Analysis of Operational Transformation Algorithms

Santosh Kumawat and Ajay Khunteta

Abstract In multiuser groupware systems, consistency maintenance and concurrency control are the most significant challenges. In groupware systems, user groups are permitted to update the shared data simultaneously. Operational transformation (OT) is a successful method for consistency maintenance in multiuser shared applications. OT, in general, supports two basic operations: insert/delete for character operations. We have done the literature review of the evolution of OT algorithms over the last 25 years since 1989. OT is discussed based on existing main OT algorithms such as dOPT, adOPT, GOT, GOTO, SDT, SOCT2, SOCT3/4, and ABT which are analyzed on the basis of the main properties as criteria of correctness, remote operation property, storage, etc. Then, categorization is done for all existing OT algorithms on the basis of major existing algorithms such as dOPT, adOPTed, GOT, GOTO, SDT, SOCT2, SOCT3/4, and ABT and then further classified on the basis of area of operation like undo, char, string, web, graph, etc. OT algorithms supporting string handling are also analyzed.

Keywords Groupware system · Distributed system · Operational transformation · Inclusion transformation · Concurrency · String handling · Consistency maintenance

Introduction

In real-time groupware systems which have multiple users, the actions of all users must be immediately propagated to all other users. Groupware systems are multiuser systems which have a shared environment and require sharing of data. The

Santosh Kumawat (✉)
Poornima University, Jaipur, Rajasthan, India
e-mail: santoshkumawat82@ymail.com

Ajay Khunteta
Poornima College of Engineering, Jaipur, Rajasthan, India
e-mail: khutetaajay@poornima.org

© Springer India 2016

N. Afzalpulkar et al. (eds.), *Proceedings of the International Conference on Recent Cognizance in Wireless Communication & Image Processing*,
DOI 10.1007/978-81-322-2638-3_2

prime requirements of groupware systems are fast response times, fine granularity, and concurrency control. Transformation algorithms are needed for consistency control in multiuser shared systems.

Traditional concurrency control methods were not suitable for distributed shared applications because they may cause the loss of user interaction results and so fail to satisfy requirement of quick local response which fulfills user intentions. Also in this case local response satisfies consistency and convergence.

In the past 25 years, OT [1, 2] is used for concurrency control in multiuser shared collaborative systems. OT achieves better user intention preservation with convergence and causality preservation. It does not reduce responsiveness and concurrent work [3] but permits users to modify shared matter at the desired time [4], which helps in development of collaborative systems in the desired collaboration medium.

Operational Transformation

In the multiuser environment, OT does replication of the shared data at all user sites. At a site, first, local operations are executed and OT transformations remote operations repair differences and inconsistencies.

OT Framework

Consider the example in which data get replicated at two shared sites. At site A and site B we have a list of cities. If we perform insert and delete operations at sites A and B and then broadcast it to another site, then without OT we get the wrong output but with OT we get the right output. The process is explained in Figs. 1 and 2.

Algorithms

This paper studies the major OT algorithms of the past 25 years for consistency maintenance in group editors, which consist of the distributed operation transformation (dOPT) algorithm [5], the SCOT2 [6], SCOT 3/4 algorithm [7], the adopted (adOPTed) algorithm [8], admissibility-based transformation (ABT) algorithm [9], ABT-undo (ABTU) algorithm [10], the state difference transformation (SDT) algorithm [11], admissibility-based sequence transformation (ABST) [12], and admissibility-based transformation with strings (ABTS) algorithm [13], the GOT optimized (GOTO) algorithm [2], and the generic operational transformation (GOT) algorithm [14].

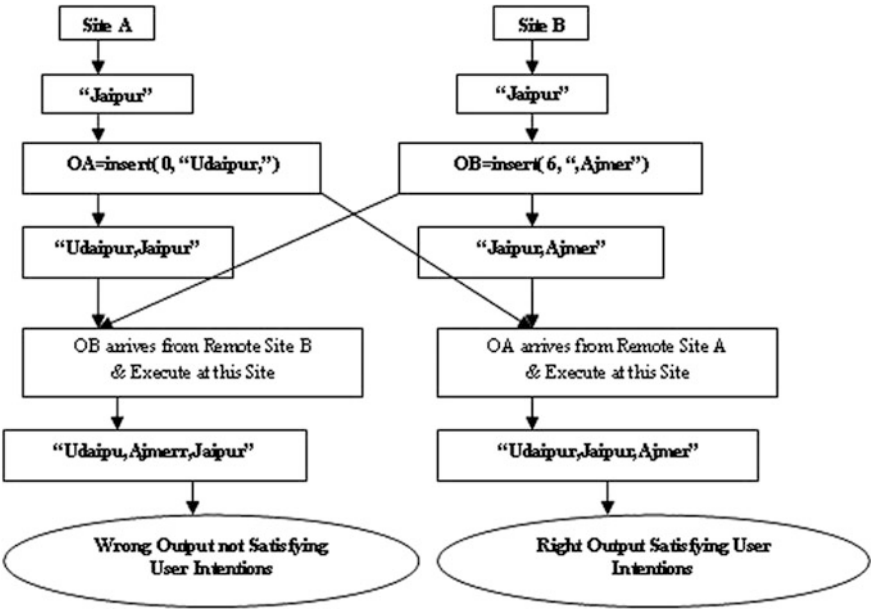

Fig. 1 Example of string operations without OT implementation

dOPT

GROVE adopted a replicated architecture of shared users to avoid a single point of failure and achieve good responsiveness in the system. GROVE has invented distributed operation transformation algorithm [5]. The dOPT-puzzle is the weakness of this algorithm. This algorithm has for local operations immediate feedback. It is distributed and does not have a central server. It fulfills the precedence property and enforces the transformation property. Properties of dOPT are explained in Fig. 3.

adOPTed

The adOPTed algorithm proposed by Ressel [8]. The adOPTed algorithm is right in general but has not proved intention preservation property. The algorithm has all the positive strengths of dOPT which have improved it. Various properties of adOPTed algorithm such as correctness criteria, etc., are explained in Fig. 4.

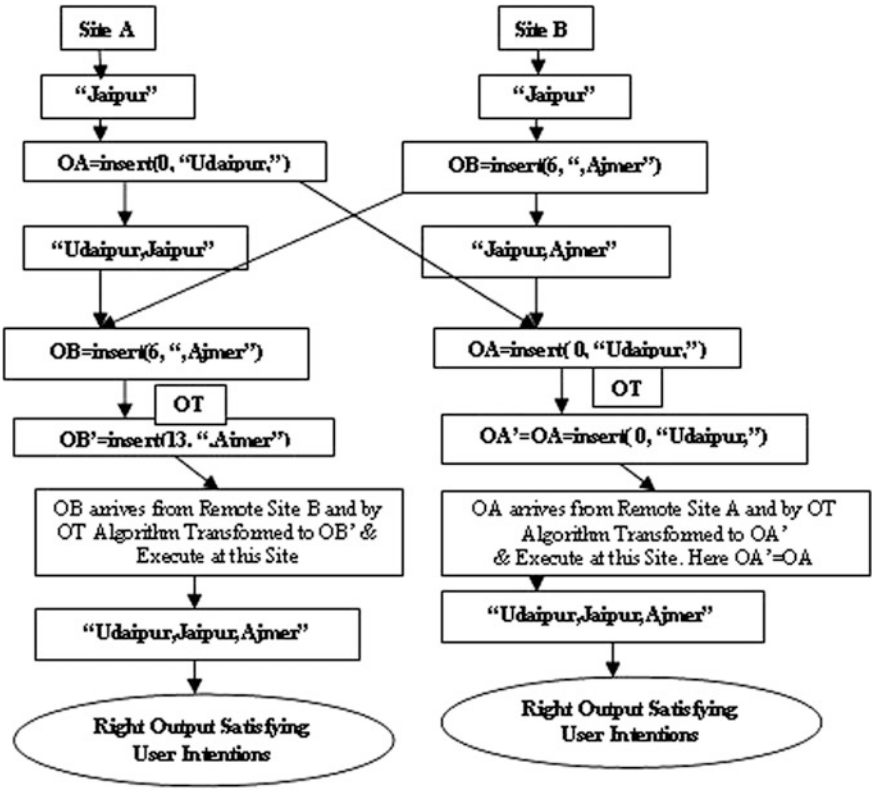

Fig. 2 Example of string operations with OT implementation

GOT

The GOT control algorithm is good for convergence. It is integrated with undo/do/redo scheme [14]. Correctness proof of algorithm is missing. It supports only basic string operations: Insert/Delete.

In this convergence TP2 is not a necessary condition. IT and ET algorithms pair has been devised for string-wise insert/delete. In a fully replicated system, convergence is achieved by GOT, which ensures at all sites cursor position maintenance and consistency maintenance. By adopting an undo/do/redo scheme GOT ensures total order. Properties of GOT can be seen in Fig. 5.

GOTO

Optimization of the GOT control algorithm is possible by two postconditions, TP1 and TP2 to reduce the number of IT/ET transformations. The optimized algorithm

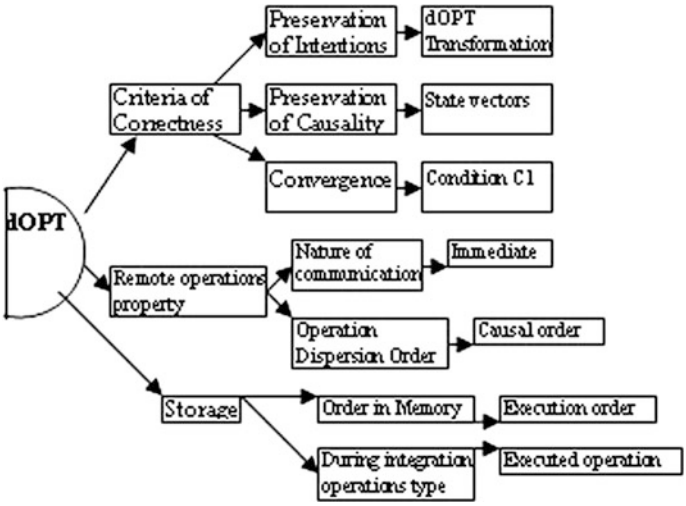

Fig. 3 Analysis of dOPT algorithm

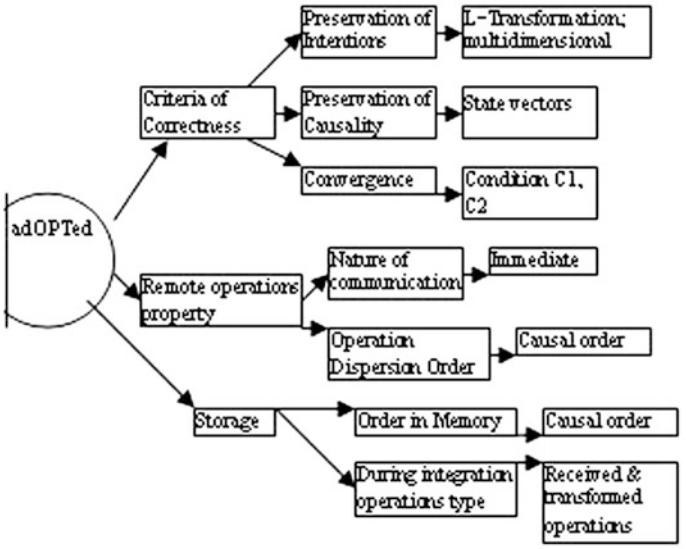

Fig. 4 Analysis of adOPTed algorithm

of the GOT control algorithm is called GOTO (GOT Optimized). To handle the so-called "lossy IT" problem GOTO [2] needs extra memory. Time complexity of the GOTO control algorithm is O (n^2). It is required to extend the algorithm to support strings. It should have integration of sequences. It fulfills the convergence property, precedence property, transformation property 1, and transformation

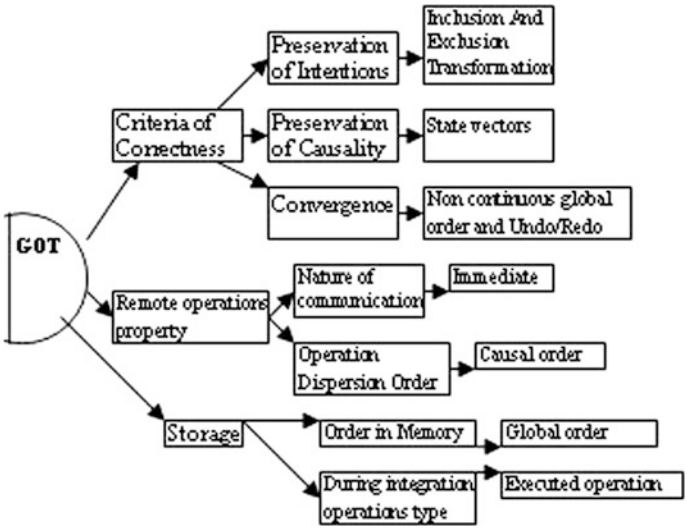

Fig. 5 Analysis of GOT algorithm

property 2 and has better undo-based operations. It is tested in editing programs such as CoMaya, CoWord, etc. Properties of GOTO control algorithm are explained in Fig. 6.

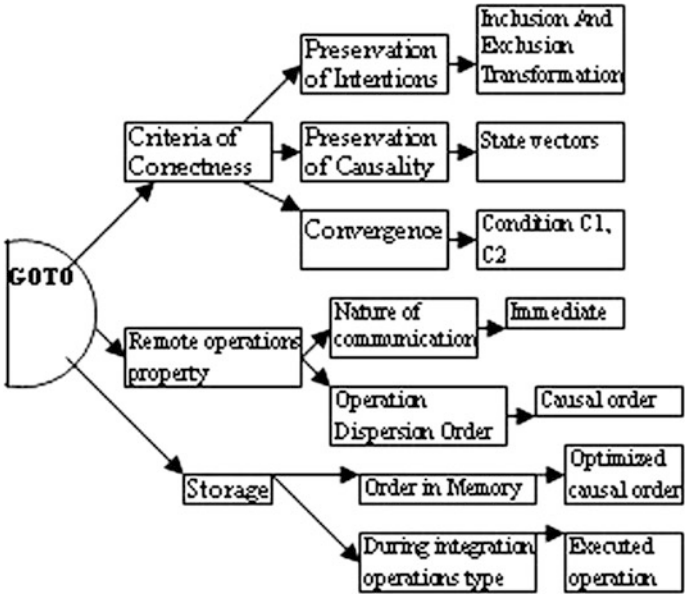

Fig. 6 Analysis of GOTO algorithm

SOCT3/SOCT4

SOCT3 has local execution, broadcast, and reception of operations. In SOCT4 and SOCT3, the operations are ordered globally by a timestamp which is given by a sequencer. In SOCT3/SOCT4 string handling are not supported. It desires implementation of algorithms in real-time applications. State vectors are not needed. Properties of SOCT3/4 are highlighted in Fig. 7.

SDT

SDT [11] has solved the TP2 puzzle. Properties of SDT can be seen in Fig. 8. Various mechanisms such as group undo, optional locking, and multi-version should be based on the concept of SDT.

ABT

ABT [9] depends on conditions like admissibility and preservation of causality and which under all conditions do not require transformation functions to work. Exclusive transformation (ET) is simple in ABT.

The string-wise ABTS algorithm by Li and Li [13] is extended from its character-wise ABT algorithm. ABTS is the first string-wise algorithm based on

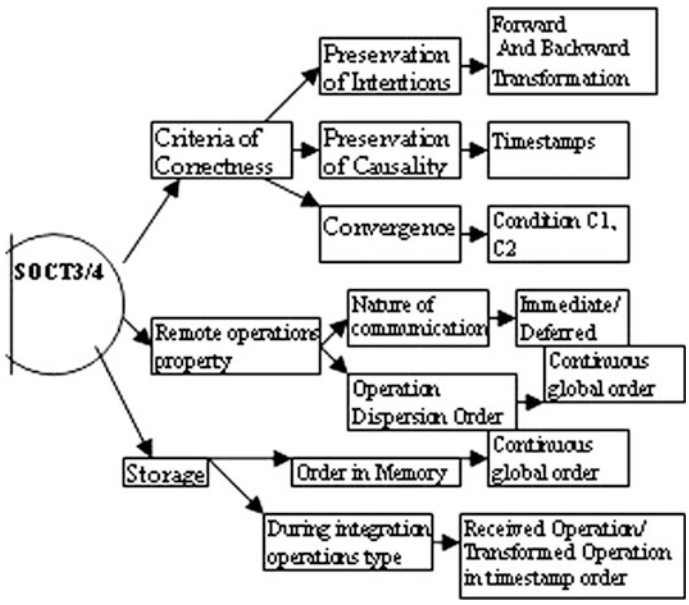

Fig. 7 Analysis of SOCT 3/4 algorithm

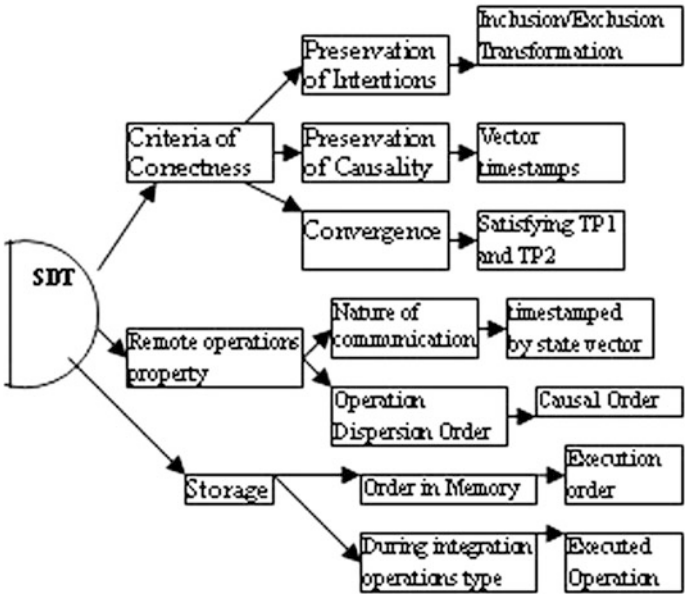

Fig. 8 Analysis of SDT algorithm

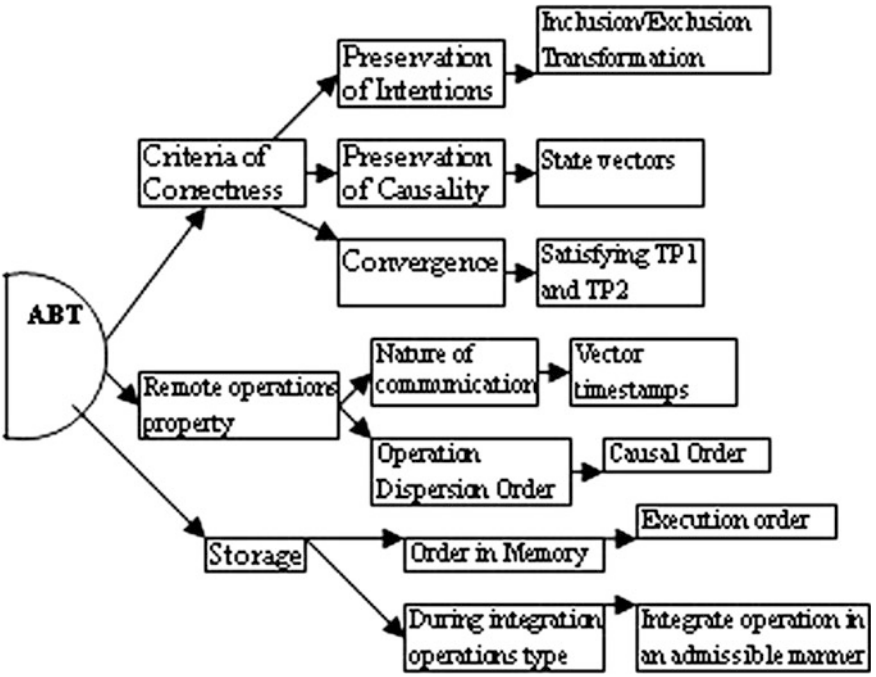

Fig. 9 Analysis of ABT algorithm

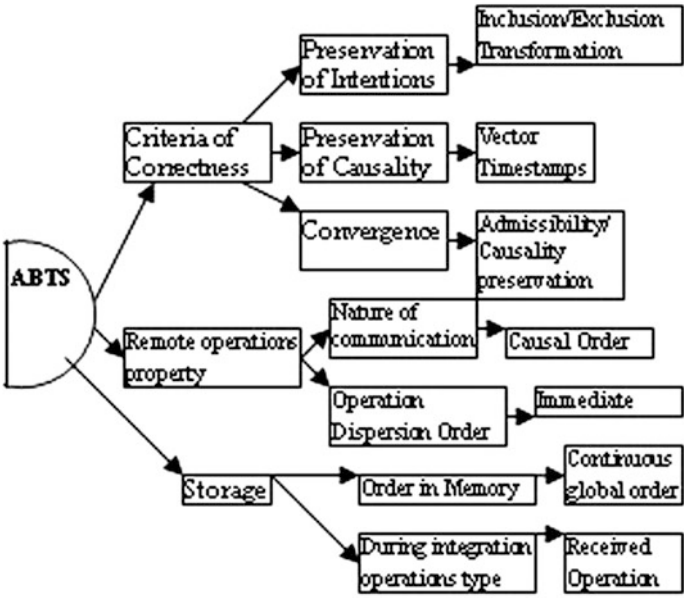

Fig. 10 Analysis of ABTS algorithm

ABT framework, which is formally proved without saving object relation. The time and space complexity of the presented ABTS algorithm is O (IHI). ABTS does not need a total order of execution and reversibility of operations. In algorithms like ABT, ABTS, and ABTU, IHI grow indefinitely. ABTU provides integrated support of do and selective undo. ABST follows concepts like ABT and hence its correctness could be formally proved. Properties of ABT can be seen in Fig. 9 and properties of ABTS are explained in Fig. 10.

Categorization of OT Algorithms

On categorizing all existing algorithms on the basis of major existing algorithms such as dOPT, adOPTed, GOT, GOTO, SDT, SOCT2, SOCT3/4, ABT, and then further classified on the basis of area of operation like undo, char, string, web,

Table 1 Categorization of OT algorithms

Operations	Major algorithm							
	dOPT	adOPTed	GOT	GOTO	SOCT2	SOCT3/4	SDT	ABT
Character	6	5	5	13	3	4	3	7
Undo/Do	2	1	1	2	1		1	2
Graph/Web-based	6			8	2		2	2
String		**1**	**1**					**1**
Total papers	14	7	7	23	6	4	6	13

Table 2 Comparison of OT algorithms

Table	dOPT	adOPTed	GOT	GOTO	SOCT2	SOCT3	SOCT4	SDT	ABT	ABTS
Time complexity	Consume more time	Less than dOPT	A bit less than adOPTed	**O(I H^2I)**	**O(I H^2I)**	A bit more than SCOT4	A bit more than ABT	O(I H^2 I) which is slower than	O(I H^2 I)	**O(|H|)**
Support string handling	No	No	**Yes**	**Yes**	No	No	No	No	No	**Yes**
Framework	CC Framework	CC Framework	CC Framework	CC Framework	CC Framework	CC Framework	CC Framework	CCI Framework	ABT Framework	ABT Framework
Space complexity	More space	A bit less than dOPT	**O(I H^2I)**	**O(I H^2I)**	O(I H^2 I)	A bit more than SCOT4	A bit more than ABT	A bit more than ABT	O(|H|)	**O(|H|)**

graph, etc., we get as in Table 1. In Table 1 is shown classification of 80 papers. In Table 1, a column is for a particular algorithm and a row is for a particular operation. The quantity in a cell shows the number of papers based on a particular algorithm supporting a particular operation. From Table 1 we find that only three algorithms support string handling—GOT, GOTO, and ABTS.

In Table 2 a comparison is done of GOT, GOTO and ABTS with respect to parameters like string handling, time, and space complexity. The second row indicates support for string handling and the values for GOT, GOTO, and ABTS are underlined. Only these three algorithms—GOT, GOTO, and ABTS support string handling which have value 'yes' and the rest all have value 'No'. In the third row space complexity is shown and the values are encircled. In the first row time complexity is shown and the values are underlined. GOT and GOTO have time and space complexity of order $O(|H^2|)$ but ABTS have time and space complexity of order $O(|H|)$. So From Table 1 we conclude that only three OT algorithms support string handling called GOT, GOTO and ABTS. From Table 2 we conclude that ABTS have support for string handling and is better than GOT and GOTO because it has less time and space complexity. Also, ABTS is based on ABT framework which can be formally proved.

Conclusion

In distributed systems groupware systems are multiuser interactive computer-based systems where users have great interaction with each other. Traditional consistency methods do not apply in groupware systems because they only consider system limited issues, serial application, and may not satisfy interaction results and user intention consistency. It is analyzed that OT is the most desired standard solution to concurrency control and consistency maintenance which satisfies intentions. In this review a study of OT techniques, over the past 25 years is presented. A comparative study of 80 papers is done of various algorithms of OT based on different parameters and it has discussed major issues, algorithms, achievements, and remaining challenges.

All existing OT algorithms are classified on the basis of main properties like criteria of correctness, remote operation property, storage, etc., which can be observed from Fig. 3 to Fig. 10. Also, a relative comparison of a number of OT algorithms including dOPT, adOPTed, GOT, GOTO, SDT, SOCT2, SOCT3, SOCT4, ABT, ABTS has been done relative to various parameters and constraints that are intention preservation, causality preservation, convergence for correctness criteria, in case of remote operations nature of communication and order of operation dispersion and memory operations like order in memory or during integration operation type in memory and also other parameters like time complexity, space complexity, support for string handling, transformation functions, and framework. All 80 papers at first get classified on the basis of major algorithms like dOPT, adopted, GOT, GOTO, ABT, SDT, SOCT2, SOCT3/4. Also, they are further are subcategorized on the basis of area of operation like string, character, undo,

web-based, graphical, etc. In the literature, only the GOT, GOTO and ABTS algorithms support string-wise operations. ABTS is the best string handling algorithm as it has less time and space complexity.

User intentions preservation should be considered more in OT algorithms. More work is required to reduce time complexity and space complexity of OT algorithms. Very few OT algorithms support insert/delete string operations so it is required development of OT algorithms supporting new composite string operations like cut-paste, find-replace, which can be formally proved. Also, integration of string handling and undo operation is still pending in OT algorithms.

References

1. Davis, A.H., Sun C., Lu, J.: Generalizing operational transformation to the standard general markup language. In: ACM (2002)
2. Sun, C., Ellis, C.: Operational transformation in real-time group editors: issues, algorithms, and achievements. In: ACM CSCW'98 (1998)
3. Sun, C., Jia, X., Zhang, Y., Yang, Y., Chen, D.: Achieving convergence, causality-preservation, and intention preservation in real-time cooperative editing systems. ACM Trans. Comput. Hum. Interact. **5**(1), 63–108 (1998)
4. Bentley, R., Dourish, P.: Medium versus mechanism: supporting collaboration through customization. In: ECSCW'95 Proceedings (1995)
5. Ellis, C.A., Gibbs, S.J.: Concurrency control in groupware systems. ACM Sigmod Rec. **18** (2): 399–407 (1989). doi:10.1145/66926. Retrieved 2007-07-26
6. Suleiman, M., Cart, M., Ferrié, J.: Concurrent operations in a distributed and mobile collaborative environment. In: Proceedings of the Fourteenth International Conference on Data Engineering, pp. 23–27, Feb 1998
7. Vidot, N., Cart, M., Ferrie, J., Suleiman, M.: Copies convergence in a distributed real-time collaborative environment. In: Proceedings of the 2000 ACM conference on Computer supported cooperative work, pp. 171–180. ACM Press New York, NY, USA (2000)
8. Ressel, M., Nitsche-Ruhland, D., Gunzenhäuser, R.: An integrating, transformation-oriented approach to concurrency control and undo in group editors. In: CSCW '96 Proceedings of the 1996 ACM conference on Computer supported cooperative work. pp. 288–297 (1996). doi:10.1145/240080.240305
9. Li, D., Li, R.: An admissibility-based operational transformation framework for collaborative editing systems. Comput. Support. Coop. Work J. Collab. Comput., Aug 2009. Accepted
10. Shao, B., Li, D. Gu, N.: An algorithm for selective undo of any operation in collaborative applications. In: ACM (2010)
11. Li, D., Li, R.: An approach to ensuring consistency in peer-to-peer real-time group editors, in Springer (2006)
12. Shao, B., Li, D. Gu, N.: A fast operational transformation algorithm for mobile and asynchronous collaboration, IEEE Trans. Parallel Distrib. Syst. **21**(12) (2010)
13. Shao, B., Li, D., Gu, N: ABTS: a transformation-based consistency control algorithm for wide-area collaborative applications. In: 5th International Conference on Collaborative Computing: Networking, Applications and Worksharing. CollaborateCom 2009, pp. 1–10. 11–14 Nov. 2009. doi:10.4108/ICST.COLLABORATECOM2009.8271
14. Sun, C., Jia, X., Zhang, Y., Yang, Y., Chen, D.: Achieving convergence, causality preservation, and intention preservation in real-time cooperative editing systems. ACM Trans. Comput. Hum. Interact. **5**(1): 63–108 (1998)

Springer

http://www.springer.com/978-81-322-2636-9

Proceedings of the International Conference on Recent Cognizance in Wireless Communication & Image Processing

ICRCWIP-2014

Afzalpulkar, N.; Srivastava, V.; Singh, G.; Bhatnagar, D. (Eds.)

2016, XXIX, 1002 p. 600 illus., 410 illus. in color.,
Hardcover
ISBN: 978-81-322-2636-9