多版本技术中的对象标识及其压缩*

窦万峰[1,2+], 李春萍[1,2]

[1](南京师范大学 数学与计算机科学学院 计算机科学系,江苏 南京 210097)

[2](南京师范大学 计算机技术研究所,江苏 南京 210097)

Object Identification and Its Compression for Multi-Versioning Technique

DOU Wan-Feng[1,2+], LI Chun-Ping[1,2]

[1](Department of Computer Science, College of Mathematics and Computer Science, Nanjing Normal University, Nanjing 210097, China)

[2](Institute of Computer Technology, Nanjing Normal University, Nanjing 210097, China)

+ Corresponding author: Phn: +86-25-83598129, E-mail: douwf@email.njnu.edu.cn, http://mcs.njnu.edu.cn

Received 2003-05-26; Accepted 2004-01-06

Dou WF, Li CP. Object identification and its compression for multi-versioning technique. *Journal of Software*, 2004,15(8):1133~1140.

http://www.jos.org.cn/1000-9825/15/1133.htm

Abstract: Collaborative editing systems support a group of collaborators to edit or view the same graphics at the same time from geographically dispersed sites connected by communication networks. The research on this is of great challenge. The multi-versioning technique in collaborative graphics editing system is discussed. This paper puts emphasis on object identification and its strategies, some existing problems. A general object identification solution for these problems is proposed, which solves their object identification and system maintenance between operations with causality relationships, and with identical relationships, that the existing strategies cannot resolve. On the other hand, the size of object identification will become larger and larger with the progress of collaborative editing works and the generation of conflicting operations, thus affect the efficiency of the system. Therefore, some rules suitable to the compression of the object identification is analyzed, an object compression method is presented and its validity is also analyzed. Analysis of some examples and experimental results show that all strategies and algorithms presented in this paper improve the efficiency of the system.

Key words: collaborative graphics editing system; operation intention preservation; multi-versioning technique; object identification; object identification compression

摘 要: 协同编辑系统支持不同地域的协作者通过网络同时地编辑同一个文本或图形对象,其研究具有挑战性.介绍了协同图形编辑中的多版本技术,重点讨论了对象标识及其策略,给出所存在的问题.提出了一个全面的对象标

* Supported by the Natural Science Foundation for Higher School of Jiangsu Province of China under Grant Nos.02KJD520006, 03KJA520066 (江苏省高校自然科学基金)

作者简介: **窦万峰**(1968－),男,陕西永寿人,博士,副教授,主要研究领域为计算机支持的协同工作,计算机辅助设计,软件工程; **李春萍**(1974－),女,助教,主要研究领域为计算机支持的协同工作.

识方案.该方案解决了以往方案不能处理具有依赖关系的操作和相同关系的操作的对象标识和系统维护问题.同时,随着编辑的进行和冲突的产生,对象标识会急剧膨胀,影响系统处理的效率,分析了对象标识压缩的规则,给出一个对象标识压缩方法,并对其有效性进行了分析.实例分析和实验结果表明,所提出的对象标识及其压缩方案提高了系统的效率.

关键词: 协同图形编辑系统;操作意愿保证;多版本技术;对象标识;对象标识压缩

中图法分类号: TP311 **文献标识码**: A

为了支持自由、自然的交互,实时协同设计系统应具有交互实时性、协作分布性、操作无约束性等特征[1,2].要满足这些特征,只有采用全复制的系统结构.研究与设计全复制结构的实时协同编辑系统(collaborative editing system,简称 CES)的关键问题之一是多个复制数据视图的一致性.

Sun 等人在前人的研究基础上提出了一个一致性模型,作为一致性维护的理论框架[3].这个一致性模型包括收敛、因果维护和意愿保证这 3 个一致性特征.该模型已成功地运用于协同文本编辑系统[4].

协同图形编辑系统处理的是图形对象,共享文档可以表示为一组具有不同属性的图形对象的集合.这些对象相互独立,对一个对象的属性的修改不会影响其他对象.协同图形编辑系统的冲突判别和处理与文本系统不同,并发操作的冲突主要发生在对同一个对象的同一个属性的操作,若不加以控制,操作的执行会导致操作意愿分离.例如,两个并发的 Move 操作同时移动同一个对象到不同的位置,执行的结果在各个复制视图上是不一致的.因此,协同图形编辑系统的不一致主要是由冲突操作的意愿分离引起的.

1 多版本技术与对象标识

如图 1 所示,Sun 等人强调操作意愿保证是协同编辑系统的根本问题,并给出了一个基于对象复制的广义的多版本模型和处理策略[1,5,6].在多版本策略下,只有当并发操作试图把同一个对象版本的同一个属性改变成不同的值时,才会发生意愿分离,并把具有这种关系的操作称为冲突操作,记作"⊗".冲突操作的执行效果不可能共存于同一个对象,因此,要保证冲突的操作在不同的协作节点上的执行效果一致并保证操作各自的意愿,有效的方法是从原始对象复制出两个对象,冲突的操作分别在各自的对象上执行.这种方法称为多版本技术(可参考文献[1]).例如,两个节点的初始视图为一个对象 G,由于在两个节点上同时产生了一个操作,分别移动该对象到不同的位置.如果直接执行,在各个节点上会得到不一致的结果,不能保证两个操作的意愿.如果对原始对象进行复制,两个冲突的操作分别作用于不同的对象版本,则可以保证操作结果和意愿一致.

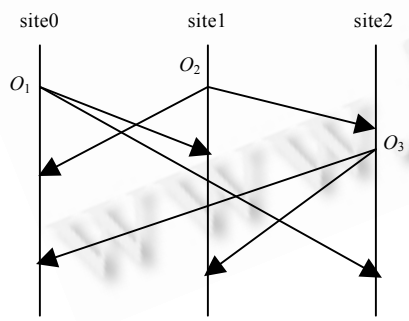

Fig.1 An example of editing operations
图 1 编辑操作示例

多版本技术虽然可以有效地解决冲突操作的意愿保证问题,但由于采用了对象复制策略,一个来自远程节点的操作在本地节点执行时,其目标对象可能是一组对象.如何唯一地标识这些复制对象以及确定操作的目标对象等,成为主要的问题.基于对象复制的多版本策略通过最大相容组集合(maximal compatible groups set,简称 MCGS)来确定待执行操作的最终的目标对象以及是否产生新的对象版本.

设给定目标对象 G 上的一组操作 GO 的最大相容组集合为 M,则有:(1) 任意一个相容组 $CG_i \in M$,都对应于一个由 G 派生的对象版本;(2) CG_i 中的操作都在对应的对象版本上执行.可以证明,对一组操作,由其生成的 MCGS 在各个节点上是唯一的,且由 MCGS 所确定的操作的执行结果也是唯一的[1].这里的相容组(compatible group,简称 CG)是指具有相容关系(也称不具有冲突关系的操作,记作"⊕")的一组操作组成的集合.构建 MCGS,有两个技术问题需要解决:一是如何在 $MCGS_{i-1}$ 上执行新操作 O_i 并生成 $MCGS_i$;二是如何标识每一个步骤中与 $MCGS_{i-1}$ 相关的所有对象的版本.针对第 1 个问题,Sun 等人提出了多对象版本增量创建算法(multiple object versions incremental creation,简称 MOVIC

算法);一致性对象标识方案(consistent object identification,简称 COID)和目标对象版本选择方案(target object version recognition,简称 TOVER)是针对第 2 个问题提出的[1].

分析发现,文献[1]给出的对象标识机制 COID 过于复杂,不但需要考虑 4 种可能的操作情况,而且涉及同一对象的所有复制版本.而随着操作的进行,对象版本标识将越来越长,维护也将更困难.同时,该方案没有考虑到相同的并发操作的情况,因而不能满足对象标识的一致性.

2 多版本中的对象标识

在分布式环境下,当远程节点上的操作到达某一节点时,其目标对象可能已发生了变化,即发生了对象复制,因此必须采用适当的对象标识机制,使得系统根据操作参数就可以识别其目标对象.多版本的对象标识应该满足:(1) 唯一性,同一个节点上的每个对象具有唯一标识;(2) 可跟踪性,复制的对象版本的标识应能反映与原对象的复制关系;(3) 一致性,不同节点的同一对象/对象版本具有相同的标识[1].

我们用 $Id(G)$和 $Id(O)$分别代表对象 G 和操作 O 的标识.由于每一个操作 O 有一个唯一的标识(可用逻辑向量实现),记作 $Id(O)$,则对象可以用创建它的操作唯一地标识,即 $Id(G)=Id(O)$.这个方案仅适合于单版本系统,而不适用于多版本系统.发生冲突时,系统将对原对象进行复制,以保证冲突操作的意愿.如果采用上述的对象标识方案,则这两个复制对象具有相同的标识,即都是原对象的标识.假设有一个远程操作,其目标对象为原对象的标识,即 $Target(O)=Id(G)$,它将无法确定两个复制对象哪一个是其最终的目标对象.

为了解决这个问题,文献[1]扩展上述方案.让两个版本对象继承原始对象的标识,同时一个版本包含引发新版本的操作 O_1 的标识,即 $Id(G(O_1))=Id(G)\cup\{Id(O_1)\}$.这个方案虽然能够正确区分不同的对象版本,并能跟踪原始的对象版本,但不能确保同一个对象的多个复制对象的标识的一致性.比如,两个冲突的操作 O_1 和 O_2 分别产生于不同的节点.由于本地操作立即执行,然后再传播到其他节点,所以不同的节点上这两个操作的执行顺序不同.在 O_1 所在的节点,先执行 O_1,然后执行 O_2,则 $Id(G\{O_1\})=Id(G),Id(G\{O_2\})=Id(G)\cup Id(O_2)$;同样,在 O_2 所在的节点,操作执行顺序为 O_2 和 O_1,对象标识结果为 $Id(G\{O_2\})=Id(G)$, $Id(G\{O_1\})=Id(G)\cup Id(O_1)$.为了解决这个问题,让每个对象版本都附加冲突操作的标识,即每个节点上的两个版本应标识为 $Id(G\{O_1\})=Id(G)\cup Id(O_1),Id(G\{O_2\})=Id(G)\cup Id(O_2)$.

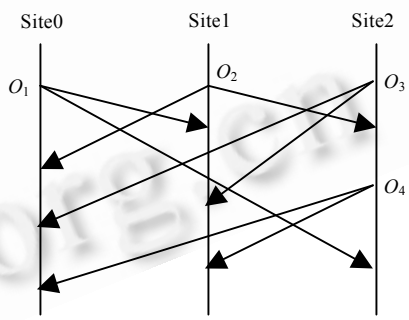

Fig.2 An example of editing operations
图 2 编辑操作示例

考虑如下的操作情形(如图 2 所示):设并发操作 O_1,O_2 和 O_3 作用于同一目标对象 G,它们的操作关系表达为 $O_1\otimes O_2,O_1\otimes O_3$ 和 $O_2\oplus O_3$.按顺序 O_1,O_2,O_3 执行:O_1 执行后,得到 $G\{O_1\}$;执行 O_2,根据冲突和对象标识策略,有 $G\{O_1\},Id(G\{O_1\})=Id(G)\cup Id(O_1)$ 和 $G\{O_2\},Id(G\{O_2\})=Id(G)\cup Id(O_2)$;执行 O_3 得到 $G\{O_1\}$和 $G\{O_2,O_3\}$,它们的对象标识不变;若执行顺序为 O_1,O_3,O_2,则 O_1,O_3 执行后,有 $G\{O_1\},Id(G\{O_1\})=Id(G)\cup Id(O_1)$ 和 $G\{O_3\},Id(G\{O_3\})=Id(G)\cup Id(O_3)$;执行 O_2 得到 $G\{O_1\}$和 $G\{O_3,O_2\}$,对象标识不变.显然,不同执行顺序下得到的相同对象版本 $G\{O_2,O_3\}$和 $G\{O_3,O_2\}$,其标识不一致.

3 改进的对象标识方案

3.1 逻辑时间向量

由于实时协同系统中的操作大多是簇发的,因此可以使用逻辑向量获取操作间的因果/并发关系[7].时间域由一组非负整数的 N 维向量来表示(N 表示协作节点的数目).每个节点 i 上都维护一个向量 $\boldsymbol{V}[1\ldots n]$,$\boldsymbol{V}[i]$是节点 i 的本地逻辑时间,$\boldsymbol{V}[j]$表示节点 i 所知道的节点 j 的最新本地时间.例如,如果 $\boldsymbol{V}[j]=x$,节点 i 知道节点 j 广播时刻的逻辑时间为 x.整个向量 $\boldsymbol{V}$ 表现了节点 i 的全局逻辑时间,节点 i 依据它为本地操作打上时间戳.节点上的逻辑时间向量随着操作的进行不断更新.逻辑时间向量更新规则如下:(1) 执行一个操作前,节点 i 更新自己的逻辑时

钟,$V[i]=V[i]+d(d>0)$,d 为增量,一般为 1;(2) 每一个操作消息都附带了远程节点发送时刻的时间戳.当节点接收到操作消息(m,Vt)时,执行以下操作:$N\geq k\geq 1$,$V[k]=\max(V[k],Vt[k])$;然后执行步骤(1).

在逻辑时间向量下,远程操作执行条件为:设操作 O 在节点 s 生成,逻辑时间向量为 $\boldsymbol{SV}_O$,则 O 在节点 $d(d<>s)$上满足因果关系(d 上的逻辑时间向量为 $\boldsymbol{SV}_d$),如果满足下列条件:(1) $\boldsymbol{SV}_O[s]=\boldsymbol{SV}_d[s]+1$;(2) $\boldsymbol{SV}_O[i]=\boldsymbol{SV}_d[i]$,$i$ 从 0 到 $N-1$,$i\neq s$.条件(1)保证 O 必须是来自节点 s 上的下一个操作,来自 s 的其他操作都已经在节点 d 上执行过.条件(2)保证 O 产生前在 s 上执行过的其他远程操作都已经在 d 上执行了.

3.2 对象标识方案

根据冲突/相容关系,不难看出操作 O_2 和 O_3 要满足上面的操作关系,只有两种情况:(1) O_2 和 O_3 具有依赖关系,即 $O_2\rightarrow O_3$;(2) O_2 和 O_3 是完全相同的操作,即 $O_2=O_3$,它们修改对象 G 的同一属性到相同的属性值.对两个相同的并发操作,在对象版本上执行任意一个即可.

对于第(1)种情况,O_2 和 O_3 具有依赖关系,如图 1 所示,则不可能存在 O_1,O_3 和 O_2 的执行顺序,因为 O_3 依赖于 O_2,则 O_3 的执行必然在 O_2 的后面(先后产生的操作必然按顺序执行,可以通过因果关系控制执行),且 O_3 的目标对象就是 O_2 的目标对象.否则,根据 $O_1\otimes O_2$ 和 $O_1\otimes O_3$ 关系和冲突定义[1]可以得到,如果 O_2 和 O_3 修改同一个对象到不同的值(相同的情况在后面分析),则 O_2 和 O_3 必然冲突,即 $O_2\otimes O_3$.分析表明,之所以会出现上面的特殊情况是因为原冲突定义没有考虑到依赖关系的操作.另一个值得注意的问题是,原有的冲突是针对单一对象版本的,多版本策略中的冲突定义必须考虑到版本因素.这里给出修改后的操作的冲突/相容关系定义.

定义 1. 给定两个操作 O_a 和 O_b,若它们满足下面的条件,则 O_a 与 O_b 冲突,记作 $O_a\otimes O_b$:(1) $O_a||O_b$;(2) $Target(O_a)\subseteq Target(O_b)$,或 $Target(O_b)\subseteq Target(O_a)$;(3) $Attr.Key(O_a)=Attr.Key(O_b)$;(4) $Attr.Value(O_a)\neq Attr.Value(O_b)$.否则,$O_a$ 与 O_b 相容,记作 $O_a\oplus O_b$.

这里,$O_a||O_b$ 表示操作具有并发关系,$Target(O_a)$表示操作的目标对象;$Attr.Key(O_a)$代表操作所修改的目标对象的属性;$Attr.Value(O_a)$表示操作修改的对象的属性值.

根据新的操作冲突/相容关系定义,可能存在上面的操作关系.但由于是依赖关系,不会出现 O_1,O_3 和 O_2 的执行顺序,因而对象标识方案得以简化.

对第(2)种情况,O_2 和 O_3 是完全相同的操作.这里首先给出操作的相同关系定义.

定义 2. 给定操作 O_a 和 O_b,若它们是操作的相同关系,记作"=",如果 $O_a||O_b$,且 $Target(O_a)\subseteq Target(O_b)$,或 $Target(O_b)\subseteq Target(O_a)$,且 $Attr.Key(O_a)=Attr.Key(O_b)$,且 $Attr.Value(O_a)=Attr.Value(O_b)$.

为了解决具有相同关系的操作所导致的版本不一致问题,对象标识机制应满足以下规则:

规则 1. 对象标识可看作是标识符的集合,所包含的操作标识按操作的全序关系(total ordering relation) "=>"[3]排序.

规则 2. 如果操作的执行引起对象标识的更新(发生对象复制),则对操作进行标记.

设上述 3 个操作的全序关系为 $O_1=>O_2=>O_3$.按 O_1,O_2 和 O_3 顺序执行,O_1 和 O_2 执行后得 $Id(G(O_1))=Id(G)\cup Id(O_1)$和 $Id(G(O_2))=Id(G)\cup Id(O_2)$;执行 O_3,由于 O_2 和 O_3 实际为具有相同关系的操作,且有 $O_2=>O_3$,不需要执行 O_3.若操作的执行顺序为 O_1,O_3 和 O_2,则 O_2 执行前有对象版本 $Id(G(O_1))=Id(G)\cup Id(O_1)$和 $Id(G(O_3))=Id(G)\cup Id(O_3)$,由于 O_3 已经执行过,O_2 不再执行,但 $O_2=>O_3$,为保持版本一致,更新对象版本及标识,用 O_2 的标识替换 O_3 的标识,即将对象标识中的 $Id(O_3)$替换成 $Id(O_2)$,得 $Id(G(O_3))=Id(G)\cup Id(O_2)$.同时,在操作历史日志中,用 O_2 替换 O_3,以保持一致.若全序关系是 $O_3=>O_2$,则忽略 O_2 操作,对象标识和操作历史日志保持不变.

基于以上分析,这里给出对象标识方案:

(1) 若对象 G 由操作 O 创建,则 $Id(G=Id(O)$.

(2) 若操作 O 与目标对象 G 上已执行过的操作相容,即 $O\oplus CG$,则 O 在 G 上的执行不影响 G 的标识.

(3) 若存在 $O_x\in CG$,且 O_x 与 O 是具有相同关系的操作,则 O 不再执行,但需作如下处理:如果 $O=>O_x$,且 O_x 与对象标识相关,则以 $Id(O)$替换 $Id(G)$中记录的 $Id(O_x)$.

(4) 若 O 在对象 G 上的执行导致对象复制,则:(i) 复制版本继承原对象 G 的标识;(ii) 对冲突操作执行的版

本,其标识应包括该操作的标识符,即 $Id(G(O_a))=Id(G)\cup Id(O_a)$,$Id(G(O_b))=Id(G)\cup Id(O_b)$,并对冲突操作进行标记.

这里需要说明的是,虽然具有相同关系的操作不执行,但仅对执行效果而言,节点仍需对逻辑时间向量进行正常的维护.由于具有相同关系的操作仅执行一次,为判断对象版本上是否存在与给定操作相同的操作,需要修改目标对象选择机制 TOVER 方案.

容易证明,这里给出的对象标识机制满足对象标识的 3 个特性.对象的创建操作是唯一的,且同一对象的不同复制版本包含有冲突操作的标识,保证对象标识唯一.MOVIC 算法可保证对给定的一组操作,所有节点上都维护相同的对象版本,且相同对象版本的标识总是一致的.所有对象版本都记录了原对象的标识,因此可追溯到原对象,满足可跟踪性.

3.3 实例分析

如图 2 所示,操作 O_1,O_2,O_3 和 O_4 的目标对象为对象 G,它们的关系为 $O_1\otimes O_2,O_1\oplus O_3,O_2\oplus O_3,O_1\|O_4,O_2\rightarrow O_4$.设操作具体为 $O_1=Move(G,X),O_2=Move(G,Y),O_3=Color(G,red),O_4=LineType(G,Pot)$.

节点 0 的初始逻辑向量为(0,0,0).O_1 先生成并执行,节点的逻辑向量为(1,0,0);当 O_2 到达节点 0 时,由于 $O_1\otimes O_2$,从 G 得到两个对象版本 $G\{O_1\}$和 $G\{O_2\}$,其标识分别为 $Id(G\{O_1\})=Id(G)\cup Id(O_1)$和 $Id(G\{O_2\})=Id(G)\cup Id(O_2)$,逻辑向量为(1,1,0);接着执行 O_3,由于 $Target(O_3)\subseteq Id(G\{O_1\})$,且 $Target(O_3)\subseteq Id(G\{O_2\})$,且 O_3 与 O_1 和 O_2 相容,因此在两个对象版本上执行得到 $G\{O_1,O_3\}$和 $G\{O_2,O_3\}$,且两个对象的标识保持不变,逻辑向量为(1,1,1);当 O_4 到达时,$Target(O_4)\subseteq Id(G\{O_1,O_3\},Target(O_4)\subseteq Id(G\{O_2,O_3\}$,但 $O_2\rightarrow O_4$,因此 O_4 应在 $G\{O_2,O_3\}$版本上执行得到 $G\{O_2,O_3,O_4\}$,且标识不变,此时逻辑向量为(1,1,2).

节点 1 的初始逻辑向量为(0,0,0).执行顺序依次是 O_2,O_1,O_3 和 O_4,其执行结果得到了两个版本 $G\{O_2,O_3,O_4\}$和 $G\{O_1,O_3\}$,它们的标识分别为[$Id(G),Id(O_2)$]和[$Id(G),Id(O_1)$],节点 1 上最终的逻辑向量为(1,1,2).

同样地,对节点 2,其初始逻辑时间向量为(0,0,0).O_3 产生并立即执行得到 $G\{O_3\}$,逻辑时间向量为(0,0,1);接着 O_2 到达,它与 O_3 目标对象相同且与 O_3 相容,得到 $G\{O_3,O_2\}$,对象表示不变,节点逻辑向量为(0,1,1);O_4 产生并执行,得到 $G\{O_3,O_2,O_4\}$,对象标识不变.当 O_1 到达节点 2 时,由于 O_1 与 O_2 冲突,且与 O_3 相容,但 O_4 与 O_3 和 O_2 具有因果关系,所以从 G 派生两个版本:$G\{O_3,O_1\}$,其对象标识为 $Id(G\{O_3,O_1\})=Id(G)\cup Id(O_1)$;$G\{O_3,O_2,O_4\}$,其标识为 $Id(G\{O_3,O_2,O_4\})=Id(G)\cup Id(O_2)$.节点 2 的逻辑向量为(1,1,2).

4 对象标识压缩

4.1 对象标识压缩技术

上面给出的对象方案可以有效地解决多版本中的对象版本标识问题.然而,从另一个方面来看,随着编辑过程的不断进行,由于对象版本标识中包含原来对象的信息,其标识会随着冲突的产生而膨胀,版本不断增多,导致系统处理效率大为下降.这就需要有一个有效的方法对对象版本标识进行压缩,以减少不必要的冗余.

在多版本技术中,由于复制的对象版本上也可能发生冲突并产生新的版本,以此类推,对象版本可以用一个二叉树来表示.二叉树的底层是实际存在的对象.不难看出,每一层对象版本的标识都包含了上一层对象的标识信息,因而对象标识会急剧增加.如果消除上一层的冗余信息,就可以使对象标识简洁,处理方便.当然,对标识进行压缩将会影响可跟踪性,但是一般情况下,顶层或上一层的对象信息对当前的编辑来说是没有必要的,可以删除,这也符合实际的应用情况.

我们知道,删除对象标识中的冗余信息不能影响对象标识的唯一性、可跟踪性和一致性,因此必须遵守一些规则.对象标识是用操作的标识来表示的,那么对象标识信息进行删除实际上就是对一些不需要的操作进行删除,其规则是该操作不会影响其他未执行的操作的执行结果.

规则 3. 要删除的操作必须在各个节点上都执行过;要删除的操作必须是各个节点上最小的(逻辑向量最小)操作.

规则 3 的要求很显然.如果要删除的操作没有在所有节点上执行,那么必然导致对象标识不一致.同时要求要删除的操作必须是标识中逻辑向量最小的,保证先执行先删除,使得对象标识满足唯一性和可跟踪性.

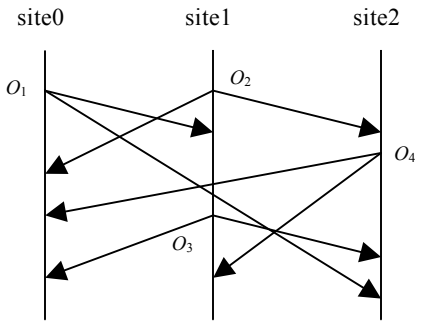

Fig.3 An example of the compression object identification

图 3 标识压缩示例

规则 4. 对象标识中要删除的操作后面的一个操作应在所有节点上执行过.

定理. 对象标识中除了创建操作以外的所有操作都是导致新版本产生的操作.

证明:可用反证法证明,在此从略. □

该规则保证导致新版本的操作在各节点都执行之后,要删除的操作才不会影响后续操作的执行和对象标识.例如,在图 3 中,假设 O_1 与 O_2 冲突,产生原始对象 G 的操作在节点 0 和 1 上分别执行后,O_1 和 O_2 可以删除.但在节点 2 上,O_1 此时还没有执行,标识为创建该对象的操作的标识,若此时对节点 0 和 1 进行标识压缩,O_4 在节点 0 和 1 上执行时可能存在找不到目标对象的危险.

规则 5. 对象标识中只有多于 1 个以上的复合标识才能进行标识压缩.

这个规则说明,只有在对象上发生冲突时才会出现复合标识,否则标识压缩会导致丢失标识的危险.如图 3 中的节点 2 上的一系列操作不会产生新版本,标识为简单标识,即为创建该对象的操作的标识.只有在执行了操作 O_1 以后才会出现冲突和多版本,才有标识压缩的可能.

结合上面的分析,这里给出一个对象标识压缩算法.

压缩算法. *IDCompress*(*G*:*GraphicObj*).

Begin

1) 设 *ObjectId* 为对象标识,有 *ObjectId*=*Id*(*G*).如果|*ObjectId*|==1,则不能进行压缩,直接返回;
2) 否则,读出对象标识中第 1 个标识所对应的操作 *O*=*ObjectId*[0];
3) 检查操作 *O* 是否已经在每个节点上执行.设节点总数为 *M*,则
   For *i*=1 to *M* Do
    If $O \notin HB[i]$ Then 返回;
   End Do
4) 读出对象标识中第 2 个标识所对应的操作 *O*=*ObjectId*[1]:检查该操作 *O* 是否已经在每个节点上执行.设节点总数为 *M*,则
   For *i*=1 to *M* Do
    If $O \notin HB[i]$ Then 返回;
   End Do
5) 删除对象标识中的第 1 个标识.令 *N*=|*ObjectId* |,则
   For *i*=0 to *N*–1 Do
    ObjectId[*i*]=*ObjectId*[*i*+1];
   End Do
6) *Id*(*G*)=*ObjectId*;

End 算法结束

算法第 1)步读取对象的标识并判断标识是否是简单标识,若是,则拒绝压缩(规则 5).第 2)步和第 3)步读取第 1 个标识的对应的操作,并检查该操作是否已经在每个节点上执行(规则 3).这里,HB(history buffer)是每个节点上维护的已经执行过的操作历史日志.若是,第 4)步继续检查操作标识下一个操作的是否也已经在各节点执行(规则 4);在满足前 4 步的情况下,第 5)步完成标识压缩工作;最后,对象 *G* 得到新的标识.

4.2 讨 论

对象标识必须满足唯一性、一致性和可跟踪性.那么,本文给出的对象标识压缩方案是否也能满足上述 3 个特性呢?首先,对象版本的产生是针对于同一个对象上的两个或多个操作之间发生冲突的情况,新的对象版本必然包含冲突操作的标识信息.因此,即使删除了对象标识中以前的操作的标识,也不会影响对象标识的唯一性.如原始对象标识为{O},发生冲突后的两个版本 $Id(G_1)=\{O,O_1\}$,$Id(G_2)=\{O,O_2\}$,假设此时操作 O 根据标识压缩规则可以删除,则两个版本的标识变为 $Id(G_1)=\{O_1\}$和 $Id(G_2)=\{O_2\}$.显然,二者是唯一的.

对象标识的一致性是指不同节点上的同一个对象的标识必须一致,是一个相对的概念.由于操作在各个节点上执行的顺序不同,版本产生的先后顺序也不一致,因而对象版本及其标识可能在某个时刻并不一致,但最终的结果是一致的.同样,对象标识压缩方案也是在某个时刻或最终完成协作时保证各个对象版本的标识一致.

可跟踪性是保证本地节点上产生的操作在远程节点执行时能够识别到正确的目标对象.由于对象复制的存在,远程操作在本地执行时可能不止一个,因此需要相应的机制获取合适的目标对象.TOVER 方案中采用对象标识来获取操作的目标对象[1].TOVER 的改进方案是通过检测操作的目标对象与候选对象的标识是否具有包含关系来确定哪些对象是最终的目标对象.这不外乎两种情况:(1) 远程节点已经对该对象的标识进行了压缩,而本地节点还没有;或者本地节点已作过压缩,而远程节点还没有,因而会出现标识相互包含的关系;(2) 两个节点上都已经完成了对象标识压缩,则对象版本标识必然相同.基于上述讨论,TOVER 方案中的条件 $Target(O)\subseteq Id(G)$为 $Target(O)\subseteq Id(G)$,或 $Target(O)\supseteq Id(G)$.这样就可以保证操作正确地选择目标对象.

当然,压缩后的对象标识的可跟踪性不同于未压缩对象标识的可跟踪性.未采用压缩方案的对象标识可以跟踪到与该对象有关的所有操作,而压缩标识只能跟踪到后续部分操作.但是,正如对象标识压缩方案中所讨论的,被删除的是冗余的操作信息,这些信息不会影响后续的操作,因而也没有跟踪这些操作的必要.以上分析说明,对象标识压缩在一定程度上能够保证对象标识的唯一性、一致性和可跟踪性.

5 结论及进一步的工作

操作意愿保证是协同编辑系统的根本问题.Sun 等人使用基于对象复制的多版本策略来解决操作意愿的不一致,但他们提出的 MOVIC 算法及其对象标识机制还有待完善[1].本文在总结已有研究成果的基础上,分析了多版本的对象标识机制,给出了一个有效的对象标识方案,同时解决了对象标识的压缩技术.本文的算法已经在一个基于对象的协同图形编辑器上实现.该系统采用 3 层物理结构,分别实现组管理、小组内通信和操作处理功能.

为了对本文的算法与策略的正确性进行验证,我们在局域网下分两组进行了实验.一组采用 3 个节点,共产生了 50 个操作.为了使实验效果明显,我们使用随机延时保证并发操作数量增加.表 1 给出 3 个节点协作的情况.另一组采用 5 个节点,共产生了 100 个操作,表 2 给出了实验的统计情况.表 3 给出了采用标识压缩和未压缩方案时系统的性能比较,表中列出每 10 个操作的系统处理时间.可以看出,当标识未压缩时,系统处理时间在逐渐增加,尤其是发生冲突时,系统处理时间增加很快;当采用标识压缩方案时,系统处理压缩需要一定的开销,但随着冲突的发生,系统的处理时间增长比较缓慢.另外,未压缩时系统的内存消耗也很大。

Table 1 Operations execution with three cooperative sites

表 1 3 个节点协作的操作情况

Number of operations	Concurrent operations	Conflict operations	Objects	Object versions	Level of object	Number of identification compression
50	24 with 4 groups	16 with 3 groups	G1 and G2	6	2 on object G1; 3 on object G2	One times on the object G1; 2 times on the object G2

Table 2 Operations execution with five cooperative sites

表 2 5 个节点协作的操作情况

Number of operations	Concurrent operations	Conflict operations	Objects	Object versions	Level of object	Number of identification compression
100	32 with 3 groups	16 with 3 groups	1	8	4	4 times

Table 3 Comparison of system performance using identification compression over un-compression strategy (ms)

表 3 标识压缩和未压缩时系统运行的性能比较 (ms)

Un-Compression	97	112	147	161	183	203	233	269	294	344
Compression	103	121	132	135	133	137	139	141	141	143
Number of operations	10	20	30	40	50	60	70	80	90	100

实验分析表明,一般冲突主要产生在一些比较重要或关键的对象,或是协作者有不同看法的情况.多版本的目的是保证各方的意愿都能够表达出来,以便更进一步地分析和讨论,所以多版本策略很适合于产品概念设计阶段.

本文的主要贡献在于:第一,提出的对象标识方案不但满足唯一性、可跟踪性和一致性要求,更重要的是,该方案与 Sun 等人给出的方案相比,对象标识简单、易于维护,同时解决了具有依赖关系的操作和相同关系的操作在 Sun 等人的方案中导致版本标识不一致的问题;考虑到多版本环境对冲突的影响,还给出修改后的冲突定义.第二,随着版本的增加,对象版本标识会越来越庞大,影响系统处理的效率.因此,本文分析了对象标识与操作的关系,首次分析并提出对象标识压缩的规则,并给出了一个对象的标识压缩算法.

本文算法的可行性和有效性在一个基于对象的协同图形编辑器上进行了验证.由于对象标识压缩技术与操作历史日志有一定的联系,它们具有包含关系,因此今后的工作是如何与操作历史日志的垃圾清除机制结合,有效地提高系统的效率.另一方面,相同关系的操作的执行,不但影响对象标识、操作历史日志的维护,而且影响各个节点的状态和操作的逻辑时间.因为在复制结构下,操作的生成和操作之间的各种关系都是通过各个节点的逻辑时间向量加以推断的,尽管相同关系的操作只要执行其中一个即可,但却影响了生成该操作的节点和其他执行节点的逻辑时间,因此在系统实现时必须全面加以考虑.

References:

[1] Sun CZ, Chen D. Consistency maintenance in real-time collaborative graphics editing systems. ACM Trans. on Computer-Human Interaction, 2002,9(1):1~41.

[2] Dou WF. Consistency preservation model for collaborative designing systems. Mechanical Design, 2002,19(11):4~7 (in Chinese with English abstract).

[3] Sun CZ, Jia X, Zhang Y, Yang Y, Chen D. Achieving convergence, causality-preservation, and intention-preservation in real-time cooperation editing systems. ACM Trans. on Computer-human Interactions, 1998,5(1):63~108.

[4] Yang GX, Shi ML. An object data model-based concurrency control for fully-replicated architecture. Chinese Journal of Computers, 2000,23(2):113~125 (in Chinese with English abstract).

[5] Sun CZ, Chen D. A multi-version approach to conflict resolution in real-time collaborative graphics editing systems. In: Christine N, ed. The 20th IEEE Int'l Conf. on Distributed Computing Systems. Taipei: IEEE Press, 2000. 316~325.

[6] Sun CZ, Ellis CA. Operational transformation in real-time group editors: Issues, algorithms, and achievements. In: Poltrock S, Gradin J, eds. Proc. of the ACM Conf. on Computer Supported Cooperative Work. Seattle: ACM Press, 1998. 59~68.

[7] Michel R, Mukesh S. Logical time: Capturing causality in distributed systems. IEEE Computer Magazine, 1996,29(4):49~56.

附中文参考文献:

[2] 窦万峰.协同设计系统的一致性保证模型.机械设计,2002,19(11):4~7.

[4] 杨光信和史美林.全复制结构下基于对象数据模型的并发控制.计算机学报,2000,33(2):113~125.