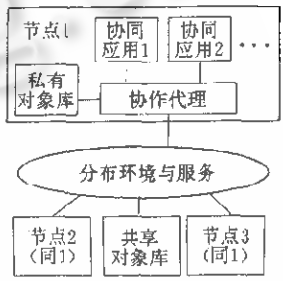

图2 CSCW系统的具体结构

1.2 CSCW 系统的具体结构

为了使 CSCW 系统容纳多种协同应用,以对群体的协作进行全面的支持,为了对 CSCW 领域的普遍性问题进行统一的处理,以弥补现有分布式系统的不足,我们在分布式系统之上引入了 CSCW 支撑平台 SPCSCW,将其作为群体参与协作的代理.整个 CSCW 系统的结构如图 2 所示,每个节点上有一个支撑平台部件,也就是图中的协作代理,各协作代理通过分布环境与服务彼此相通,协作代理之上是各种协同应用部件,各节点中相同的协同应用部件通过协作代理组成完整的协同应用.

各协作代理与各协同应用部件组成了系统中的全部运行实体.系统中除了各种运行实体之外,还有存放各种信息的数据库.为了便于表示与操作,信息都是以对象的形式存在的.在系统中每个节点中的协作代理管理一个私有对象库.而整个系统中的共享对象由分布式数据库管理系统进行维护,各节点的协同应用部件可通过协作代理对其进行共享操作,以完成基于共享对象的协作.

1.3 系统要素

从上述 CSCW 系统结构可以看出,与协作支持及系统本身相关的要素如下:

• 用户及身份 用户通过某个节点参与协作,成为协作群体的一员,为了使协作过程顺利进行,群体中的每个成员都有一定的身份,该身份表明了他在协作中的权利与义务.

• 支撑平台(协作代理) 系统的每个节点中有一个支撑平台部件,它是用户参与协作的代理.它不仅要连通不同节点中的协同应用部件,管理各种协同应用的关系,而且还要维护用户间的动态协作关系.在协作过程中,每个用户都只与一个节点中的协作代理相关.因此在讨论系统本身的问题时,我们用协作代理表示与之相关的用户,这样用户在协作中的身份,也就表现为协作代理的身份.

• 协同应用及其状态 各节点中相同的协同应用部件组成完整的分布式协同应用,可为群体成员完成具体协作任务提供支持.每个协同应用部件在协作过程中都会处于某种状态,此状态表明当时用户使用该应用程序的情况.

• 信息对象及其共享 从协作的角度看,每个信息对象包含两个部分:信息对象本身以及各种身份的群体成员对它的操作权限.此外,在协作过程中,每个共享对象都处于一种状态,该状态表示各应用程序部件对它的操作情况.

• 系统实体间的交互途径 系统中各实体间的直接交互只有两种:一种是各协作代理之间的交互,另一种是各应用程序部件同与其相连的协作代理的交互.系统中的所有其它形式的交互,都是由这两类直接的交互构成的.

• 系统的状态 系统的状态包括 4 个方面:①协作状态,也就是协作的阶段,如发起、召集、进行、结束等;②协作群体的各成员及其身份;③各应用的状态;④各共享信息对象的状态.

2 基于时序逻辑的描述模型

建立 CSCW 系统描述模型时,不仅要体现以上各种要素,而且应该抓住系统行为的核心,也就是系统各运行部件

之间的交互.由于在交互过程中,各运行部件的行为是时间相关性很强的时序行为,并且其关系本质上是并行的,为了对交互进行形式化的描述,我们运用了时序逻辑语言 XYZ/E 中的一些概念及描述方法.

2.1 时序逻辑及 XYZ/E 语言

时序逻辑系统在经典一阶逻辑的连接词、量词(¬(非),∧(与),∨(或),→(蕴含),==(等于),$A(全称量词),$E(存在量词),$T(真),$F(假))之外引入了一些时序逻辑算子,如□(一直),◇(最终),$O(下一时刻),$U(直到)等.其语义可参见文献[5].

XYZ/E[5]语言把时序逻辑算子溶进程序设计语言之中,使其既是逻辑系统又是程序设计语言.在 XYZ/E 的时序逻辑公式中,有一种称为状态转换等式,其形式为 Ox=exp,它表示下一时刻变量 x 的值等于表达式 exp 当前的值.它与普通高级语言中的赋值语句相似.该语言还设置了一个系统变量 LB,称为控制变量,用于指明程序当前的标号,这样 $OLB=1 就表示程序的下一执行语句的标号为 1,其语义就是"goto 1",这里"1"称为转出标号.与之相应的另一等式 LB=1 表明"1 的定义出现",这里"1"称为定义标号.

通过设置控制变量,XYZ/E 将各个分离的时序逻辑公式连成了统一的整体.以此为基础,它给出进程的概念,进程是可以并发运行并通过消息的传递协调各自行为的程序模块.从一个进程到另一个进程之间消息的传递是通过两个进程之间的通道实现的,通道可以动态决定.在 XYZ/E 中有两条与通道相关的消息通信命令,即输出命令和输入命令,其形式为 ch ! y 与 ch ? x.并发进程之间的通信分为同步通信与异步通信两种,分别用"SYN"和"ASYN"表示,其语义可参见文献[4].当然除了这些之外,XYZ/E 中还提出了很多其它的概念,并给出了程序的各种证明规则,从而使其成为一种对各类系统进行形式化抽象描述和验证的有力工具.

2.2 描述模型 TLDM_CSCW

鉴于时序逻辑在分布式系统的描述与验证上的作用,我们据此建立了 CSCW 系统的描述模型,整个模型是个八元组 TLDM_CSCW=(Q,P,A,O,C,S,T,I).

- Q 为系统可能出现的所有协作状态的集合.
- $P=(P,R)$. P 为系统中所有协作代理的集合,$P=\{p_i \mid i=1,2,\ldots,n, n$ 为协作成员数$\}$,p_i 表示第 i 个节点中的协作代理. R 为协作成员可能的身份集合,也就是协作代理的可能身份.
- $A=(A,U)$. A 为系统中所有应用程序部件的集合,$A=\{a_{ij} \mid i=1,2,\ldots,n, n$ 为协作成员数$, j=1,2,\ldots,m, m$ 为系统中不同应用程序数$\}$,a_{ij} 表示节点 i 中第 j 个应用程序的部件.为便于讨论,我们用集合 $A_j=\{a_{ij} \mid i=1,2,\ldots,n, n$ 为协作成员数$\}$,表示完整的分布式协同应用 j;用集合 $A'_i=\{a_{ij} \mid j=1,2,\ldots,m, m$ 为系统中不同应用程序数$\}$,表示节点 i 中的所有协同应用程序部件. U 为协同应用可能出现的所有状态的集合.
- $O=(O,V,W)$. O 为系统中所有信息对象的集合;V 为信息对象的所有可能的操作权限的集合;W 为协作中信息对象的所有可能的操作状态的集合.
- C 为系统中各种通道的集合.系统中各运行实体表示为 XYZ/E 中的并行进程,实体间的直接交互途径表示为进程间的通道.根据 CSCW 系统中各实体间的直接交互情况可将通道分为 3 类:$C=CH\_PP+CH\_AP+CH\_PA$.

$CH\_PP=\{ch\_pp_{ij} \mid i,j=1,2,\ldots,i\neq j\}$,表示各协作代理间传递信息的通道,$ch\_pp_{ij}$ 表示协作代理 i 向 j 传递消息的通道.$CH\_AP=\{ch\_ap_{ji} \mid i=1,2,\ldots,j=1,2,\ldots,m\}$,为节点中应用程序向协作代理传递信息的通道,$ch\_ap_{ji}$ 表示节点 i 中的应用程序 j 向协作代理传递消息的通道.$CH\_PA=\{ch\_pa_{ij} \mid i=1,2,\ldots,j=1,2,\ldots,m\}$,为节点中协作代理向应用程序传递信息的通道,$ch\_pa_{ij}$ 表示节点 i 中的协作代理向应用程序 j 传递消息的通道.

- $S=(\Sigma,Sb,Se)$. Σ 为系统的所有可能的状态的集合.$Sb\in\Sigma$ 为系统初始状态,$Se\in\Sigma$ 为系统的终止状态.由于系统状态涉及 4 个方面,每个方面的不同都导致不同的系统状态,因此系统可能的状态很多,为方便起见,我们用一个系统变量 S 来表示系统的当前状态,用 $S(c)$ 表示当前协作的状态,$S(p)$ 表示当前 p 的身份,$S(a)$ 表示当前 a 的状态,$S(o,r)$ 表示当前身份为 r 的协作成员对信息对象 o 的操作权限,$S(o,a)$ 表示当前协同应用部件 a 对 o 的操作状态.
- T 为各种逻辑算子和谓词的集合,谓词是对用户的操作、系统中的事件以及实体行为的抽象.
- I 为系统中所有可能的交互的集合,有 $I=\{interaction_i \mid i=1,2,\ldots\}$,其中每个 *interaction* 的定义如下

interaction::=%*INTERACTION InterName* ==[*units*]
units::=*unit* {; *units*}
unit::=*Entity* : □[*ces*] WHERE *cons*
ces::=*ce* {, *ces*}

这里{x}表示 x 或空. *ce* 为 XYZ/E 中的条件元. *unit* 与 XYZ/E 中的单元类似,其中的 *Entity* 表示某个运行实体,这里 *unit* 表示 *Entity* 实体在 *InterName* 交互中的行为. *interaction* 称为交互,它首先由记号 %*INTERACTION* 加以

标识，其后是该交互的名称，交互的主要部分是一些单元，也就是说，一次完整的交互过程是由系统中各实体的行为组成的. 交互中的各个单元可以看成 XYZ/E 中的并发进程，其中的 *Entity* 为进程的名称，进程间通过通道进行信息交换，用这些信息交换来协调各并发进程(也就是各运行实体)的行为，使得交互得以进行.

系统中的每种交互都是在系统处于某种状态下，用户进行了某种允许的操作后进行的，交互是系统中的完整过程，它可能会导致整个系统状态的改变，而使用户的协作过程向前推进.

通过对系统在所有状态下，用户进行各种允许的操作后所产生的交互过程进行描述，一方面可以体现系统的整体状态转换关系，另一方面又可刻画系统内部的行为，这样就能比较全面地表示系统的各个方面.

2.3 CSCW 系统行为的描述

CSCW 系统所提供的各种功能都是由系统中的运行实体 P 和 A 以及它们之间的交互完成的. 下面给出系统通过实体间的交互来维护群体协作关系的情况. 群体成员要进行协作，必须首先建立某种形式的协作关系. 在协作过程中，群体成员可以动态地加入和退出协作. 在综合的 CSCW 系统中，这种动态协作结构的维护是由协作代理统一进行的，在关系的改变结束之后，协作代理才把改变后的情况通知给所有的协同应用. 下面以“加入”为例来看看在维护协作关系时系统各实体间的交互.

$$
\begin{aligned}
&\%INTERACTION\ Join == [\\
&p_s:\ \square\ [\\
&\quad LBp_s = Join\_STARTp_s \wedge PB\_Join(p_s) \Rightarrow \$OLBp_s = lp_{s1};\\
&\quad LBp_s = lp_{s1} \wedge S(c) == q_r \Rightarrow \$Och\_pp_{sc}!\ Join\_Req(p_s) \wedge \$OLBp_s = lp_{s2};\\
&\quad LBp_s = lp_{s1} \wedge S(c) != q_r \Rightarrow \$OLBp_s = Join\_STOPp_s;\\
&\quad LBp_s = lp_{s2} \Rightarrow \$OTD = 0 \wedge \$OLBp_s = lp_{s3};\\
&\quad LBp_s = lp_{s3} \wedge TD < t \wedge \neg ch\_pp_{sc}?\ x \Rightarrow \$OTD = TD + 1 \wedge \$OLBp_s = lp_{s3};\\
&\quad LBp_s = lp_{s3} \wedge TD < t \wedge ch\_pp_{sc}?\ x \Rightarrow \$OLBp_s = lp_{s4};\\
&\quad LBp_s = lp_{s3} \wedge TD \geqslant t \Rightarrow \$OLBp_s = Join\_STOPp_s;\\
&\quad LBp_s = lp_{s4} \wedge x! = Join\_Ack(p_s) \Rightarrow \$OLBp_s = Join\_STOPp_s;\\
&\quad LBp_s = lp_{s4} \wedge x == Join\_Ack(p_s) \Rightarrow \$OLBp_s = lp_{s5};\\
&\quad LBp_s = lp_{s5} \Rightarrow \$OAction(Join\_Ack(p_s), p_s) \wedge \$OLBp_s = lp_{s6};\\
&\quad LBp_s = lp_{s6} \Rightarrow \$A\ (a_{sk} \in A'_s)\ \$Och\_pa_{sk}!\ Join\_Ack(p_s) \wedge \$OLBp_s = lp_{s7};\\
&\quad LBp_s = lp_{s7} \Rightarrow \$OLBp_s = Join\_STOPp_s;]\\
&p_c:\ \square\ [\\
&\quad LBp_c = Join\_STARTp_c \wedge ch\_pp_{sc}?\ x \Rightarrow \$OLBp_c = lp_{c1};\\
&\quad LBp_c = lp_{c1} \wedge x == Join\_Req(p_s) \Rightarrow \$OLBp_c = lp_{c2};\\
&\quad LBp_c = lp_{c1} \wedge x! = Join\_Req(p_s) \Rightarrow \$OLBp_c = Join\_STOPp_c;\\
&\quad LBp_c = lp_{c2} \Rightarrow \$OAction(Join\_Req(p_s), p_c) \wedge \$OLBp_c = lp_{c3};\\
&\quad LBp_c = lp_{c3} \wedge \neg Join\_Yes(p_s) \Rightarrow \$OLBp_c = lp_{c4};\\
&\quad LBp_c = lp_{c3} \wedge Join\_Yes(p_s) \Rightarrow \$OLBp_c = lp_{c5};\\
&\quad LBp_c = lp_{c4} \Rightarrow ch\_pp_{sc}!\ Join\_NA(p_s) \wedge \$OLBp_c = Join\_STOPp_c;\\
&\quad LBp_c = lp_{c5} \Rightarrow ch\_pp_{sc}!\ Join\_Ack(p_s) \wedge \$OLBp_c = lp_{c6};\\
&\quad LBp_c = lp_{c6} \Rightarrow \$A\ (a_{ck} \in A'_c)\ \$Och\_pa_{ck}!\ Joined(p_s) \wedge \$OLBp_c = lp_{c7};\\
&\quad LBp_c = lp_{c7} \Rightarrow \$A\ (p_k \in P - \{p_c\})\ \$Och\_pp_{ck}!\ Joined(p_s) \wedge \$OLBp_c = lp_{c8};\\
&\quad LBp_c = lp_{c8} \Rightarrow \$OP = P + \{p_s\} \wedge \$OLBp_c = Join\_STOPp_c;]\\
&\$A\ p_k \in P - \{p_c\}:\ \square\ [\\
&\quad LBp_k = Join\_STARTp_k \wedge ch\_pp_{ck}?\ x \Rightarrow \$OLBp_k = lp_{k1};\\
&\quad LBp_k = lp_{k1} \wedge x == Joined(p_c) \Rightarrow \$OLBp_k = lp_{k2};\\
&\quad LBp_k = lp_{k1} \wedge x! = Joined(p_c) \Rightarrow \$OLBp_k = Join\_STOPp_k;\\
&\quad LBp_k = lp_{k2} \Rightarrow \$OAction(Joined(p_c), p_k) \wedge \$OLBp_k = lp_{k3};\\
&\quad LBp_k = lp_{k3} \Rightarrow \$A\ (a_{kj} \in A'_k)\ \$Och\_pa_{kj}!\ Joined(p_c) \wedge \$OLBp_k = lp_{k4};\\
&\quad LBp_k = lp_{k4} \Rightarrow \$OLBp_k = Join\_STOPp_k;]\\
&\$A\ a_{sk} \in A'_s:\ \square\ [\\
&\quad LBa_{sk} = Join\_STARTa_{sk} \wedge ch\_pa_{sk}?\ x \Rightarrow \$OLBa_{sk} = la_{sk1};\\
&\quad LBa_{sk} = la_{sk1} \wedge x == Join\ \ Ack(p_s) \Rightarrow \$OLBa_{sk} = la_{sk2};\\
&\quad LBa_{sk} = la_{sk1} \wedge x! = Join\_Ack(p_s) \Rightarrow \$OLBa_{sk} = Join\_STOPa_{sk};\\
&\quad LBa_{sk} = la_{sk2} \Rightarrow \$OAction(Join\_Ack(p_s), a_{sk}) \wedge \$OLBa_{sk} = Join\_STOPa_{sk};]\\
&\$A\ a_{jk} \in A:\ \square\ [\\
&\quad LBa_{jk} = Join\_STARTa_{jk} \wedge ch\_pa_{jk}?\ x \Rightarrow \$OLBa_{jk} = la_{jk1};\\
&\quad LBa_{jk} = la_{jk1} \wedge x == Joined(p_c) \Rightarrow \$OLBa_{jk} = la_{jk2};\\
&\quad LBa_{jk} = la_{jk1} \wedge x! = Joined(p_c) \Rightarrow \$OLBa_{jk} = Join\_STOPa_{jk};\\
&\quad LBa_{jk} = la_{jk2} \Rightarrow \$OAction(Joined(p_c), a_{jk}) \wedge \$OLBa_{jk} = Join\_STOPa_{jk};]]
\end{aligned}
$$

以上是某个用户按下"加入"按钮后,整个系统的动作.其中 p_s 表示提出"加入"申请的节点;p_c 表示主席节点;$PB_Join(p_s)$是表示用户 s 按下"加入"按钮的谓词,$S(c)==q_r$ 表示系统当前的协作状态为 q_r,$q_r\in Q$ 是允许用户执行"加入"操作的协作状态;$Join_Req(p_s)$表示 p_s 提出的"加入"请求,其中包含节点 s 中的各种信息;TD 为时延变量;t 为等待时间;$Join_Ack(p_s)$表示对 p_s 的"加入"请求的应答,其中包含系统的各种信息;$Join_NA(p_s)$表示对 p_s 的"加入"请求的拒绝;$Action(Join_Ack(p_s),p_s)$表示 p_s 在得到应答时所作的操作;$Action(Join_Req(p_s),p_c)$表示 p_c 在接到 p_s 的"加入"请求时所作的操作;$Join_Yes(p_s)$表示主席允许 p_s 加入协作;$Joined(p_s)$表示 p_s 已加入协作的事件,其中包含节点 s 中的各种信息;$Action(Joined(p_s),p_k)$表示 p_k 对 p_s 加入协作的事件进行的处理.

当然,对于一个完整的 CSCW 系统而言,其行为包括众多的交互,通过对系统中的所有可能的交互进行形式化的描述,就可以全面地刻画系统的行为,这里省略了系统中其它交互的描述.

3 结束语

为了清晰地描述 CSCW 系统的行为,以对系统进行分析与验证,本文基于时序逻辑,提出了一个 CSCW 系统的抽象描述模型.此模型全面地概括了 CSCW 系统的各个方面,对系统中的多种要素进行了合理的抽象.它将 CSCW 系统的主要行为表示为系统中各种运行实体之间的交互,这些交互是由协作中的用户对系统的操作引起的,它们可导致整个系统状态的改变,从而使用户的协作得以进行.鉴于时序逻辑语言 XYZ/E 在系统的描述和验证方面的功能,这些交互过程是以 XYZ/E 进行描述的,这为 CSCW 系统各种性质的验证打下了基础.

参考文献

1 Tarney K. Protocol specification and testing. New York and London: Plenum Press, 1991

2 史美林. CSCW:计算机支持的协同工作. 通信学报,1995,16(1):55~61
(Shi Mei-lin. CSCW: computer supported cooperative work. Journal of China Institute of Communications, 1995,16(1):55~61)

3 Ruruta R, Stotts P D. Interpreted collaboration protocols and their use in groupware prototyping. In: Proceedings of CSCW'94, Chapel Hill, NC, USA, 1994. 121~131

4 Takeda K, Inaba M, Sugibara K. User interface and agent porotoyping for flexible working. IEEE Multimedia, Summer 1996. 40~50

5 马华东,刘慎权. 基于时序逻辑的动画描述模型. 计算机学报,1995,18(11):814~821
(Ma Hua-dong, Liu Shen-quan. Temporal logic based animation description model. Chinese Journal of Computers, 1995,18(11):814~821)

6 柳军飞,唐稚松. 软件过程建模语言研究. 软件学报,1996,7(8):449~457
(Liu Jun-fei, Tang Zhi-song. A study on scftware modeling languages. Journal of Software, 1996,7(8):449~457)

7 沈武威,唐稚松. XYZ 系统在电信领域中的应用. 软件学报,1996,7(6):321~330
(Shen Wu-wei, Tang Zhi-song. Application of XYZ system in telecommunication. Journal of Software, 1996,7(6):321~330)

8 韩俊刚,王岩冰,沈武威. 用 XYZ/E 语言描述和验证硬件的行为. 软件学报,1996,7(11):676~682
(Han Jun-gang, Wang Yan-bing, Shen Wu-wei. Describing and verifying the behavior of hardware in XYZ/E language. Journal of Software, 1996,7(11):676~682)

9 唐稚松,赵琛. 一种面向软件工程的时序逻辑语言. 软件学报,1994,5(12):1~16
(Tang Zhi-song, Zhao Chen. A temporal logic language oriented toward software engineering. Journal of Software, 1994,5(12),1~16)

Temporal Logic Based Description Model for Computer Supported Cooperative Work Systems

WANG Guo-yi SHI Yuan-chun XU Guang-you

(*Department of Computer Science and Technology Tsinghua University Beijing* 100084)

Abstract To enable group members carry out work cooperatively, CSCW(computer supported cooperative work) systems may deal with distribution issues, solve the application-specific problems, and provide user-centered cooperation support as well. These result in the complexities of CSCW systems. However, describing the behavior of systems formally is a key step in the process of system development. In order to describe the behavior of CSCW systems clearly and make it possible to verify some specific properties of the system, this paper proposes a temporal logic based description model for CSCW systems. In this model, a CSCW system consists of running entities and information objects, and the system behavior is represented by the interactions between entities described using a temporal logic language——XYZ/E. This model can guide the analysis and design of CSCW systems.

Key words CSCW(computer supported cooperative work), temporal logic, description mode.

Class number TP311.1