支持操作意图一致性的实时协同编辑算法综述

何发智[1),2)] 吕 晓[2),3)] 蔡维纬[2)] 程 媛[2)]

[1)](武汉大学软件工程国家重点实验室 武汉 430072)

[2)](武汉大学计算机学院 武汉 430072)

[3)](海军工程大学计算机工程系 武汉 430033)

摘 要 有别于传统分布式系统,实时协同编辑系统强调自然和谐的人人交互和人机交互,允许不同地点的协同用户同时编辑同一共享对象.为了给协同用户提供良好的响应性,实时协同编辑系统有必要采用全复制式体系结构,但又给共享对象的一致性维护问题带来巨大挑战,这一直是协同计算学科的研究热点.近年研究重点逐步从结果一致性发展到操作意图一致性.该文以操作意图一致性为主线,对支持操作意图一致性的实时协同编辑算法进行深入和全面的比较、分析和总结.首先,以Lamport事件偏序关系为起点,对协同编辑系统的因果关系、简单并发关系和偏并发关系概念进行整理.同时,给出实时协同编辑系统中全序关系的分类,操作的全序和操作对象的全序.在对全序关系和优先级进行剖析的基础上,阐述了实时协同编辑系统中的三类一致性模型:CC(Causality-preservation,Convergence)模型、CCI(Causality-preservation,Convergence,Intention-preservation)模型和CA(Causality-preservation,Admissibility)模型.特别地,文中按照因果一致性、结果一致性和操作意图一致性的分类,给出操作意图一致性的维护路线图.然后,分别综述了各类实时协同编辑算法的研究进展和现状,包括OT(Operational Transformation)算法、AST(Address Space Transformation)算法和CRDT(Commutative Replicated Data Type)算法.进一步,文中给出了各类操作意图一致性算法的基本原理和执行框架,并基于一个代表性协同工作场景和算例对典型算法进行详细解析.接下来,文中从操作意图一致性、支持操作意图一致性的典型算法、操作转换函数和算法时间复杂度这4个关键方面对实时协同编辑算法进行归纳和对比.最后对全文小结,并指出为了进一步发展实时协同编辑系统,未来还需要在优先级策略、算法伸缩性、复杂操作语义类型、粗粒度操作对象以及选择性撤销机制等方面深入开展一些研究工作.

关键词 实时协同编辑;一致性模型;操作意图一致性;操作转换;可交换的复制数据类型

中图法分类号 TP311 **DOI号** 10.11897/SP.J.1016.2018.00840

Survey of Real-Time Collaborative Editing Algorithms Supporting Operation Intention Consistency

HE Fa-Zhi[1),2)] LÜ Xiao[2),3)] CAI Wei-Wei[2)] CHENG Yuan[2)]

[1)](*State Key Laboratory of Software Engineering, Wuhan University, Wuhan* 430072)

[2)](*School of Computer Science, Wuhan University, Wuhan* 430072)

[3)](*College of Computer Engineering, Naval University of Engineering, Wuhan* 430033)

Abstract Different from traditional distributed systems, real-time collaborative editing systems support natural and harmonious human to computer interactions and human to human interactions. Real-time collaborative editing systems allow multiple geographically dispersed collaborative users to view and edit the shared object over computer networks. In order to provide high responsiveness for collaborative users, real-time collaborative editing systems adopt the fully-replicated architecture, which brings the great challenge for consistency maintenance of the shared object. Consistency

收稿日期:2016-05-13;在线出版日期:2017-01-05.本课题得到国家自然科学基金(61472289,61502353)、国家重点研发计划(2016YFC0106305)资助.**何发智**,男,1968年生,博士,教授,中国计算机学会(CCF)会员,主要研究领域为协同计算、CAD图形图像、多核众核并行计算.E-mail:fzhe@whu.edu.cn.**吕 晓**,女,1983年生,博士研究生,讲师,主要研究方向为CSCW、协同CAD.**蔡维纬**,男,1988年生,博士研究生,主要研究方向为CSCW、协同CAD.**程 媛**,女,1983年生,博士,讲师,主要研究方向为CSCW、协同CAD.

maintenance is one of the most essential challenges in the design and implementation of real-time collaborative editing systems, which has been a continuous hot topic in the field of collaborative computing for over 25 years. In recent years, the research focus has gradually moved from result consistency to operation intention consistency. In this paper, from the perspective of operation intention consistency, we give a deep and comprehensive comparison, analysis and summary of existing real-time collaborative editing algorithms. First of all, based on Lamport's partial ordering relation of events, we discuss and generalize the basic concepts and terminologies of the causal ordering relation, the simple concurrent relation and the partial concurrent relation. Then, we give the classification of the total ordering relation, such as the total ordering relation of operations and the total ordering relation of operation objects. Moreover, the total ordering relation of operations is further categorized as the centered total ordering relation or the distributed total ordering relation. After analyzing the total ordering relation and the priority, we investigate three types of consistency models, including CC (Causality-preservation, Convergence) model, CCI (Causality-preservation, Convergence, Intention-preservation) model and CA (Causality-preservation, Admissibility) model. Specially, we propose a road map to maintain operation intention consistency according to the categorization of causality preservation, convergence and operation intention preservation. Next, we discuss the research status and progress of existing real-time collaborative editing algorithms supporting operation intention consistency. Until now, there are three representative classes of real-time collaborative editing algorithms to support operation intention consistency, including OT (Operational Transformation) algorithms, AST (Address Space Transformation) algorithms and CRDT (Commutative Replicated Data Type) algorithms. Furthermore, we dig out and analyze the basic principles and overall framework for each class of real-time collaborative editing algorithms, such as representative algorithms proposed by Sun, typical algorithms presented by Li, key algorithms developed by Shao, significant algorithms invented by Gu and most existing CRDT algorithms. In the mean time, we present the integrated procedures of typical collaborative editing algorithms in detail based on the same representative collaborative work scenario and sample. Then, we summarize and compare different kinds of typical real-time collaborative editing algorithms from four major aspects, including the summary of operation intention consistency, the comparison of typical algorithms, the discussion of key operational transformation functions and the theoretical analysis of time complexity. Finally, we draw a conclusion and give some examples and possible future research directions to promote further development of real-time collaborative editing systems, particularly, the development of flexible priority strategy, the improvement of scalability, supporting sophisticated operation semantic types, integrating coarse-grained operation objects, providing selective undo mechanisms and so on.

Keywords　real-time collaborative editing; consistency model; operation intention consistency; operational transformation; commutative replicated data type

1　引　言

2013 年图灵奖得主 Leslie Lamport① 的主要贡献在于 1978 年发表了一篇关于分布式系统中基本事件关系的论文[1]. 实时协同工作系统包括实时协同编辑系统，在其研究过程中的主流文献充分继承和引用了 Lamport 的基本事件关系[1-4]. 与传统分布式系统的区别在于，实时协同编辑系统强调的是自然和谐的人人交互和人机交互[2-7]，体现了 ACM

① http://amturing.acm.org/award_winners/lamport_1205376.cfm

学科分类中以“人为中心的计算(Human-Centered Computing,HCC)”①[8-9],是 CSCW(Computer Supported Cooperative Work)领域的一个重要的研究方向②.

实时协同编辑系统允许不同地理位置的协同工作者通过计算机和网络共同编辑/设计共享对象,包括文本、图像、图形、XML 文件和复杂 CAD 模型等.与单用户的编辑系统相比,实时协同编辑系统的优势是可以有效地发挥群体智慧.最早具有代表性的协同编辑系统是 Ellis 等人[2]在 1989 年提出.此后,实时协同编辑系统得到了大量的研究和探索,典型代表如 Google Wave/Docs③④,Codoxware⑤ 和 IBM OpenCoWeb⑥ 等.这些协同应用系统提高了群体的工作效率和质量,并且进一步促进了协同应用系统及其相关问题的探索.

实时协同编辑系统的相关研究面临若干难题②.为了实现系统的高响应性和高并行性,通常采用全复制式结构,但是又给共享对象的一致性维护带来了巨大挑战[2-7].在实时协同编辑领域近 30 年的研究中,该问题一直是研究者持续关注的热点.

传统的一致性维护方法例如锁机制(locking)[10]和串行化(serialization)方法[11]不适合实时协同编辑,主要有两方面原因:(1)传统方法的操作响应时间过长,难以支持自由、和谐和自然的人机/人人交互;(2)传统方法重点考虑共享对象的结果一致性,而忽略了用户的操作意图.

为了解决上述问题,支持操作意图一致性的实时协同编辑算法逐步成为目前的研究热点[3-5].该类算法在较高的网络延迟下也能保证操作的高响应性,尽可能维护多个用户的操作意图,简化冲突检测和消解,因而逐步成为首选的一致性维护方法.目前,该类算法的相关研究成果已经在各个领域都有广泛的应用.例如,支持协同办公[12-15];支持多媒体图形图像协同编辑[16-23];支持软件协同开发[24-25];支持协同 CAD[26-29]及支持 p2p(peer to peer)协同编辑[30]等.

理论研究方面,Ellis 和 Gibbs[2]最早在 1989 年以 GROVE 系统为背景首先提出操作转换(Operational Transformation,OT)的基本模型和算法,来维护实时协同编辑过程中共享对象的一致性.以此为开端,OT 算法在 Ressel、Sun 和 Li 等人[5,12,16,18,31-40]的指引和推动下得到了大量的探索,发表在 ACM CSCW 会议、ACM Siggroup 会议、ACM TOCHI 期刊、IEEE TPDS 期刊、IEEE TOC 期刊以及国内的《计算机学报》等期刊上.OT 算法的主要思想是本地产生的操作立刻执行,接收到的远程操作需要与本地操作历史中已执行的并发操作进行操作转换后再执行.与传统的锁机制和串行化方法相比,OT 算法的优势是具有很好的本地响应性[41-51].同时,OT 算法也面临一些技术挑战,例如,设计正确的操作转换函数具有困难,不断有算法被找出特定协同场景下的“puzzle”;大部分 OT 算法没有良好的伸缩性等[38-40,52-61].

地址空间转换算法作为另一类支持操作意图一致性的协同编辑算法,为共享对象的一致性维护提供了新的思路,典型代表为 AST(Address Space Transformation)算法[4,62].与 OT 算法相比,这类算法在处理远程操作时,并不是通过修正操作来获取操作的正确执行位置,而是通过计算文档的地址空间来将文档状态回退到操作产生时的状态来获取操作的执行位置.为实现文档地址空间的正确转换,采用标记和回溯的方法来实现.因此,该类方法又被称为标记和回溯(Mark&Retrace)方法.基于 AST 算法,各种相关算法被提出,发表在 WWW 会议、ACM CSCW 会议以及 ACM Siggroup 等会议上[4,62-69].例如,文献[64]提出了用来解决大规模协同的一致性维护方法.文献[65]提出了用于解决 Web2.0 环境下基于 XML 结构的镜像站点同步方法.文献[66]提出了用于解决协同评论中部分复制树形结构数据的一致性维护方法.文献[67]提出了用于支持异构协同编辑云服务的一致性维护方法.

可交换的复制式的数据类型(Commutative Replicated Data Type,CRDT)作为一类新的支持操作意图一致性的协同编辑算法被提出并得到了深入的研究.以法国的研究机构 INRIA 为代表,提出了一系列 CRDT 相关算法,发表在 ACM CSCW 会议、ACM Siggroup 会议以及 IEEE TPDS 期刊等上[30,70-81].CRDT 算法不需要保存操作历史,并发操作之间不需要进行操作转换.通过分配给所有操作对象唯一的标识符 ID(Identifier),使得并发操作之间可交换执行.大部分 CRDT 算法具有良好的伸缩性,适合应用于大规模的 p2p 协同编辑领域[77-80].

① http://dl.acm.org/ccs/ccs.cfm
② http://cscw.acm.org/2014/program_workshop.html#W9
③ http://www.waveprotocol.org/whitepapers/org/whitepapers/operational-transform
④ http://www.codecommit.com/blog/ja-va/understanding-and-applying-operational-transformation
⑤ http://www.codoxware.com
⑥ http://github.com/opencoweb/coweb#readme

CRDT 算法也面临一些技术挑战，例如，如何设计并发操作的可交换执行、如何给操作对象分配全局唯一的 ID 以及如何减少 ID 的空间开销等.

作为已有文献[52]的进一步拓展，文中主要进展体现在两个方面：(1) 以支持操作意图一致性为主线，对实时协同编辑算法进行全面综述；(2) 讨论的算法不仅限于 OT 算法，还包括地址空间转换算法和 CRDT 算法.

本文第 2 节在对基本问题、概念、术语和标志性研究成果进行分析的基础上，给出操作意图一致性的路线图，对支持操作意图一致性的算法进行分类；基于上述路线图和分类，第 3、4、5 节综述各类算法的原理和研究现状；第 6 节给出各类算法的对比与小结；第 7 节进行全文小结和未来研究展望.

2　基本概念和术语

2.1　偏序、因果关系、并发关系

操作间的偏序最早源于 Lamport 事件偏序关系，即 happened before 和 concurrent[1] 的逻辑时钟 (Logical Clocks). 基于 Lamport 的偏序事件关系，文献[3,61]给出协同编辑系统中的因果关系和并发关系，详见定义 1 和定义 2.

定义 1.　因果关系. 给定任意两个分别位于站点 i 和站点 j 的操作 O_a 和 O_b，当且仅当 O_a 和 O_b 满足下列 3 个条件之一时，称 O_a 和 O_b 存在因果关系 (记作 $O_a \rightarrow O_b$)：① $i=j$，O_a 发生在 O_b 之前；② $i \neq j$，O_a 在站点 j 的执行先于 O_b 的产生；③ 存在操作 O_c，有 $O_a \rightarrow O_c$ 并且 $O_c \rightarrow O_b$.

定义 2.　并发关系. 给定任意两个操作 O_a 和 O_b，当且仅当 O_a 和 O_b 既不满足 $O_a \rightarrow O_b$，又不满足 $O_b \rightarrow O_a$ 时，称 O_a 和 O_b 存在并发关系 (记作 $O_a \parallel O_b$).

基于文献[3-4,38-40,52,61]中的讨论，文中给出了简单并发关系和偏并发关系的定义，详见定义 3 和定义 4.

定义 3.　简单并发关系. 当满足并发关系的两个操作 O_a 和 O_b 产生于相同的文档状态时，称 O_a 和 O_b 是简单并发关系.

定义 4.　偏并发关系. 当满足并发关系的两个操作 O_a 和 O_b 产生于不同的文档状态时，称 O_a 和 O_b 是偏并发关系.

图 1 用时空图 (time-space) 给出操作之间的因果和并发关系. 其中，O_1 与 O_2 和 O_1 与 O_3 为因果关系，O_2 与 O_3、O_1 与 O_4、O_2 与 O_4 以及 O_3 与 O_4 为并发关系. 其中，O_2 与 O_3 和 O_1 与 O_4 为简单并发关系，O_2 与 O_4 和 O_3 与 O_4 为偏并发关系.

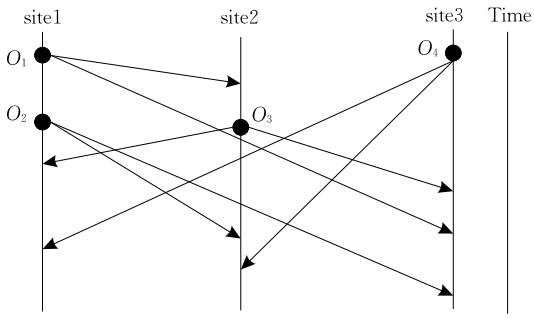

图 1　偏序关系的时空图

图2采用因果执行图(Causally Executable Graph)给出操作之间的因果和并发关系[75]. 图中的顶点表示各站点产生的操作，操作之间的有向边表示操作的因果关系，其中，有向边的起点是有向边终点的因操作. 操作之间的无向边表示操作的并发关系，包括简单并发关系和偏并发关系. 如图 2 所示，$O_1 \rightarrow O_2$，$O_1 \rightarrow O_3$，$O_2 \parallel O_3$，$O_1 \parallel O_4$，$O_2 \parallel O_4$，$O_3 \parallel O_4$.

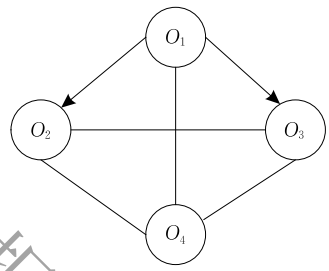

图 2　偏序关系的因果执行图

除了上述偏序定义之外，Duchien 等人[82] 给出协同系统中分布式对象内部和对象之间的行为或方法的偏序关系；Sun 等人[36-37] 基于操作上下文给出原始操作和转换操作的上下文依赖和并发关系，该关系兼容了 Sun 的因果关系和并发关系；Oster 等人[70] 基于操作语义，给出了操作之间基于语义的因果(依赖)关系和并发关系；Imine 等人[53] 也基于操作语义给出了插入(删除)操作与插入操作之间的依赖关系，即所有操作仅仅与插入操作存在依赖关系，而删除操作之间不存在依赖关系.

2.2　全序

相比偏序关系定义的客观性，全序关系的定义更依赖于某种“假设”. 例如，Lamport[1] 提出的全序关系，如果逻辑时钟相等，假设可以通过进程 ID 号来确定次序，从而得到一种全序关系. 类似地，实时协同编辑算法采用不同假设和策略来定义全序，基于某种“假设”来对操作或者操作对象进行定序.

2.2.1 操作的全序

操作的全序是给实时协同编辑系统中产生的所有操作赋予唯一的全局次序,并按照全序进行操作转换或按照全序执行.操作的全序有两类具体方法.

分布式全序.代表性的方法有:GOT 算法基于 SV 和站点编号(siteID)定义了操作间的一种全序[33];TIBOT/TIBOT2.0 以线性的时间间隔 TI(Time Interval)和 siteID 定义了来定义操作间一种的全序[35,38].

集中式全序.又可分为中心服务器取全序和中心序列产生器取全序.前者将各协同站点产生的操作发送到中心服务器,中心服务器采用先进先出(First in First out)策略将接收操作序列化之后转发给其它站点,如 Jupiter 系统[32]、Google Wave/Docs(参见本文第 3 页脚注③④)和 NICE[34].后者将中心序列产生器产生的连续的正整数赋予给各站点产生的操作,如 SOCT3/4 通过一个中心 Sequencer 定义了操作间全序[11,31].某些算法系统,例如 COT 既可采用分布式取全序,也可采用集中式取全序[36-38].

2.2.2 操作对象的全序

操作对象的全序指在协同编辑系统中,操作对象在系统内部数据结构的全局位置 ID 标识.

协同编辑算法通过维护内部数据结构中操作对象的全序位置关系来维护共享对象的一致性.代表性算法有 AST 算法,采用了地址空间转换技术维护每个站点字符节点间一致的顺序[4,62].ABT 家族算法通过操作转换技术维护各站点操作对象间相同的全序位置关系[46-51].CRDT 家族算法通过给操作对象分配全局有序的唯一 ID,并通过 ID 将操作对象全序的保存到内部数据结构中实现各站点内部数据结构中操作对象间全序的位置关系[30,70-76].

2.2.3 全序假设与优先级

全序的定义,在前面若干判定条件相等的情况下,往往假设可以用到优先级(Priority)定序.上述操作全序中的分布式全序方法和操作对象的全序方法,主要采用 siteID 作为优先级来对全序定义进行完备化.例如,在 GOT 中,当两个操作的 SV 的和相同时,siteID 小的操作全序在前[33].在 TIBOT/TIBOT2.0 中,当两个操作的 TI 相同时,siteID 小的操作全序在前[35,38].在 SDT、SDTO、LBT 及 ABT 中,当两个并发的插入操作在同一位置插入不同操作对象时,siteID 小的操作对象全序在前[42-46].在 AST 中,当两个插入字符的 SV 和相同时,siteID 小的插入字符位置在前[4,62].在 WOOT 中,siteID 小的操作对象全序在前[70].在 TreeDoc 中,当两个操作对象的逻辑时钟相同时,siteID 小的操作对象的 ID 全序小[73].在 RGA 中,当两个操作对象的 session 相同且 SV 的和相同时,siteID 小的操作对象的 ID 全序小[75].

这些方法,本质上与 Lamport 的假设可以把进程 ID 号作为优先级进行全序完备化方法相似.

2.3 一致性模型

支持操作意图一致性的协同编辑算法可参照一致性模型来设计和开发.

(1) CC 模型

1996 年,Ressel 等人在文献[83]中明确阐述了协同编辑算法要满足两个一致性条件:因果一致性(Causality Preservation)和结果一致性(Convergence),并提出了 CC 模型.

(2) CCI 模型

后续研究发现 CC 模型不足以完整约束一个操作转换系统的行为.因此,Sun 等人[3,61]在 CC 模型的基础上进行了完善,指出一个协同编辑系统除了要满足因果一致性和结果一致性外,还需要做到操作意图一致性,并给出 OT 算法应遵循的 CCI 模型:因果一致性、结果一致性以及操作意图一致性(Intention Preservation).大多数算法是参照 CCI 模型设计和开发的.

(3) CA 模型

Li 等人在文献[46]中指出 CCI 模型中意图保持定义的模糊性(ambiguity),提出了可以被形式化证明的 CA 模型:因果一致性和 Admissibility 属性.其中,Admissibility 属性蕴含了操作意图一致性和结果一致性这两方面含义.参照 CA 模型设计的协同编辑算法可以被形式化证明其算法的正确性.

2.3.1 因果一致性和结果一致性

文献[3,61]给出了因果一致性和结果一致性的描述,如定义 5 和定义 6.

定义 5. 因果一致性.给定任意一对操作 O_a 和 O_b,如果 $O_a \rightarrow O_b$,那么在所有站点,O_a 在 O_b 之前执行.

定义 6. 结果一致性.当一个协同会话在静默状态时,所有站点共享文档的副本是一致的.

因果一致性一般可通过 SV 定义和实现.

结果一致性一般可通过两种方法实现:(1)设计算法的转换函数满足转换属性 TP1(Transformation Property 1)和 TP2(Transformation Property 2)[54];(2)设计算法的控制过程避免 TP1/TP2 的约束,即可以构建唯一全序的操作转换路径[31-38].已有研究

成果表明，设计转换函数满足 TP1/TP2 非常困难，相继有学者找出 dOPT 不能满足 TP1 和 TP2 的“puzzle”，adOPTed 不满足 TP2 的“puzzle”，SOCT2 不满足 TP1 的“puzzle”以及 GOTO 不满足 TP1 的“puzzle”[40,55-56]. 相比之下，第 2 类方法更容易实现结果一致性.

为了解决 TP1 和 TP2 的“puzzle”问题，很多学者做了大量的研究和探索. Sun 在 2014 年文献[56]中从转换函数层面上总结了 OT 算法的“puzzle”问题，包括 do 操作的 TP1/TP2 的“puzzle”问题以及 undo 操作的 IP1 和 IP2/IP3 的“puzzle”问题. 针对 do 和 undo 的“puzzle”问题，Sun 认为可以从转换函数的层面上解决，但需要结合具体的应用来设计相应的转换函数；也可以从算法的控制层面上考虑设计避免 TP1/TP2、IP1 和 IP2/IP3 约束的 OT 算法. 此外，Randolph 等人[40]在 2015 年 TOC 上发表的文献中总结了已有算法存在的“puzzle”问题，并利用自动机合成了满足 TP1 和 TP2 的操作转换函数.

2.3.2　操作意图一致性

Sun 等人[3,61]最早通过操作的“执行效果(execution effect)”给出“操作意图(intention of an operation)”的一个笼统定义，并未严格定义、度量和说明“执行效果”. Sun 等人[11]认为操作意图的具体定义要依赖于具体应用中的“操作语义(operation's semantics)”，并且不能通过串行化方法达到.

Sun 等人[3,61]从两个层面上考虑维护操作意图的一致性：(1)考虑算法的控制过程，Sun 等人给出了维护操作意图一致性的一个通用框架，即可采用维护操作的因果一致性和操作的全序来构建操作转换路径；(2)考虑算法的操作转换函数，Sun 等人指出通过设计操作转换函数来维护操作的意图需要依赖于具体应用中的操作语义.

基于以上讨论，Sun 的操作意图的维护，结合了 Sun 同时代多版本的含义[57]，多种结果只要获得其中的任意一个一致性结果，即认为满足操作意图一致性.

廖斌等人在文献[52]中指出 Sun 的操作意图问题描述的不足，提出在算法框架中“意图保持指每一步操作都要解决这两类并发问题”.

Li 等人在文献[46]也指出 Sun 的意图保持的模糊性，给出协同文本编辑的意图保持的一种可以被量化和形式化的 CA 模型. 模型中的 Admissibility 属性为每个操作在执行状态时都不会违背之前已经建立的操作效果关系(Operation Effects Relation)[46]. Li 认为操作效果关系为操作对象之间的位置关系，只要每一步操作的执行都不违背之前建立的操作效果关系，最终各站点内部数据结构保存的操作对象之间有相同的全序位置关系，即维护了操作意图的一致性.

基于 Li 等人提出的 Admissibility 属性，Shao 等人[41]给出了操作效果序(Effects Relation Order，ERO)概念. ERO 指在某个文档状态上出现过的所有原子对象(或字符)都可以通过它们之间的相对位置得到一种全序关系. Shao 等人认为，各站点产生操作(issue operation)时，操作对象之间建立了某种 ERO，通过执行调度算法和转换函数，只要确保每一步操作的执行不违背已建立的 ERO，那么就可保证，当所有操作在各站点都执行后，各站点内部数据结构保存的操作对象之间有相同的 ERO，即维护了操作意图的一致性.

尽管上述研究对操作意图的理解不同，维护方法不同，但都采用了操作转换技术来维护操作意图一致性.

不同于上述研究，Gu 等人[4,62]率先给出了维护操作意图一致性的新思路—地址空间转换技术. Gu 等人[4,62]认为意图保持可以通过转换文档的地址空间实现. 当处理并发操作时，并不需要转换操作，而是将文档的地址空间回溯到操作产生时的状态，在该状态下执行操作可以获取正确的执行位置. 这种方法可以保证当所有的操作在各站点执行结束后，各站点线性结构存储的操作字符的顺序是一致的，实现了结果一致性和操作意图一致性.

与 OT 技术和地址空间转换技术不同，CRDT 方法维护操作意图一致性需要给每个操作对象分配全局唯一的 ID，结合不同的调度算法将操作对象全序的映射到内部的数据结构中，确保内部数据结构中操作对象的全序位置关系，从而实现操作意图一致性，并收敛于一致性的结果.

基于上述讨论，综合相关文献，文中给出操作意图一致性维护的研究路线图如下：

(1)建立维护操作意图一致性的整体框架，即每一步的执行都要维护操作的意图.

(2)针对某种“应用”，基于某种“效果”，描述或者定义某种“操作意图”.

(3)基于(2)中操作意图的描述或者定义，设计相应的操作意图一致性的维护算法.

虽然各类算法对操作意图的理解不同，定义不同，维护方法不同，应用效果不同，但到目前为止，基

本上都能够被上述路线图所覆盖.

基于上述对操作意图及一致性问题的讨论,可将代表性的操作意图一致性的维护算法大体分为三类,操作转换算法、地址空间转换算法和可交换的复制式的数据类型,分别见第 3、4、5 节.

3 操作转换算法的进展和现状

3.1 支持 Sun 的操作意图一致性的相关算法

3.1.1 Sun 的操作意图一致性

Sun 基于操作的"执行效果"描述"操作意图",给出了实现操作意图一致性需要满足两个条件:(1)操作在产生时执行和在远程站点执行具有相同的执行效果;(2)操作和与其并发操作的执行效果互不干涉[3,61].

图 3 为一个 session 中 3 个站点产生操作和接收操作的场景.文档初始状态为"abc",各站点产生操作的情况为:site1 产生的本地操作为 $O_1=del(1)$,$O_4=del(0)$;site2 产生的本地的操作为 $O_2=insert(2,x)$,$O_5=del(0)$;site3 产生的本地操作为 $O_3=insert(1,y)$,$O_6=insert(2,z)$.当 site1 和 site2 经过操作转换并执行完所有操作后,site1 的文档状态为"xzyc",site2 的文档状态为"yzxc".按照 Sun 的操作意图,最终 3 个站点都收敛于"xzyc"或者 3 个站点都收敛于"yzxc",都符合 Sun 的操作意图一致性.

Sun 的这种操作意图一致性接近于多版本的含义,这就是 Sun 意图一致性的模糊性所在.与多版本区别在于,Sun 的操作意图一致性通过某个全序来自动选择其中的一个一致性结果.而多版本方法首先保证多个版本的有效正确的分裂与管理,然后让用户交互选择其中的一个一致性结果.

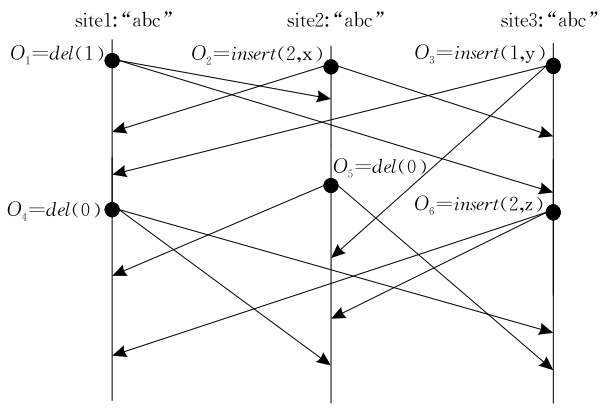

图 3 3 个站点初始状态为"abc"的协同编辑场景

3.1.2 Sun 的操作意图的相关算法

支持 Sun 的操作意图一致性算法主要采用操作的全序(见文中 2.2.1 节)方法实现,分为分布式全序方法和集中式全序方法.

分布式全序方法中,基于算法的控制过程可进一步分为两类.

(1)采用 undo/redo 并发控制策略,使算法的调度严格按全序进行操作转换并按照全序执行,典型代表是 GOT 和 TIBOT[33,35],具体执行的详细步骤见文献[52].这类算法的特点是操作转换函数的设计简单,不需要满足 TP1 和 TP2.但是这类算法的控制过程复杂,需要构建唯一的操作转换路径.近年来研究指出,第一类方法中的 undo/redo 并发控制策略并不高效,而且可能会引起异常的交互界面[38-39].

(2)算法严格按全序进行操作转换,但并不严格按全序执行,典型代表是 SOCT3/4、COT 和 TIBOT2.0[31,36-38].不同于第 1 类方法,第 2 类方法的特点是所有操作不需要按照全序进行调度和执行,而仅仅是在与并发操作序列进行操作转换时,将并发操作序列按照操作的全序排列后逐一进行操作转换后再执行.与第 1 类方法相比,这类方法的控制过程简单,不需要构建唯一的操作执行路径.由于各站点操作的执行顺序不同,为得到一致性的结果,这类方法的操作转换函数的设计需要满足 TP1,因此,这类方法的操作转换函数的设计比第 1 类方法复杂.作者给出了按全序进行操作转换算法的执行步骤,如算法 1 所示.

算法 1. 按全序进行操作转换算法.

本地操作集成

输入:本地产生的操作 O

输出:传播到其它站点的操作 $O_{request}$

1. O 产生后立刻执行,将 O 集成到 H 中.
2. 获取 O 的传播形式 $O_{request}$.
3. 将 $O_{request}$ 向各个站点传播.

远程操作集成

输入:来自其它站点的操作 $O_{request}$

输出:操作 $O_{request}$ 的执行形式 $EO_{request}$

1. 选择具备调度条件的远程操作 $O_{request}$.
2. 找与 $O_{request}$ 并发的操作,并将 $O_{request}$ 与其并发操作按全序进行操作转换后得到 $EO_{request}$.
3. 执行 $EO_{request}$,将 $EO_{request}$ 集成到 H 中.

在本地操作集成中,SOCT3 和 COT 的集成过程比 TIBOT2.0 和 SOCT4 简单.原因是 SOCT3 和 COT 的本地操作 O 执行后立刻传播,即 $O_{request}=O$.而 TIBOT2.0 和 SOCT4 的本地操作 O 执行后延迟传播,对 O 进行了预处理,传播形式 $O_{request}$ 需要包含

全序在 O 前所有并发操作对文档产生的效果. 在远程操作的集成中，TIBOT2.0 和 SOCT4 的集成过程比 SOCT3 和 COT 简单，原因是在传播前已经包含了全序在 O 前的并发操作的执行效果，因而在远程站点仅需考虑全序在 O 后的并发操作对文档产生的效果.

集中式全序方法，典型代表为 Jupiter 系统[32]、Nice 系统[34]和 Google Wave/Docs(参见本文第 3 页脚注③④). 上述系统的各协同站点都与中心服务器相连接，协同站点执行本地操作后将操作传送到中心服务器. 中心服务器负责全序操作(必要时进行操作转换后)，并将操作向各站点传播. 协同站点接收到服务器传播的操作后，进行操作转换后执行，并将转换后的操作存储到内部的数据结构中. 其中，Jupiter 系统的服务器和协同站点采用了 2D 的状态空间存储执行的操作，空间开销较大. Google Wave/Docs 的服务器采用了线性表来存储操作，空间开销比 Jupiter 系统小. Nice 系统的服务器和协同站点都采用了线性表存储操作，空间开销比 Jupiter 系统和 Google Wave/Docs 小. 上述讨论的算法的服务器端都需要进行操作转换，会导致 OT 算法调度过程复杂，伸缩性较差[39].

Xu 和 Sun[39]在 2016 年发表在 TPDS 上一篇文献中提出了一种新的 OT 算法 POT(Pattern-Based OT)，该算法也采用了集中式全序方法来实现操作意图的一致性. 与上述讨论的算法不同，POT 的服务器端不需要进行操作转换，所有的操作转换都是在协同站点执行. POT 的优点是处理远程操作的时间复杂度为 $O(N)$，其中 N 为并发操作的数量.

3.1.3　Sun 的操作意图下的协同工作场景的算例对比分析

为了更清晰阐述 Sun 的操作意图下的相关算法的执行异同，本节结合图 3 中具体的协同工作场景中的算例，给出各典型算法在 site2 的执行异同和 Sun 的操作意图一致性的应用效果. 假设 siteID 从小到大为 site1＜site2＜site3，其中 H 为操作历史，S 为文档状态. 表 1 列出 Sun 的操作意图下各典型算法在 site2 的执行异同. 各典型算法在 site2 都收敛一致性的结果“xzyc”，基于图 3 中的算例，Sun 的操作意图一致性的应用效果为“xzyc”.

典型代表 GOT 和 TIBOT 严格按照操作的全序进行操作转换和执行，即操作转换路径等于操作执行路径. 在 TIBOT2.0、SOCT3/4 和 COT 算法中，仅仅操作转换按照全序进行，而操作执行路径不是按照全序构建. 其中，TIBOT2.0 和 COT 的操作的执行路径按照操作的接收顺序构建，而 SOCT3/4 的操作的执行路径除了按照操作的接收顺序构建，还需要确保接收的远程操作按照 CGO(Continuous Global Order)进行调度. 由表 1 可知，Sun 的典型算法得到了一致性的操作对象的全序位置关系为“x”≺“z”≺“y”≺“c”.

表 1　Sun 的操作意图下的典型算法在 site2 的执行异同

典型算法	操作的全序	操作执行路径	H	操作转换路径	S
GOT	基于 SV，操作的全序为 $O_1 \prec O_2 \prec O_3 \prec O_5 \prec O_4 \prec O_6$	$O_1, O_2', O_3', O_5, O_4', O_6'$ $O_2' = insert(1,x), O_4' = \varnothing,$ $O_3' = insert(2,y),$ $O_6' = insert(1,z)$	$O_1, O_2', O_3', O_5, O_4', O_6',$ $O_2' = insert(1,x),$ $O_4' = \varnothing,$ $O_3' = insert(2,y),$ $O_6' = insert(1,z)$	$O_1, O_2', O_3', O_5, O_4', O_6',$ $O_2' = insert(1,x),$ $O_4' = \varnothing,$ $O_3' = insert(2,y),$ $O_6' = insert(1,z)$	“xzyc”
TIBOT	令 TI=0 产生 O_1, O_2, O_3，TI=1 产生 O_5，TI=2 产生 O_4, O_6，基于 TI，操作的全序同 GOT				
TIBOT2.0	同 TIBOT	$O_2, O_1', O_5, O_3', O_6', O_4'$ $O_1' = del(1), O_3' = insert(1,y),$ $O_6' = insert(1,z), O_4' = \varnothing$			
SOCT3 SOCT4	基于 CGO，操作的全序同 GOT	$O_2, O_1', O_5, O_3', O_4', O_6'$ $O_1' = del(1), O_3' = insert(1,y),$ $O_4' = \varnothing, O_6' = insert(1,z)$			
COT	可采用集中式全序或分布式全序，操作的全序同 GOT	$O_2, O_1', O_5, O_3', O_6', O_4'$ $O_1' = del(1), O_3' = insert(1,y)$ $O_6' = insert(1,z), O_4' = \varnothing$	$O_2, O_1, O_5, O_3, O_6, O_4$		

3.2　支持 Li 的操作意图一致性的相关算法

3.2.1　Li 的操作意图一致性

Li 将操作意图描述为操作效果关系，即操作对象之间的前后位置关系[46]. 只要每个操作在执行状态时都不会违背之前已经建立的操作效果关系，就能实现操作意图的一致性，收敛于正确结果.

基于文献[42-46]的讨论，结合图 3 所示的场景，各站点产生操作 O_1, O_2, O_3 时，建立的操作效果关系为“a”≺“y”≺“b”≺“x”≺“c”. 当 site3 产生 O_6 时，建立的操作效果关系为“a”≺“y”≺“z”≺“x”≺“c”. 按照 Li 的操作意图一致性，当各站点都执行完所有操作后，只有三站点都收敛于“yzxc”才符合 Li 的操作意图一致性. Li 等人[46]引入操作效果关系这种语义，对全序假设的完备性进行了修正(不完全依

赖 siteID),使得操作意图一致性不再模糊,并且符合某种语义关系.

3.2.2 Li 的操作意图的相关算法

支持 Li 的操作意图一致性的算法可采用操作对象的全序方法实现(见本文 2.2.2 节),进而分 3 类:(1)构建无方向的操作转换路径;(2)构建双向的操作转换路径;(3)构建优化的历史记录 H,插入操作序列 H_i 在前,删除操作序列 H_d 在后.

第 1 类方法典型代表是 SDT 和 SDTO[42,44-45]. 这类方法的特点是各站点在集成远程操作 O 时,O 与各站点的操作历史中存储的已经执行的并发操作的转换顺序不唯一,即各站点从相同的文档状态出发,沿着不同的操作转换路径进行操作转换后再执行. 为了确保各站点收敛一致性的结果,这类方法需要设计转换函数同时满足 TP1 和 TP2. 虽然 SDT 和 SDTO 声称其设计的操作转换函数满足 TP1 和 TP2,但不满足 TP2 的"puzzle"被文献[54]指出. 近年来的研究表明,在采用非全序的 OT 算法中,目前只有 TTF 和 Randolph 等人[40,54]经过自动机合成的操作转换函数可以满足 TP1 和 TP2.

第 2 类方法典型代表是 LBT[43],这类方法的特点是构建了双向的操作转换路径,如图 4 中的①和②. 采用双向操作转换路径的原因是在非全序的 OT 算法中,为了实现结果一致性,需要设计操作转换函数同时满足 TP1 和 TP2. 然而,设计插入操作的 IT 函数和 ET(Exclusive Transformation)函数满足 TP2 非常困难. 因此,在集成远程操作时,如果 O 为删除操作,从 s 出发,沿着图 4 中①进行操作转换,即按照操作历史中存储的已执行的并发操作序列进行操作转换;如果 O 为插入操作,沿着图 4 中②进行操作转换,即将 O 排除其因操作序列 P_b 中的删除操作的效果回退到状态 s'',然后从 s'' 出发,构建转换路径 P_f,得到了 O 在 s' 下的正确的执行形式 O'.

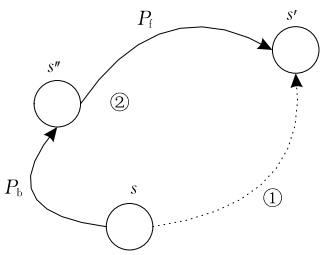

图 4 双向操作转换路径

第 3 类方法的典型代表是 ABT[46],ABT 是第 1 个经过严格形式化证明能够实现意图保持的 OT 算法. 这类方法的特点是构建了优化操作历史记录 $H=H_i \cdot H_d$,其中 H_i 代表操作历史中按照操作的执行顺序存储的插入操作序列,H_d 代表操作历史中按照操作的执行顺序存储的删除操作序列. 每当执行一个本地或远程操作后,都根据操作的类型,将其集成到 H_i 或 H_d 中. 在操作历史 H 中,H_i 排列在 H_d 前的原因是排除删除操作对插入操作产生的影响,维持操作对象的前后位置关系,确保各站点收敛一致性的结果. 作者给出了 ABT 算法的执行步骤,如算法 2 所示.

算法 2. ABT 算法.

本地操作集成

输入:本地产生的操作 O

输出:传播到其它站点的操作 O_{request}

1. O 产生后立刻执行.
2. 将 O 与 H_d 进行交换转换得到 O_{request} 和 H'_d.
3. 将 O_{request} 向各个站点传播.
4. 如果 O 为插入操作,$H=H_i \cdot O_{\text{request}} \cdot H'_d$;如果 O 为删除操作,$H=H_i \cdot H_d \cdot O$.

远程操作集成

输入:来自其它站点的 O_{request}

输出:O_{request} 的执行形式 EO_{request}

1. 选择具备因果保持的远程操作 O_{request},将 H_i 分为 O_{request} 的因操作 H_{ih} 和与 O_{request} 并发的插入操作 H_{ic}.
2. 将 O_{request} 与 H_{ic} 进行 IT 得到 O''_{request}.
3. 将 O''_{request} 与 H_d 进行 IT 得到 O'_{request}.
4. $EO_{\text{request}}=O'_{\text{request}}$,执行 EO_{request}.
5. 如果 O 为插入操作,$H'_d=IT(H_d, O''_{\text{request}})$,$H=H_i \cdot O''_{\text{request}} \cdot H'_d$;如果 O 为删除操作,$H=H_i \cdot H_d \cdot O'_{\text{request}}$.

3.2.3 Li 的操作意图下的协同工作场景的算例对比分析

为了更清晰阐述 Li 的操作意图下的相关算法的执行异同,结合图 3 中的协同工作场景中的算例,给出各典型算法在 site2 的执行情况和 Li 的操作意图一致性的应用效果. 表 2 列出了 Li 的操作意图下各典型算法在 site2 的执行异同. 各典型算法在 site2 都收敛一致性的结果"yzxc",即 Li 的操作意图一致性的应用效果为"yzxc".

典型代表 SDT、LBT 和 ABT 的操作的执行路径相同,都是按照操作的接收顺序构建的. SDT 的操作历史中操作的排列顺序、操作的转换路径及操作的执行路径都相同. LBT 的操作历史中操作的排列顺序等于操作的执行路径. 而 LBT 的操作转换路径则是根据操作类型构建了双向路径. 当接收的是删除操作,操作转换路径等于操作历史中并发操作的排列顺序;当接收的插入操作,构建转换路径 P_b 和 P_f,然后沿着 P_b+P_f 的路径进行操作转换. ABT

的操作历史中操作的排列顺序和操作的转换路径相同，都是按照插入操作序列在前，删除操作序列在后构建的. 表 2 列出的 Li 的典型算法得到了一致性的操作对象的全序的位置关系“y”≺“z”≺“x”≺“c”.

表 2　Li 的操作意图下的典型算法在 site2 的执行异同

典型算法	操作执行路径	H	操作转换路径	S	操作对象的全序关系
SDT	$O_2, O_1', O_5, O_3', O_6', O_4'$ $O_1'=del(1)$, $O_3'=insert(0,y)$, $O_6'=insert(1,z)$, $O_4'=\varnothing$	$O_2, O_1', O_5, O_3', O_6', O_4'$ $O_1'=del(1), O_3'=insert(0,y)$, $O_6'=insert(1,z), O_4'=\varnothing$	$O_2, O_1', O_5, O_3', O_6', O_4'$ $O_1'=del(1), O_3'=insert(0,y)$, $O_6'=insert(1,z), O_4'=\varnothing$	“yzxc”	“y”≺“z”≺“x”≺“c”
LBT			删除操作：$O_2, O_1', O_5, O_3', O_6', O_4'$ 插入操作：$P_b=O_1', O_5, O_4'$, $P_f=O_2, O_3', O_6', O_1', O_5, O_4'$		
ABT		$O_2, O_3, O_6, O_1'', O_5, O_4'$ $O_1''=del(3), O_4'=\varnothing$	$O_2, O_3, O_6, O_1'', O_5, O_4'$ $O_1''=del(3), O_4'=\varnothing$		

3.3　支持 Shao 的操作意图一致性的相关算法

3.3.1　Shao 的操作意图一致性

Shao 等人将操作意图描述为操作效果序(ERO)，只要确保每一步操作的执行不违背已建立的 ERO，当所有操作在各站点都执行后，各站点内部数据结构保存的操作对象之间有相同的 ERO，即维护了操作意图的一致性，收敛于正确的结果. 实际上，Shao 的操作意图和 Li 的操作意图本质上相同，都是操作对象之间的前后位置关系，通过维护操作对象之间全序的位置关系来维护操作意图的一致性[41,46].

基于文献[47-51]的讨论，结合图 3 所示的场景，当各站点都执行完所有操作后，只有 3 站点都收敛于“yzxc”才符合 Shao 的操作意图一致性. Shao 等人引入操作效果序这种语义，对全序假设的完备性进行了修正(不完全依赖 siteID)，使得操作意图一致性不再模糊，并且符合某种语义关系.

3.3.2　Shao 的操作意图的相关算法

Shao 的操作意图一致性的算法可采用操作对象的全序方法实现(见本文 2.2.2 节). 在 ABT 算法的基础上，Shao 等人[47]提出了 ABTS 算法. ABTS 将 ABT 处理的对象从单个字符扩展为 string，在操作转换过程中可以处理 string 的重叠和级联分裂等情况. 基于 ABTS，优化版本 ABTSO 被提出[48]，ABTSO 中的操作预先按照 ERO 排序，在操作历史中找并发操作的时间为线性时间. ABTSO 的不足在于处理 string 的删除操作时，不支持 string 的级联分裂和重叠，而是将 string 的删除操作转换为面向字符的删除操作来处理. 上述讨论的 ABT、ABTS 和 ABTSO 处理的对象为单个操作，而 ABST[49-50]处理的对象为操作序列. 不仅预先将 H_i 和 H_d 按照 ERO 排序，同时一个操作序列 T 用来存储本地未发送的插入操作序列 T_i 和未发送的删除操作序列 T_d，且 T_i 和 T_d 也分别按 ERO 排序. 与 ABT、ABTS 和 ABTSO 相比，ABST 更适合应用在移动设备中和异步协同编辑领域. 基于 ABT 算法，Shao 等人[51]提出了支持 selective undo 功能的 ABTU 算法，ABTU 通过预先对 H_i 和 H_d 按照 ERO 排序，使处理远程操作的时间复杂度从 COT 的指数级降到线性时间.

3.3.3　Shao 的操作意图下的协同工作场景的算例对比分析

与 Sun 和 Li 的操作意图下的典型算法不同，Shao 的操作意图下的典型算法 ABTS 和 ABTSO 的操作对象都是 string，为了清晰阐述 Shao 的操作意图下典型算法执行异同，这里结合图 5 中的协同工作场景和算例，给出 ABTS 和 ABTSO 在 site2 的执行情况和 Shao 的操作意图一致性的应用效果.

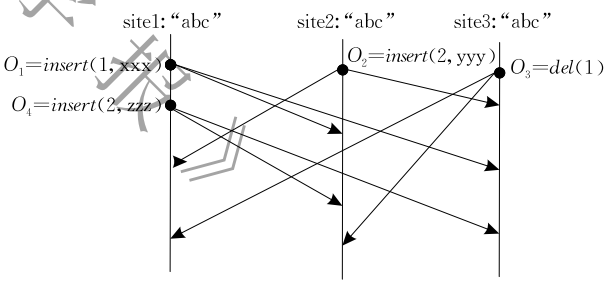

图 5　支持 string 操作的协同编辑场景

图 5 为一个 session 中 3 个站点产生操作和接收操作的工作场景. 文档初始状态为“abc”，各站点产生操作的情况为 site1 产生 $O_1=insert(1,xxx)$，$O_4=insert(2,zzz)$；site2 产生 $O_2=insert(2,yyy)$；site3 产生 $O_3=del(1)$. site1 产生 O_1 和 O_4 时，site1 本地操作历史 H_d 中没有删除操作，因而 O_1 和 O_4 不需要排除删除操作的效果，可直接向其它站点传播. site3 产成 O_3 时，site3 本地操作历史 H_d 中没有删除操作，因而 O_3 直接向其它站点传播.

表 3 列出了 Shao 的操作意图下的 ABTS 和 ABTSO 算法在 site2 的执行异同. 典型代表 ABTS 和 ABTSO 的操作的执行路径相同，都是按照操作

的接收顺序构建的.ABTS的操作历史中的插入操作序列和删除操作序列都是按照操作执行顺序排列.而ABTSO的操作历史中的插入操作序列和删除操作序列按照操作效果序排列.表3列出的Shao的ABTS和ABTSO算法得到了一致性的结果"axzzzxxbyyyc".操作对象的全序的位置关系"a"≺"x"≺"z"≺"z"≺"z"≺"x"≺"x"≺"b"≺"y"≺"y"≺"y"≺"c".

表3 Shao的操作意图下的典型算法在site2的执行异同

典型算法	操作执行路径	H		操作转换路径	S	操作对象的全序关系
		H_i	H_d			
ABTS	O_2,O_1,O_4,O_3', $O_3'=del(4)$	O_2,O_1,O_4	O_3' $O_3'=del(4)$	O_2,O_1,O_4,O_3' $O_3'=del(4)$	"axzzzxxbyyyc"	"a"≺"x"≺"z"≺"z"≺"z"≺"x"≺"x"≺"b"≺"y"≺"y"≺"y"≺"c"
ABTSO	O_2,O_1,O_4,O_3' $O_3'=del(4)$	O_1,O_4,O_2' $O_2'=insert(8,yyy)$	O_3' $O_3'=del(7)$	O_1,O_4,O_2',O_3' $O_2'=insert(8,yyy)$ $O_3'=del(7)$		

4 地址空间转换算法的进展和现状

4.1 Gu的操作意图一致性

Gu等人[4,62]率先给出了地址空间转换算法,通过引入了字符节点间顺序这种语义,使操作意图一致性不再模糊,可以实现复制式结构下的CCI特性.不同于OT技术,地址空间转换不是对操作本身转换,而是对文档的状态进行转换.这种方法可以保证当所有的操作在各站点执行结束后,各站点线性结构中存储的操作字符节点的顺序是一致的,实现了操作意图一致性,收敛于一致性的结果.

基于文献[4,62-69]中的讨论,结合图3所示的场景,各站点都执行完所有操作后,按照Gu等人的操作意图,当各站点的操作字符节点顺序为"a"≺"y"≺"b"≺"z"≺"x"≺"c",即"yzxc"才符合Gu的操作意图一致性.其中,字符节点"a"和"b"在用户视图上不可见的字符.Gu等人引入字符节点间的顺序这种语义,对全序假设的完备性进行了修正(不完全依赖siteID),使得操作意图一致性不再模糊,并且符合某种语义关系.

4.2 Gu的操作意图的相关算法

Gu等人提出的AST算法的每个协同站点都维护了一个线性的数据结构Doc_s,用于存储所有操作的字符节点.执行远程操作时,将当前的文档状态回退到远程操作产生时的状态,执行远程操作后,再将文档状态恢复到当前状态,见算法3.

算法3. AST算法.

本地操作集成.

输入:站点i产生的本地操作O

输出:传播到其它站点的操作O_{request}

1. O产生后立刻执行,将其存储到文档Doc_s中O的操作节点中.
2. 标记O的时间戳后作为O_{request}向其它站点传播.

远程操作集成.

输入:本地站点i接收到远程站点j的O_{request}

输出:文档Doc_s

1. 选择具备调度条件的远程操作O_{request}.
2. Doc_s回溯到O_{request}的产生状态后执行O_{request}.
3. 将O_{request}存储到Doc_s中的O_{request}的操作节点中.
4. 站点的$SV_i[j]=SV_i[j]+1$.
5. 将Doc_s回溯到当前$SV_i[j]$的文档状态.

与OT算法相比,这类算法的优点是控制过程简单,避免了OT算法中并发操作之间复杂的操作转换过程.由于在文档中保留了所有操作的字符节点的位置,可以支持selective undo功能.此外,这类算法的执行效率通过采用红黑树可以优化为$O(\log N)$,N为字符节点的数量.

基于AST算法,文献[63]提出了支持selective undo的AST-Undo算法.该方法引入一个计数器counter来记录每个字符节点的操作意图的数量,根据counter值来决定最终字符节点的有效性.

文献[64]基于AST算法,提出了一种新的逻辑时钟,直接因果向量(Direct Causal Vector,DCV),来检测因果关系和并发关系.DCV的每个分量是由站点号和逻辑时钟构成的,大小不依赖于站点的数量,适合应用于大规模的动态协同编辑.

基于AST算法,文献[65]提出了在Web2.0环境下基于XML结构的镜像站点的同步方法.与传统的锁机制方法相比,该法可提供更短的响应时间,实现良好的负载平衡.

文献[66]对AST算法进行了改进,开发了一个原型系统Hydra,用于支持移动设备中的协同评论系统的数据一致性维护.该方法基于中心服务器设计了一个标量的时间戳来检测因果关系和并发关系,更适合应用在大规模移动设备中.

文献[67]基于AST算法,提出了支持异构协同编辑云服务的一致性维护方法.该方法实现了不同协同编辑云服务间的数据的共享、操作和同步.

文献[68]对AST算法进行扩展,提出了支持string操作的AST-String算法.该方法是将AST

中的操作对象从单个字符扩展为字符串，可以支持 string 的重叠和分裂.

文献[69]对 AST 算法进行改进，应用到协同旅游路线规划中. 该方法支持协同用户随时加入或离开，实现协同旅游路线的一致性.

4.3 Gu 的操作意图下的协同工作场景的算例对比分析

为了更清晰阐述 Gu 的操作意图下的典型算法的执行情况，结合图 3 中的协同工作场景中的算例，给出 AST 算法在 site2 的执行情况和 Gu 的操作意图一致性的应用效果. 表 4 为 AST 算法在 site2 的执行情况. 其中，字符 β 代表在用户视图上不可见的字符. AST 算法在 site2 都收敛一致性的结果“yzxc”，即 Gu 的操作意图一致性的应用效果为“yzxc”. 当 site2 所有操作执行完，AST 的 Doc_s 结构如图 6 所示.

表 4　AST 算法的各操作在 site2 的执行

	本地操作	SV_O	远程操作	SV_O	$Retracing(Doc_s, SV_O)$的文档状态	在 SV_O下执行远程操作后的文档状态	$Retracing(Doc_s, SV_s)$的文档状态	执行完操作后的 Doc_s
O_2	$O_2=insert(2,x)$	⟨0,1,0⟩						Doc_s=“abxc”
O_1			$O_1=del(1)$	⟨1,0,0⟩	“ab~~x~~c”	“a~~bx~~c”	“a~~b~~xc”	Doc_s=“a~~b~~xc”
O_5	$O_5=del(0)$	⟨1,2,0⟩						Doc_s=“~~a~~bxc”
O_3			$O_3=insert(1,y)$	⟨0,0,1⟩	“ab~~x~~c”	“ayb~~x~~c”	“ay~~b~~xc”	Doc_s=“ay~~b~~xc”
O_6			$O_6=insert(2,z)$	⟨1,1,2⟩	“ay~~b~~xc”	“ay~~b~~zxc”	“ay~~b~~zxc”	Doc_s=“ay~~b~~zxc”
O_4			$O_4=del(0)$	⟨2,1,1⟩	“ay~~bz~~xc”	“ay~~bz~~xc”	“ay~~b~~zxc”	Doc_s=“ay~~b~~zxc”

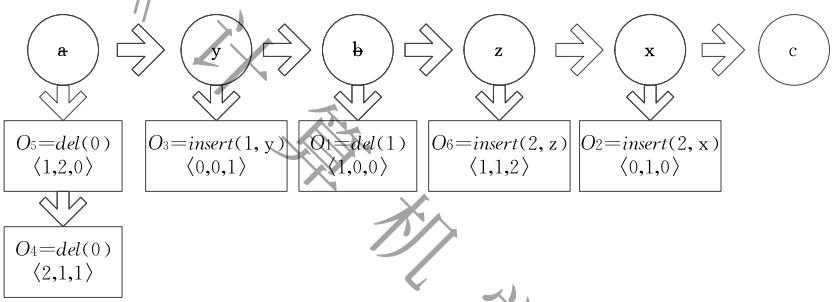

图 6　site2 操作执行后 AST 算法的 Doc_s 结构

5　可交换的复制式的数据类型的进展和现状

5.1　CRDT 的操作意图一致性

综合相关文献[30,70-81]，CRDT 算法的操作意图可描述为操作对象在数据结构中的全局位置，由分配给操作对象唯一全序的 ID 映射.

结合图 3 所示的协同编辑场景，各操作产生时，不同的 CRDT 算法结合不同的 ID 分配方法，给操作对象分配唯一的全序 ID，通过 ID 将操作对象映射到相应数据结构中，得到操作对象的全序位置为“y”≺“z”≺“x”≺“c”. CRDT 的操作意图一致性的编辑算法可以保证各操作在所有站点执行结束之后，各站点内部数据结构中保存的操作对象具有相同的全序位置. 因此，三站点收敛于“yzxc”符合 CRDT 的操作意图一致性.

CRDT 通过全局 ID 来界定内部数据结构中操作对象的全局位置，通过引入该语义来修正全序假设的完备性(不完全依赖 siteID)，使得意图一致性不再模糊，并且符合该语义关系.

5.2　CRDT 的操作意图的相关算法

支持 CRDT 的操作意图一致性的算法采用操作对象的全序方法实现(见本文 2.2.2 节)，典型代表是 WOOT、TreeDoc、RGA 和 LOGOOT. 这类方法的特点是通过给每个操作对象分配全局唯一的 ID，使得并发操作之间可以交换执行. 为了确保各站点收敛一致性的结果，这类方法基本思想是基于相应的数据结构和分配给操作对象的 ID，将操作对象全序的保存到内部的数据结构中. 这类方法的主要挑战是如何给操作对象分配全局唯一的 ID，如何减少墓碑所占的空间开销. 作者给出了 CRDT 算法的执行步骤，如算法 4 所示.

算法 4.　CRDT 算法.

本地操作集成

输入：本地产生的操作 O

输出：传播到其它站点的操作 $O_{request}$

1. O 产生后，结合不同数据结构，采用不同策略给 O 的操作对象分配唯一 ID.
2. 基于已分配的 ID，找到 O 在相应数据结构中的操作位置后执行 O.

3. 获取 O 的传播形式 $O_{request}$，将 $O_{request}$ 向各站点传播.

远程操作集成

输入：来自其它站点的 $O_{request}$

输出：$O_{request}$ 的操作位置 $P_{request}$

1. 选择具备调度条件的远程操作 $O_{request}$.
2. 基于 $O_{request}$ 的操作对象 ID，采用不同策略，在相应的数据结构中找 $P_{request}$.
3. 在 $P_{request}$ 位置执行 $O_{request}$.

Oster 等人[70]提出的 WOOT 是第一个 CRDT 算法. 每个协同站点都维护存储操作对象的线性表 S 和存储接收操作的集合 Pool. $\langle ns, ng\rangle$（siteID 和逻辑时钟）作为操作对象的唯一 ID. 基于 $O_{request}$ 的操作对象的 ID，在 S 中找到操作位置 $P_{request}$ 后执行操作. 与其它主流的 OT 算法相比，WOOT 并没有采用 Lamport 的 happened before 关系调度远程操作，而是定义了基于操作语义依赖的 Precondition 条件来维护因果的一致性. WOOTO[80]是对 WOOT 的扩展，将操作对象从单个字符扩展为行字符.

WOOT/WOOTO 的优点是使用逻辑时钟代替 SV，使得算法具有很好的伸缩性. 缺点是算法将删除对象置为墓碑，却没给出墓碑清除策略，随着墓碑数量的增加，算法的时空开销都会随之增加. 相关文献表明，WOOT/WOOTO 执行本地和远程插入操作的时间复杂度为 $O(k^2)$，其中 k 为并发插入结点的数量[80].

Yu 和 Ignat 等人[78-79]对 WOOT 算法进行了改进和优化，提出了支持 string 和 selective undo 功能的 WOOT-String，该算法采用了与 WOOT 相同的方式排序并发插入的字符串，理论上与 WOOT 有相同的时间复杂度.

Preguica 和 Marques 等人提出了 TreeDoc[73]，每个站点维护一个二叉树和操作对象的线性历史（buffer）. 基于哈夫曼编码和二叉树中序遍历，将 buffer 中的操作对象与二叉树中的结点对应. 逻辑时钟和站点号（counter，siteID）作为操作对象的 ID. 无论集成本地操作 O 还是远程操作 $O_{request}$，根据操作对象的 ID，找到其在二叉树中的操作位置 $P_{request}$ 后执行操作.

TreeDoc 的优点是采用了压缩策略（flatten）将二叉树压缩成线性表存储，同时采用了垃圾回收（garbage-collection）策略清除墓碑，因此，这算法在一定程度上减少了空间开销. TreeDoc 的缺点是若插入的对象总是在文本最后，导致构建的二叉树为非平衡二叉树，树的深度会快速增加，因而 garbage-collection 将花费更多的时间开销. 此外，相关文献指出，garbage-collection 策略并不适合应用在 p2p 的协同编辑环境[72,80].

Roh 等人[75]提出的 RGA，每个协同站点都维护哈希表（hash）和链表（list）. 用 S4vector 作为操作对象的唯一 ID. 集成本地操作需要遍历 list 找到相应的操作位置 $P_{request}$. 集成远程操作 $O_{request}$ 则需要通过 hash 和 S4vector 找到 list 中的 $P_{request}$. 若并发插入操作已在 $P_{request}$ 后插入了不同结点，需要将 $O_{request}$ 的 S4vector 与其它结点的 S4vector 比较后获取正确插入位置.

RGA 算法的优点是在主流的 CRDT 算法中，具有最好的平均计算性能. 由于使用了 hash 来唯一定位 list 中的操作对象，使远程删除操作的时间复杂度为 $O(1)$. 远程插入操作的时间复杂度为 $O(m)$，m 为并发插入操作对象的个数. RGA 算法的缺点是定义 S4vector 时使用了 SV，因而随着协同站点数量的增加，算法的伸缩性较差.

Weiss 等人[71]提出了 LOGOOT. 每个协同站点都维护操作对象的 ID 所组成的 IDTable 和多行字符组成的 Document. 分给每行字符 $\langle i, s, c\rangle$（随机数、siteID 和逻辑时钟）作为唯一 ID. 由于 IDTable 中 ID 大小与 Document 中操作对象的位置一一映射，因此在集成远程操作时，需要在 IDTable 中找到操作对象的 ID，即可快速映射到 Document 中的操作位置 $P_{request}$.

LOGOOT 的优势是在主流的 CRDT 算法中，LOGOOT 具有最好的本地操作的计算性能，尤其是本地删除操作的时间复杂度为 $O(1)$. 此外，LOGOOT 没有使用 SV，因此更适合应用在 p2p 的协同编辑环境中. LOGOOT 缺点是 ID 的长度占用很大的空间开销，尤其是当大量的并发插入操作发生在 Document 中的同一位置，导致 ID 的长度无限增加，空间开销很大. 此外，LOGOOT 没有明确给出如何实现因果一致性[77,80].

Weiss 等人[72]对 LOGOOT 算法进行了优化和扩展，提出了支持 selective undo 功能的 LOGOOT-Undo 算法，支持用户在任何时刻、任何位置撤销操作. 之后，André 等人[80]提出了支持 string 操作的 LOGOOT-String，可应用在大规模粗粒度的 p2p 协同编辑环境中.

Wu 等人[74]提出的 PPDS，适合无结构文本的协同编辑. 通过给每个操作对象分配一个有理数 S，并基于有理数大小全序操作对象位置. 虽然该算法指出可以实现结果一致性和意图保持，但并未给出严格的形式化证明.

5.3 CRDT 的操作意图下的协同工作场景的算例对比分析

为了更清晰阐述 CRDT 的操作意图下的相关算法的执行异同，结合图 3 中的协同工作场景中的算例，给出各典型算法在 site2 的执行情况和 CRDT 的操作意图一致性的应用效果. 表 5 给出 CRDT 的操作意图下各典型算法在 site2 的执行异同，其中 β 代表 β 的墓碑. 各典型算法在 site2 都收敛一致性的结果“yzxc”，即 CRDT 的操作意图一致性的应用效果为“yzxc”.

表 5　CRDT 的操作意图下的典型算法在 site2 的执行异同

典型算法	ID	操作对象的历史	S	操作对象的全序关系（β 表示 β 为墓碑）
WOOT	$ID(O_1)=(1,1)$，$ID(O_2)=(2,1)$，$ID(O_3)=(3,1)$，$ID(O_4)=(1,2)$，$ID(O_5)=(2,2)$，$ID(O_6)=(3,2)$	线性表：a y b z x c	“yzxc”	“a”≺“y”≺“b”≺“z”≺“x”≺“c”
TreeDoc	ID(a)=[0]，ID(b)=[ ]，ID(c)=[1] ID(x)=[10]，ID(y)=[01]，ID(z)=[011]	二叉树：a y z b x c		“a”≺“y”≺“z”≺“b”≺“x”≺“c”
RGA	$S4vector(O_1)=(1,1,1,1)$，$S4vector(O_2)=(1,2,1,0)$，$S4vector(O_3)=(1,3,1,0)$，$S4vector(O_4)=(1,1,4,2)$，$S4vector(O_5)=(1,2,3,2)$，$S4vector(O_6)=(1,3,4,1)$	链表：a y z b x c		“a”≺“y”≺“z”≺“b”≺“x”≺“c”

6　支持操作意图一致性的实时协同编辑算法对比与小结

为了对各类操作意图下的相关算法进行全面的对比、分析与总结，我们按照操作意图一致性、操作意图一致性的典型算法、算法中使用的核心操作转换函数和典型算法的时间复杂度分别讨论.

6.1 操作意图一致性的对比

基于上述 3、4、5 节的讨论，Sun 的操作意图一致性的应用效果有别于 Li、Shao、Gu 和 CRDT 的操作意图一致性的应用效果. 主要原因在于 Sun 的操作意图一致性更加宽泛，是通过某种全序自动选择多种一致性结果中的任意一个一致性结果即可. 其它类别的操作意图一致性通过引入了某种操作语义，使得操作意图一致性不再模糊. 例如，Li 的操作意图一致性引入的操作语义是操作效果关系；Shao 的操作意图一致性引入的操作语义为操作效果序；Gu 的操作意图一致性引入的操作语义为字符节点的顺序；CDRT 的操作意图一致性引入的操作语义为操作对象在数据结构中的全局位置. Li、Shao、Gu 和 CRDT 的操作意图一致性的维护都可以通过维护操作对象的全序位置关系来实现. 其中，Li 和 Shao 的操作意图一致性采用了操作转换技术来维护了操作对象的全序位置关系，Gu 的操作意图一致性采用了地址空间转换技术来维护字符节点的一致顺序，CRDT 的操作意图一致性通过给操作对象分配全局 ID，结合内部的数据结构，将操作对象全序的映射到内部的数据结构中来维护操作对象的全序位置.

操作意图一致性对比如图 7 所示. 各类操作意图一致性的前提都是确保结果一致性，在此基础上，Sun 的操作意图一致性选择其中的一种全序结果. 而 Li、Shao、Gu 和 CRDT 的操作意图一致性都是选择唯一的操作对象的全序位置. 相比较下，通过引入

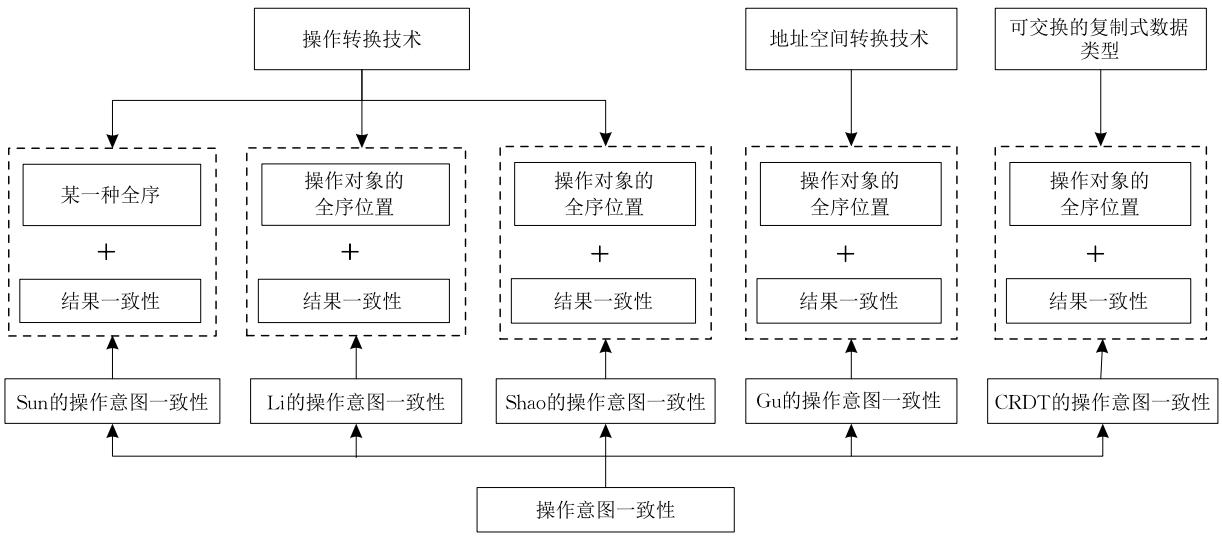

图 7　操作意图一致性对比

某种语义关系，Li、Shao、Gu 和 CRDT 的操作意图一致性是更严格、更形式化和更量化的操作意图一致性.

6.2 各类操作意图一致性的典型算法对比

表 6 列出了支持 Sun 的操作意图一致性的典型算法对比. 为了实现因果一致性，GOT 和 COT 使用了 SV 和 CV，当参与协同的站点数量很大时，维护 SV 和 CV 需要花费很大的时空开销. TIBOT/TIBOT2.0 都使用了 TI 来实现因果一致性，但并未明确给出 TI 长度如何设置以及各站点的 TI 如何同步. SOCT3/SOCT4 采用集中式的时间戳 TS(Time Stamps)来实现因果一致性，但是 TS 的设置需要 Sequencer 进行操作的序列化，因而获取 TS 需要花费很大的时间开销.

为了实现结果一致性，GOT 和 TIBOT 构建全序的操作转换路径和操作执行路径，并严格按照操作的全序调度和执行. TIBOT2.0、COT 和 SOCT3/4 构建了全序的操作转换路径，但操作的执行路径不是按照全序构建的.

表 6 中的算法都采用线性数据结构来存储操作历史. 其中，为了更好支持 selective undo 功能，COT 的操作历史中存储的是已经执行的原始操作，而其它典型算法的操作历史存储的是经过操作转换后可执行的操作.

表 6 支持 Sun 的操作意图一致性的典型算法对比

对比方法		GOT	TIBOT	TIBOT2.0	COT	SOCT3	SOCT4
因果一致性		满足/SV	满足/TI	满足/TI	满足/CV	满足/TS	满足/TS
结果一致性		满足 按全序转换并执行	满足 按全序转换并执行	满足 按全序转换	满足 按全序转换	满足 按全序转换	满足 按全序转换
操作转换路径		全序	全序	全序	全序	全序	全序
线性数据结构	存储的操作	已执行操作	已执行操作	已执行操作	已执行的原始操作	已执行操作	已执行操作
	操作的排序	全序＝执行顺序	全序＝执行顺序	全序≠执行顺序	非全序＝执行顺序	全序≠执行顺序	全序≠执行顺序
操作转换函数		*IT*、*ET*、*LIT*、*LET*	*IT*	*IT*	*IT*	*FT*、*BT*	*FT*
操作转换函数是否需要满足 TP1、TP2		不需要满足 TP1 和 TP2	不需要满足 TP1 和 TP2	需满足 TP1	需满足 TP1	需满足 TP1	需满足 TP1
全序类型/优先级		分布式取全序/siteID	分布式取全序/siteID	分布式取全序/siteID	分布式取全序/siteID 集中式取全序/N	集中式取全序/N	集中式取全序/N
是否支持 selective undo		不支持	不支持	不支持	支持	不支持	不支持
是否支持 string 操作		支持	不支持	不支持	不支持	不支持	不支持
是否适合 p2p 协同编辑		不适合	不适合	不适合	不适合	不适合	不适合

表 6 中的算法都使用了操作转换函数，这些函数的具体细节将在 6.3 节讨论. 其中，GOT 和 TIBOT 的操作转换函数不需要满足转换属性 TP1 和 TP2，因为算法严格按照操作的全序调度和执行. TIBOT2.0、COT 和 SOCT3/4 的操作转换函数需要满足转换属性 TP1，而不需要满足 TP2，因为这些算法都按照唯一全序的操作转换路径进行操作转换，避免了 TP2 的约束.

表 6 中的算法采用了不同策略定义操作全序. 其中，GOT 和 TIBOT/TIBOT2.0 采用了分布式取全序方法，而 SOCT3/4 采用集中式全序方法，COT 既可以采用分布式取全序也可以采用集中式取全序.

在表 6 中的算法中，只有 COT 支持 selective undo 功能，只有 GOT 支持 string 的操作. 由于表 6 中的算法使用 SV 或等同于 SV 的时间戳，较适合应用于站点数固定且小规模的协同编辑系统中，不适合应用于大规模的 p2p 协同编辑领域.

表 7 中列出了支持 Li 的操作意图一致性的典型算法对比. 为了实现因果一致性，表 7 中的所有算法都遵循 Lamport 的 happened before 关系，采用了 SV 检测因果关系和并发关系.

为了实现结果的一致性，SDT 和 SDTO 按照操作的执行顺序构建操作的转换路径. 遗憾的是，SDT/SDTO 被文献[54]发现了特定场景下的“puzzle”，因而无法实现结果一致性. LBT 为了实现结果的一致性，选择构建了双向的操作转换路径. ABT 构建了插入操作序列 H_i 在前，删除操作序列 H_d 在后的操作转换路径.

表 7 的典型方法都采用了线性的数据结构来存储操作历史. SDT/SDTO 和 LBT 的操作历史按照操作的执行顺序构建. 不同于 SDT/SDTO 和 LBT，ABT 的操作历史都是按照插入操作序列 H_i 在前，删除操作序列 H_d 在后构建，并且 H_i 和 H_d 中的操作是按照操作的执行顺序排列.

表 7 中典型算法都使用了操作转换函数，这些函数的具体细节将在 6.3 节中讨论. 其中，SDT/SDTO 的操作转换函数需要满足 TP1 和 TP2，虽然 SDT/SDTO 声称设计的操作转换函数满足 TP1 和 TP2.

表 7　支持 Li 的操作意图一致性的典型算法对比

方法对比		SDT/SDTO	LBT	ABT
因果一致性		满足/SV	满足/SV	满足/SV
结果一致性		不满足	满足 双向的操作转换路径	满足 操作对象的全序位置
操作转换路径		非全序/无方向	非全序/双向	非全序/$H_i \cdot H_d$
线性数据结构	存储的操作	已执行操作	已执行操作	已执行操作
	操作的排序	执行顺序	执行顺序	$H=H_i \cdot H_d$，H_i和 H_d按执行顺序排序
操作转换函数		*IT*、*ET*、*SQIT*、*ETSQ*	*IT*、*ET*	*IT*、*ET*、*Swap*
操作转换函数是否需要满足 TP1、TP2		需满足 TP1 和 TP2	删除操作：需要满足 TP1 和 TP2 插入操作：需要满足 TP1	需满足 TP1
全序定义使用的优先级		操作效果关系＋siteID	操作效果关系＋siteID	操作效果关系＋siteID
是否支持 selective undo 操作		不支持	不支持	不支持
是否支持 string 操作		不支持	不支持	不支持
是否适合 p2p 协同编辑		不适合	不适合	不适合

但是，后续研究发现了不满足 TP1 和 TP2 的“puzzle”. 在 LBT 算法中，如果为删除操作，操作转换函数需要满足 TP1 和 TP2；如果为插入操作，操作转换函数需要满足 TP1，原因是构建了特殊的操作转换路径避免了 TP2 的约束. ABT 的操作转换函数只需要满足 TP1，因为构建的操作转换路径中 H_i 在前，H_d 在后，从而避免了 TP2 的约束.

在全序定义中，表 7 列出的典型算法通过引入操作效果关系＋siteID 对全序进行修正. 表 7 列出的典型算法中，没有算法支持 selective undo 功能和 string 操作. 由于表 7 中的所有算法都使用了 SV 维护因果一致性，不具有良好的伸缩性，所以不适合应用于大规模的 p2p 协同编辑.

表 8 中列出了支持 Shao 的操作意图一致性的典型算法对比. 表 8 中列出的所有算法都遵循 Lamport 的 happened before 关系，采用了 SV 检测因果关系和并发关系，并维护因果一致性.

为了实现结果一致性，ABTS、ABTSO、ABST 和 ABTU 的操作转换路径按照插入操作序列 H_i 在前，删除操作序列 H_d 在后来维护操作对象全序的位置关系.

在算法采用的数据结构中，ABTS、ABTSO、ABST 和 ABTU 都采用了线性的数据结构来存储操作历史. 其中，ABTS 中的 H_i 和 H_d 中的操作是按照操作的执行顺序排列，而 ABTSO、ABST 和 ABTU 中的 H_i 和 H_d 中的操作按照操作效果序排序. 表 8 中列出的算法的操作历史中存储的都是经过操作转换后的可执行操作.

表 8　支持 Shao 的操作意图一致性的典型算法对比

方法对比		ABTS	ABTSO	ABST	ABTU
因果一致性		满足/SV	满足/SV	满足/SV	满足/SV
结果一致性		满足 操作对象的全序位置	满足 操作对象的全序位置	满足 操作对象的全序位置	满足 操作对象的全序位置
操作转换路径		非全序/$H_i \cdot H_d$	非全序/$H_i \cdot H_d$	非全序/$H_i \cdot H_d$	非全序/$H_i \cdot H_d$
线性数据结构	存储的操作	已执行操作	已执行操作	已执行操作	已执行操作
	操作的排序	$H=H_i \cdot H_d$ H_i和 H_d按执行顺序排序	$H=H_i \cdot H_d$ H_i和 H_d按 *ERO* 排序	$H=H_i \cdot H_d$ H_i和 H_d按 *ERO* 排序	$H=H_i \cdot H_d$ H_i和 H_d按 *ERO* 排序
操作转换函数		*ITLL*、*ITOSq*、*SwapDsq*、*SwapLL*	*ITLL*、*SwapDsq*、*SwapLL*	*ITSQ*、*SwapSQ*	*IT*、*ET*、*Swap*
操作转换函数是否需要满足 TP1、TP2		需满足 TP1	需满足 TP1	需满足 TP1	需满足 TP1
全序定义使用的优先级		操作效果序＋siteID	操作效果序＋siteID	操作效果序＋siteID	操作效果序＋siteID
是否支持 selective undo		不支持	不支持	不支持	支持
是否支持 string 操作		支持	支持	不支持	不支持
是否适合 p2p 协同编辑		不适合	不适合	不适合	不适合

表 8 中列出的算法都使用了操作转换函数，这些函数的具体细节将在 6.3 节中讨论. 其中，ABTS、ABTSO、ABST 和 ABTU 的操作转换函数只需要满足 TP1，因为构建的操作转换路径中 H_i 在前，H_d 在后，从而避免了 TP2 的约束.

在全序定义中，表 8 列出的典型算法通过引入操作效果序＋siteID 对全序进行修正. 在表 8 列出的典型算法中，只有 ABTU 算法支持 selective undo

功能.只有 ABTS 和 ABTSO 算法支持 string 操作.ABTSO 比 ABTS 高效,处理远程操作时,将 ABTS 的时间复杂度从平方级优化为线性时间.但是,ABTSO 算法在支持 string 的删除操作时,将基于 string 的删除操作转化为基于字符的删除操作序列后逐一处理.表 8 列出的典型算法中,没有同时支持 selective undo 功能和 string 操作的算法.

表 8 中的所有算法都使用了 SV 维护因果一致性,不具有良好的伸缩性,所以不适合应用于大规模的 p2p 协同编辑.

表 9 列出了支持 Gu 的操作意图一致性的典型算法对比.为了实现因果一致性,AST、AST-Undo 和 AST-String 都采用了 SV 来检测因果和并发关系.而 Hydra 基于中心服务器设计了一个标量的时间戳来检测因果关系和并发关系.

为了实现结果一致性,表 9 中的算法都采用了地址空间转换技术来维护各个站点的操作字符节点间的一致顺序.由于地址空间转换是对文档的状态进行转换,而不是对操作本身进行转换,因而不需要设计操作转换函数.

表 9 中的 AST、AST-Undo 和 AST-String 都是采用了全复制式的线性结构来存储操作对象,而 Hydra 采用了部分复制的树形结构来存储操作对象.

在全序定义中,表 9 列出的典型算法通过引入操作字符节点顺序+siteID 对全序进行修正,使得全序的定义不完全依赖 siteID.表 9 列出的典型算法中,只有 AST 和 AST-Undo 算法支持 selective undo 功能.只有 AST-String 算法支持 string 操作,可以处理字符串的重叠和分裂,没有算法同时支持 selective undo 功能和 string 操作.

表 9 中的 AST、AST-Undo 和 AST-String 都使用了 SV 维护因果一致性,不具有良好的伸缩性,所以不适合应用于大规模的 p2p 协同编辑.而 Hydra 由于采用了标量的时间戳检测因果关系和并发关系,所以适合应用于大规模的移动设备及 p2p 协同编辑环境中.

表 9 支持 Gu 的操作意图一致性的典型算法对比

方法对比	AST	AST-Undo	AST-String	Hydra
因果一致性	满足/SV	满足/SV	满足/SV	满足/scalar time
结果一致性	满足 地址空间转换 实现字符节点全序	满足 地址空间转换 实现字符节点全序	满足 地址空间转换 实现字符节点全序	满足 地址空间转换 实现字符节点全序
是否需要操作转换	不需要	不需要	不需要	不需要
数据结构	全复制式线性结构 Docs	全复制式线性结构 Docs	全复制式线性结构 Docs	部分复制式的树形结构
全序定义使用的优先级	字符节点顺序+siteID	字符节点顺序+siteID	字符节点顺序+siteID	字符节点顺序+siteID
是否支持 selective undo	支持	支持	不支持	不支持
是否支持 string 操作	不支持	不支持	支持	不支持
是否适合 p2p 协同编辑	不适合	不适合	不适合	适合

表 10 为支持 CRDT 的操作意图一致性的典型算法对比.首先,这些方法采用不同的策略实现因果一致性.TreeDoc 和 LOGOOT、LOGOOT-Undo 以及 LOGOOT-String 都是假设采用已有的机制来实现因果一致性.例如,TreeDoc 假设可以采用 SV 或 VV(Version Vector)来实现因果一致性,而 LOGOOT、LOGOOT-Undo 和 LOGOOT-String 假设可以采用因果传播机制(Causal Dissemination Mechanism)来实现因果一致性.WOOT、WOOTO 和 WOOT-String 定义了 Precondition 条件来实现因果一致性.而 Precondition 条件有别于 Lamport 的 happened before 关系,前者主要考虑操作语义上的因果依赖关系,而后者主要考虑操作时间上的先后关系.RGA 定义了 Precedence Relation 关系来实现因果一致性.该关系不仅兼容了 Lamport 的 happened before 关系,而且还定义了并发操作之间操作执行的优先权.PPDS 算法的有理数分配机制已经考虑了操作之间的因果关系.

为了实现结果的一致性,表 10 中的算法都是基于 ID 将操作对象全序的保存到相应的数据结构中,从而维护了操作对象全序的位置关系.

表 10 中的算法都采用了非线性的数据结构存储操作对象和操作对象的 ID,因而花费的空间开销较大.与表 6 到表 8 列出的典型算法相比,表 10 中的算法不需要构建操作转换路径,并发操作间不需要操作转换,不需要设计操作转换函数.

表 10 中的算法,需要将删除的操作对象保存为墓碑的有 TreeDoc、WOOT/WOOTO、RGA、PPDS 和 WOOT-String.其中,TreeDoc 和 RGA 给出了墓碑清除策略.WOOT/WOOTO 没有墓碑清除策略,

表 10　支持 CRDT 的操作意图一致性的典型算法对比

方法对比	TreeDoc	WOOT WOOTO	RGA	LOGOOT	LOGOOT-Undo	LOGOOT-String	PPDS	WOOT-String
因果一致性	满足/SV 或 VV	Precondition	Precedence Relation	满足/causal dissemination mechanism	满足/causal dissemination mechanism	满足/causal dissemination mechanism	不满足	Precondition
结果一致性	满足 操作对象的 全序	满足 操作对象的 全序	满足 操作对象的 全序	满足 操作对象的 全序	满足 操作对象的 全序	满足 操作对象的 全序	满足 操作对象的 全序	满足 操作对象的 全序
是否需要 操作转换	不需要	不需要	不需要	不需要	不需要	不需要	不需要	不需要
数据结构	Binary Tree Buffer Log	S Pool	hash list Patch	IDTable Documnet Patch	IDTable Documnet Patch	LOGOOTSPLITNAIVE LOGOOTSPLITSTRING LOGOOSPLITTAVL	PPS LEQ、REQ Buffer	View Model log
是否存储 墓碑	是	是	是	否	否	否	是	是
ID	huffman code+ 〈counter,siteID〉	〈ns,ng〉	S4Vector	〈i,s,c〉	〈i,s,c〉	〈base:offset〉	S:rational numbers	〈pid,pun, offset〉
全序定义 使用的 优先级	操作对象的 全局 位置+siteID	操作对象的 全局 位置+siteID	操作对象的 全局 位置+siteID	操作对象的 全局 位置+siteID	操作对象的 全局 位置+siteID	操作对象的 全局 位置+siteID	操作对象的 全局 位置+siteID	操作对象的 全局 位置+siteID
是否支持 selective undo	不支持	不支持	不支持	不支持	支持	不支持	不支持	支持
是否支持 string 操作	不支持	不支持	不支持	不支持	不支持	支持	不支持	支持
是否适合p2p 协同编辑	不适合	适合	不适合	适合	适合	适合	适合	适合

随着删除操作数量的增加，墓碑占用的空间开销大. LOGOOT、LOGOOT-Undo 和 LOGOOT-String 没有使用墓碑，而是直接将删除对象从相应的数据结构中清除. 原因是这些算法分配给操作对象的 ID 的大小和操作对象在相应数据结构中的排列顺序一一映射，即使不保留墓碑，也能确保操作对象间正确的全序位置关系.

表 10 中的 CRDT 算法采用不同的策略分配 ID. 其中，TreeDoc、WOOT、WOOTO、LOGOOT 和 LOGOOT-Undo 在分配 ID 时使用了站点号和逻时钟取代了 SV，因而适合应用于 p2p 的协同编辑环境中. 而 RGA 定义的 S4Vector 使用了 SV，因此不适合应用于大规模的 p2p 的协同编辑环境中. 此外，LOGOOT-String 和 WOOT-String 在定义 ID 时都使用了偏移量(offset)，目的是为了唯一标识每个被分裂的 string. PPDS 通过设计唯一的有理数来标识操作对象的位置，因此 PPDS 也适合应用在大规模的 p2p 的协同编辑环境中.

表 10 中列出的算法中，只有 LOGOOT-Undo 算法和 WOOT-String 算法支持 selective undo 功能. 其中，LOGOOT-Undo 仅支持基于 line 的 selective undo 功能，不支持 line 的分裂和重叠. WOOT-String 的操作对象为 string，同时支持 string 的 selective undo 功能和 string 的重叠和分裂.

在 p2p 协同编辑方面，除了 RGA 和 TreeDoc 算法，其它算法都可应用于 p2p 的协同编辑领域.

6.3　典型算法的核心操作转换函数对比

表 6 到表 8 中使用的核心操作转换函数对比如图 8. 其中，操作转换函数分为 3 类：(1) 元操作之间的转换函数，如包含转换 IT、排斥转换 ET、前向转换 FT、后向转换 BT 和交换转换 $Swap$；(2) 元操作与操作序列之间的转换函数，包括包含转换 LIT、$SQIT$ 和 $ITOSq$，排斥转换 LET 和交换转换 $SwapDsq$；(3) 操作序列之间的转换函数，包括包含转换 $ITLL$ 和 $ITSQ$，交换转换 $SwapLL$ 和 $SwapSQ$.

IT 函数定义了两个操作之间的包含转换规则，经过操作转换后，一个操作包含另一个操作的执行效果[33]. IT 函数执行的一个重要前提条件是两个并发操作处于相同的文档状态，即两个并发操作为简单并发关系. FT 函数的执行效果等同于 IT. 为了处理偏并发操作，IT 的逆过程排斥转换函数 ET 被提出[33]. ET 定义了两个操作之间的排斥转换规则. 与 IT 相反，ET 是在一个操作中排除掉另一个操作的执行效果，目的是为了将不同文档状态下的操作转换为相同的文档状态. BT 和 $Swap$ 的功能相同，定义了两个操作之间的交换转换的规则. 为了使两个操作交换后能正确执行，BT 和 $Swap$ 封装了 ET 和 IT 函数，先调用 $ET(O_2,O_1)=O_2'$，然后调用 $IT(O_1,O_2')=O_1'$，得到了等价的执行序列 $[O_1,O_2]=[O_2',O_1']$.

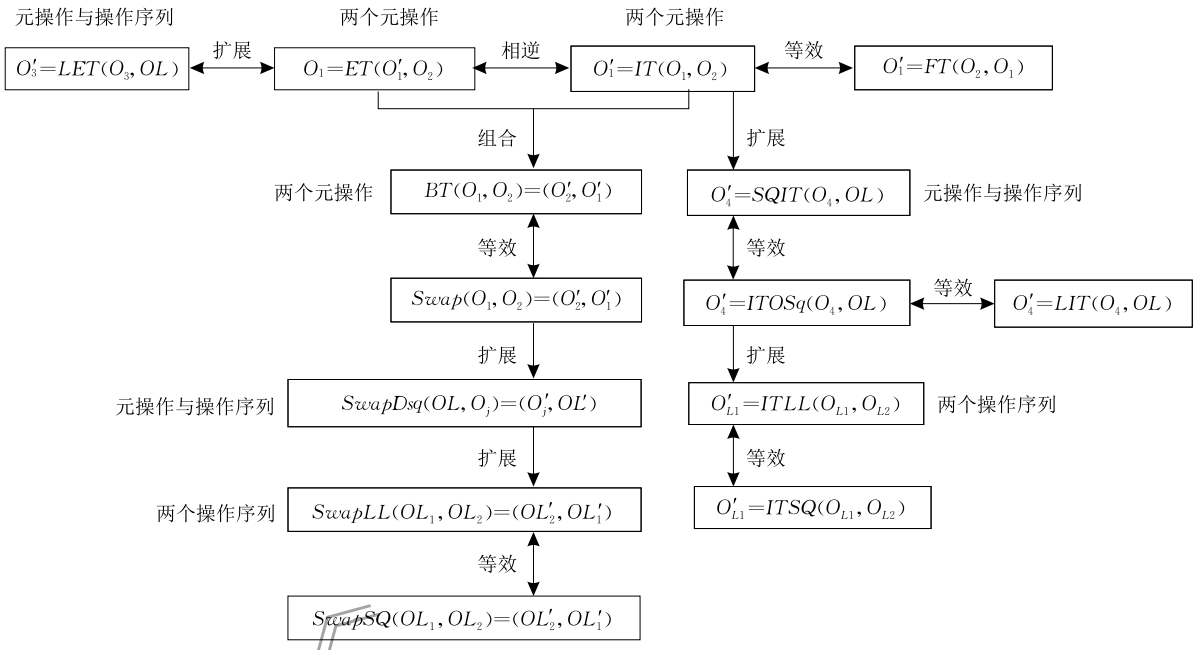

图 8 核心操作转换函数对比

基于 IT 函数，可将其扩展为元操作和操作序列之间的包含转换函数 LIT、$SQIT$ 和 $ITOSq$，这些函数的功能相同，经过操作转换后，一个操作包含了一个操作序列中所有操作的执行效果. LIT、$SQIT$ 和 $ITOSq$ 函数执行的一个重要前提条件是 O 与 OL 中第 1 个操作在相同的文档状态. 基于 ET 函数，可将其扩展为元操作与操作序列之间的排斥转换函数 LET. 经过 LET 后，元操作排除了操作序列中所有操作的执行效果. 基于 $Swap$ 函数，可以扩展为元操作与操作序列之间的交换转换函数 $SwapDsq$.

基于元操作和操作序列之间的包含转换函数 LIT，可以进一步扩展为两个操作序列之间的包含转换函数 $ITLL$ 和 $ITSQ$，这两个函数的功能相同，经过包含转换后，一个序列包含了另一个序列中所有操作的执行效果. 基于元操作与操作序列之间的交换转换函数 $SwapDsq$，可以进一步扩展为两个操作序列之间的交换转换函数 $SwapLL$ 和 $SwapSQ$. 这两个函数功能相同，交换两个操作序列的执行顺序.

6.4 典型算法的时间复杂对比

表 11 给出了各类操作意图一致性的典型算法的时间复杂度的对比. 其中，各符号代表的含义为 $|H|$ 代表操作历史的大小；$|H_i|$ 代表插入操作序列的大小；$|H_d|$ 代表删除操作序列的大小；$|T|$ 代表本地未发送的操作序列的大小；N 为操作对象的数量；d 为左右目标对象间的操作对象的数量，用于 WOOT/WOOTO 算法；m 代表并发插入操作对象的数量，用于 RGA；k 代表分配给操作对象的 ID 的大小，用于 TreeDoc、LOGOOT 和 LOGOOT-Undo；n 代表文档的大小，用于 LOGOOT.

Sun 的操作意图一致性的典型算法具有很好的本地响应性，因为本地操作产生后立即执行，本地操作的时间复杂度为 $O(1)$. 远程操作的执行需要花费较大的时间开销，尤其是 COT 和 GOT 的远程操作的时间复杂度为指数级. SOCT3 执行远程操作的时间复杂度为 $O(|H|^2)$，在处理远程操作时，使用 SV 在 H 中找并发的操作序列，并用后向转换函数 BT 将 H 重排为因操作序列在前，并发操作序列在后，所以时间开销为 $O(|H|^2)$. SOCT4 的远程操作的时间复杂度为 $O(|H|)$，由于接收到的远程操作已经包含了全序在其前的并发操作的执行效果，所以不需要在 H 中找并发的操作序列，仅需要与全序在其后的操作序列进行操作转换. TIBOT/TIBOT2.0 的远程操作的时间复杂度为 $O(|H|)$，在处理远程操作时，采用了 TI 在 H 中找并发的操作序列.

在 Li 的操作意图一致性的典型算法中，SDT、SDTO 和 LBT 的本地操作的时间复杂度为 $O(1)$. SDT 执行远程操作的时间复杂度为 $O(|H|^3)$，原因是在处理远程操作时，需要在 H 中找并发的操作序列，时间开销为 $O(|H|^2)$. 此外，还需要计算并发操作序列中的每个操作与远程操作的操作效果序，时间开销 $O(|H|)$，因此，总的时间开销为 $O(|H|^3)$. SDTO 是 SDT 的优化版本，执行远程操作的时间复杂度为 $O(|H|^2)$，只有当出现了位置参数相同时，才计算操作效果序，因此时间开销为 $O(|H|^2)$.

LBT 的远程操作的时间复杂度为 $O(|H|^2)$，在处理远程操作时，在 H 中找并发的操作序列需要花费 $O(|H|^2)$. 在 ABT 算法中，ABT 处理本地操作 O 的时间复杂度为 $O(|H_d|)$，包括本地操作的执行时间和生成 O 的传播形式 O_{request} 的时间. ABT 在处理远程操作的时间复杂度为 $O(|H_i|^2)$，由于接收到的远程操作已经排除了 H_d 中删除操作的效果，所以只需要在 H_i 中找并发的操作序列，因而时间开销为 $O(|H_i|^2)$.

在 Shao 的操作意图一致性的典型算中，ABTS 算法的时间复杂度同 ABT，处理本地操作 O 的时间复杂度为 $O(|H_d|)$，ABTS 在处理远程操作的时间复杂度为 $O(|H_i|^2)$. ABTSO 和 ABTU 的本地和远程操作的时间复杂度都为 $O(|H|)$，由于 H 中的 H_i 的 H_d 都按照 ERO 排序，因此找并发操作的时间开销为 $O(|H|)$. ABST 处理的对象为操作序列，因而本地操作和远程操作的时间复杂度都为 $O(|H|+|T|)$，其中 H 中的 H_i 的 H_d 以及 T 都是按照 ERO 排序的.

在 Gu 的操作意图一致性的典型算法中，AST 算法的本地操作的时间复杂度为 $O(1)$. AST 处理远程操作时主要包括回溯过程和操作执行过程. 为了优化回溯和操作执行的效率，AST 将存储字符节点的线性结构调整为存储字符节点的红黑树结构，结合红黑树的性质和优化策略，使得处理的远程插入和删除操作的平摊代价为 $O(\log N)$.

在 CRDT 操作意图一致性的典型算法中，WOOT 本地插入操作和远程插入操作的时间复杂度都为 $O(N\times d^2)$，首先基于 Precondition 条件，从 N 个操作对象中找出左右目标对象，然后以左右目标对象为边界，递归的在 d 个操作对象中找插入的位置，因而总的时间开销为 $O(N\times d^2)$. WOOT 本地删除操作和远程删除操作的时间复杂度都为 $O(N)$，在 N 个操作对象中找出目标对象将其置为墓碑，因而时间开销为 $O(N)$. WOOTO 是 WOOT 的优化版本，采用 degree 代替 WOOT 的 Precondition 条件在 N 个操作对象中找左右目标对象，因而在处理远程插入操作的时间复杂度为 $O(N+d^2)$. RGA 本地操作的执行时间为 $O(N)$，无论是本地插入操作还是本地删除操作，都需要遍历链表中的 N 个操作对象，找到目标对象，因而时间开销为 $O(N)$. RGA 的远程删除操作的时间复杂度为 $O(1)$，通过 hash 和操作对象的 ID 可以在 $O(1)$ 的时间内定位到 list 中的目标对象. RGA 的远程插入操作的时间复杂度为 $O(m)$，首先在 $O(1)$ 的时间内找到链表中的目标对象，然后与目标对象后存在的 m 个并发插入的操作对象比较各自的 S4vector 后找到插入位置，因而时间开销为 $O(m)$. TreeDoc 的本地和远程操作的时间复杂度都为 $O(k)$，根据 k 的编码在二叉树中找目标对象. LOGOOT 和 LOGOOT-Undo 的操作对象为行字符，执行本地插入操作的时间复杂度为 $O(k)$，根据 k 的值，在文档中找插入的位置后插入操作对象. LOGOOT 和 LOGOOT-Undo 的本地删除操作的时间复杂度为 $O(1)$，直接在文档中找到目标行后删除. LOGOOT 和 LOGOOT-Undo 的远程操作的时间复杂度为 $O(k\times\log(n))$，基于 k 的值在 IDTable 中找到操作位置，基于该操作位置，在文档中采用二分搜索找到文档中的操作位置后执行操作.

从表 11 可知，AST 算法和 CRDT 算法的时间复杂度主要依赖于操作对象的数量 N，与操作历史 H 的大小无关，即使 H 的规模增大，只要操作对象的数量 N 保持稳定，执行操作的时间也不会随着 H 的规模增大而增加. 与地址空间转换算法和 CRDT 算法相比，OT 算法的时间复杂度主要依赖于操作历史 H 的大小，随着 H 的规模增大，执行操作的时间也会随之增加.

表 11　各类操作意图一致性的典型算法时间复杂度对比

典型算法		本地操作	远程操作
Sun 的操作意图一致性的典型算法	COT	$O(1)$	$O(a^{\|H\|})$，a 为常量
	GOT	$O(1)$	$>O(a^{\|H\|})$
	SOCT3	$O(1)$	$O(\|H\|^2)$
	SOCT4	$O(1)$	$O(\|H\|)$
	TIBOT	$O(1)$	$O(\|H\|)$
	TIBOT2.0	$O(1)$	$O(\|H\|)$
Li 的操作意图一致性的典型算法	SDT	$O(1)$	$O(\|H\|^3)$
	SDTO	$O(1)$	$O(\|H\|^2)$
	LBT	$O(1)$	$O(\|H\|^2)$
	ABT	$O(\|H_d\|)$	$O(\|H_i\|^2)$
Shao 的操作意图一致性的典型算法	ABTS	$O(\|H_d\|)$	$O(\|H_i\|^2)$
	ABTSO	$O(\|H\|)$	$O(\|H\|)$
	ABTU	$O(\|H\|)$	$O(\|H\|)$
	ABST	$O(\|H\|+\|T\|)$	$O(\|H\|+\|T\|)$
Gu 的操作意图一致性的典型算法	AST	$O(1)$	$O(\log N)$
CRDT 的操作意图一致性的典型算法	WOOT	Insert：$O(N\times d^2)$	Insert：$O(N\times d^2)$
		Delete：$O(N)$	Delete：$O(N)$
	WOOTO	Insert：$O(N\times d^2)$	Insert：$O(N+d^2)$
		Insert：$O(N)$	Delete：$O(N)$
	RGA	$O(N)$	Insert：$O(m)$
			Delete：$O(1)$
	TreeDoc	$O(k)$	$O(k)$
	LOGOOT LOGOOT-Undo	Insert：$O(k)$	$O(k\times\log(n))$
		Delete：$O(1)$	

7 小结与展望

7.1 小结

文中以 Lamport 事件间的偏序关系为起点，对实时协同编辑系统的因果关系、并发关系、简单并发关系和偏并发关系概念进行整理. 特别地，对全序概念和优先级进行深入剖析. 在此基础上，按照一致性模型中的因果一致性、结果一致性和操作意图一致性的分类，给出了操作意图一致性的维护路线图. 以该路线图为指南，综述了各类操作意图下的典型算法，阐述了各种算法的执行过程、优势和局限性，并结合算法实例进行了对比和分析. 最后，本文从 4 个层面对各类操作意图一致性的协同编辑算法进行全面的对比与总结.

7.2 展望

对将来可能研究方向的展望如下：

(1) 建立更加灵活的优先级策略来维护用户的操作意图

目前，绝大部分实时协同文本编辑算法采用固定的优先级策略全序操作或者操作对象来维护用户的操作意图，致使算法缺乏灵活性. 例如，文中表 6 到表 10 中的绝大部分算法在全序的定义时，当前面若干判定条件相等时，都用 siteID 大小作为优先级给操作或者操作对象定序. 实际上，操作意图的维护与应用是高度相关的，特定的应用有特定的操作意图表现形式，可以结合不同的应用背景中的操作语义来定义优先级，维护用户的操作意图. 例如，文献[84]通过将本体论(Ontology)引入到协同文本编辑领域，构建了一个可以维护单词语义一致性的协同编辑算法，有效的维护了用户的操作意图. 文献[18]采用了多版本技术来解决协同图形编辑中用户操作意图的冲突问题. 文献[21]结合了图形编辑中操作的静态和动态语义，解决了在协同图形编辑中用户操作语义的冲突问题. 文献[26]使用 CAD 模型体积作为优先级依据来维护用户的操作意图.

(2) 探索伸缩性更强的适合 p2p 的 OT 算法

相关研究表明，p2p 协同编辑系统最主要的特点是具有很强的伸缩性，支持协同站点动态的加入或离开，即使参与协同站点数量很大时，也能保证良好的性能[85]. 绝大部分 OT 算法都使用固定大小的 SV 来检测操作之间的因果关系和并发关系. SV 的大小由参与协同站点的数量决定，当参与协同的站点数量很大时，维护 SV 需耗费很大的时空开销. 因此，如何对现有的 OT 算法进行改进和优化，使之更适合 p2p 协同编辑是该领域一个重要的研究内容. 目前，只有少数 OT 算法适合应用到 p2p 的协同编辑领域. 例如，文献[64]提出了一种新的因果检测方法，适合大规模动态的 p2p 协同编辑. 文献[86]基于多版本技术，提出了一种支持 p2p 的异步协同的 OT 算法. 文献[53]基于文本的操作语义，提出了一种新的因果依赖关系，可以支持协同用户动态的加入或离开协同会话，适合应用于 p2p 的协同编辑环境.

(3) 探索更多更高效地支持 selective undo 的实时协同编辑算法

在实时协同编辑系统中，selective undo 是一个重要的功能，可撤销任何用户、在任何位置和在任何时刻所执行的操作[58]. 目前，在协同文本编辑中，只有少数算法[4,36-37,51,58,62-63,72,78-79,83,87] 支持 selective undo 功能. 其中，文献[87]是第 1 个支持 undo 的协同编辑算法，可以支持包括文本、图形以及多媒体等多种类型的 undo 操作. 但是该算法集成一个 undo 操作需要平方级的时间复杂度. 文献[83]仅仅支持 undo 本地操作，并且已经被后续研究发现了特殊协同工作场景中的“puzzle”问题[40]. 文献[36-37,58]支持 undo 历史中的任何操作，但是执行一个 undo 操作需要指数级的时间复杂度. 文献[51]通过将操作历史按照操作效果序排列后使得集成一个 undo 操作的时间复杂度优化为线性时间. 文献[4,62-63]通过采用红黑树的数据结构并结合相应的优化策略，可以将 undo 操作的时间复杂度优化为对数时间. 虽然上述协同编辑算法都支持 undo 操作，但是 undo 操作的计算效率可以进一步优化. 此外，上述讨论的大部分算法都只支持原子字符的 selective undo 功能，支持 string 的 selective undo 功能的算法很少[78-79]. 实际上，在实时协同文本编辑系统中，支持 string 的 selective undo 操作非常常见，例如撤销一段文本，撤销文本的粘贴等等. 因此，如何构建高效率的支持 selective undo 功能，尤其是支持 string 的 selective undo 功能的算法，对实时协同编辑领域来说意义重大.

(4) 探索支持更多具有直接执行复杂操作语义类型的实时协同编辑算法，而不是用基本的插入和删除操作来间接地合成复杂操作

尽管实时协同编辑算法在不同的应用领域支持的操作类型不同，但目前绝大多数协同文本编辑算法只支持两个基本的原子操作 insert 和 delete，少数算法支持 update 操作[59,75]. 虽然，理论上 update 操作可以由 delete 和 insert 操作组合实现. 但是在

实际应用中，组合操作对算法的性能有很大影响. 尤其是当用户执行的 update 操作数量增加时，将其转化为 delete 和 insert 的组合操作数量将成倍增加，使算法耗费更多的时空开销. 因此，探索直接支持复杂操作语义类型的实时协同编辑算法是需要进一步深入研究的内容. 例如，文献[59]采用了多版本技术将所有 update 操作的执行效果保留为多个版本，通过对多个版本的管理和分裂，让用户通过交互方式来选择其中一个版本. 文献[75]通过给每个 update 操作分配唯一全局的 ID，当有多个冲突的 update 操作时，通过比较 ID 的大小来保留其中的一个 update 操作的效果.

(5) 探索针对粗粒度对象而不是基本字符的实时协同编辑算法

在过去的 20 多年里，大部分协同编辑算法只考虑了如何支持面向字符的操作. 然而，在真实的协同工作系统中，操作的对象往往是粗粒度的. 例如，在协同文本编辑中，文本的剪切和复制. 在协同软件开发系统中，程序员编辑的对象是由多行代码组成的程序段. 当将这些支持字符操作的算法应用到粗粒度协同编辑系统上时，通常是将该粗粒度的操作简化为面向字符的操作序列来模拟实现，导致系统需要执行更多的操作，影响算法执行效率[41]. 近年来，随着大数据和云计算的发展，实时协同编辑系统也逐渐趋向于大规模的智能化协同[4,88-89]. 在智能化协同工作系统中，支持粗粒度对象的协同有利于协同用户更好的交流思想，分享经验、知识和智慧[90]. 在大规模的协同工作系统中，由于参与协同的用户数量增多，致使对共享对象的操作的数量也增多，理论上集成更多的操作通常会影响算法的响应性和交互性[91]. 因此，算法的计算效率在大规模的协同中至关重要. 相关研究表明，支持粗粒度对象的一致性维护算法在计算效率上更有优势[76-80]. 因此，如何设计支持粗粒度对象的协同编辑算法是该领域一个重要的研究方向.

(6) 充分利用多核 CPU/众核 GPU 体系来加速协同编辑算法

在实时协同编辑系统中，算法的性能是关键，尤其是算法的时间开销. 如果用户的操作不能得到快速响应，用户会因为系统的时滞而退出协同会话. 目前，主流的 OT 算法处理远程操作的时间复杂度为 $O(|H|^2)$. 少数算法通过采用优化的数据结构，将处理远程操作的时间开销优化为 $O(|H|)$. 尽管 OT 算法的计算性能被不断的优化，但是绝大部分 OT 算法的计算时间都依赖于操作历史的大小. 随着处理操作的数量的增加，操作历史的大小随之增加，处理远程操作的时间开销也随之增长. 因此，如何利用多核 CPU/众核 GPU 体系来加速现有的协同编辑算法，是该领域需要进一步研究和探索的难题. 例如，文献[92]给出了一个基于 GPU 的加速协同计算框架，该框架可用于改善和优化 CSCW 工作系统的性能. 文献[93]提出了一个并行的 OT 算法来批量的处理远程操作. 该算法利用并发操作的可交换性，将远程操作对操作历史的转换过程分配给多个并行线程完成，大大提高了系统的吞吐量.

(7) 探索有机融合 OT 和 CRDT 的实时协同编辑算法

相关研究表明，OT 算法的最大优势是具有很好的本地响应性[3-4]. OT 算法面临的最大挑战是伸缩性较差. 与 OT 算法相比，大部分的 CRDT 算法具有良好的伸缩性，适合应用于 p2p 的协同编辑领域. 然而，绝大部分 CRDT 算法多采用非线性的数据结构来存储操作对象，因而空间开销较大[77]. 因此，如何充分利用这两类算法的优势，探索出具有高响应和伸缩性强的实时协同编辑算法是该领域未来的研究内容之一. 目前，相关研究其仍处于初步阶段[81]，还需要进一步的深入研究.

(8) 探索支持实时协同编辑的云计算/服务计算

随着云计算/服务计算的发展，通过云平台实现信息的共享和编辑成为重要技术手段. 例如，谷歌服务 Google Drive，内置了 Google Docs，不仅提供了海量存储空间，而且支持大规模的实时协同办公. 因此，探索在云平台上的服务计算来支持多用户协同工作是未来发展趋势. 例如，文献[67]提出了一种支持异构协同编辑云服务之间的一致性维护方法，可以支持异构协同编辑的云服务之间的实时数据交换和共享. 文献[94]提出了一种面向服务的异构 CAD 系统的基于特征的数据交换方法. 该方法可以在云设计与制造平台的环境下实现异构 CAD 系统之间协同产品的设计.

(9) 将实时协同编辑算法应用到更广泛的热点领域

实时协同编辑系统编辑的对象包括文本、图形图像和复杂 CAD 模型等. 本文重点讨论了基于文本的实时协同编辑算法. 实际上，围绕着图形图像及复杂 CAD 模型的协同编辑算法不断涌现，形成了一批重要的研究成果[16-29,94-104].

在基于对象的图形图像协同编辑系统和协同

CAD 系统中,不同系统的对象数据结构不同,相应的协同编辑算法存在差异,例如:离散表示的图像协同编辑[17,19]和结构化表示的图形协同编辑[18,21]之间存在差异;二维图形系统[16,19]与三维图形系统[22-23]存在差异;三维网格图形数据[20]与三维 CAD 实体图形数据[95]存在差异;三维 CAD 实体图形数据[95]与三维特征 CAD 图形数据[29]存在差异[97].

以 Sun 等人经典文献[18]为例,采用了多版本技术来解决基于对象的图形编辑中用户操作的冲突问题,将用户冲突操作的执行效果保留为多个版本,通过对多个版本的分裂和管理,由用户通过交互的方式选择其中的一个版本.

以 Gao 等人 2016 年文献[19]为例,将多版本方法与 undo/redo 方法相结合,提出了基于位图的协同图像编辑的一致性维护方法.

以 Cheng 等人 2016 年文献[29]为例,根据特征 CAD 系统拓扑元素命名机制[95-96],对构成 CAD 特征图形的拓扑元素进行命名,提出了一种支持协同工作的拓扑元素数据结构(Topological Entity Structure Tree,TES_Tree),给出了三维特征 CAD 的操作冲突的检测方法及一致性维护方法.

异构 CAD 系统的集成与协同工作是目前的热点应用[27,98-104].

以文献[99-100]为例,提出了系统建模操作(system modeling operations)和中性模型命令(neutral modeling commands),来支持异构 CAD 系统之间的实时同步协同工作.

以文献[27]为例,提出了协同特征依赖图(Collaborative Feature Dependent Graph)和协同特征操作列表(Collaborative Feature Operational List),来支持各种遗产类(Legacy)CAD 系统之间实时协同工作,并进行灵活的并发控制.

除了同步协同工作之外,还包括异构 CAD 系统异步协同工作与集成技术.文献[101]分析了过程表示与过程获取的差异,提出了基于过程恢复的异构 CAD 集成与协同工作方法.文献[102]提出的拓扑元素匹配方法,既考虑同步协同工作,又考虑异步协同工作.文献[103-104]从奇异特征(singular feature)问题出发,提出了一种支持异构 CAD 系统的数据交换与异步协同工作方法.

最后,协同编辑系统及协同 CAD 的最新发展还包括:基于多核/众核平台[105-106]加速协同编辑算法[93];考虑安全措施的协同编辑系统[107];支持异构协同编辑云服务的一致性维护方法[67];大数据云制造背景下协同 CAD 系统的安全问题[108]与服务集成问题[94]等.

参 考 文 献

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

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

[3] Sun C, Yang Y, Zhang Y, Chen D. A consistency model and supporting schemes for real-time cooperative editing systems//Proceedings of the 19th Australian Computer Science Conference. Melbourne, Australia, 1996: 582-591

[4] Gu N, Yang J, Zhang Q. Consistency maintenance based on mark & retrace technique in groupware systems//Proceedings of the International ACM SIGGROUP Conference on Supporting Group Work. Florida, USA, 2005: 264-273

[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] Saito Y, Shapiro M. Optimistic replication. ACM Computing Surveys, 2005, 37(1): 42-81

[7] Jiang Jin-Lei, Shi Mei-Lin. Object synchronization and merging in CSCW. Journal of Computer Research and Development, 2003, 40(9): 1313-1318(in Chinese)
(姜进磊, 史美林. CSCW 中的对象同步与合并. 计算机研究与发展, 2003, 40(9): 1312-1318)

[8] Jain R. Eventweb: Developing a human-centered computing system. Computer, 2008, 41(2): 42-50

[9] Tomlinson B, Blevis E, Nardi B, et al. Collapse informatics and practice: Theory, method, and design. ACM Transactions on Computer-Human Interaction, 2013, 20(4): 1-26

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

[11] Reed D P, Kanodia R K. Synchronization with eventcounts and sequencers. Communications of the ACM, 1979, 22(2): 115-123

[12] Davis A H, Sun C, Lu J. Generalizing operational transformation to the standard general markup language//Proceedings of the ACM Conference on Computer Supported Cooperative Work. New Orleans, USA, 2002: 58-67

[13] Sun C, Xia S, Sun D, et al. Transparent adaptation of single-user applications for multi-user real-time collaboration. ACM Transactions on Computer-Human Interaction, 2006, 13(4): 531-582

[14] Xia S, Sun D, Sun C, et al. Leveraging single-user applications for multi-user collaboration: The coword approach// Proceedings of the ACM Conference on Computer Supported Cooperative Work. Chicago, USA, 2004: 162-171

[15] Lin K, Chen D, Sun C, Dromey G. Leveraging single-user Microsoft Visio for multi-user real-time collaboration// Proceedings of the CDVE. Shanghai, China, 2007: 353-360

[16] Sun C, Wen H, Fan H. Operational transformation for orthogonal conflict resolution in real-time collaborative 2D editing systems//Proceedings of the ACM Conference on Computer Supported Cooperative Work. Seattle, USA, 2012: 1391-1400

[17] Wang X, Bu J, Chen C. Achieving undo in bitmap-based collaborative graphics editing systems//Proceedings of the ACM Conference on Computer Supported Cooperative Work. New Orleans, USA, 2002: 68-76

[18] Sun C, David C. Consistency maintenance in real-time collaborative graphics editing systems. ACM Transactions on Computer-Human Interaction, 2002, 9(1): 1-41

[19] Gao L, Yu F, Chen Qi, Xiong N. Consistency maintenance of do and undo/redo operations in real-time collaborative bitmap editing systems. Cluster Computing, 2016, 19(1): 255-267

[20] Liu F, Xia S, Shen H, Sun C. CoMaya: Incorporating advanced collaboration capabilities into 3d digital media design tools// Proceedings of the ACM Conference on Computer Supported Cooperative Work. San Diego, USA, 2008: 5-8

[21] Jiang B, Bu J, Chen C, Wang B. Semantic consistency maintenance in collaborative graphics design systems// Proceedings of the 12th IEEE International Conference on Computer Supported Cooperative Work in Design. Xi'an, China, 2008: 35-40

[22] Sun C, Dong X. Operational transformation for dependency conflict resolution in real-time collaborative 3D design systems// Proceedings of the ACM Conference on Computer Supported Cooperative Work. Washington, USA, 2012: 1401-1410

[23] Sun C. Dependency-conflict detection in real-time collaborative 3D design systems//Proceedings of the ACM Conference on Computer Supported Cooperative Work. San Antonio, USA, 2013: 715-728

[24] Yang G, Shi M. Cova: A programming language for cooperative applications. Science in China Series F: Information Sciences, 2001, 44(1): 73-80

[25] Jiang Jin-Lei, Shi Mei-Lin. CovaTM and its implementation. Chinese Journal of Computers, 2003, 26(4): 438-445 (in Chinese)
(姜进磊，史美林. CovaTM 及其实现. 计算机学报，2003, 26(4): 438-445)

[26] Liu H, He F, Li X, Huang Z. Consistency maintenance in collaborative CAD system. Chinese Journal of Electronics, 2013, 22(1): 15-20

[27] Cai X, Li X, He F, et al. Flexible concurrency control for legacy CAD to construct collaborative CAD environment. Journal of Advanced Mechanical Design, Systems, and Manufacturing, 2012, 6(3): 324-339

[28] Cheng Y, He F, Cai X, et al. A group undo/redo method in 3D collaborative modelling systems with performance evaluation. Journal of Network and Computer Applications, 2013, 36(6): 1512-1522

[29] Cheng Y, He F, Wu Yi, Zhang D. Meta-operation conflict resolution for human—human interaction in collaborative feature-based CAD systems. Cluster Computing, 2016, 19(1): 237-253

[30] Weiss S, Urso P, Molli P. Wooki: A p2p Wiki-based collaborative writing tool//Proceedings of the 8th International Conference on Web Information Systems Engineering. Nancy, France, 2007: 503-512

[31] Vidot N, Cart M, Ferrié J, Suleiman M. Copies convergence in a distributed real-time collaborative environment//Proceedings of the ACM Conference on Computer Supported Cooperative Work. Philadelphia, USA, 2000: 171-180

[32] Nichols D A, Curtis P, Dixon M, Lamping J. High-latency, low-bandwidth windowing in the Jupiter collaboration system// Proceedings of the 8th Annual ACM Symposium on User Interface and Software Technology. Pittsburgh, USA, 1995: 111-120

[33] Sun C, Zhang Y, Jia X, Yang Y. A generic operation transformation scheme for consistency maintenance in real-time cooperative editing systems//Proceedings of the ACM Conference on Supporting Group Work. Phoenix, USA, 1997: 425-434

[34] Shen H, Sun C. Flexible notification for collaborative systems// Proceedings of the ACM Conference on Computer Supported Cooperative Work. New Orleans, USA, 2002: 77-86

[35] Li R, Li D, Sun C. A time interval based consistency control algorithm for interactive groupware applications//Proceedings of the 10th International Conference on Parallel and Distributed Systems. Newport Beach, USA, 2004: 429-436

[36] Sun D, Sun C. Operation context and context-based operational transformation//Proceedings of the 20th Anniversary Conference on Computer Supported Cooperative Work. Alberta, Canada, 2006: 279-288

[37] Sun D, Sun C. Context-based operational transformation in distributed collaborative editing systems. IEEE Transactions on Parallel and Distributed Systems, 2009, 20(10): 1454-1470

[38] Xu Y, Sun C, Li M. Achieving convergence in operational transformation: Conditions, mechanisms and systems// Proceedings of the 17th ACM Conference on Computer Supported Cooperative Work & Social Computing. Baltimore, USA, 2014: 505-518

[39] Xu Y, Sun C. Conditions and patterns for achieving convergence in OT-based co-editors. IEEE Transactions on Parallel and Distributed Systems, 2016, 27(3): 695-709

[40] Randolph A, Boucheneb H, Imine A, Quintero A. On synthesizing a consistent operational transformation approach. IEEE Transactions on Computers, 2015, 64(4): 1074-1089

[41] Shao Bin. Research on Efficient Consistency Maintenance Approaches of Operational Transformation [Ph. D. dissertation]. Fudan University, Shanghai, 2010(in Chinese)
(邵斌. 高效的操作转换一致性维护方法研究[博士学位论文]. 复旦大学, 上海, 2010)

[42] Li D, Li R. Preserving operation effects relation in group editors//Proceedings of the ACM Conference on Computer Supported Cooperative Work. Chicago, USA, 2004: 457-466

[43] Li R, Li D. A landmark-based transformation approach to concurrency control in group editors//Proceedings of the 2005 International ACM SIGGROUP Conference on Supporting Group Work. Florida, USA, 2005: 284-293

[44] Li D, Li R. An approach to ensuring consistency in peer-to-peer real-time group editors//Proceedings of the ACM Conference on Computer Supported Cooperative Work. San Diego, USA, 2008: 553-611

[45] Li D, Li R. An operational transformation algorithm and performance evaluation//Proceedings of the ACM Conference on Computer Supported Cooperative Work. San Diego, USA, 2008: 469-508

[46] Li D, Li R. An admissibility-based operational transformation framework for collaborative editing systems//Proceedings of the ACM Conference on Computer Supported Cooperative Work. Savannah, USA, 2010: 1-43

[47] Shao B, Li D, Gu N. ABTS: A transformation-based consistency control algorithm for wide-area collaborative applications//Proceedings of the 5th International Conference on Collaborative Computing: Networking, Applications and Worksharing. Washington, USA, 2009: 1-10

[48] Shao B, Li D, Gu N. An optimized string transformation algorithm for real-time group editors//Proceedings of the 15th International Conference on Parallel and Distributed Systems (ICPADS). Shenzhen, China, 2009: 376-383

[49] Shao B, Li D, Gu N. A fast operational transformation algorithm for mobile and asynchronous collaboration. IEEE Transactions on Parallel and Distributed Systems, 2010, 21(12): 1707-1720

[50] Shao B, Li D, Gu N. A sequence transformation algorithm for supporting cooperative work on mobile devices//Proceedings of the ACM Conference on Computer Supported Cooperative Work. Savannah, USA, 2010: 159-168

[51] Shao B, Li D, Gu N. An algorithm for selective undo of any operation in collaborative applications//Proceedings of the 16th ACM Conference on Supporting Group Work. Sanibel Island, USA, 2010: 131-140

[52] Liao Bin, He Fa-Zhi, Jing Shu-Xu. Survey of operational transformation algorithms in real-time computer-supported cooperative work. Journal of Computer Research and Development, 2007, 44(2): 326-333(in Chinese)
(廖斌, 何发智, 荆树旭. 实时协同工作系统中操作转换算法综述. 计算机研究与发展, 2007, 44(2): 326-333)

[53] Imine A. Flexible concurrency control for real-time collaborative editors//Proceedings of the 28th International Conference on Distributed Computing Systems Workshops. Beijing, China, 2008: 423-428

[54] Oster G, Molli P, Urso P, Imine A. Tombstone transformation functions for ensuring consistency in collaborative editing systems//Proceedings of the International Conference on Collaborative Computing: Networking, Applications and Worksharing. Atlanta, USA, 2006: 1-10

[55] Imine A, Molli P, Oster G, Rusinowitch M. Proving correctness of transformation functions in real-time groupware//Proceedings of the 8th European Conference of Computer-Supported Cooperative Work. Helsinki, Finland, 2003: 277-293

[56] Sun C, Xu Y, Agustina A. Exhaustive search of puzzles in operational transformation//Proceedings of the 17th ACM Conference on Computer Supported Cooperative Work & Social Computing. Baltimore, USA, 2014: 519-529

[57] Sun C, Chen D. A multi-version approach to conflict resolution in distributed groupware systems//Proceedings of the 20th International Conference on Distributed Computing Systems. Taipei, China, 2000: 316-325

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

[59] Sun D, Xia S, Sun C, Chen D. Operational transformation for collaborative word processing//Proceedings of the ACM Conference on Computer Supported Cooperative Work. Chicago, USA, 2004: 437-446

[60] Cai Wei-Wei, He Fa-Zhi, Lv Xiao. An efficient preserving intention operational transformation for real-time collaborative editing. Chinese Journal of Computers, 2015, 38(10): 2041-2053(in Chinese)
(蔡维纬, 何发智, 吕晓. 一种高效率的实时协同编辑中的意图保持操作转换算法. 计算机学报, 2015, 38(10): 2041-2053)

[61] Sun C, Jia X, Zhang Y, Chen D. Achieving convergence, causality-preservation, and intention preservation in real-time cooperative editing system. ACM Transactions on Computer-Human Interaction, 1998, 5(1): 63-108

[62] Gu Ning, Yang Jiang-Ming, Zhang Qi-Wei. Consistency maintenance based on the address space transformation technique in group editors. Chinese Journal of Computers, 2007, 30(5): 763-774(in Chinese)
(顾宁, 杨江明, 张琦炜. 协同组编辑中基于地址空间转换的一致性维护方法. 计算机学报, 2007, 30(5): 763-774)

[63] Yang Jiang-Ming, Gu Ning, Wu Xiao-Yuan. Address space transformation based on method supporting group Undo. Journal on Communications, 2006, 27(3): 48-56(in Chinese)
(杨江明, 顾宁, 吴筱媛. 基于地址空间转换方法的 Undo 操作支持. 通信学报, 2006, 27(3): 48-56)

[64] Gu N, Zhang Q, Yang J, Ye W. DCV: A causality detection approach for large-scale dynamic collaboration environments //Proceedings of the International ACM Conference on Supporting Group Work. Sanibel Island, USA, 2007: 157-166

[65] Yang J, Wang H, Gu N, et al. Lock-free consistency control for web2.0 applications//Proceedings of the 17th International Conference on World Wide Web. Beijing, China, 2008: 725-734

[66] Xia H, Lu T, Shao B, et al. A partial replication approach for anywhere anytime mobile commenting//Proceedings of the 17th ACM Conference on Computer Supported Cooperative Work & Social Computing. Baltimore, USA, 2014: 530-541

[67] Xia H, Lu T, Shao B, et al. Hermes: On collaboration across heterogeneous collaborative editing services in the cloud// Proceedings of the 18th IEEE International Conference on Computer Supported Cooperative Work in Design. Shanghai, China, 2014: 655-660

[68] Gao L, Tang W. High efficient consistency maintenance strategy of real-time string text editing systems. International Journal of Hybrid Information Technology, 2015, 8(10): 383-394

[69] Yang D, Lu T, Xia H, et al. Making itinerary planning collaborative: An ast-based approach//Proceedings of the 20th IEEE International Conference on Computer Supported Cooperative Work in Design. Nanchang, China, 2016: 257-262

[70] Oster G, Urso P, Molli P, Imine A. Data consistency for p2p collaborative editing//Proceedings of the 20th Anniversary Conference on Computer Supported Cooperative Work. Banff, Canada, 2006: 259-268

[71] Weiss S, Urso P, Molli P. Logoot: A scalable optimistic replication algorithm for collaborative editing on p2p networks// Proceedings of the 29th IEEE International Conference on Distributed Computing Systems. Montreal, Canada, 2009: 404-412

[72] Weiss S, Urso P, Molli P. Logoot-undo: Distributed collaborative editing system on p2p networks. IEEE Transactions on Parallel and Distributed Systems, 2010, 21(8): 1162-1174

[73] Preguica N, Marques J M, Shapiro M, Letia M. A commutative replicated data type for cooperative editing//Proceedings of the 29th IEEE International Conference on Distributed Computing Systems. Montreal, Canada, 2009: 395-403

[74] Wu Q, Pu C, Ferreira J E. A partial persistent data structure to support consistency in real-time collaborative editing// Proceedings of the 26th IEEE International Conference on Data Engineering (ICDE). Long Beach, USA, 2010: 776-779

[75] Roh H G, Jeon M, Kim J S, Lee J. Replicated abstract data types: Building blocks for collaborative applications. Journal of Parallel and Distributed Computing, 2011, 71(3): 354-368

[76] Lv X, He F, Cai W, Cheng Y. A string-wise CRDT algorithm for smart and large-scale collaborative editing systems. Advanced Engineering Informatics. doi: 10.1016/j.aei.2016.10.005 (in Press)

[77] Ahmed-Nacer M, Ignat C L, Oster G, Roh H G. Evaluating CRDTs for real-time document editing//Proceedings of the 11th ACM Symposium on Document Engineering. Mountain View, USA, 2011: 103-112

[78] Yu W, André L, Ignat C L. A CRDT supporting selective undo for collaborative text editing//Proceedings of the International Federated Conference on Distributed Computing Techniques. Grenoble, France, 2015: 193-206

[79] Yu W. Supporting string-wise operations and selective undo for peer-to-peer group editing//Proceedings of the 18th International Conference on Supporting Group Work. Sanibel Island, USA, 2014: 226-237

[80] André L, Martin S, Oster G, Ignat C L. Supporting adaptable granularity of changes for massive-scale collaborative editing// Proceedings of the 9th International Conference Conference on Collaborative Computing: Networking, Applications and Worksharing (Collaboratecom). Austin, Texas, 2013: 50-59

[81] Mehdi A N, Urso P, Balegas V, Perguica N. Merging ot and CRDT algorithms//Proceedings of the 1st Workshop on Principles and Practice of Eventual Consistency. New York, USA, 2014: 8-11

[82] Duchien L, Florin G, Seinturier L. Partial order relations in distributed object environments. ACM SIGOPS Operating Systems Review, 2000, 34(4): 56-75

[83] Ressel M, Ruhland N, Gunzenhauser R. An integrating, transformation-oriented approach to concurrency control and undo in group editors//Proceedings of the ACM Conference on Computer Supported Cooperative Work. Boston, USA, 1996: 288-297

[84] Gu N, Xu J, Wu X, et al. Ontology based semantic conflicts resolution in collaborative editing of design documents. Advanced Engineering Informatics, 2005, 19(2): 103-111

[85] Androutsellis-Theotokis S, Spinellis D. A survey of peer-to-peer content distribution technologies. ACM Computing Surveys, 2004, 36(4): 335-371

[86] Cart M, Ferrie J. Asynchronous reconciliation based on operational transformation for p2p collaborative environments// Proceedings of the International Conference on Collaborative Computing: Networking, Applications and Worksharing. New York, USA, 2007: 127-138

[87] Prakash A, Knister M J. A framework for undoing actions in collaborative systems. ACM Transactions on Computer-Human Interaction, 1994, 1(4): 295-330

[88] Ackerman M S, Dachtera J, Pipek V, Wulf V. Sharing knowledge and expertise: The CSCW view of knowledge management. Computer Supported Cooperative Work, 2013, 22(4-6): 531-573

[89] Fischer J, Porcheron M, Lucero A, et al. Collocated interaction: New challenges in 'same time, same place' research//Proceedings of the 19th ACM Conference on Computer Supported Cooperative Work and Social Computing Companion. New York, USA, 2016: 465-472

[90] Negrao A P, Costa J, Ferreira P, Veiga L. Interest aware consistency for cooperative editing in heterogeneous environments. International Journal of Cooperative Information Systems, 2014, 23(1): 42-75

[91] Ignat C L, Oster G, Fox O, et al. How do user groups cope with delay in real-time collaborative note taking//Proceedings of the 14th European Conference on Computer Supported Cooperative Work. Oslo, Norway, 2015: 223-242

[92] Chen G, Li G, Wu B, Pei S. A GPU-based computing framework for CSCW//Proceedings of the 14th International Conference on Computer Supported Cooperative Work in Design. Shanghai, China, 2010: 100-103

[93] Cai W, He F, Lv X. Multi-core accelerated operational transformation for collaborative editing//Proceedings of the 11th International Conference on Collaborative Computing: Networking, Applications and Worksharing. Wuhan, China, 2015: 121-128

[94] Wu Y, He F, Zhang D, Li X. Service-oriented feature-based data exchange for cloud-based design and manufacturing. IEEE Transactions on Services Computing, online, doi: 10.1109/TSC.2015.2501981

[95] Jing S, He F, Han S, Liu H. A method for topological entity correspondence in a replicated collaborative CAD system. Computers in Industry, 2009, 60(7): 467-475

[96] Jing Shu-Xu, He Fa-Zhi, Liu Hua-Jun. A survey of persistent naming problem for topological entities. Journal of Computer-Aided Design & Computer Graphics, 2007, 19(5): 545-552(in Chinese)
(荆树旭, 何发智, 刘华俊. 拓扑元素永久命名综述. 计算机辅助设计与图形学学报, 2007, 19(5): 545-552)

[97] He F, Han S. A method and tool for human-human interaction and instant collaboration in CSCW-based CAD. Computers in Industry, 2006, 57(8): 740-751

[98] Gao Shu-Ming, He Fa-Zhi. A survey of heterogeneous CAD system integration. Journal of Computer-Aided Design & Computer Graphics, 2009, 21(5): 561-568(in Chinese)
(高曙明, 何发智. 异构 CAD 系统集成技术综述. 计算辅助设计与图形学学报, 2009, 21(5): 561-568)

[99] Gao S. Real-time exchange of CAD models based on neutral modelling commands. International Journal of Product Lifecycle Management, 2010, 4(4): 331-337

[100] Li M, Gao S, Fuh J, Zhang Y. Replicated concurrency control for collaborative feature modelling: A fine granular approach. Computers in Industry, 2008, 59(9): 873-881

[101] Li X, He F, Cai X, Zhang D. CAD data exchange based on the recovery of feature modelling procedure. International Journal of Computer Integrated Manufacturing, 2012, 25(10): 874-887

[102] Li X, He F, Cai X, et al. A method for topological entity matching in the integration of heterogeneous CAD systems. Integrated Computer-Aided Engineering, 2013, 20(1): 15-30

[103] Zhang De-Jun, He Fa-Zhi, Wu Yi-Qi. Singular feature interoperability of heterogeneous CAD model based on directed mutation particle swarm optimization. Science in China, Series F: Information Sciences, 2015, 45(5): 634-649(in Chinese)
(张德军, 何发智, 吴亦奇. 一种基于定向变异粒子群算法的异构 CAD 模型奇异特征互操作方法. 中国科学: 信息科学, 2015, 45(5): 634-649)

[104] Zhang D, He F, Han S, Li X. Quantitative optimization of interoperability during feature-based data exchange. Integrated Computer-Aided Engineering, 2016, 23(1): 31-50

[105] Zhou Y, He F, Qiu Y. Optimization of parallel iterated local search algorithms on graphics processing unit. The Journal of Supercomputing, 2016, 72(6): 2394-2416

[106] Zhou Y, He F, Qiu Y. Dynamic strategy based parallel ant colony optimization on GPUs for TSPs. Science China Information Sciences, 2016, doi: 10.1007/s11432-015-0594-2

[107] Yeh S, Su M, Chen H, Lin C. An efficient and secure approach for a cloud collaborative editing. Journal of Network and Computer Applications, 2013, 36(6): 1632-1641

[108] Wu Y, He F, Chen Y. A service-oriented secure infrastructure for feature-based data exchange in cloud-based design and manufacture. Procedia CIRP, 2016, 56: 55-60

HE Fa-Zhi, born in 1968, Ph. D., professor. His research interests include collaborative computing, CAD graphics and images, multi-core CPU/many-core GPU computing.

LÜ Xiao, born in 1983, Ph. D. candidate, lecturer. Her research interests include Computer Supported Collaborative Work (CSCW), collaborative CAD.

CAI Wei-Wei, born in 1988, Ph. D. candidate. His research interests include Computer Supported Collaborative Work (CSCW), collaborative CAD.

CHENG Yuan, born in 1983, Ph. D., lecturer. Her research interests include Computer Supported Collaborative Work (CSCW), collaborative CAD.

Background

This paper provides a complete survey of real-time collaborative editing algorithms supporting operation intention consistency, which lays a solid foundation for us to develop related algorithms. Real-time collaborative editing systems can support natural and harmonious human to human interactions, which are different from traditional distributed systems. A great challenge of real-time collaborative editing systems is the consistency maintenance, which has been a hot research point in collaborative computing. Over nearly three decades, many consistency maintenance algorithms have been proposed, such as OT algorithms, AST and CRDT algorithms.

At first, most existing algorithms more focus on how to achieve the result consistency. There are two strategies: one is to design transformation functions capable of preserving TP1/TP2; the other is to design generic control algorithms capable of avoiding TP1/TP2. Past research has found that it is relatively easy to construct a total ordering transformational path to avoid TP2 by using control mechanisms.

Even if the result consistency can be achieved, the consistency result may be contrary to users' operation intentions. Therefore, the consistency maintenance gradually focuses on supporting operation intention consistency. Over nearly three decades, how to maintain operation intention consistency has been a challenging issue in real-time collaborative editing systems.

In this paper, a survey of real-time collaborative editing algorithms supporting operation intention consistency is presented. From the point of Lamport's partial ordering relations of events, the concepts and terminologies of causal ordering relations, concurrent relations and partial concurrent relations in real-time collaborative editing systems are discussed and generalized. Specially, after analyzing the total order hypothesis and priority in depth, the route map of maintaining operation intention consistency is proposed according to the classification of causality, convergence and operation intention preservation. Meanwhile, the typical algorithms are categorized into different classes according to operation intention consistency. Then, the main principle, the whole framework and the same representative collaborative work scenario and sample are discussed in each class algorithm. After comparing and analyzing these algorithms individually, a conclusion is drawn. Some possible research directions in future are given.

This work is supported by the National Natural Science Foundation of China (Nos. 61472289 and 61502353) and the National Key Research and Development Project (No. 2016YFC0106305).