协同编辑中维护操作意愿的文档标注方法*

何鸿君 吴泉源 罗 莉

(长沙工学院计算机系 长沙 410073)

摘要 在实时协同编辑系统中,操作意愿一致性维护是国际上提出的新概念,是协同系统一致性维护的重要方面.文档标注文法通过对共享文档进行恰当标注,屏蔽并发操作对共享文档造成的影响,使得操作在任意协作点上执行时的环境与操作产生的环境一致,从而有效地维护操作意愿一致性.结合实际应用中的具体操作,重点论述文档标注方法以及相应的控制算法.

关键词 CSCW,实时协同编辑,操作意愿维护,文档标注方法.

中图法分类号 TP311

实时协同编辑系统支持物理上分布的多个协作者同时编辑同一文档,这是一类重要的CSCW(computer-supported cooperative work)应用系统.实时协同编辑系统是典型的分布式应用系统,除要求对用户操作有很快的响应速度外,通常还有以下特点:对同一时刻进行操作的协作用户数目没有限制,对协作用户可以编辑的文档部分也不加制约,用户可在任意时刻编辑共享文档的任意部分,如同GROVE,REDUCE等系统所倡导的那样[1~3];所有结点都在本地存储器保留有共享文档的整个拷贝,结点之间传送的信息主要是操作信息和参数以及必要的控制信息,这种体系结构允许操作首先在本地执行,操作在本地的响应时间很短.

维护操作意愿(Intention Preservation of Operation)是国际上提出的新概念,旨在使用户产生该操作时希望达到的效果与该操作实际执行后所产生的效果一致,属于并发操作的一致性维护范畴.分布式操作系统和数据库系统等,对并发操作及其一致性维护问题进行了广泛而深入的研究,有大量可资使用的成果.但是,随着网络计算与协同工作研究的发展,这种一致性维护问题因操作的并发和网络传输延迟的不确定性而大大复杂化了.操作意愿的维护便是在这种背景下提出的.目前,维护操作意愿已成为协同工作,特别是实时协同编辑系统研究的挑战性研究课题[1~5].本文提出的文档标注方法,是实现操作意愿维护的有效途径.

本文首先讨论操作意愿问题的若干重要概念,简单介绍目前国际上解决操作意愿维护问题的主要成果,即操作变换方法的基本思想.然后,提出文档标注方法,结合具体实例进行说明,并详细论述该方法在实时协同编辑系统中维护操作意愿的控制算法.最后,总结文档标注方法的特点,并给出进一步的研究计划.

1 操作意愿问题

或许是由于其隐蔽性,或许是由于网络计算的发展时间还不长,操作意愿维护问题长期以来没有得到足够的重视,对问题的本质和复杂性缺乏深刻了解.C. Sun博士于近年在世界上首先明确地提出了操作意愿(Intention of Operation)的概念,并将操作变换方法应用于实时协同文本编辑系统中操作意愿的维护[4,5].

定义 1. 操作意愿.

操作O的操作意愿是指在O产生时的环境下,操作O所要完成的动作.在操作O产生时的文档状态下,执行O得到的结果显然满足操作意愿.

---

* 本文研究得到国家863高科技项目基金资助.作者**何鸿君**,1968年生,博士,讲师,主要研究领域为分布计算技术,多媒体技术.**吴泉源**,1942年生,教授,博士生导师,主要研究领域为专家系统,分布计算技术.**罗莉**,女,1971年生,博士,讲师,主要研究领域为人工神经网络,分布计算技术.

本文通讯联系人:何鸿君,长沙 410073,长沙工学院计算机系601教研室

本文1997-10-22收到原稿,1998-02-17收到修改稿

在实时协同编辑系统中,一个操作在远程结点执行产生的效果,若不加以调整和控制,则可能与操作在本地执行时产生的效果不一样,即与用户希望达到的效果不一致.不妨假定 O_1 和 O_2 分别是在结点 A,B 上同时产生的操作,并且 A,B 结点的文档状态在 O_1 和 O_2 产生前是相同的.那么,根据实时协同编辑系统的特点,O_1 和 O_2 产生后首先在本地结点执行,然后广播到所有其他协作结点并执行.当 O_1 到达结点 B 时,结点的文档状态由于执行了 O_2 而发生了变化,与 O_1 产生时的文档状态不一致,若简单执行 O_1,则很可能产生错误的结果.同样,O_2 到达结点 A 时,也将遇到类似的问题.

现考察一个具有插入、删除操作的实时协同文本编辑系统.如图1所示,考虑 O_1,O_2 在结点 N_0 和 N_1 上的执行情况.假设共享文档的初始状态为"*ABCDEF*",$O_1=O_2=remove[1,3]$,表示删除位置3(记第1个字符的位置为0)开始的1个字符,即都想删除字符"*D*".显然,所有结点上应该得到的正确结果是"*ABCEF*".然而,得到的结果很可能是"*ABCF*".实际上,这类例子俯拾皆是.仍以图1为例,且共享文档的初始状态仍为"*ABCDEF*",设 $O_1=insert[$"11",1$]$,$O_2=insert[$"22",3$]$,$O_3=remove[3,0]$,为3个不同结点上同时产生的并发操作,分别表示在位置1、3之前插入字符串"11"、"22"以及删除"*ABC*".正确的结果是"1122*DEF*",而按 O_1,O_2,O_3 的6种排列中的任何一种顺序执行,都不可能得到正确结果.操作意愿违背问题,本质上与操作的执行顺序无关,它是由于操作的并发而引起的,而网络分布计算使问题变得更加复杂.可以看出,这种来自实时协同编辑系统的一致性维护问题,具有相当的挑战性,现有分布式计算的控制算法不能直接用来解决问题[6~8].

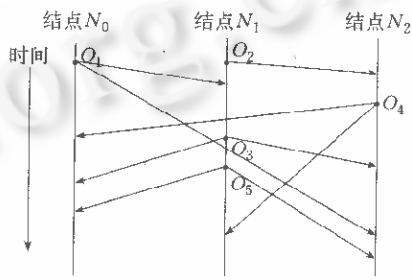

图1　实时协同编辑系统的操作关系示例

2　维护操作意愿的现有方法

操作变换方法是目前国际上研究操作意愿维护而提出的一种有效方法.将操作变换思想用于实时协同编辑系统,C. A. Ellis 等人做了先驱性的工作[1,2].其后,C. Sun 博士等人在其基础上继续深入研究,发现了原来方法存在的问题,并首次明确提出了维护操作意愿的概念,操作变换方法得到了进一步的发展和完善[3~5].

操作变换的基本思想是:操作执行前对操作的各参数形式进行调整,以补偿由于执行了其他操作而引起的文档变化.有两种主要的操作变换,分别称为包含变换(Inclusion Transformation)和剔除变换(Exclusion Transformation)[1,3~5].包含变换的作用是把一个操作对文档造成的影响有效地包含进另一个操作里,即变换后的操作参数已经考虑了另一个操作对文档造成的影响.剔除变换的作用是消除一个操作对另一个操作的影响.操作变换的实现与具体应用相关,需根据实际的操作语义来设计相应的算法.不同的应用,对于任一操作,如何得到操作的正确变换形式,是对变换操作进行调度的过程,其控制算法是通用的[1,4,5].

举个简单例子,如图1所示.设共享文档的初始状态为"*ABCDEF*",$O_1=insert[$"11",1$]$,即在"*A*"与"*BCDEF*"之间插入"11",设 $O_2=remove[3,2]$,即删除"*CDE*".O_1 在结点 N_0 执行后,文档变为"*A*11*BCDEF*",O_2 到达结点 N_0 时,通过操作变换后变为 $O'_2=remove[3,4]$,执行得到文档"*A*11*BF*".

应该指出的是,复杂的操作关系使得实施操作变换并不容易.例如,将上面例子中的 O_1 改为 $insert[$"11",3$]$,则 O_2 变换后应得到两个操作 $remove[1,2]$和 $remove[2,5]$.

3　文档标注方法

可以看出,操作变换方法着眼于操作本身,试图通过变换操作的参数形式来实现操作意愿维护.由于操作变换算法与操作的语义紧密相关,特别是,由于网络传输延迟的不确定性,当结点规模稍大时,操作之间的可能关系变得非常复杂,考虑全面很不容易,并且实施操作变换时还会遇到不少需要特别处理的情况.[5]因此,操作变换算法设计的难度和技巧性要求都很高.

文档标注方法则立足于共享文档本身.无论某一操作执行前发生了多少并发操作,通过对共享文档加标注的手段,把并发操作执行所引起的文档变化部分屏蔽起来,使该操作执行时的文档状态仍然与该操作产生前瞬

间的文档状态一致，从而实现操作意愿的维护.具体方法如下：设操作O产生前瞬间的文档状态为DOC，当O传送到远程结点执行时，若有O的并发操作在该远程结点上已经先执行，则通过对文档加标注的手段，隐藏由于先执行的并发操作所引起的文档的变化部分，使得该远程结点的文档状态仍然是DOC，然后执行O，最后，去除所加的标注，以恢复隐藏的文档部分.

根据我们目前的研究，文档被标注部分的类型有两种：(1) 不可见.表示从操作的角度看，这部分文档是不存在的，而实际上这部分文档是存在的，即对于操作而言是不可见的.(2) 可见但实际不存在.表示从操作的角度看，这部分文档是可见的，但实际上已经被前面执行的某个操作删除了.

删除操作需要重点关注.执行删除操作时，如果将文档的相应部分真正删除掉了，那么其后将该删除部分恢复出来是很不方便的.其原因在于：操作的执行结果和当时的系统状态是相关的，而执行操作后，系统状态即发生了改变；操作的逆操作要正确执行，往往还需要用到这些系统状态数据.一种可行的做法是：当执行删除操作时，并不真正将文档的对应部分删除掉，而是做上相应标记.不少文件系统、数据库系统为了提供被删除文件、数据记录的恢复功能都采用了类似技术.

我们通过上节中的例子来说明文档标注方法.如图1所示，设共享文档的初始状态为"$ABCDEF$"，$O_1=insert$["11", 3]，$O_2=remove$[3, 2]，考察O_1和O_2在结点N_0和N_1上的执行情况.假定用下划线代表"不可见"标注，用删除线代表"可见但实际不存在"标注.

结点N_0：(1) O_1执行后，文档状态变为"$ABC11DEF$"；(2) O_2到达本结点，由于O_1是O_2的并发操作，因此，需要屏蔽O_1所引起的文档变化部分，即"11".对"11"加下划线标注，文档状态变为"$ABC\underline{11}DEF$".执行O_2，删除位置2开始的3个字符，由于"$\underline{11}$"带有下划线标注，不在删除之列，因此，删除的字符是"CDE"，从而得到"$AB\underline{11}F$".然后，去掉"$\underline{11}$"上的标注.

上一节中提到，采用操作变化方法，O_2变换后应得到两个操作.而这里采用文档标注方法，O_2不需作任何变化，执行时只需记住带下划线标注的字符不在删除之列.

结点N_1：1) O_2执行后，用户看到的文档为"ABF".由于O_2是删除操作，所以，如前所述，并不实际把"CDE"从文档中除去，文档状态仍为"$ABCDEF$"，但需要在"CDE"的各字符上标明操作O_2，表示被操作O_2所删除，并且不被显示出来；2) O_1到达本结点，由于O_2是O_1的并发操作，因此，需要屏蔽O_2所引起的文档变化部分.对"CDE"加上删除线标注，文档状态变为"$ABC\sout{DE}F$".执行O_1，将"11"插入位置3(即D)之前，得到"$ABC11\sout{DE}F$".接着，去掉"C"、"DE"上的删除线标注，用户看到的文档为"$AB11F$".

4 控制算法

文档标注方法的实现可分为两个部分.一部分是如何对文档进行标注，算法与具体应用相关.另一部分则是高层控制算法，它与具体应用无关，负责根据当前到达的操作与已经执行了的操作的关系，调度文档标注算法，以使操作所看到的文档状态与该操作产生时本地结点的文档状态相同，然后执行操作，满足操作意愿要求.

定义 2. 结点的状态NS.

结点的状态由在该结点上已经执行了的操作序列表示.例如，结点i上已经按顺序执行了操作O_1，O_2和O_3，那么，结点i的状态NS_i可以表示为一个表$[O_1,O_2,O_3]$.表尾元素是最近执行的操作.

定义 3. 操作的上下文CT.

操作的上下文定义为操作产生时本地结点已经执行了的操作序列.例如，操作O在结点i上产生的瞬间，结点i上已经按顺序执行了操作O_1，O_2和O_3，那么，操作O的上下文CT_O可以表示为一个表$[O_1,O_2,O_3]$.

在控制算法中需要用到操作的上下文信息，因此，操作的上下文信息连同操作本身打包在一起，然后传送到各个协作结点.

不同的应用和操作，给文档加标注的算法的具体实现不同，但功能思想是一致的.为讨论方便，定义以下两个函数.

(1) *Document markDoc*(*Document doc*, *Operation op*)；

前提条件：结点的文档状态为doc，执行了的操作为op.

功能描述:将操作 op 对文档所造成的影响进行标注,以隐藏起 op 执行后文档所发生的变化.

结果条件:文档将发生变化到一个新的状态 doc'. 设 op 执行前文档的状态为 doc'',则 $markDoc()$执行后,从操作的角度看,文档状态仍然是 doc'',即看不出 doc' 与 doc''的区别.

(2) $Document\quad demarkDoc(Document\quad doc,\quad Operation\quad op)$;

前提条件:结点的文档状态为 doc,执行了的操作为 op,并且操作 op 对文档 doc 造成的影响已经通过调用 $markDoc()$被隐藏起来.

功能描述:将文档中针对 op 所加的标注除去.

结果条件:文档将发生变化到一个新的文档状态 doc',在 doc' 中操作 op 对文档造成的影响被显示出来.

上述两个函数满足以下关系:

$$demarkDoc(markDoc(DOC,op),op) = DOC.$$

任意一个操作,如果产生于本地结点,则立即执行;如果产生于远程结点,其执行由以下算法控制.

控制算法　操作 O 产生于远程结点,设当前结点的状态为 NS,文档状态为 DOC,操作 O 的上下文为 CT_O,按以下步骤执行:

(1) 如果 $NS=CT_O$,则执行 O. 算法结束;

(2) 如果(op 是 CT_O 的元素,并且不是 NS 的元素),则延迟执行 O. 算法结束;

(3) (a) 置 $ListTmp$ 为空表.

(b) 从右向左扫描 NS,如果(op 是 NS 的元素,并且不是 CT_O 的元素),则

{$DOC=markDoc(DOC,op)$;

将 op 放入 $ListTmp$ 的表尾;

}

(4) 执行 O;

(5) 从右向左扫描 $ListTmp$,对于 $ListTmp$ 的元素 op,执行

$$DOC = demarkDoc(DOC,op);$$

(6) 算法结束.

当一个操作到达结点时,如果需要进行文档标注,除了操作的上下文信息外,还应当知道结点的状态信息,才能使标注操作得以正确实施. 如果在结点上保存所有操作的信息,会造成信息大量冗余,消耗过多内存资源,效率降低越来越严重. 那么,该如何确定需保存的操作信息呢? 我们有下面的定理.

定理 1. 设集合 S 所包含的操作在所有协作结点上都被执行,则结点可以不再保存集合 S 所包含的操作的信息.

根据上述定理,可以构造不同形式的算法,以实现操作信息的保存.

我们由上节中的例子来进一步说明. 如果确信 O_1 和 O_2 已经在所有结点上被执行,那么,根据定理 1,在结点 N_1 和 N_2 上,"CDE"可以正式从文档中除去.

5　结束语

维护操作意愿是一项复杂的、颇具挑战性的课题,国际上的研究也刚刚起步,我们目前所取得的成果只是迈向成功的一小步. 文档标注方法是维护操作意愿的一种新方法,它具有易于理解、通用的特点,算法设计也比较直观,但还不完善. 在今后的研究中,我们将进一步发展和完善文档标注方法,并考虑把它应用到分布式计算的其他领域.

致谢　C. Sun 博士在华讲学的内容对我们启发很大,特别是对本文的写作提出了很多宝贵的具体意见. 另外,我们与王怀民博士还进行过多次有益的讨论. 在此,谨向他们表示我们诚挚的谢意.

参考文献

1 Ellis C A, Gibbs S J. Concurrency control in groupware systems. In: Baltimore ed. Proceedings of ACM SIGMOD Conference on Management of Data. New York: ACM Press, 1989. 399～407

2 Ellis C A, Gibbs S J, Rein G L. Groupware: some issues and experiences. Communications of ACM, 1991,34(1):39～58

3 Sun C *et al*. REDUCE: a prototypical cooperative editing system. In: Smith M J *et al* eds. Proceedings of the 7th International Conference on Human-Computer Interaction. San Fransisco: Elsevier Press, August, 1997. 89～92

4 Ressel M, Nitsche-Ruhland D, Gunzenbauser R. An integrating transformation-oriented approach to concurrency control and undo in group editors. In: Turner J ed. Proceedings of ACM Conference on Computer Supported Cooperative Work. 1996. 288～297

5 Sun C *et al*. A generic operation transformation scheme for consistency maintenance in real-time cooperative editing system. In: Payne S C, Prinz W eds. Proceedings of the International Conference on Supporting Group Work (GROUP'97). Phoenix, Arizona USA, Nov. 1997. 425～434

6 Tanenbaum A S. Distributed Operating System. Beijing: Tsinghua University Press, 1996. 119～158

7 Gerard Tel. Introduction to Distributed Algorithms. London: Cambridge University Press, 1994

8 Stefano Ceri *et al*. Distributed Databases: Principles and Systems. New York: McGraw-Hill Book Company, 1984. 209～244

Document Marking Scheme for Preserving Intention of Operation in Cooperative Editing System

HE Hong-jun WU Quan-yuan LUO Li

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

Abstract Intention preservation of operation is a new proposed concept in the field of CSCW (computer-supported cooperative work), and the key feature of consistency control. A new called document marking scheme for intention preservation is proposed in this paper. By properly marking the sharing document, the document marking scheme can hide the effect of all concurrent operations of one operation. Therefore, operation can be performed at any cooperative side under the document environment just like it produced. Associating operations in a real-time cooperative editing system, the idea and the high-level control algorithm of the new method are presented in detail.

Key words CSCW(computer-supported cooperative work), real-time cooperative editing, intention preservation of operation, document marking scheme.