

一种新的多版本增创算法

杨 君 窦万峰

(南京师范大学数学与计算机科学学院 南京 210097)

(南京师范大学虚拟地理环境教育部重点实验室 南京 210097)

摘 要 现有的对象复制技术是实时协同图形编辑系统中符合一致性模型的并发控制策略,但难以解决非几何属性以及三维绘图系统中的并发冲突情况. 介绍了一个全新的一致性维护模型——基于版本复制的多版本技术,当并发操作发生冲突时,利用简化规则从目标版本派生出叶子物理版本,分别应用冲突操作到不同的叶子版本,有效地解决了非调和冲突操作的意愿保证. 通过研究多版本技术的相关支持算法,在算法 MOVIC 的基础上,从各个冲突操作的最大共同相容操作着手,提出了基于相容冲突组的快速多版本增创算法 FMVIC(Fast Multiple Versions Incremental Creation),减少了相容操作的比较次数,缩小了去冗余操作的范围.

关键词 协同图形编辑; 并发控制; 版本复制; 一致性模型; 相容冲突组

中图法分类号 TP391

A New Multiple Versions Incremental Creation Algorithm

YANG Jun DOU Wan-Feng

(College of Mathematics and Computer Science, Nanjing Normal University, Nanjing 210097)

(Key Laboratory of Virtual Geographic Environment, Nanjing Normal University, Nanjing 210097)

Abstract The multi-versioning technique based on object replication strategy can meet consistency model while the conflict operations occur in the cooperative graphics editing systems(CG-ES), but it does not efficiently solve conflict between non-geometry attribute operations and complex graphics objects. A new distributed multi-versioning model based on version replication is proposed for consistency maintenance of discordant conflicting operations according to the conflicting features and types of concurrent operations in CGES. To guarantee intention of conflicting operations, leaf-versions must be replicated from the original version based on the reduce strategy, and then applying conflicting operations to the corresponding object of the different replicated versions. The leaf-versions can be created by MOVIC algorithm in which the comparison-number of compatible operations is too much. A Fast Multiple Versions Incremental Creation (FMVIC) algorithm based on the compatible and conflict group is presented in this paper. The final versions constructed by the FMVIC algorithm is the same as that constructed by the MOVIC algorithm, but it can decrease the comparison-number of compatible operations and narrow the scope of comparison operations for the removing redundant versions.

Keywords cooperative graphics editing; concurrency control; version replication; consistency model; compatible and conflict groups

1 引 言

协同图形编辑系统支持不同地域的多个协作人员通过网络实时地操作同一个图形,共同高效地完成设计工作.为支持自由、自然的交互,协同编辑系统应当满足交互实时性、协作分布性和操作无约束性^[1-2],为此必须采用全复制式的系统结构.其中保证各个复制数据视图的一致性这类系统的关键技术之一.传统的并发控制策略,如加锁和序列化都无法真正达到“自然、自由交互”的目标.

操作转换方法 dOPT 算法最早由 Ellis 提出^[3],并由 Sun 等人进行了深入研究和改进^[4].它通过操作转换函数保证对象的一致性及用户操作意图.但是 dOPT 主要针对文本/字符对象,对图形对象的处理存在难以克服的问题,例如难以构造复杂图形对象的转换函数.因为转换函数的包含转换和排除转换本质上是一系列的 Undo/Do/Redo 操作,其对于复杂图形几何求解不易实现.oodOPT 算法^[5]扩展了 dOPT 所能支持的数据结构,但没有涉及图形对象的特殊语义.

多版本方法是一种重要的用以实现协同图形编辑的一致性维护技术.对象复制技术^[6-7]是由 Sun 等人提出的一种多版本策略,其完全符合一致性模型^[4],当操作发生冲突时,将目标对象派生出多个复制对象,然后分别应用这些冲突操作到不同的复制对象.文献^[8-9]进一步讨论并分别提出了保持上下文意愿策略和基于语义的多版本策略.但基于对象复制的相关策略不能有效解决非几何属性操作引起的冲突,因为各复制对象可能处于相同的位置而被相互覆盖,因而操作意愿无从体现.另一方面,对象复制技术难以将复杂的三维图形对象恢复到冲突操作执行前的状态,由于多个复制对象仅存在于一个物理版本,而协作的最终版本只能存在一个对象,所以其需要大量的仲裁工作.

基于版本复制的协同多版本技术我们提出了一个新思路,通过物理版本的复制来保证冲突操作的意愿^[10].由于每个物理版本对应一个实际版本,每个版

本可以直接被保存作为协作的编辑结果,从而避免了版本内的仲裁工作. MVIC 算法^[10]将所有父版本和各子版本都用版本树的形式在各站点表示,这种方式使得大量冗余版本被存储.本文提出简化规则,在版本复制技术中仅保留根版本和叶子版本.多版本增创算法 MOVIC^[6] (Multiple Object Versions Incremental Creation)利用操作之间的冲突和相容关系动态确定最大相容组集即最终叶子版本.通过分析,发现当多个冲突操作具有相同的相容操作时, MOVIC 算法将相同的相容操作多次存储在不同的相容组内,导致重复比较计算及生成大量的冗余相容组.文献^[11]通过压缩多版本中的对象标识以优化其存储空间,但没有改进原算法的计算性能.本文在 MOVIC 算法的基础上进行了改进,从各个冲突操作的最大共同相容操作着手,提出了一种新的多版本增创算法 FMVIC(Fast Multiple Versions Incremental Creation)以确定最终叶子版本,它减少了相容操作的比较次数,缩小了去冗余操作的范围.

本文第 2 节介绍基于版本复制的多版本技术,并提出简化规则;第 3 节介绍并分析用以生成最终叶子版本的 MOVIC 算法;第 4 节给出相容冲突组定义,并提出新的 FMVIC 算法;第 5 节通过实验比较 MOVIC 和 FMVIC 的性能;第 6 节为结论.

2 基于版本复制的协同多版本技术

多版本技术强调操作意愿的保证,当发生操作冲突时,从目标文档版本派生出多个子版本,分别应用冲突操作到不同的子版本以得到多个版本.基于对象复制的多版本技术仅有一个物理版本,发生冲突时仅仅是对目标对象进行复制,然后使冲突操作作用于不同的复制对象,复制对象共存于同一个文档.基于版本复制的协同多版本技术则不同,当两个并发操作发生冲突时,从原始版本复制出两个子版本,然后将冲突操作分别应用到不同复制版本的相应对象上,协同编辑的共享文档表示为一组物理版本.协同多版本技术的一个示例如图 1 所示,当操作

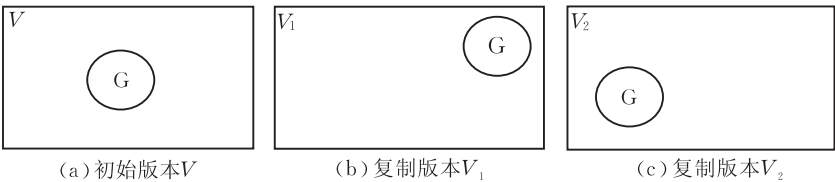


图 1 协同多版本技术示例

O_a 和 O_b 均作用于初始版本 V 中的对象 G ,且将对象 G 向两个不同方向移动时,为解决冲突,协同多版本技术将初始版本 V 另复制出两个新版本 V_1, V_2 ,并且将冲突操作 O_a 和 O_b 分别作用于新版本 V_1, V_2 .此时所有节点共享的文档由三个物理版本构成:版本 V 、版本 V_1 和版本 V_2 .

2.1 基本概念

设 O 是一个操作, G 为 O 的目标对象, $Attr(O)$ 代表操作 O 所作用的目标对象 G 的某个属性, $Attr.type(O)$ 代表目标对象的属性类型, $Attr.value(O)$ 代表目标对象的属性值, $Version(O)$ 代表操作 O 的目标版本.

定义 1. 因果关系“ \rightarrow ”.

设节点 i 和 j 生成的两个操作 O_a 和 O_b ,如果符合下面条件之一,则 O_a 和 O_b 存在因果关系,表示为 $O_a \rightarrow O_b$.

(1) $i=j$,且操作 O_a 在操作 O_b 之前生成;

(2) $i \neq j$,且 O_a 在节点 j 上的执行发生在操作 O_b 生成之前;

(3) 存在一个操作 O_x ,有 $O_a \rightarrow O_x$ 且 $O_x \rightarrow O_b$.

定义 2. 依赖关系和并发关系“ \parallel ”.

设两个任意的操作 O_a 和 O_b ,如果 O_a 和 O_b 存在因果顺序关系 $O_a \rightarrow O_b$,则说明 O_b 依赖于 O_a ;如果既不存在 $O_a \rightarrow O_b$,也不存在 $O_b \rightarrow O_a$,则 O_a 和 O_b 存在并发关系,记作: $O_a \parallel O_b$.

逻辑时间向量可用于维护操作之间的依赖和并发关系^[4]:每个协作节点都维护一个 n 元组的逻辑时间向量 $\mathbf{V}=(v_1, v_2, \dots, v_n)$,其中 n 为协作节点的个数.逻辑时间向量仅适合每个节点只有一个版本的情况,当存在多个版本时,可以用逻辑时间矩阵,即给逻辑时间向量加上版本列,将其扩展为二维矩阵以记录每个节点上各个版本的逻辑时间.

定义 3. 冲突关系“ \otimes ”和相容关系“ \odot ”.

给定两个操作 O_a 和 O_b ,它们作用于同一个物理版本上的同一个图形对象,如果它们具有冲突关系,记作: $O_a \otimes O_b$,当且仅当

(1) $Version(O_a) = Version(O_b)$,

(2) $O_a \parallel O_b$,

(3) $Attr.type(O_a) = Attr.type(O_b)$,

(4) $Attr.value(O_a) \neq Attr.value(O_b)$.

否则, O_a 和 O_b 具有相容关系,记作: $O_a \odot O_b$.

从定义可知,只有当两个并发操作应用于同一版本的相同对象的相同属性,并且并发改变该属性至不同属性值时,才会导致冲突.

2.2 版本复制技术

当一个操作在执行时所产生的实际结果不同于其在生成时想要的结果,则出现操作意愿分离.出现这种不一致现象的原因是由于并发执行使得操作执行时的状态和定义时的状态不一致.我们应用版本复制技术以使操作的意愿得到保证.

图2给出一个具体应用实例.设操作 O_1, O_2 和 O_3 具有相同的目标版本 V 和目标对象 G ,并满足关系: $O_1 \parallel O_2 \parallel O_3$, $O_1 \otimes O_2$, $O_1 \odot O_3$, $O_2 \odot O_3$.假设 O_1 向上移动 G , O_2 向下移动 G , O_3 设置 G 的颜色属性为红色.

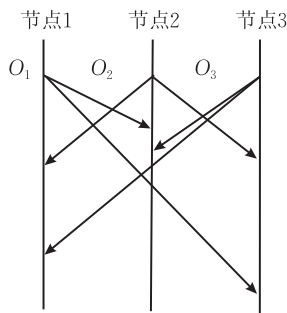


图2 并发应用实例

节点1首先执行 O_1 并产生版本 $V(O_1)$;接着 O_2 到达并执行,因为 $O_1 \otimes O_2$,由原版本 V 产生两个复制版本 V_1 和 V_2 ,即 $V_1(O_1)$ 和 $V_2(O_2)$,原版本 V 恢复为操作 O_1 执行前的状态 $V()$; O_3 到达并执行,此时 O_3 的目标版本是 V, V_1 和 V_2 ,由于 O_3 与 O_1 和 O_2 都相容,故 O_3 在 V, V_1 和 V_2 上均执行,最终共享文档为由 $V(O_3), V_1(O_1, O_3)$ 和 $V_2(O_2, O_3)$ 组成的一组物理版本.

容易看出,节点2与节点1情况相同.

在节点3上依次执行操作 O_3 和 O_2 后结果为 $V(O_2, O_3)$; O_1 到达并执行时,由于 $O_1 \otimes O_2, O_1 \odot O_3$, O_1 与 O_2 的冲突导致产生子版本 $V_1(O_1, O_3)$ 和 $V_2(O_2, O_3)$,原版本 $V(O_2, O_3)$ 恢复为状态 $V(O_3)$.最终,节点3与节点1上的执行结果也完全相同.

随着协同编辑的进行,版本的层次结构将不断延伸成一个树状结构,我们称其为版本树.以某一叶子结点为目标版本的一组并发操作所产生的树称为此组并发操作的版本子树.并发操作的原始目标版本称为此组操作的根版本;此根版本的叶子结点对应于那些未被复制的包含所有并发操作的各个叶子版本;其余的结点对应的版本称为中间版本.中间版本的保留使版本集合不断扩充,最终形成目标版本的子树结构,这使得冗余操作信息在各个中间版本重复存储多次,占据大量的空间.而实际协作过程

中,协作者主要关注于此组并发操作的叶子版本和根版本,其中叶子版本为并发操作最终的各个复制版本,根版本用于支持用户视图的回溯.所以,我们引入简化规则只保留一组并发操作的根版本和叶子版本.

最终叶子版本通过执行版本递增创建算法确定,MOVIC 算法通过生成最大相容组确定各个版本.根版本需要包含同一目标版本上的所有操作,为满足根版本的表示需求,引进新的版本表示形式——目标相容组 TCG(Target Compatible Group).若不存在冲突操作,TCG 和文献[6]中定义的 CG 在内容上是相同的,均记录同一个目标版本上的各个相容操作.当冲突发生时,TCG 需要包含相互冲突的操作,所以扩展引入冲突操作组 COG(Conflict Operation Group)概念及冲突操作组集合 COGS(Conflict Operation Group Set)概念.对应每个版本,COG 记录了导致该版本被复制的众多冲突操作组中的一组,COGS 则包含了导致该版本被复制的所有冲突操作组,并以特殊符号 I 加以标志,即 $COGS = \{I(COG_1), I(COG_2), \dots, I(COG_m)\}$.

例如,给定目标版本 V 上的 4 个并发操作 O_1, O_2, O_3, O_4 且 $O_1 \otimes O_2, O_3 \otimes O_4$,操作执行顺序为 O_1, O_2, O_3, O_4 . O_1 和 O_2 的依次执行导致产生叶子版本 $V_1(O_1)$ 和 $V_2(O_2)$,根版本为 $V(I(O_1, O_2))$.接着 O_3 执行,因为 O_3 和 O_1, O_2 都相容,所以,有 $V(O_3, I(O_1, O_2)), V_1(O_1, O_3), V_2(O_2, O_3)$.最后 O_4 执行, $O_3 \otimes O_4$ 导致版本 V_1 和 V_2 发生复制从而转变为中间版本 $V_1(O_1, I(O_3, O_4)), V_2(O_2, I(O_3, O_4))$,生成新的叶子版本 $V_{1.1}(O_1, O_3), V_{1.2}(O_1, O_4), V_{2.1}(O_2, O_3), V_{2.2}(O_2, O_4)$,根版本更新为 $V(I(O_1, O_2), I(O_3, O_4))$,版本复制的层次结构如图 3 所示.通过简化规则,最终共享版本为 $V(I(O_1, O_2), I(O_3, O_4)), V_{1.1}(O_1, O_3), V_{1.2}(O_1, O_4), V_{2.1}(O_2, O_3)$ 和 $V_{2.2}(O_2, O_4)$.

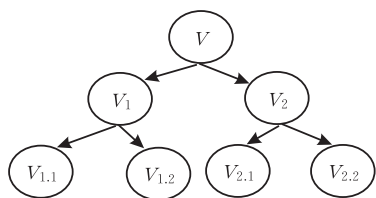


图 3 版本层次结构

2.3 任意操作组的复合执行效果

针对协同操作的特性,为保证任意一组操作在所有协同节点上的操作效果都相同,Sun 和 Chen 等人使用相容操作组合及正确性校验规则来确定版

本^[6].根据 2.1 节协同多版本技术中的冲突定义,两个并发操作相冲突的必要条件之一就是这两个操作具有相同的目标版本.在此前提下,最终的叶子版本可以用最大相容组集合 MCGS^[6]来表示,通过多版本增创算法 MOVIC 确定相同目标对象上一组并发操作的复合执行效果.若给定目标对象 G 上的一组操作 GO 的最大相容组集合 M ,则(1)任意一个最大相容组 $MCG_i \in M$,其对应一个由 G 派生的叶子版本;(2) MCG_i 中的操作都在对应的版本上执行. MCGS 对于一组操作是唯一的,所以由 MCGS 确定的操作效果也是唯一的.

3 多版本增创算法 MOVIC

在实时协同编辑系统中,由于网络传输的原因,多个协同用户的并发操作可能以不同顺序到达协同节点,因此需要一个分布算法递增地为各协同节点构建最大相容组集合 MCGS. MCGS 的唯一性保证这组操作在协同节点上的执行效果总是一致的,即 MOVIC 算法具有操作执行顺序无关性.

3.1 MOVIC 算法描述

设一组目标对象相同的 n 个操作在某一个站点以 O_1, O_2, \dots, O_n 的顺序执行.此算法应产生一系列有序的 MCGS: $MCGS_1, MCGS_2, \dots, MCGS_n$. MOVIC 算法主要解决某一操作 O_i 作用于 $MCGS_{n-1}$ 生成 $MCGS_n$ 的过程.

下面简单描述此算法.

设 O_i 为待执行操作, $O_i \odot CG_x$ 代表 O_i 与 CG_x 中所有操作都相容, $O_i \otimes CG_x$ 代表 O_i 与 CG_x 中所有操作都冲突.

1. 将 $MCGS_i$ 初始化为空,记计数器 C 为 $MCGS_{i-1}$ 中相容组的数目.
2. 从 $MCGS_{i-1}$ 中取出一个相容组 CG_x ,若操作 $O_i \odot CG_x$,则 $CG_x = CG_x + \{O_i\}$;若 $O_i \otimes CG_x$,则 $C = C - 1$;若 O_i 和 CG_x 中操作既有冲突又有相容时,取出 CG_x 中所有和 O_i 都相容的操作与 O_i 组成一个新的 CG_{new} ,并将 CG_{new} 加入到 $MCGS_i$ 中;将 CG_x 加入到 $MCGS_i$ 中;重复以上步骤直到遍历完 $MCGS_{i-1}$ 中所有的相容组.
3. 如果 $C = 0$,则将操作 O_i 作为 CG_{new} 加入到 $MCGS_i$ 中.
4. 清除 $MCGS_i$ 中的冗余相容组.对于步 2 和步 3 生成的 CG_{new} ,如果其被 $MCGS_i$ 中其它的相容组 CG_y 所包含,则将其视为冗余相容组并从 $MCGS_i$ 中删去.

3.2 实例描述

用实例简单描述 MOVIC 算法的具体实现过程及其操作执行顺序无关性.假设有一组目标对象相同的 4 个操作 $(O_1 \otimes O_2 \otimes O_3) \odot O_4$ 在两个站点的执

行顺序分别为 O_1, O_2, O_3, O_4 和 O_1, O_2, O_4, O_3 .

在第一种执行顺序的情况下,算法实现步骤如下:

O_1 执行后, $MCGS_1 = \{O_1\}$; O_2 执行后, 由于 $O_1 \otimes O_2$ 则发生版本复制后的 $MCGS_2 = \{\{O_1\}, \{O_2\}\}$; O_3 执行后, 由于 $O_1 \otimes O_2 \otimes O_3$, 同理 $MCGS_3 = \{\{O_1\}, \{O_2\}, \{O_3\}\}$; O_4 执行后, 由于 O_4 和其他三个操作都相容, 所以 $MCGS_4 = \{(O_1, O_4), (O_2, O_4), (O_3, O_4)\}$.

在第二种执行顺序的情况下,算法实现步骤如下:

$MCGS_1 = \{O_1\}$; $MCGS_2 = \{\{O_1\}, \{O_2\}\}$; 当 O_4 执行后, 由于 O_4 和 O_1, O_2 都相容, $MCGS_3 = \{(O_1, O_4), (O_2, O_4)\}$; 当 O_3 作用于 $MCGS_3$ 时, O_3 首先和 (O_1, O_4) 比较, 由于 $O_1 \otimes O_3$ 而 $O_4 \odot O_3$, 产生一个新的相容组 (O_3, O_4) , 同理 O_3 和 (O_2, O_4) 比较后, 产生相容组 (O_3, O_4) , $MCGS_4 = \{(O_1, O_4), (O_3, O_4), (O_2, O_4), (O_3, O_4)\}$; 去冗余操作后结果为 $MCGS_4 = \{(O_1, O_4), (O_2, O_4), (O_3, O_4)\}$.

按照 MOVIC 算法, 两种不同的操作执行顺序最终创建了一致的 $MCGS_4$.

3.3 MOVIC 算法分析

仔细研究可以发现, 当并发操作相容冲突关系比较复杂时, 基于相容组的 MOVIC 算法步 2 将导致以下两个问题:

(1) 当 n 个冲突操作拥有相同的相容操作时, 将生成 n 个 CG 且相同的相容操作将被存储 n 遍, 如果再执行一个新的操作 O_i , O_i 会多次和这些相同操作进行比较, 导致重复计算. 如上例第二种执行顺序的情况下, 由于 O_1, O_2 两个冲突操作都和 O_4 相容, 则 $MCGS_3$ 的每个 CG 中都有操作 O_4 , 因此 O_3 需要与 O_4 进行 2 次比较.

(2) 由于多个 CG 中含有相同的相容操作, 必然引起 CG 冗余问题, 导致最后一步去除冗余相容组的包含性检查次数增多. 如 O_3 执行于 $MCGS_3$ 时产生了两个相同的 CG, 去除冗余相容组时, 冗余的 CG 增加了一倍的相容组比较次数.

本文首次提出相容冲突组, 将多个冲突操作和它们共同的相容操作存储在一个相容冲突组中, 从而避免多次重复存储, 减少比较次数和去冗余时的包含性检查次数.

4 快速多版本增创算法 FMVIC

多版本技术的关键是使用多版本增创算法对一组并发操作进行分析以创建最终叶子版本, 本节将详细描述基于相容冲突组的快速多版本增创算法

(FMVIC).

定义 4. 冲突组 (Conflict Group).

设存在任意两个或两个以上的操作, 若这些操作相互冲突, 则将这些操作记为一个冲突操作组.

定义 5. 相容冲突组 (Compatible and Conflict Groups).

给定一组操作 GO , 如果其子集可被划分为相容组集和冲突组两部分, 前半部分相容组集包含一个或多个相容组, 各个相容组之间存在冲突关系, 但和后半部的冲突组操作都相容, 相容组集用 R_s 表示; 后半部为一个冲突组存放相互冲突的操作, 用 C 表示. 则此子集被记为一个相容冲突组.

定义 6. 相容冲突组集 (Compatible and Conflict Groups Set).

给定一组操作 GO , 若这组操作的多个相容冲突组能包含所有操作并能反映出这组操作之间的所有冲突相容关系, 则将这些相容冲突组记为一个相容冲突组集.

定义 7. 最大相容冲突组 (Maximal Compatible and Conflict Groups).

给定一组操作 GO 上的一个相容冲突组 CCG_x , 若任意属于 GO 但不属于 CCG_x 的操作都和 $CCG_x.R_s$ 的某一操作冲突, 则此 CCG_x 为一个最大相容冲突组.

定义 8. 最大相容冲突组集 (Maximal Compatible and Conflict Groups Set).

给定一组操作 GO 上的一个 $CCGS$, 若这个 $CCGS$ 中所包含的相容冲突组都是最大相容冲突组 $MCCG$ 并且包含所有属于 GO 的 $MCCGS$, 则此相容冲突组集记为最大相容冲突组集.

性质 1 (唯一性). 给定版本 V 上的一组操作 GO , 则存在与之对应的唯一的 $MCCGS$.

4.1 基于相容冲突组的分布式快速多版本增创算法

设版本 V 的一组操作以 O_1, O_2, \dots, O_n 的顺序执行. 此分布递增算法应产生一系列有序的 $MCCGS$: $MCCGS_1, MCCGS_2, \dots, MCCGS_n$. 下面具体描述快速多版本递增创建算法.

Algorithm FMVIC ($O_i, MCCGS_{i-1}$): $MCCGS_i$.

1. $MCCGS_i = \{\}$;

2. if $i = 1$, then $R_1 = \{O_1\}$; $CCG_{new}.R_s = \{R_1\}$;

$CCG_{new}.C = \emptyset$; $MCCGS_i = \{CCG_{new}\}$;

else

for every $CCG_m \in MCCGS_{i-1}$

(a) if $CCG_m.C = \emptyset$, then

if $O_i \odot CCG_m.R_1$, then

$CCG_m.R_1 = CCG_m.R_1 + \{O_i\};$
 else if there is only one operation that $\{O_i | (O \in CCG_m.R_1) \wedge (O \otimes O_i)\}$, then
 $CCG_m.R_1 = CCG_m.R_1 - O_i;$
 $CCG_m.C = \{O_i\} + \{O\};$
 else
 $R_{new} = \{O | (O \in CCG_m.R_x) \wedge (O \odot O_i)\};$
 $CCG_{new}.Rs = R_{new}; CCG_{new}.C = \{O_i\};$
 $MCCGS_i = MCCGS_i + \{CCG_{new}\}.$
 (b) else if $O_i \odot CCG_m.C$, then
 for every $R_x \in CCG_m.Rs$
 if $CCG_m.R_x \odot O_i$, then
 $CCG_m.R_x = CCG_m.R_x + \{O_i\};$
 else
 $R_{new} = \{O | (O \in CCG_m.R_x) \wedge (O \odot O_i)\} + \{O_i\};$
 $Rs_{temp} = Rs_{temp} + \{R_{new}\};$
 for every $R_{new} \in Rs_{temp}$
 if $R'_{new} \in Rs_{temp}$ and $R_{new} \subseteq R'_{new}$, then
 $Rs_{temp} = Rs_{temp} - R_{new};$
 $CCG_m.Rs = CCG_m.Rs + Rs_{temp}.$
 (c) else if $O_i \otimes CCG_m.C$, then
 if $O_i \odot CCG_m.Rs$, then
 $CCG_m.C = CCG_m.C + \{O_i\};$
 else
 for every $R_x \in CCG_m.Rs$
 if $O_i \odot CCG_m.R_x$, then
 $Rs_{temp} = Rs_{temp} + \{R_x\};$
 else
 $R_{new} = \{O | (O \in CCG_m.R_x) \wedge (O \odot O_i)\};$
 $Rs_{temp} = Rs_{temp} + \{R_{new}\};$
 for every $R_{new} \in Rs_{temp}$
 if $R'_{new} \in Rs_{temp}$ and $R_{new} \subseteq R'_{new}$, then
 $Rs_{temp} = Rs_{temp} - R_{new};$
 $CCG_{new}.Rs = Rs_{temp}; CCG_{new}.C = \{O_i\};$
 $MCCGS_i = MCCGS_i + \{CCG_{new}\}.$
 (d) else
 $C_1 = \{O | (O \in CCG_m.C) \wedge (O \odot O_i)\};$
 $C_2 = \{O | (O \in CCG_m.C) \wedge (O \otimes O_i)\};$
 if $O_i \odot CCG_m.Rs$, then $CCG'_m.C = C_1;$
 $CCG_m.C = CCG_m.C - C_1;$
 for every $R_x \in CCG_m.Rs$
 $R_{new} = R_x + \{O_i\};$
 $CCG'_m.Rs = CCG'_m.Rs + R_{new};$
 $MCCGS_i = MCCGS_i + \{CCG'_m\};$
 else
 for every $R_x \in CCG_m.Rs$,
 if $O_i \odot R_x$, then

$CCG_m.Rs = CCG_m.Rs - R_x;$
 $CCG'_{new}.Rs = CCG'_{new}.Rs + R_x;$
 $R_{new} = R_x + \{O_i\};$
 $Rs_{temp} = Rs_{temp} + \{R_{new}\};$
 else
 $R_{new} = \{O | (O \in R_x) \wedge (O \odot O_i)\} + \{O_i\};$
 $Rs_{temp} = Rs_{temp} + \{R_{new}\};$
 for every $R_{new} \in Rs_{temp}$
 if $R'_{new} \in Rs_{temp}$ and $R_{new} \subseteq R'_{new}$,
 then $Rs_{temp} = Rs_{temp} - R_{new};$
 $CCG_{new}.Rs = Rs_{temp};$
 $CCG_{new}.C = C_1;$
 $CCG'_{new}.C = C_2;$
 $MCCGS_i = MCCGS_i + \{CCG_{new}\} + \{CCG'_{new}\};$
 (e) $MCCGS_i = MCCGS_i + \{CCG_m\};$

3. We should check the redundant groups in $MCCGS_i$ if there are two situations as follows.

(a) if $O_i \in CCG_x.C$, and there is CCG_y in which the groups that include O_i also include $CCG_x.Rs$, then $CCG_x.C = CCG_x.C - O_i$;

(b) if $CCG_x.C \cap CCG_y.C = C_i \neq \emptyset$, $CCG_x.Rs \cap CCG_y.Rs = R_i \neq \emptyset$, C_i should be removed and CCG_{new} is created that $CCG_{new}.Rs = CCG_x.Rs \cup CCG_y.Rs$, $CCG_{new}.C = C_i$.

可以发现：最大相容冲突组的结构可以最大程度地保证提取出各组版本的相同相容操作，所以待执行操作和相同相容操作之间的比较次数将大大减少；由于各个组中重复存储相同操作的现象减少，所以需要进行包含性检查的相容冲突组的范围被有效地缩小了；尤其当所有的相互冲突的操作都拥有相同的相容操作关系时，各操作没有重复且仅被比较一次；将 $MCCGS$ 中各最大相容冲突组 $MCCGs$ 的 Rs 集合的各个相容组分别和 C 中各个元素相结合，便可以得到相应的各个版本即 $MCGS$ 。所以，FMVIC 算法得出的最终版本与 MOVIC 算法一样，并且其同样适用于复杂的扩展性冲突关系，但是性能比 MOVIC 算法好。

需要注意的是为了保证各个站点的存储结构的一致性，当操作 O_x 加入到组中时必须按操作的全序关系进行排列，特别是在合成新的相容冲突组时尤为重要。

4.2 快速多版本增创算法的正确性和次序无关性

定理 1(正确性). $GO = \{O_1, O_2, \dots, O_n\}$ 应用于同一个对象上，以 O_1, O_2, \dots, O_n 的顺序执行。由 FMVIC 算法得到的最大相容冲突组集 $MCCGS_n$ 是

GO 的 MCCGS.

证明. 讨论 $MCCGS_i (1 \leq i \leq n)$.

当 $i = 1$ 时, $MCCGS_1 = \{\{(O_1) | \}\}$ 是操作组 $GO_1 = \{O_1\}$ 唯一的 MCCGS.

假设 $i = m$ 时, $MCCGS_m$ 是操作组 $GO_m = \{O_1, O_2, \dots, O_m\}$ 的唯一的 MCCGS. 下面需证明 $i = m + 1$ 时, $MCCGS_{m+1}$ 是操作组 $GO_{m+1} = \{O_1, O_2, \dots, O_m, O_{m+1}\}$ 的唯一的 MCCGS.

(1) 首先证明 $MCCGS_{m+1}$ 中的每一个 CCG 都是 GO_{m+1} 的 MCCG.

因为 $MCCGS_m$ 是操作组 $GO_m = \{O_1, O_2, \dots, O_m\}$ 的唯一的 MCCGS. 所以, 根据 MCCGS 的定义可得, $MCCGS_m$ 中的所有 CCG 均是 GO_m 的 MCCG. 操作 O_i 到达时, 需根据 O_i 与 $MCCGS_m$ 中各个 CCG 之间的相容或冲突关系, 决定 $MCCGS_{m+1}$ 的生成情况. 但是, 无论 O_i 与其目标版本对应 CCG 之间具有什么关系, 根据 FMVIC 算法, O_i 总是要被结合到对应其目标版本的 CCG 中且仍满足 MCCG 定义. 因此, $MCCGS_{m+1}$ 中的每一个 CCG 都是 $GO_{m+1} = \{O_1, O_2, \dots, O_m, O_{m+1}\}$ 的 MCCG.

(2) 证明 GO_{m+1} 的所有 MCCG 都包含在 $MCCGS_{m+1}$ 中.

设 $MCCG_x$ 是 GO_{m+1} 的任意一个最大相容冲突组. 反设 $MCCG_x$ 不在 $MCCGS_{m+1}$ 中, 则 $MCCG_x$ 一定包含 O_{m+1} , 因为 GO_{m+1} 的所有不含 O_{m+1} 的 MCCG 继承了 $MCCGS_m$ 的特性, 已经直接包含在 $MCCGS_{m+1}$ 中了. 设 MCG_x 是 $MCCG_x$ 中对应的包含 O_{m+1} 的最大相容组, $MCG_x = CG_x + \{O_{m+1}\}$, 其中 CG_x 是 GO_m 的一个相容组. 则一定存在 GO_m 的一个 MCG_y , 有 $CG_x \subseteq MCG_y$, 并且包含 MCG_y 的 $MCCG_y$ 一定含在 $MCCGS_m$ 中. 根据 FMVIC 算法, O_{m+1} 应与 $MCCG_y$ 包含的操作进行比较从而建立 $MCCGS_{m+1}$, 其中一定有包含 $MCG_x = CG_x + \{O_{m+1}\}$ 的 $MCCG_x$, 与 $MCCG_x$ 不含在 $MCCGS_{m+1}$ 中的反设矛盾.

综上所述, 当 $i = m + 1$ 时, 定理成立.

性质 2 (一致性). 给定具有相同目标对象的一组操作 GO, 无论以什么次序执行, 由 FMVIC 算法得到的最大相容冲突组集合 $MCCGS_n$ 都是一致的.

证明. 由 FMVIC 算法的正确性定理及 MCCGS 的唯一性 (性质 1) 可直接得到此结论.

FMVIC 算法的执行顺序无关性保证了各个协同节点最终都可得到相同的数据结构和用户视图.

4.3 实例分析

本节针对上一节中的例子, 描述 FMVIC 算法的实现过程并与 MOVIC 算法进行比较. 同样给定 4 个并发操作 $(O_1 \otimes O_2 \otimes O_3) \odot O_4$, 其在两个站点的执行顺序分别为 O_1, O_2, O_3, O_4 和 O_1, O_2, O_4, O_3 .

在第一种执行顺序的情况下, 算法实现步骤如下: O_1 执行后, 将 O_1 加入到第一个相容冲突组的相容组中, 生成 $MCCGS_1 = \{\{(O_1) | \}\}$; O_2 执行后, 由于 $O_1 \otimes O_2$ 则将其加入到 $CCG_1.C$ 中, $MCCGS_2 = \{\{(O_1) | O_2\}\}$; O_3 执行后, 由于 $O_1 \otimes O_2 \otimes O_3$, 同理 $MCCGS_3 = \{\{(O_1) | O_2, O_3\}\}$; O_4 执行后, 由于 $O_4 \odot CCG_1.C$ 且 $CCG_1.Rs = \emptyset$, 所以 $CCG_1.R_1 = O_4$, $MCCGS_4 = \{\{(O_4) | O_1, O_2, O_3\}\}$.

在第二种执行顺序的情况下: $MCCGS_1 = \{\{(O_1) | \}\}$; $MCCGS_2 = \{\{(O_1) | O_2\}\}$; 当 O_4 执行后, 由于 O_4 和 O_1, O_2 都相容, $MCCGS_3 = \{\{(O_4) | O_1, O_2\}\}$; 当 O_3 作用于 $MCCGS_3$ 时, O_3 首先和 $CCG_1.C$ 即 (O_1, O_2) 比较, 再和 Rs 中的相容组 $R_1(O_4)$ 比较, 由于 $O_3 \otimes CCG_1.C$ 并且 $O_3 \odot CCG_1.Rs$, 所以将 O_3 加入到 $CCG_1.C$ 中, $MCCGS_4 = \{\{(O_4) | O_1, O_2, O_3\}\}$.

将 $MCCGS_4$ 中的 Rs 集合中的相容组分别和 C 集合中的操作进行结合, 即可得到最终各个版本: (O_1, O_4) , (O_2, O_4) , (O_3, O_4) .

通过和上一节中此例的执行过程比较可以发现, 新算法最终得到的各个版本和原算法一致, 其同样具有操作执行顺序无关性, 但在操作比较次数上少于原算法, 且此例中 FMVIC 算法无需进行去冗余操作, 性能明显得到提高.

5 模拟实验

我们用模拟实验对 MOVIC 算法和 FMVIC 算法的实现性能进行比较, 针对两个冲突关系分布不同的操作集 (11 个操作, 互相冲突对数为 10, 按照 O_1, O_2, \dots, O_{11} 的执行顺序), 分别进行 MOVIC 算法和 FMVIC 算法的模拟实验. 算法的运行时间主要由操作间的比较次数决定, 所以我们计算操作间的比较次数以比较算法的性能.

图 4 是两个冲突用例的冲突关系和与其相对应的实验结果示意图.

冲突用例 1 (冲突较集中于开始的操作).

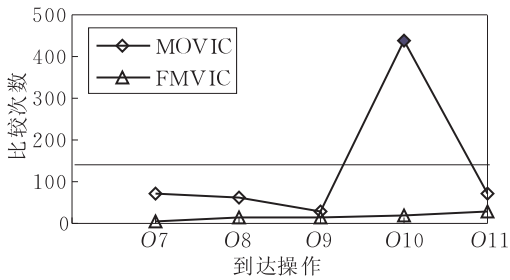
$O_1 \otimes O_2, O_1 \otimes O_3, O_1 \otimes O_4, O_2 \otimes O_3, O_2 \otimes O_4, O_3 \otimes O_4, O_4 \otimes O_6, O_4 \otimes O_8, O_5 \otimes O_7, O_9 \otimes O_{10}$.

冲突用例 2 (冲突较集中于中间的操作).

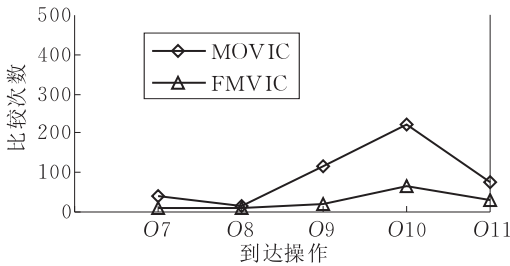
$O_3 \otimes O_4, O_3 \otimes O_5, O_3 \otimes O_{10}, O_4 \otimes O_5, O_4 \otimes O_6,$

$O_4 \otimes O_7, O_5 \otimes O_6, O_5 \otimes O_7, O_6 \otimes O_7, O_8 \otimes O_9$.

实验结果表明，不管冲突如何分布，FMVIC 算法比之 MOVIC 算法减少了操作间的比较次数. 随着操作数的增多，冲突操作对比较次数影响较小，这种性能的提高尤为重要.



(a) 基于冲突用例 1 的比较次数



(b) 基于冲突用例 2 的比较次数

图 4 MOVIC 算法和 FMVIC 算法随着执行到达操作的比较次数变化曲线

6 总 结

并发操作的意愿保证是协同图形编辑系统的核心技术之一. 本文在总结已有研究成果的基础上，介绍了基于版本复制的协同多版本技术，并根据简化规则提出新的策略. 为了确定最终叶子版本，在 MOVIC 算法的基础上，提出一种面向相容冲突集组的快速多版本增创算法 FMVIC. 通过理论和实验证明，FMVIC 算法不仅和 MOVIC 算法所构造的最终版本一致且同样适合复杂的扩展性冲突关系，但操作比较次数少，计算速度有很大提高，更适合应用于大规模的实时协同编辑系统中. 当版本过多时，如何智能化地选择版本、删除版本及合并版本都是

多版本技术中有待研究的问题.

参 考 文 献

[1] Ellis C A, Gibbs S J. Concurrency control in groupware systems//Proceedings of the ACM Conference on the Management of Data 1989. Portland Oregon, 1989: 399-407

[2] Sun C Z, Ellis C A. Operational transformation in real-time group editors: Issues, algorithms, and achievements//Proceedings of the ACM Conference on Computer Supported Cooperative Work. Seattle, USA, 1998: 59-68

[3] Ellis C A, Gibbs S J, Rein G L. Groupware: Some issues and experiences. Communications of ACM, 1991, 34(1): 39-58

[4] Sun C Z, Jia X, Zhang Y, Yang Y, Chen D. Achieving convergence, causality-preservation, and intention-preservation in real time cooperative editing systems. ACM Transaction on Computer-Human Interactions, 1998, 5(1): 63-108

[5] Yang Guang-Xin, Shi Mei-Lin. Object data model based concurrency control in fully-replicated architecture. Chinese Journal of Computers, 2000, 23(2): 113-125(in Chinese) (杨光信, 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, 23(2): 113-125)

[6] Sun C Z, Chen D. Consistency maintenance in real-time collaborative graphics editing systems. ACM Transactions on Computer-Human Interaction, 2002, 19(1): 1-41

[7] Sun C Z, Chen D. Intention preservation by object replication in cooperative graphics editing systems. Journal of Applied Systems Studies, 2000, 11(3): 345-360

[8] Xue L, Orgun M. A multi-versioning algorithm for intention preservation in distributed real-time group editors//Proceedings of the 25th Australasian Computer Science Conference. Australia, 2003: 19-28

[9] Bu J, Jiang B, Chen C. Maintaining semantic consistency in real-time collaborative graphics editing systems. International Journal of Computer Science and Network Security, 2006, 6(4): 57-61

[10] Dou W F, Zhu M, Shen Q. Cooperative multi-versioning technique based on version replication//Proceedings of the 10th International Conference on Computer Supported Cooperative Work in Design. Nanjing, 2006: 159-164

[11] Dou Wan-Feng, Li Chun-Ping. Object identification and its compression for multi-versioning technique. Journal of Software, 2004, 15(8): 1133-1140(in Chinese) (窦万峰, 李春萍. 多版本技术中的对象标识及其压缩. 软件学报, 2004, 15(8): 1133-1140)



YANG Jun, born in 1982, M. S. . Her research interests include CSCW and collaborative CAD.

DOU Wan-Feng, born in 1968, Ph. D. , associate professor. His research interests include CSCW, distributed computing and multimedia.

Background

Real-time collaborative graphics editing systems allow a group of users to view and edit the same graphics document at the same time from geographically dispersed sites connected via the Internet. To interact freely and naturally, the collaborative graphics editing systems should meet the following requirements: Interactive in real-time, collaborative distribution and non-restriction in operations. Consistency maintenance in the face of concurrent access to shared objects is one of the core issues in the design of these types of systems. It seems that the only way is to adopt a replicated architecture for the storage of shared documents. To support concurrent editing in the replicated architecture, consistency maintenance is one of the core issues.

There are four strategies to solve conflicts among operations, lock mechanism, serialization, dOPT and object replication. The multi-versioning technique based on object replication strategy can meet consistency model while the conflict operations occur in the cooperative graphics editing systems (CGES), but it does not efficiently solve conflict between non-geometry attribute operations and complex graphics objects. A new distributed multi-versioning model based on version replication and reduce strategy are proposed to guar-

antee intention of conflicting operations. Based on MOVIC algorithm, a Fast Multiple Versions Incremental Creation (FMVIC) algorithm based on the compatible and conflict group is presented in this paper. The final versions constructed by the FMVIC algorithm is the same as that constructed by the MOVIC algorithm, but it can decrease the comparison-number of compatible operations and narrow the scope of comparison operations for the removing redundant versions.

The research work is supported by the Natural Science Foundation for Higher School of Jiangsu Province of China under grant No. 07KJD520112. The project is aimed at related techniques in real-time collaborative editing systems. The research team has gained a fruitful achievement in some related areas, such as collaboration transparency, heterogeneous synchronized collaborative design. In consistency maintenance area, they have proposed a cooperative multi-versioning approach, developed different strategies and researched the object identification scheme on the multiple object versions. The work in this paper focuses on a new strategy about the version replication and improving efficiency of distributed algorithm for incremental creation of multiple versions.