

文章编号:1001-9081(2006)11-2749-04

一种扩展单用户应用程序实现协同编辑的方法

李 林,曾广周

(山东大学 计算机科学与技术学院,山东 济南 250061)

(lilin_jiamhere@mail.sdu.edu.cn)

摘 要:为了实现协同编辑,提出了一种对单用户应用程序进行扩展的方案。方案支持多个用户间无限制的协同编辑,使用操作变换机制维护数据的一致性。为了能充分利用操作变换机制,在单用户应用程序的外部设置了一个线性表用以索引文档中的对象;同时在控制算法层对 David Sun 的扩展 OT 算法进行了改进,以更好的支持对具有复杂对象结构的文档的共享控制。

关键词:协同编辑;单用户应用程序;操作变换

中图分类号: TP311;TP399 **文献标识码:** A

A collaborative editing approach based on single-user application extending

LI Lin, ZENG Guang-zhou

(School of Computer Science and Technology, Shandong University, Jinan Shandong 250061, China)

Abstract: An approach to extend single-user application was proposed to support collaborative editing. The approach supported unconstrained collaboration, and used operational transformation to maintain data consistency. In order to make best use of operational transformation, a linear sequence outside single-user application was used to index the objects in documents. David Sun's extending operational transformation algorithm was improved on the control layer to support documents sharing control of complex object structure.

Key words: collaborative editing; single-user application; operational transformation

0 引言

计算机支持的协同工作(Computer Supported Cooperative Work, CSCW)技术的发展促成了很多群件性质的协同编辑工具的诞生,如 Grove^[1], REDUCE^[2], GRACE^[3]。一些 workflow 管理系统中也提供了具有协同交互能力的表单工具,如 Wowww! 系统^[4,5]。这些协同工具可以很好地支持多人之间的协作,但作为编辑工具其功能和可用性却远不如一些常用的单用户应用程序,如 MS Word。所以在办公、生产中,人们往往不情愿放弃好用的单用户应用程序而去使用相对简陋的协同编辑工具。

对单用户应用程序进行扩展使其能支持协同编辑是一个很好的解决方案。在不改变源代码的前提下,使多个用户使用单用户应用程序进行协同编辑,既能够支持多人间的交互和协作,又保留了单用户应用程序的强大的功能和良好的可用性。

在协同编辑中,为了使多个用户能并发地,不受其他用户干扰地对共享文档进行编辑,必须支持无限制的协同。所谓无限制的协同^[6],是指在协同编辑期间的任何时刻,每个用户都能够并发地,并且是自由地编辑共享文档的任意部分。

现在大多数的单用户应用程序共享系统,无论是采用有中心结构还是复制结构,无论是采用视图级共享还是命令级共享,都使用令牌轮转机制来分配控制权,即在同一时间只能有一个用户(主用户)拥有控制权,其他用户(从用户)只能被

动地接受主用户执行的操作。一个用户要想获得控制权,必须首先获得地板控制权(floor-control)。这样很容易保持数据的全局一致性,但极大地限制了用户的自由操作,多个用户不能并发地进行操作,从而只能提供有限的协作支持,不能满足无限制协作的要求。

要实现无限制的协同,可以借鉴 Grove, REDUCE 这些群件的协同机制。在这些系统里,一种被称作操作变换(Operational Transformation, OT)^[7]的乐观的并发控制机制,被用来维护文档在全局的一致性。在每个客户端都拥有共享文档的一个副本,所有的用户都可以自由地对文档进行编辑,这些操作在本地立即执行,然后把操作广播给其他的客户端,在远程客户端执行操作之前先要进行操作变换。

Du Li 提出了 ICT(Intelligent Collaboration Transparency)方法^[8]将操作变换机制引入到单用户应用程序共享系统中。在该方法中,用户的鼠标/键盘操作被截获下来,转换成带有语义的应用级操作(如插入字符、删除字符)。这些应用级操作被发送到其他客户端,并利用操作变换机制进行一致性维护,在远程客户端执行时要把应用级操作重新转换成鼠标/键盘事件进行回放。ICT 方法只针对文本文档的协同编辑,给出了变换方法,没有说明如何支持图像等复杂对象的编辑,大大限制了它在实际中的应用。

受 ICT 的启发,David Sun 提出了 CoWord 方法^[9],它不仅能处理文本,而且能处理像 Word 文档这样复杂的对象结构。David Sun 在 GOTO 算法^[5]的基础上对 OT 算法进行了扩

收稿日期:2006-05-15;修订日期:2006-07-05

作者简介:李林(1982-),男,山东聊城人,硕士研究生,主要研究方向:协同编辑、工作流;曾广周(1947-),男,山东郓城人,教授,博士生导师,主要研究方向:计算机支持的协同工作、智能计算与移动计算、工作流管理。

展^[10],增加了一个“Update”元操作以支持对复杂对象的操作,在控制算法层他沿用了 GOTO 算法^[5]的策略。但 Abdessamad Imine 等指出 GOTO 算法在某些特定情况下不能保证数据在全局的一致性^[11],其原因就在于 GOTO 算法的控制层算法并不完备。因此有必要对 David Sun 的扩展 OT 算法进行改进,使之能很好地维持数据在全局的一致性。

本文提出了一个对单用户应用程序进行扩展的方案,用于支持协同编辑,同时给出了一个改进的扩展 OT 算法,以维护协同编辑中具有复杂对象结构的文档在全局的一致性。

1 扩展单用户应用程序实现无限制协同

要实现无限制协同,关键就在于利用操作变换机制维持数据的全局一致性。操作变换要求共享文档中的对象存储在一维或多维的地址空间中,并且可以使用位置作为索引定位对象^[9]。在 MS Word 提供的 API 中,一个 Word 文档对应于一个 Document 对象,一块连续的 Word 文档区域对应一个 Range 对象。在 CoWord 方法中,利用 Range 对象中包含的线性表来对文档内部的对象进行定位,从而达到了操作变换所要求的条件。然而在不同的单用户应用程序中,文档内部对象的存储方式是各不相同的。为了对尽可能多的单用户应用程序进行扩展,本文的方法是在应用程序的外部设置一个线性表用来索引文档中的对象,其结构如图 1 所示。在每个客户端都有一个单用户应用程序的实例(包含了共享文档的副本),以及一个用于索引对象的线性表。

位置	1	2	3	...	N
对象指针	对象1指针	对象2指针	对象3指针		对象N指针

图 1 线性表

操作变换的作用对象是对线性表的操作。对线性表的操作有三种: Insert(pos, obj), Delete(pos) 和 Update(pos, name, value)。其中 Update 的参数 name 代表对象属性的名称, value 代表属性的值。本地客户端将对线性表的操作广播给其他的客户端,利用操作变换机制维护各个客户端的线性表的一致性。由于最终的目的是对单用户应用程序中的文档进行协同,线性表要同应用程序中的文档保持同步。一方面,用户通过单用户应用程序对本地文档进行了修改,线性表中的内容也要进行相应的改变;另一方面,从其他客户端来的对线性表的操作,也得引起单用户应用程序中的文档进行相应更新。

用户利用单用户应用程序对文档的操作是多种多样,如剪切粘贴一段文本,改变某个图像的大小,改变文本的字体,删除某个表格等。这些操作最终都可以转化为三种元操作:插入对象、删除对象和更新对象属性。这三种元操作映射到线性表上,分别表现为:插入节点、删除节点和更新节点的属性。从图 2 中我们看到这种映射关系。

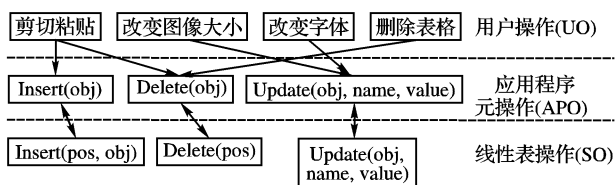


图 2 不同层次操作之间的映射

当用户利用单用户应用程序对文档进行操作时,截获其鼠标/键盘事件。调用应用程序的 API,根据鼠标/键盘事件提供的参数解析出被操作对象的指针以及被修改的属性名和

属性值,从而映射为应用程序元操作。然后在线性表中查找到对象该所在的位置,进一步映射为线性表操作。对线性表执行了相应操作以后,需要将线性表操作广播给其他客户端。

当某个客户端收到其他客户端发来的线性表操作时,首先对其进行操作变换,然后根据线性表操作的位置参数 pos 查找到相应的对象指针,从而将线性表操作映射为应用程序元操作,最后调用相应的 API 对文档进行更新。

在客户端之间传播的是线性表操作,操作变换对线性表操作进行并发控制,以维持文档在全局的一致性。传统的 OT 算法只能处理插入和删除操作,为了支持更新对象属性的操作,必须将 OT 算法进行扩展。

2 改进的扩展 OT 算法

OT 算法可以分成两层,下层是变换函数,上层是控制算法。变换函数决定了一个操作针对另一个操作如何进行变换,操作参数怎样进行调整;控制算法则负责对本地的操作队列进行管理,一个远程操作到达时与队列中的哪些操作进行变换,以何种顺序与这些操作进行变换。从整体上来看,控制算法要对变换函数进行调用。David Sun 在变换函数层对传统的 OT 算法进行了扩展,下文将首先介绍这种扩展,然后在控制算法层对 David Sun 的算法进行改进。

2.1 扩展变换函数

在 GOTO 算法中变换函数分为两类。一类是 Inclusion Transformation 函数 $IT(O_1, O_2)$, O_1 针对 O_2 做变换,使 O_1 在不影响自身操作效果的基础上能包含 O_2 的操作效果;另一类是 Exclusion Transformation 函数 $ET(O_1, O_2)$, 使 O_1 在不影响自身操作效果的基础上能去除 O_2 的操作效果。

上文提到对线性表的操作有三种: Insert(pos, obj), Delete(pos) 和 Update(pos, name, value)。GOTO 算法的变换函数只能支持 Insert 和 Delete 两种操作。David Sun 在此基础上提出了扩展的变换函数,可以支持 Update 操作。下面给出扩展后的 IT 和 ET 的算法描述。为了叙述方便,在不改变算法流程的基础上采用了比 David Sun 更为紧凑的描述方式。O 代表操作, O. type 代表操作的类型(可以为 INSERT, DELETE, UPDATE), O. p 代表操作的位置参数, O. obj 代表操作的对象, O. name 和 O. value 代表操作的属性名参数和属性值参数, O. id 代表操作者的用户标识。

算法 1 扩展的 IT 函数

```

IT(  $O_1, O_2$  ):  $O_1'$ 
 $O_1' = O_1$ ;
if (  $O_1.p > O_2.p$  )
    if (  $O_2.type = INSERT$  )
         $O_1'.p += 1$ ;
    else if (  $O_2.type = DELETE$  )
         $O_1'.p -= 1$ ;
else if (  $O_1.p = O_2.p$  )
    if (  $O_1.type = INSERT$  and
         $O_2.type = INSERT$  and  $O_1.id < O_2.id$  )
         $O_1'.p += 1$ ; //利用比较用户 id 来解决插入冲突
    else if ( (  $O_1.type = DELETE$  or  $O_1.type = UPDATE$  ) and
         $O_2.type = INSERT$  )
         $O_1'.p += 1$ ;
    else if ( (  $O_1.type = DELETE$  or  $O_1.type = UPDATE$  ) and

```

```
O2.type = DELETE)
O1'.p = NOP; //空操作
else if (O1.type = UPDATE and
O2.type = UPDATE and O1.name = O2.name and
O1.id < O2.id)
O1'.value = O2.value; //比较用户 id 来解决插入冲突
```

算法 2 扩展的 ET 函数

```
ET(O1, O2): O1'
O1' = O1;
if (O1.p > O2.p)
if (O2.type = INSERT)
O1'.p -= 1;
else if (O2.type = DELETE)
O1'.p += 1;
else if (O1.p = O2.p)
if ((O1.type = DELETE or O1.type = UPDATE) and
O2.type = INSERT)
throws Exception; //无效的变换
else if (O2.type = DELETE)
O1'.p += 1;
```

在 IT 中,当 $O_1.p = O_2.p$ 时需要解决两种冲突。一种是多个用户同时同一位置插入对象的冲突,采用比较用户 id 的方法解决冲突,当 $O_1.id < O_2.id$ 时,将 O_1 的位置后移。一种是多个用户并发更新同一对象的同一属性的冲突,采用类似的方法解决冲突,当 $O_1.id < O_2.id$ 时,将 O_1 的 value 替换为 O_2 的 value。

在 ET 中,如果 O_1 是删除或更新 O_2 插入的对象时, O_1 针对 O_2 的 ET 变换是无意义的。

2.2 控制算法

David Sun 的扩展 OT 算法在控制算法层沿用了 GOTO 算法的控制策略。在 GOTO 算法中,每个客户端都维持一个历史操作队列(HB)。当一个远程操作 O 到达时,首先判断它的前序操作是否都已经执行,如果没有则需要等待。然后将 O 针对 HB 中所有与 O 并发的操作进行变换。GOTO 中采用的策略是先将 HB 中与 O 并发的操作移到右侧,即将 HB 分成左右两部分: O 的前序操作的队列 Q_h 和 O 的并发操作的队列 Q_c 。接下来将 O 针对 Q_c 中的操作依次进行变换,得到 O' ,执行 O' 并将 O' 添加到 HB 的末尾。

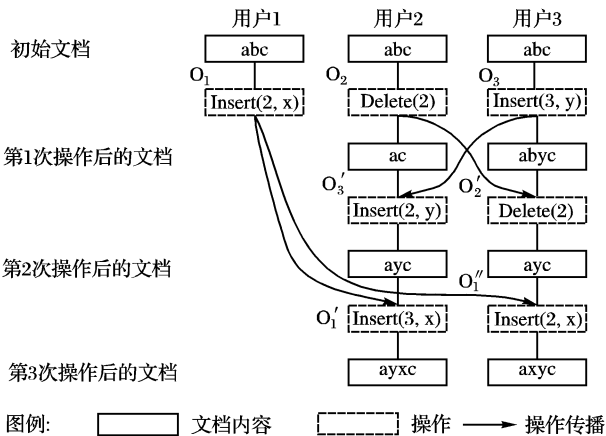


图 3 文档在全球不一致的例子

GOTO 的这种策略是不完备的,在特定情况下不能保证文档在全球的一致性。考虑这样一种情况,初始文档为

“abc”,用户 1,2,3 在同一时间分别执行了操作 $O_1: Insert(2, x), O_2: Delete(2), O_3: Insert(3, y)$,假设用户 1,2,3 的用户标识号依次增大。假设操作在用户 2,3 的客户端上的到达顺序分别是 $O_2 O_3 O_1, O_3 O_2 O_1$ 。结果如图 3 所示。

文档在用户 2 和 3 的客户端上出现了不一致的情况,其原因是 O_2 删除了两个 Insert 操作 O_1 和 O_3 之间的内容,导致在不同的路径($O_2 O_3$ 和 $O_3 O_2$)中针对 O_3 对 O_1 进行变换时,变换函数采取了不同的策略。一种是根据位置调整参数(在用户 3),另一种则是根据比较用户之间的 id 来调整参数(在用户 2)。由于第二种策略可以采用两种相反的方式,必然有一种与第一种策略存在冲突。当 O_1 和 O_3 同时为 Update 操作时,会发生类似的情况。

为了消除这种情况的发生,可以将 Q_c 中的 Delete 操作移动到右侧,即将 Q_c 再分为左右两部分:包含 Insert 和 Update 操作的队列 Q_{cin} 和包含 Delete 操作的队列 Q_{cd} ,然后再将 O 针对 Q_c 中的操作依次进行变换。

GOTO 算法给出了互换两个操作的执行顺序又同时保持两者的操作效果的函数 $Transpose(O_1, O_2)$ ^[5]。

算法 3 Transpose 函数

```
Transpose(O1, O2): (O1', O2')
O2' = ET(O2, O1)
O1' = IT(O1, O2')
```

下面利用 Transpose 实现对 Q_c 中的 Delete 操作右移的函数 RightShiftDel。

算法 4 RightShiftDel 函数

```
RightShiftDel(Q): Q'
Qin = []; Qd = [];
for (i = 1; i <= |Q|; i++){
if (Q[i].type = DELETE)
Qd = [Qd, Q[i]]; //将 Q[i] 添加到 Qd 的末尾
else {
for (j = |Qd|; j >= 1; j--)
(Q[i], Qd[j]) = Transpose(Qd[j], Q[i]);
Qin = [Qin, Q[i]]; //将 Q[i] 添加到 Qin 的末尾
}
}
Q' = [Qin, Qd];
```

需要注意的是 RightShiftDel 中在计算 $ET(O_1, O_2)$ 时 O_2 总是 Delete 操作,所以不会出现 ET 函数中无效变换的情况。

实现了对 Q_c 中的 Delete 操作右移之后,下面给出改进后的控制算法 TransformRO。输入是历史操作队列 HB 和新到的远程操作 O ,输出是变换后可以执行的操作 O' 。

算法 5 控制算法 TransformRO

```
TransformRO(HB, O): O'
O' = O;
将 HB 分为左右两部分, O 的前序操作的队列 Qh 和 O 的并发操作的队列 Qc;
if (|Qc| > 0){
if (O.type = INSERT)
Qc = RightShiftDel(Qc);
for (i = 1; i <= |Qc|; i++)
O' = IT(O', Qc[i]);
}
```

在 TransformRO 中将 HB 中与 O 并发的操作右移,其详细

实现参见 GOTO 算法^[5]。当远程操作是 Insert 操作时, 必须再将 Q_c 中的 Delete 操作移到右侧, 才能保证变换的正确性。

3 一个应用的例子

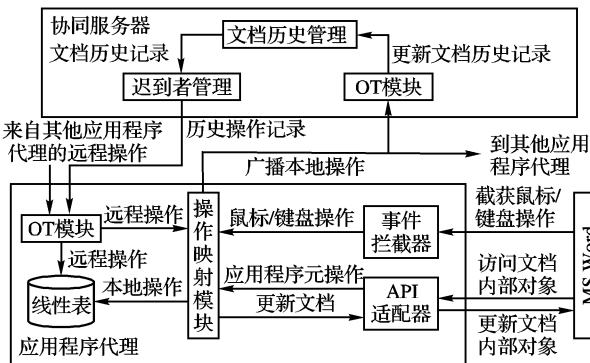


图4 一个利用 MS Word 进行协同编辑的系统的框架

上文论述了如何对单用户应用程序进行扩展, 使其支持无限制的协同编辑。下面给出一个应用的例子: 一个群组利用 MS Word 进行协同编辑。整个系统的框架包括三大部分组成: 协同服务器、客户端和单用户应用程序, 即 MS Word。

MS Word 作为用户的编辑工具。应用程序代理负责对不同用户的 MS Word 进行协同, 以支持协同编辑。协同服务器集中存放共享文档, 并负责维护文档的操作历史, 处理迟到者的加入。下面对应用程序代理和服务器的各个模块的功能作简要的说明。

3.1 应用程序代理

API 适配器 对 MS Word 的 API 进行了封装。由于该模块与具体的应用程序相关, 应具有可插拔性。该部分以插件的形式存在, 协同不同的应用程序要启动不同的插件。

事件拦截器 负责截获用户对 MS Word 的鼠标键盘操作, 并交给操作映射模块处理。

操作映射模块 客户端本地发生操作时, 负责调用 API 适配器封装的 API, 把事件拦截器截获的对 MS Word 的操作映射成对线性表的操作, 然后广播出去。在其他客户端发送的远程操作到达时, 负责解析 OT 模块处理过的远程操作, 然后调用 API 适配器封装的 API 对 Word 文档进行更新。

OT 模块 负责对其他客户端发来的远程操作进行操作变换, 对线性表进行更新, 然后交给操作映射模块进行处理。

3.2 协同服务器

文档历史管理 维护共享文档的操作历史, 对文档版本进行管理。

OT 负责对客户端发来的操作进行操作变换。因为协同服务器要保存协同过程的历史记录, 所以每个参与者的本地操作除了要发送给其他参与者以外, 也要发送给协同服务器。

迟到者管理 负责处理迟到者加入问题, 使迟到者能得到最新的共享文档版本以及文档操作历史记录。

4 结语

本文提出了一个对单用户应用程序进行扩展以支持协同编辑的方案。在不改变源代码的前提下, 使多个用户使用单

用户应用程序进行无限制的协同编辑。既支持多人间的交互和协作, 又保留了单用户应用程序的强大的功能和良好的可用性。方案中在单用户应用程序外部设置了一个线性表用来索引文档中的对象, 利用操作变换机制维持线性表之间的同步, 从而保持共享文档在全局的一致性。方案中提出了改进的扩展 OT 算法, 以维护协同编辑中具有复杂对象结构的文档在全局的一致性。

以线性表为桥梁, 操作变换机制间接地作用在单用户应用程序文档上, 使得多个用户可以使用单用户应用程序对共享文档进行无限制的协同编辑。在协同编辑期间, 一个用户可以对共享文档的任意部分进行编辑、修改, 而不受其他用户的干扰。用户对共享文档的操作的效果会立即在本地界面上表现出来, 不需要其他用户的客户端进行确认。客户端同时会把用户对文档的操作映射为对线性表的操作, 并把线性表操作广播给其他客户端。其他客户端收到线性表操作以后, 先进行操作变换, 然后映射为应用程序元操作, 调用 API 对文档进行更新。因为是在后台使用 API 对文档进行更新, 不需要抢夺本地用户的鼠标键盘控制权, 所以用户感觉不到更新文档带来的干扰, 从而给用户带来更大的方便, 使用户在不受其他编辑者干扰的情况下对文档进行更自由的编辑。

参考文献:

- [1] ELLIS CA, GIBBS SJ. Concurrency control in groupware systems [J]. SIGMOD Record, 1989, 18(2): 399-407.
- [2] YANG Y, SUN C, ZHANG Y, et al. Real-time cooperative editing on the internet [J]. IEEE Internet Computing, 2000, 4(3): 18-25.
- [3] SUN C, CHEN D. Consistency maintenance in real-time collaborative graphics editing systems [J]. ACM Transactions on Computer-Human Interaction, 2002, 9(1): 1-41.
- [4] 史美林, 杨光信, 向勇, 等. 一个基于 Web 的工作流管理系统 [J]. 软件学报, 1999, 10(11): 1148-1155.
- [5] 史美林, 向勇, 杨光信, 等. 计算机支持的协同工作理论与应用 [M]. 北京: 电子工业出版社, 2001.
- [6] SUN C, JIA X, ZHANG Y, et al. Achieving convergence, causality-preservation, and intention-preservation in real-time cooperative editing systems [J]. ACM Transactions on Computer-Human Interaction, 1998, 5(1): 63-108.
- [7] SUN C, ELLIS CA. Operational transformation in real-time group editors: issues, algorithms, and achievements [A]. Proceedings of the ACM Conference on Computer-Supported Cooperative Work [C]. Seattle WA (USA): ACM Press, 1998. 59-68.
- [8] LI D, LI R. Transparent sharing and interoperation of heterogeneous single-user applications [A]. Proceedings of the ACM Conference on Computer-Supported Cooperative Work [C]. New Orleans, Louisiana, USA: ACM Press, 2002. 246-255.
- [9] XIA S, SUN D, SUN C, et al. Leveraging single-user applications for multi-user collaboration: the cword approach [A]. Proceedings of the 2004 ACM Conference on Computer Supported Cooperative Work [C]. Chicago, Illinois, USA: ACM Press, 2004. 162-171.
- [10] SUN D, XIA S, SUN C, et al. Operational transformation for collaborative word processing [A]. Proceedings of ACM Conference on CSCW 2004 [C]. Chicago, USA: ACM Press, 2004. 437-446.
- [11] IMINE A, MOLLI P, OSTER G, et al. Proving Correctness of Transformation Functions in Real-Time Groupware [A]. 8th European Conference of Computer-supported Cooperative Work [C]. Helsinki, Finland: Kluwer Academic Publishers, 2003. 14-18.