计算机研究与发展
Journal of Computer Research and Development

ISSN 1000-1239/CN 11-1777/TP
44(2): 326～333, 2007

实时协同工作系统中操作转换算法综述

廖 斌 何发智 荆树旭
(武汉大学计算机学院 武汉 430072)
(fzhe@whu.edu.cn)

Survey of Operational Transformation Algorithms in Real-Time Computer-Supported Cooperative Work

Liao Bin, He Fazhi, and Jing Shuxu
(*School of Computer Science, Wuhan University, Wuhan* 430072)

Abstract The partial relations of distributed operations and the consistency model are introduced and analyzed. Based on the development of typical operational transformation algorithms, basic operational transformation functions are discussed and summarized. The intention problems are carefully analyzed and the algorithm guides for intention preservation are presented. Open issues and future research directions are also discussed.

Key words computer-supported cooperative work; collaborative editing; operational transformation; consistency maintenance; concurrency control

摘 要 首先分析了实时协同工作系统中的事件关系,讨论了一致性模型及其收敛问题,然后以典型实时协同工作系统中操作转换算法的研究进展为线索,将算法中最核心的调度函数剥离出来,进行了详细的分析.对意图维护问题进行了解析,将其归结为在立即模式下对并发问题的处理,得出了意图维护问题在算法层面的准则.最后对基本转换函数和典型转换算法进行了归纳和总结,并对进一步研究方向进行了探讨.

关键词 计算机支持的协同工作;协同编辑;操作转换;一致性维护;并发控制

中图法分类号 TP301.6

与传统分布式系统不同,实时协同工作系统强调自然与和谐的人人交互,对协同交互过程中的并行性和响应性有要求[1-4]:并行性意味着多个用户可以同时操作共享她据,从而提高工作效率;响应性则要求尽可能快地看到本地用户和远程用户的操作效果,从而缓解用户交互的时滞现象.作为一类典型的实时协同工作系统,ACM 专门成立协同编辑的特殊兴趣小组 (SIGCE),近 6 年来连续举行了 6 届研讨会[5].实时协同编辑系统虽然可以通过采用全复制式结构来获得高并行性和高响应性,但也使一致性维护问题复杂起来.为此,科罗拉多州大学的 Ellis 和 Gibbs 在 ACM SIGMOD 99 会议上首先提出了操作转换(distributed operational transformation, dOPT)的思路[1],该研究一直持续到最近.

1 一致性模型

1.1 偏序关系定义

大多数操作转换算法都基于一种偏序关系来进行调度,故先引出.少数算法不同程度地运用了全

收稿日期:2005-07-04;**修回日期**:2006-06-13
基金项目:国家自然科学基金项目(60303029);国家自然科学基金委国际合作项目 NSFC/KOSEF(60510627);浙江大学 CAD&CG 国家重点实验室开放课题基金项目(A0501)

序关系,将在相关算法中讨论. 基于 Lamport 状态向量 SV(state vectors)[6]可定义两两操作之间偏序意义上的因果关系[1-2,4].

定义 1. 因果关系. 给定任意两个分别位于站点 i 和站点 j 上的操作 O_a 和 O_b,称 O_a 和 O_b 存在因果关系(记做 $O_a \rightarrow O_b$),当且仅当 O_a 和 O_b 满足下列 3 个条件之一:① $i=j$ 并且操作 O_a 发生在 O_b 之前;② $i \neq j$ 并且操作 O_a 在站点 j 的执行先于操作 O_b 的产生;③存在操作 O_x,并且有 $O_a \rightarrow O_x$ 和 $O_x \rightarrow O_b$.

定义 2. 依赖关系和非依赖关系(或称并发关系,或称独立关系). 给定任意两个操作 O_a 和 O_b,称 O_a 和 O_b 具备依赖关系,当且仅当 O_a 和 O_b 满足 $O_a \rightarrow O_b$ 或者 $O_b \rightarrow O_a$;称 O_a 和 O_b 具备非依赖关系(记做 $O_a \parallel O_b$),当且仅当 O_a 和 O_b 既不满足 $O_a \rightarrow O_b$,又不满足 $O_b \rightarrow O_a$.

定义 2 实际上是定义 1 的另一种表述方式. 对于存在依赖关系的操作可以按照因果次序来定序. 对于具有非依赖关系的操作还可进一步分类.

定义 3. 偏并发关系. 给定任意两个具有并发关系操作 O_a 和 O_b,称 O_a 和 O_b 存在偏并发关系,当且仅当 O_a 和 O_b 生成时的文档状态不同,即生成 O_a(或 O_b)时的状态比生成 O_b(或 O_a)时的状态多了任意操作 O 对文档状态产生的影响.

图 1 所示的 O_1 和 O_3 就是一例偏并发关系.

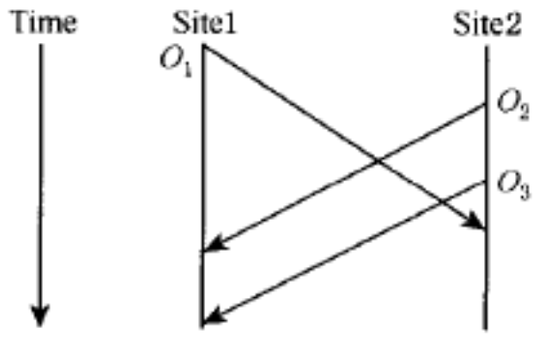

Fig. 1　An example of partial concurrency.

图 1　偏并发关系示例

1.2　事件关系分类和层次结构

设 O_a 和 O_b 为两个操作,记"·"表示操作的连续合成执行序列;记"≡"表示两种执行序列 $O_a \cdot O_b$ 和 $O_b \cdot O_a$ 相等,即当在相同文档状态被执行时它们将产生相同的结果.

根据操作语义(从另一个角度来考察事件关系)可以确定两两事件之间是否具有可交换性,即对于两个并发操作 O_a 和 O_b,称 O_a 和 O_b 具备可交换关系(记做 $O_a /\!/ O_b$),当且仅当 $O_a \cdot O_b \equiv O_b \cdot O_a$;否则,它们具备不可交换关系(记做 $O_a \perp O_b$).

事件关系的层次结构如图 2 所示:

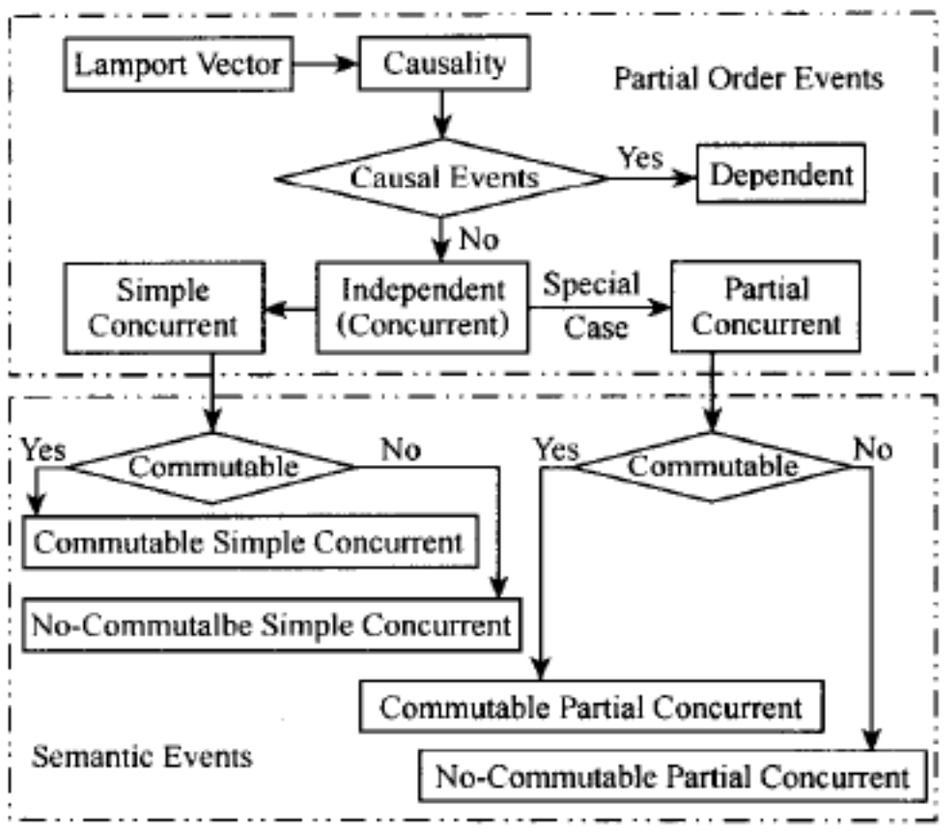

Fig. 2　Hierarchical relations of distributed operations.

图 2　事件关系的层次结构

1.3　不一致性问题

不失一般性,如图 3 所示,假设各个操作的执行顺序如下:站点 0:O_1, O_2, O_4, O_3. 站点 1:O_2, O_1, O_3, O_4. 站点 2:O_2, O_4, O_3, O_1.

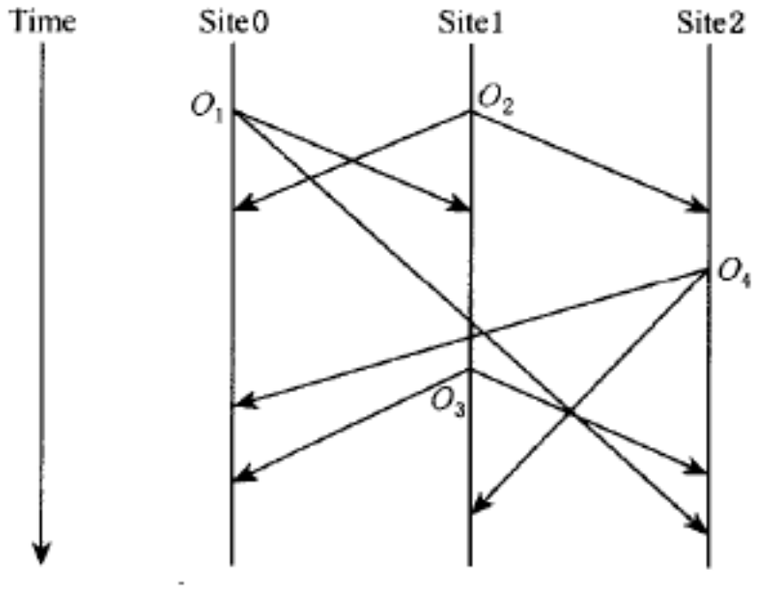

Fig. 3　A scenario of collaborative editing.

图 3　一个协同编辑工作场景

问题 1. 因果不一致. 在站点 1,O_1 的到达在 O_3 生成之前,即 $O_1 \rightarrow O_3$;而在站点 2,O_3 到达时,O_1 并未到达,O_1 和 O_3 没有按照操作之间的因果次序到达和执行,这就产生因果不一致.

问题 2. 结果不一致. 如果操作 O_1, O_2, O_3, O_4 具有不可交换关系,就会导致各个站点的结果不一致.

Sun 等人通过"执行效果"来定义操作"意图"[2]. 文中是从以下两个方面来解析 Sun 等人提出的意图. 首先,按照 Sun 等人本身观点,意图不一致与结果不一致的性质不同. 结果收敛可以通过简单串行化方法(保持操作的原始参数形式不变)来获得,而意图收敛不可能通过这种简单串行化方法获得. 其次,文中通过对比 Ressel 等人关于转换性质的研究

(见本文第5节分析),两种基本转换单元分别对应两种并发问题(简单并发和偏并发). 这样,如果通过一系列的基本转换单元(而不是简单的串行化)来实现结果收敛,在这样的总体收敛过程中的每一步(即通过基本转换单元)都能保持意图. 这样就需要考虑第3类不一致性问题.

问题3. 意图不一致. 在站点1时,O_2是对原有文档进行操作;而在站点0,O_2到达后对已经被O_1处理过的文档进行操作,O_2在站点0进行操作时原有意图有可能不能实现,产生意图不一致.

1.4 评判一致性的3个条件

根据上述定义和问题分析可以得出评判协同编辑系统一致性的3个条件.

评判条件1. 因果保持. 对任意一对操作O_a和O_b,如果$O_a \rightarrow O_b$,那么在所有站点O_a先于O_b执行;

评判条件2. 结果收敛. 所有站点都执行完相同操作集后,所有站点的共享文档的副本是一致的;

评判条件3. 意图保持. 对任意一个操作O,O的本地和远程执行效果与O的意图相同,并且如果存在O_x且$O_x \parallel O$,那么O_x的执行效果与O的执行效果互不影响.

2 基本操作转换算法及其转换函数

首先本地用户生成的操作立即被执行,然后与**SV**关联建立因果关系(SOCT4和TIBOT除外)后发往其他站点. 由于网络延迟,当一个操作O达到远程站点之后,接收的次序未必就能满足因果关系,需要等待直到满足因果关系之后操作转换方法才决定如何调度该操作.

Ellis和Gibbs所提出的操作转换算法(dOPT)流程如图4所示[1]. 其中日志文件*Log*按照先后顺序保存所有已执行的n个操作,$Log[i]$为*Log*中第i个操作. EO为O的执行形式.

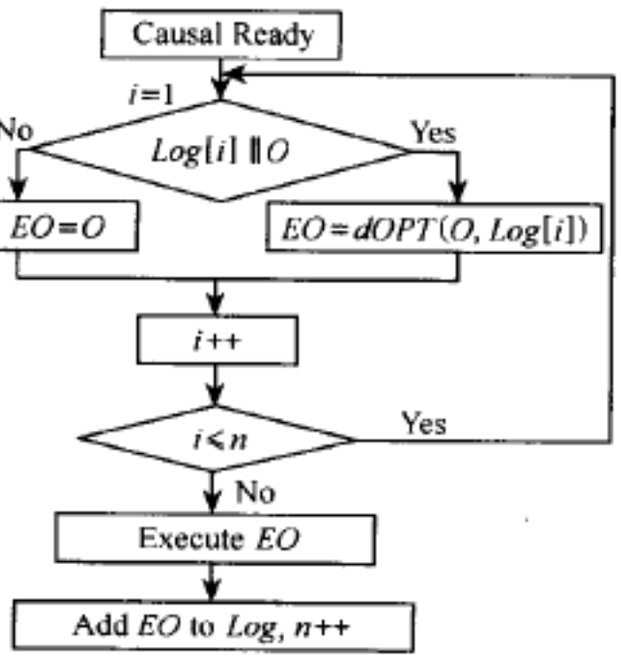

Fig. 4 dOPT algorithm.

图4 dOPT算法总体流程

当转换函数dOPT把O对$Log[i]$进行转换时,是基于这两个操作之间的操作语义(插入和删除)和操作参数(在第几个位置插入或者删除)来决定是否或者如何修改O的参数(即O的插入或者删除位置参数增加一位,减少一位或者不变),从而得到转换之后的EO.

在对O进行转换时存在两种特例. 第1种特例,当两个插入操作在同一位置插入或者删除相同字符,因为$Log[i]$和O表达的是同一种操作语义,且$Log[i]$已经执行,故O转换之后的EO为空操作(即不予执行).

第2种特例,当两个插入操作在同一位置插入不同字符,对于这种语义无法兼容的情形,引入站点号大小作为优先级来处理. 当O优先级低于$Log[i]$时,转换之后的EO的参数要调整增加一个位置,这样EO的执行不影响到$Log[i]$插入字符的所在位置. 当O优先级高于$Log[i]$时,转换之后的EO的参数不变,即相当于O被直接执行. 由于EO的执行,自然将$Log[i]$所插入的字符推后一个位置,即相当于$Log[i]$的执行参数增加一个位置.

记在位置p插入字符x为$insert[x,p]$;在位置p删除字符为$delete[p]$. O对$Log[i]$的转换有4种:$T_{II}(insert[x,p],insert[y,q])$(记$O$和$Log[i]$对应生成的站点号为$S_1$和$S_2$),$T_{DD}(delete[p],delete[q])$,$T_{ID}(insert[x,p],delete[q])$和$T_{DI}(delete[p],insert[y,q])$. 4种转换结果如图5所示:

			T_{II}	T_{DD}	T_{ID}	T_{DI}
$p<q$			p	p	p	p
$p>q$			$q+1$	$p-1$	$p-1$	$p+1$
$p=q$	$x=y$		空	空	p	$p+1$
	$x\neq y$	$S_1<S_2$	$p+1$			
		$S_1>S_2$	p			

Fig. 5 Parameter adjustment in dOPT algorithm.

图5 dOPT转换函数对操作参数的调整

这样,通过调用dOPT转换函数,将O对日志中所有与之具有并发关系的操作进行若干次转换,得到EO之后被执行. 但是dOPT算法不完备,其转换函数仅仅考虑简单并发问题,于是Sun等人提出了GOT算法[7-8]和GOTO算法[2,9],以解决偏并发问题.

3　包含转换函数,排斥转换函数与位移转换函数

3.1　GOT 算法中的包含转换和排斥转换

针对 dOPT 算法的不足,Sun 等人提出 GOT 算法[7-8],调度函数流程如下:

① 按某种全序关系将 *Log* 中的操作分成两部分(一部分全序在 *O* 前,另一部分全序在 *O* 后),但是各个部分还是按全序排列;

② 将 *Log* 中全序在 *O* 后的操作 Undo 掉;

③ 将 *O* 对 *Log* 中全序在 *O* 前的操作进行转换,得到 *EO*;

④ 将 *EO* 置入 *Log*;

⑤ 按全序取出①中被 Undo 掉的一个操作对 *EO* 进行转换,得到该操作的执行形式,Redo 之并加入 *Log*;

⑥ 重复⑤直到所有 Undo 掉的操作被转换,Redo 和加入 *Log*.

对于任意站点生成的所有操作,可以赋予其全局惟一的先后次序. GOT 中通过首先比较各个操作生成时的 ***SV*** 之和;当大小相等时,再比较站点大小(类似于 dOPT 算法的优先级),从而确定一种全序关系. GOT 中的全序用于确定 *O* 在 *Log* 中的位置以及基于该位置的相关运算次序,并非决定是否调度 *O* 的依据.

GOT 算法第③步中的转换考虑到两种情况. 第 1 种情况,*O* 和 *Log*[*i*]是简单并发关系. 为了在考虑到并发操作 *Log*[*i*]已经对文档状态产生影响的情况下,实现 *O* 的执行效果,需要把 *O* 对 *Log*[*i*]进行转换,使得 *Log*[*i*]对文档产生的影响被 *EO* 所包含. GOT 把这种转换称之为包含转换(inclusion transformation, IT). IT 要求转换时,*O* 和 *Log*[*i*]都针对同一文档状态.

第 2 种情况,如果 *O* 和 *Log*[*i*]是偏并发关系. GOT 处理该情况如下:①进行排斥转换(exclusion transformation, ET),即把 *O* 对因操作转换得到 *O*′,使得因操作对文档状态产生的影响被 *O*′排斥在外;②把 *O*′对 *Log*[*i*]进行 IT 得到 *EO*.

GOT 通过 ET 解决偏并发问题. 下面从运算形式上把 IT 和 ET 与 dOPT 进行对比.

为了得到正确的执行形式 *EO*,IT 把 *O* 对 *Log*[*i*]进行转换,以考虑 *Log*[*i*]已经对文档状态产生的影响,故 IT 与 dOPT 中转换函数参数变化规则相同,即与表 1 相同.

为了使 *O* 得到和 *Log*[*i*]相同的文档状态,ET 把 *O* 对因操作进行转换,以排除因操作已经对文档状态产生的影响,故 ET 与 dOPT 中转换函数参数变化规则相反,即与表 1 相反.

GOT 算法第⑤步中的转换(即把 Undo 掉的操作对 *EO* 进行转换时)同样考虑到这两种情形. GOT 中每次调度一个满足因果关系的远程操作时都要经历上述 6 个步骤,效率较低.

3.2　GOTO 算法中的位移转换

尽管 GOTO 声称是对 GOT 的优化改进,但实际上 GOTO 的总体流程与 dOPT 相同,在遇到偏并发问题时,采用了所谓的位移转换(transpose transformation, TT)来构造临时运算的日志文件 *Log*′,通过对 *Log*′进行转换得到 *EO*[2,9]. 调度函数流程如下:①扫描 *Log*,查找是否存在 *O* 的并发操作,如果没有,则 $EO = O$,转入⑥;如果有,转入②;②如果都是简单并发操作,则 *O* 对它们进行 IT 得到 *EO*,转入⑥,否则,转入③;③将 *Log* 拷贝为 *Log*′,作为临时运算使用;④通过 TT 将 *Log*′分为两部分 SEQ_1 和 SEQ_2,其中 SEQ_1 包含 *O* 的因操作,SEQ_2 包含 *O* 的并发操作;⑤*O* 对 SEQ_2 进行 IT 得到 *EO*;⑥执行 *EO* 将其添加到 *Log* 末端.

TT 通过 IT 和 ET 来实现,其形式化表达为 $TT(O_a, O_b) = (O_b', O_a')$,其中 $O_b' = ET(O_b, O_a)$,$O_a' = IT(O_a, O_b')$.

通过 TT,虽然 *Log* 和 *Log*′的执行次序不同,但执行效果相同. 这样对 *Log*′进行转换就等效于对 *Log* 进行转换. 由于在 *Log*′中 *O* 的因操作全部在 *O* 的并发操作之前,这样就使得 *O* 和 SEQ_2 中 *Log*′[*i*]只存在简单并发关系,从而解决偏并发问题.

fastdOPT[10]算法对 GOTO 进行了改进,是在发送端把 *O* 对因操作的原始形式进行 ET,得到与偏并发操作相同的文档状态,减少了转换次数,提高了效率.

4　前向转换函数和后向转换函数

4.1　SOCT2 算法中的操作转换

O 和 *Log* 中的操作有 3 种情况:*Log* 只包含 *O* 的因操作;*Log* 只包含 *O* 的因操作和简单并发操作;*Log* 包含 *O* 的因操作、简单并发操作和偏并发操作.

为了采用统一流程来处理以上 3 种情况,在

Suleiman 等人提出的 SOCT2 算法中,每次调度 O 时,都要进行类似于 TT 的后向转换(backward transposition, BT)构造临时运算的 Log',对 Log' 进行转换得到 EO[11],流程如下:①BT 将 Log' 分为两部分 SEQ_1 和 SEQ_2,其中 SEQ_1 包含 O 的因操作,SEQ_2 包含 O 的并发操作;②把 O 对 SEQ_2 进行类似于 IT 的前向转换(forward transposition, FT)得到 EO;③执行 EO 并置入 Log 末端.

杨等人把 SOCT2 和 dOPT 扩展到采用了面向对象模型的 list 对象[4,12].

FT 的出发点和 IT 类似,即为了得到执行形式 EO,需要把 O 对 $Log[i]$ 进行转换,以便考虑 $Log[i]$ 已经对文档状态产生的影响,使 EO 的执行效果与 O 在原有文档下的执行效果相同.

BT 由两个所谓 FT 组成,其形式化表达为 $BT(O_a, O_b) = (O_b', O_a')$. 其中,$O_b'$ 采用隐式转换函数来表达,即为了获得 O_b' 必须满足 $O_b = FT(O_a, O_b')$;O_a' 采用显式转换函数来表达,即 $O_a' = FT(O_b', O_a)$. 这样,获得 O_b' 和获得 O_a' 的过程正好相逆.

4.2 SOCT3 和 SOCT4 算法中的操作转换

在 Vidot 等人提出的 SOCT3 和 SOCT4 算法中,给每个站点上的每个操作都赋予一个惟一的序号,也称时间戳(time stamps, TS),该序号连续增长(从 1 开始到 N,1,2,3,…,N). 通过 TS 也可以定义一种全序关系,被称为持续全局次序(continuous global order, CGO)[13].

与前面所有方法中调度远程操作的依据不同,SOCT3 和 SOCT4 按照严格全序来决定是否调度所接收的远程操作 O. 如日志文件图 6 所示,当 $TS(O) = TS(O_i) + 1$ 时才调度 O,否则 O 要等待.

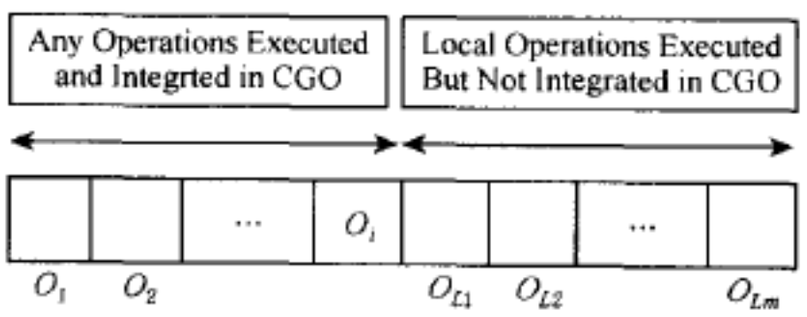

Fig. 6 *Log* in SOCT 3 and SOCT 4.

图 6 SOCT3 和 SOCT4 日志文件

SOCT3 和 SOCT4 中日志分两部分. 第 1 部分 $O_j(1 \leqslant j \leqslant i)$ 是远程或本地已执行且按 CGO 集成的操作. 第 1 部分中的操作保持 CGO,即各操作按全序连续排列. 第 2 部分 $O_{Lk}(1 \leqslant k \leqslant m)$ 是本地已执行但未集成到第 1 部分的操作. 第 2 部分中的操作不一定保持 CGO,但是按 TS 升序排列.

通过以上预处理,SOCT3 和 SOCT4 中接下来的主要工作是:①获得 O 的正确的执行形式;②集成 O 到日志中正确的位置;③在日志中的正确的存放形式.

SOCT3 中是这样来完成上述工作的:①将 Log 拷贝为 Log',作为临时运算使用;②BT 将 Log' 分为两部分 SEQ_1 和 SEQ_2,其中 SEQ_1 包含 O 的因操作,SEQ_2 包含 O 的并发操作;③O 对 SEQ_2 进行 FT 得到 EO;④执行 EO(相当于在日志尾部执行);⑤通过 BT 将 EO 对 O_{L1}. O_{L2},…,O_{Lm} 转换得到 EO',从而将 EO' 按照 CGO 集成到 $i+1$ 的位置(相当于位移到 $i+1$ 的位置,并获得正确的存放形式 EO').

SOCT4 对本地操作的广播进行了延迟(deferred broadcast),分为本地流程(①,②和③)和远程流程(④,⑤,⑥和⑦):①本地操作满足 $TS(O_{L1}) = TS(O_i) + 1$ 时才开始准备发送 O_{L1};②把 O_{L1} 对本地日志第 1 部分中的操作进行 FT,得到 O_{L1}';③广播 O_{L1}';④远端站点接收到 O_{L1}' 以后,若满足 $TS(O_{L1}') = TS(O_i) + 1$ 时才开始准备调度 O_{L1}';⑤把 O_{L1}' 对远端站点中日志第 2 部分的操作逐个进行 FT 得到 EO_{L1}';⑥执行 EO_{L1}';⑦把远端站点中日志第 2 部分的操作逐个对 EO_{L1}' 进行 FT,从而将 O_{L1}' 按照 CGO 集成到远端站点日志中 $i+1$ 的位置.

由于 SOCT4 中操作在发送时已经考虑了 CGO 在其之前的操作对文档产生的影响,故当其到达远端时,只需对远端站点的日志第 2 部分的操作进行转换.

基于 SOCT4 的整体流程,Molli 等人将上述思路扩展到文件系统的数据同步,并对特定转换问题及其转换函数进行了探索[14]. TIBOT[15] 以时间间隔(time interval, TI)为主来定义全序. 在将本地同一个 TI 内的操作集 $|O|$ 对外广播之前,把 $|O|$ 对 $TI < |O|$ 的操作进行 IT 得到 $|O'|$,类似于 SOCT4. 远端站点接收并调度 $|O'|$ 时,则进行 Undo/Redo 和 IT 处理,类似于 GOT.

GOT 和 TIBOT 采用分布式方法来取全序. SOCT3 和 SOCT4 既可以采用分布式方法,又可以采用集中式方法来取全序[16-17].

5 L 转换函数、交互图、转换性质和意图问题小结

5.1 L 转换函数、交互图和转换性质

Ressel 等人在 adOPTed 算法中提出了 L 转换(L-transformation, LT)函数和多维交互图

(multidimensional interaction graph, MIG)[18]. 该图的顶点用来表示文档状态,有向边表示操作,如图 7 所示:

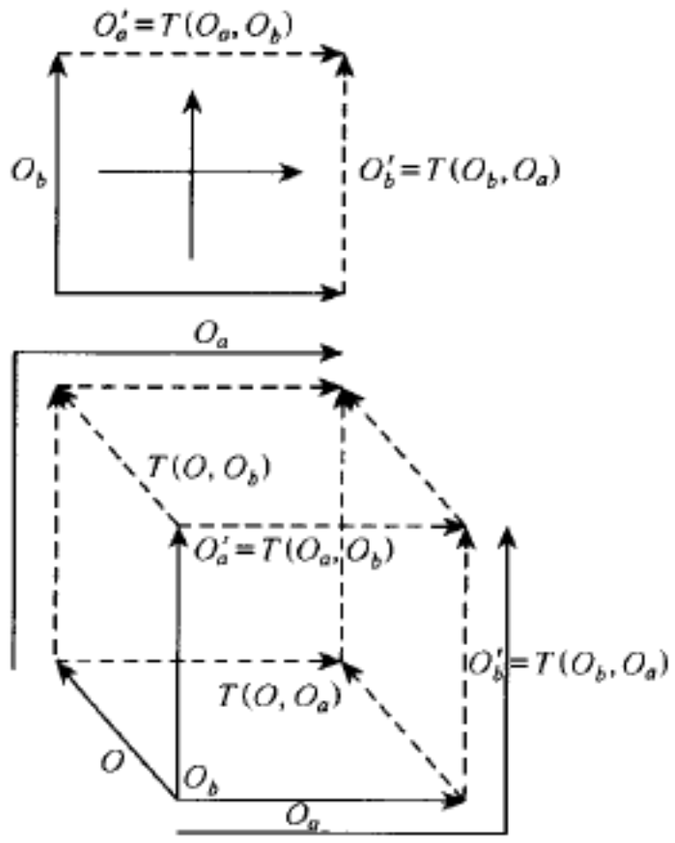

Fig. 7　L-transformation and interaction graph.

图 7　L 转换和多维交互图

从图 7 中可以看出,正确的转换必须基于同一个顶点. 简单并发关系已经基于同一个顶点,可以直接转换,关键是偏并发关系.

以文中图 1 所示的偏并发问题为例,MIG 的解决方法如图 8 所示. 首先把 O_1转换为 O_1',使 O_3和 O_1'处于同一个顶点,然后再把 O_3对 O_1'进行转换. Ressel 等人把上述性质定义为 TP1(对应简单并发关系)和 TP2(对应偏并发关系),并得出实现结果收敛的充分必要条件是满足转换性质 TP1 和 TP2[18].

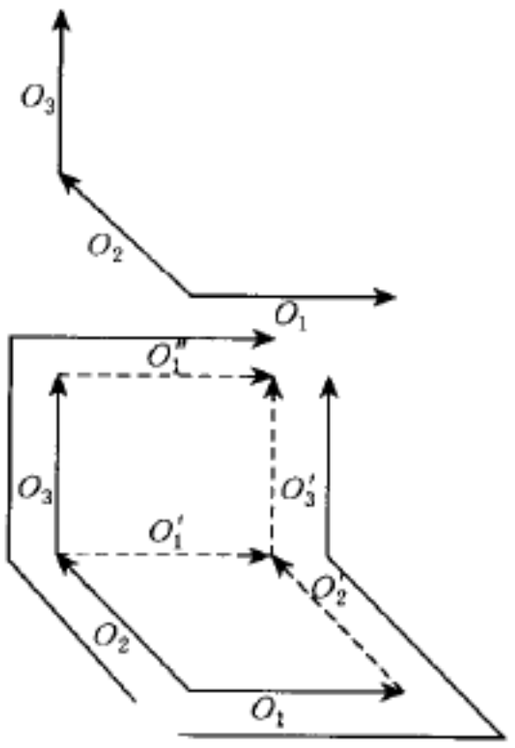

Fig. 8　Partial concurrency problem solved by MIG.

图 8　MIG 解决偏并发问题

实际上,这种情况下的结果收敛也能实现意图保持,尽管 Ressel 等人没有提及意图问题.

5.2　意图维护问题小结

综上所述,意图保持仍然是解决两种并发问题. 与结果收敛的区别在于,意图保持指每一步操作都要解决这两类并发问题,是一种立即模式.

至此,文中关于意图保持问题的小结如下:第 1 类意图保持问题,在立即模式下解决普通并发问题;第 2 类意图保持问题,在立即模式下解决偏并发问题.

Sun 在 GOT 算法中认为 Ellis 的 dOPT 算法没有实现意图维护,应该指的没有解决偏并发关系. 实际上,对于存在偏并发关系的操作集,dOPT 算法既不能实现意图保持,也不能实现结果收敛. 除非偏并发关系已经在预处理中避免了(例如 SOCT4 和 TIBOT 算法),这样只须解决普通并发问题就可以实现意图保持,并且能够实现结果收敛.

通过以上分析可以归纳出一个普适性概念,基本转换单元:第 1 类基本操作单元以两个操作为一个基本单元,处理具有简单并发关系的操作转换单元;第 2 类基本操作单元以 3 个操作为一个基本单元,处理具有偏并发关系的操作转换单元.

这样就可以得出意图维护问题在算法层面的准则:准则 1,通过构造基本转换单元,将任意次序的操作序列归约为上述两种基本转换单元;准则 2,由基本转换单元所形成的执行序列既能保证结果收敛,又能实现意图保持.

6　转换函数和转换算法的对比总结

基本操作转换函数的对比,如图 9 所示(其中,O'_a,O'_b为 O_a,O_b进行转换后的形式).

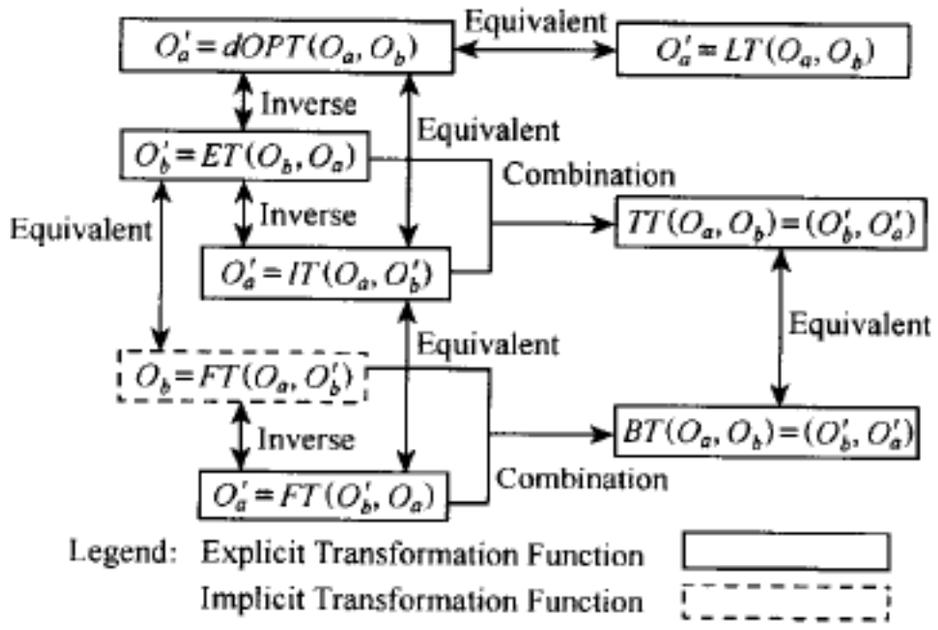

Fig. 9　Comparison of transformation functions.

图 9　基本转换函数之间的对比

图 9 中 dOPT, LT, IT 和 $O'_a=FT(O'_b,O_a)$对应的是基本操作转换(第 1 种转换单元,解决简单并发问题). 在此基础上,进行组合可以实现第 2 种转

换单元(解决偏并发问题).

根据图 9 所示的关系,基本转换函数及其相逆的转换函数的具体实现参考表 1(相同或者相反).

以上述基本转换函数为核心的典型操作转换算法的对比见图 10. 结果收敛和意图保持均要考虑简单并发问题和偏并发问题. 结果收敛指的是整体结果收敛,而意图保持指的是每一步操作意图都要保持. 这样,结果收敛和意图保持既存在联系,又存在区别.

Comparison	dOPT	adOPTed	GOT	GOTO	SOCT2	SOCT3	SOCT4	TIBOT
Scheduling	prtial	prtial	prtial	prtial	prtial	total	total	total + |O|
Log	main	main	main	main	main	main	main	main
				temporary	temporary	temporary		temporary
Causality	***SV***	***SV***	***SV***	***SV***	***SV***	*TS*	*TS*	*TI*
Convergence	TP1	TP1	Undo/ Redo	TP1	TP1	TP1 + CGO	TP1 + CGO	Undo/Redo
	no	TP2		TP2(TT)	TP2(BT)	CGO + BT	avoid	avoid
Intention	dOPT	LT	IT	IT	FT	FT	FT	IT
	no	MIG	ET	ET	BT	BT	avoid	avoid

Fig. 10 Comparison of typical operational transformation algorithms.

图 10 典型操作转换算法对比

dOPT 不能保证偏并发情况下的结果收敛和意图保持. GOT 采用了 Undo/Redo 方案,在结果收敛方面将两类并发问题统一处理. SOCT4 和 TIBOT 在发送端进行了预处理,从而避免了偏并发问题.

7 结束语

文中对典型实时协同工作系统中操作转换算法进行了总结. 进一步的研究方向可以分为 2 个方面:一方面,就协同编辑的操作转换而言,可以参照图 9 和图 10,对操作的发送、接收、转换、执行、存放等构造出不同的调度模式,从而达到了异曲同工的目的;另一方面,将操作转换从协同文本编辑向其他实时协同系统领域扩展. 首先如何将数据结构从一维的线性数据结构扩展到复杂数据结构. 其次,将简单语义扩展到复杂语义. 这需要根据特定应用背景和操作语义来对原始操作中的参数值进行调整,即得到一张类似于表 1 的参数转换调整表. 最后,在一维的线性数据结构中,用位置来标识字符,并且基于位置参数来进行转换,字符本身没有名字. 但在复杂操作语义中往往要对操作对象进行命名,这样就需要对名字进行转换. 相信随着研究工作的深入,会在上述两个方面做出一些创新和突破.

参 考 文 献

[1] C A Ellis, S J Gibbs. Concurrency control in groupware systems [C]. ACM *Int'l Conf on Management of Data* '89, Seattle, 1989

[2] C Sun, C A Ellis. Operational transformation in real-time group editors: Issues, algorithms, and achievements [C]. ACM Int'l Conf on Computer Supported Cooperative Work '98, Seattle, 1998

[3] Feng Jian, Lin Zongkai. Research on the human-human interaction interface of coeditor as a cooperative editing [J]. Journal of Computer-Aided Design & Computer Graphics, 1999, 11(3): 225-227 (in Chinese)
(冯键, 林宗楷. 协同编辑系统 CoEditor 的人-人交互界面研究[J]. 计算机辅助设计与图形学学报, 1999, 11(3): 225-227)

[4] Yang Guangxin, Shi Meilin. Object data model based concurrency control in fully-replicated architecture [J]. Chinese Journal of Computers, 2000, 2(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[J]. 计算机学报, 2000, 2(2): 113-125)

[5] S N Humbad. Freetext: A consistency algorithm for group text editors [C]. The 6th Int'l Workshop on Collaborative Editing Systems, Chicago, 2004

[6] L Lamport. Time, clocks, and the ordering of events in a distributed system [J]. Communication of ACM, 1978, 21(7): 558-565

[7] C Sun, X Jia, Y Yang, *et al*. A generic operation transformation schema for consistency maintenance in real-time cooperative editing systems [C]. ACM Int'l Conf on Supporting Group Work '97, Phoenix, 1997

[8] C Sun, X Jia, Y Zhang, *et al*. Achieving convergence, causality preservation, and intention preservation in real-time cooperative editing systems [C]. ACM Trans on Computer-Human Interaction, 1998, 5(1): 63-108

[9] C Sun. Undo as concurrent inverse in group editors [J]. ACM Trans on Computer-Human Interaction, 2002, 9(4): 309-361

[10] L Yu, Z Lin, Y Gao, *et al*. Consistency control in synchronous cooperative work [C]. The 6th Int'l Conf on Computer Supported Cooperative Work in Design, London, Ontario, Canada, 2001

[11] M Suleiman, M Cart, J Ferrié. Serialization of concurrent operations in distributed collaborative environment [C]. ACM Int'l Conf on Supporting Group Work '97, Phoenix, 1997

[12] Yang Wuyong, Shi Meilin, Jiang Jinlei. An operation transformation based concurrency control integrating multicast Agent [C]. Journal of Software, 2004, 15(4): 497-503 (in Chinese)
(杨武勇, 史美林, 姜进磊. 一种集成组播代理和操作转换的并发控制方法[J]. 软件学报, 2004, 15(4): 497-503)

[13] N Vidot, M Cart, J Ferrié, *et al*. Copies convergence in a distributed real-time collaborative environment [C]. ACM Int'l Conf on Computer Supported Cooperative Work'00, Philadelphia, 2000

[14] P Molli, G Oster. Using the transformational approach to build a safe and generic data synchronizer [C]. ACM Int'l Conf on Supporting Group Work'03, Sanibel Island, Florida, 2003

[15] R Li, D Li, C Sun. A time interval based consistency control algorithm for interactive groupware applications [C]. The 10th Int'l Conf on Parallel and Distributed Systems'04, Newport Beach, 2004

[16] G L Lann. Algorithms for distributed data sharing systems which use tickets [C]. The 3rd Workshop on Distributed Data Management and Computer Networks, Berkeley, 1978

[17] J S Banino, C Kaiser, H Zimmermann. Synchronization for distributed systems using a single broadcast channel [C]. The 1st Int'l Conf on Distributed Computing Systems, Huntsville, 1979

[18] M Ressel, D Nitsch-Ruhlmrd, R Gunzenbauser. An integrating, transformation-oriented approach to concurrency control and undo in group editors [C]. ACM Conf on Computer Supported Cooperative Work'96, Boston, 1996

Liao Bin, born in 1979. Ph. D. candidate. His main research interests are CSCW and collaborative CAD.
廖　斌,1979 年生,博士研究生,主要研究方向为 CSCW 和协同 CAD.

He Fazhi, born in 1968. Ph. D., associate professor, and Ph. D. supervisor. His main research interests are CAD/CG/CSCW.
何发智,1968 年生,博士,副教授,博士生导师,主要研究方向为 CAD/CG/CSCW (fzhe@whu.edu.cn).

Jing Shuxu, born in 1978. Ph. D. candidate. His main research interests are CSCW and CAD.
荆树旭,1978 年生,博士研究生,主要研究方向为 CSCW 和 CAD.

Research Background

Computer-supported cooperative work (CSCW) reflects a trend from using computer to solve computing problem to using computer to support human to human interaction. A less-constraint and fast-response interaction can be achieved only in replicated CSCW architecture. However, the concurrency control in replicated CSCW systems is more complicated. This paper begins with the analysis of the operation relations and the consistency models. Then the basic operational transformation functions, such as inclusion transformation, exclusion transformation, transpose transformation, forward transposition, backward transposition, L-transformation, are discussed and summarized. The intention problems are carefully analyzed and the algorithm guides for intention preservation are presented. Finally, open issues and future research directions are discussed. Our work is supported by the National Natural Science Foundation of China (60303029) and International Cooperation Project of National Science Foundation of China/Korea Science and Engineering Foundation (NSFC/KOSEF, 60510627).