# Operational transformation

**Operational transformation** (**OT**) is a technology for supporting a range of collaboration functionalities in advanced collaborative software systems. OT was originally invented for consistency maintenance and concurrency control in collaborative editing of plain text documents. Its capabilities have been extended and its applications expanded to include group undo, locking, conflict resolution, operation notification and compression, group-awareness, HTML/XML and tree-structured document editing, collaborative office productivity tools, application-sharing, and collaborative computer-aided media design tools.[1] In 2009 OT was adopted as a core technique behind the collaboration features in Apache Wave and Google Docs.

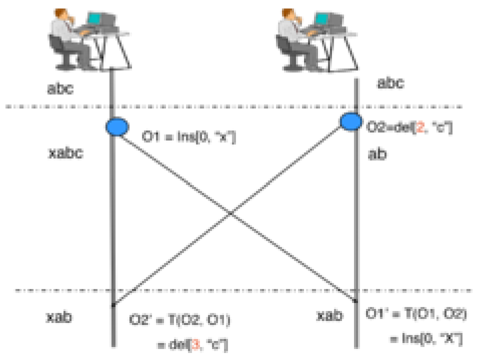
## History

Operational Transformation was pioneered by C. Ellis and S. Gibbs[2] in the GROVE (GRoup Outline Viewing Edit) system in 1989. Several years later, some correctness issues were identified and several approaches[3][4][5][6] were independently proposed to solve these issues, which was followed by another decade of continuous efforts of extending and improving OT by a community of dedicated researchers. In 1998, a Special Interest Group on Collaborative Editing[7] was set up to promote communication and collaboration among CE and OT researchers. Since then, SIGCE holds annual CE workshops in conjunction with major CSCW (Computer Supported Cooperative Work) conferences, such as ACM, CSCW, GROUP and ECSCW.

## System architecture

Collaboration systems utilizing Operational Transformations typically use replicated document storage, where each client has their own copy of the document; clients operate on their local copies in a lock-free, non-blocking manner, and the changes are then propagated to the rest of the clients; this ensures the client high responsiveness in an otherwise high-latency environment such as the Internet. When a client receives the changes propagated from another client, it typically transforms the changes before executing them; the transformation ensures that application-dependent consistency criteria (invariants) are maintained by all sites. This mode of operation results in a system particularly suited for implementing collaboration features, like simultaneous document editing, in a high-latency environment such as the web.

## Basics



The basic idea of OT can be illustrated by using a simple text editing scenario as follows. Given a text document with a string "abc" replicated at two collaborating sites; and two concurrent operations:

1. O1 = Insert[0, "x"] (to insert character "x" at position "0")
2. O2 = Delete[2, "c"] (to delete the character "c" at position "2")

generated by two users at collaborating sites 1 and 2, respectively. Suppose the two operations are executed in the order of O1 and O2 (at site 1). After executing O1, the document becomes "xabc". To execute O2 after O1, O2 must be transformed against O1 to become: O2' = Delete[3, "c"], whose positional parameter is incremented by one due to the insertion of one character "x" by O1. Executing O2' on "xabc" deletes the correct character "c" and the document becomes "xab". However, if O2 is executed without transformation, it incorrectly deletes character "b" rather than "c". The basic idea of OT is to transform (or adjust) the parameters of an editing operation according to the effects of previously executed concurrent operations so that the transformed operation can achieve the correct effect and maintain document consistency.

## Consistency models

One functionality of OT is to support consistency maintenance in collaborative editing systems. A number of consistency models have been proposed in the research community, some generally for collaborative editing systems, and some specifically for OT algorithms.

### The CC model

In Ellis and Gibbs 1989 paper "Concurrency control in groupware systems",[2] two consistency properties are required for collaborative editing systems:

- **C**ausality preservation: ensures the execution order of causally dependent operations be the same as their natural cause-effect order during the process of collaboration. The causal relationship between two operations is defined formally by Lamport's "happened-before" relation. When two operations are not causally dependent, they are concurrent. Two concurrent operations can be executed in different order on two different document copies.
- **C**onvergence: ensures the replicated copies of the shared document be identical at all sites at quiescence (i.e., all generated operations have been executed at all sites).

Since concurrent operations may be executed in different orders and editing operations are not commutative in general, copies of the document at different sites may diverge (inconsistent). The first OT algorithm was proposed in Ellis and Gibbs's paper[2] to achieve convergence in a group text editor; the state-vector (or vector clock in classic distributed computing) was used to preserve the precedence property.

## The CCI model

The CCI model was proposed as a consistency management in collaborative editing systems.[4][8] Under the CCI model, three consistency properties are grouped together:

- **C**ausality preservation : the same as in the CC model.
- **C**onvergence: the same as in the CC model.
- **I**ntention preservation: ensures that the effect of executing an operation on any document state be the same as the intention of the operation. The intention of an operation O is defined as the execution effect which can be achieved by applying O on the document state from which O was generated.

The CCI model extends the CC model with a new criterion: intention preservation. The essential difference between convergence and intention preservation is that the former can always be achieved by a serialization protocol, but the latter may not be achieved by any serialization protocol if operations were always executed in their original forms. Achieving the nonserialisable intention preservation property has been a major technical challenge. OT has been found particularly suitable for achieving convergence and intention preservation in collaborative editing systems.

The CCI model is independent of document types or data models, operation types, or supporting techniques (OT, multi-versioning, serialization, undo/redo). It was not intended for correctness verification for techniques (e.g. OT) that are designed for specific data and operation models and for specific applications. In,[4] the notion of intention preservation was defined and refined at three levels: First, it was defined as a generic consistency requirement for collaborative editing systems; Second, it was defined as operation context-based pre- and post- transformation conditions for generic OT functions; Third, it was defined as specific operation verification criteria to guide the design of OT functions for two primitive operations: string-wise insert and delete, in collaborative plain text editors.

## The CSM model

The condition of intention preservation was not formally specified in the CCI model for purposes of formal proofs. The SDT[9] and LBT[10] approaches try to formalize an alternative conditions that can be proved. The consistency model proposed in these two approaches consist of the following formal conditions:

- ***C**ausality*: the same definition as in CC model
- ***S**ingle-operation effects*: the effect of executing any operation in any execution state achieves the same effect as in its generation state
- ***M**ulti-operation effects*: the effects relation of any two operations is maintained after they are both executed in any states

### The CA model

The above CSM model requires that a total order of all objects in the system be specified. Effectively, the specification is reduced to new objects introduced by insert operations. However, specification of the total order entails application-specific policies such as those to break insertion ties (i.e., new objects inserted by two current operations at the same position). Consequently, the total order becomes application specific. Moreover, in the algorithm, the total order must be maintained in the transformation functions and control procedure, which increases time/space complexities of the algorithm.

Alternatively, the **CA** model is based on the **admissibility theory**.[11] The CA model includes two aspects:

- ***C**ausality*: the same definition as in CC model
- ***A**dmissibility*: The invocation of every operation is admissible in its execution state, i.e., every invocation must not violate any effects relation (object ordering) that has been established by earlier invocations.

These two conditions imply convergence. All cooperating sites converge in a state in which there is a same set of objects that are in the same order. Moreover, the ordering is effectively determined by the effects of the operations when they are generated. Since the two conditions also impose additional constraints on object ordering, they are actually stronger than convergence. The CA model and the design/prove approach are elaborated in the 2005 paper.[11] It no longer requires that a total order of objects be specified in the consistency model and maintained in the algorithm, which hence results in reduced time/space complexities in the algorithm.

## OT system structure

OT is a system of multiple components. One established strategy of designing OT systems[2][3][4][5][12][13] is to separate the high-level transformation control (or integration) algorithms from the low-level transformation functions.

|   |
|---|
| **OT Control Algorithms**<br>(determine which operations are transformed against others according to their concurrency/context relations) |
| **OT properties and conditions**<br>(divide responsibilities between algorithms and functions) |
| **OT Transformation Functions**<br>(determine how to transform a pair of primitive operations according to operation types, positions, and other parameters) |

The transformation control algorithm is concerned with determining:

1. Which operation should be transformed against a causally ready new operation
2. The order of the transformations

The control algorithm invokes a corresponding set of transformation functions, which determine how to transform one operation against another according to the operation types, positions, and other parameters. The correctness responsibilities of these two layers are formally specified by a set of transformation properties and conditions. Different OT systems with different control algorithms, functions, and communication topologies require maintaining different sets of transformation properties. The separation of an OT system into these two layers allows for the design of generic control algorithms that are applicable to different kinds of application with different data and operation models.

The other alternative approach was proposed in.[11] In their approach, an OT algorithm is correct if it satisfies two formalized correctness criteria:

1. Causality preservation
2. Admissibility preservation

As long as these two criteria are satisfied, the data replicas converge (with additional constraints) after all operations are executed at all sites. There is no need to enforce a total order of execution for the sake of achieving convergence. Their approach is generally to first identify and prove sufficient conditions for a few transformation functions, and then design a control procedure to ensure those sufficient conditions. This way the control procedure and transformation functions work synergistically to achieve correctness, i.e., causality and admissibility preservation. In their approach, there is no need to satisfy transformation properties such as TP2 because it does not require that the (inclusive) transformation functions work in all possible cases.

## OT data and operation models

There exist two underlying models in each OT system: the data model that defines the way data objects in a document are addressed by operations, and the operation model that defines the set of operations that can be directly transformed by OT functions. Different OT systems may have different data and operation models. For example, the data model of the first OT system[2] is a single linear address space; and its operation model consists of two primitive operations: character-wise insert and delete. The basic operation model has been extended to include a third primitive operation update to support collaborative Word document processing[14] and 3D model editing.[15] The basic OT data model has been extended into a hierarchy of multiple linear addressing domains,[16][17][18] which is capable of modeling a broad range of documents. A data adaptation process is often required to map application-specific data models to an OT-compliant data model.[19][20]

There exist two approaches to supporting application level operations in an OT system:

1. Generic operation model approach: which is to devise transformation functions for three primitive operations: insert, delete, and update.[19] This approach needs an operation adaptation process to map application operations to these primitive operations. In this approach, the OT operation model is generic, so transformation functions can be reused for different applications.
2. Application-specific operation model approach: which is to devise transformation functions for each pair of application operations.[20][21] For an application with m different operations, m x m transformation functions are needed for supporting this application. In this approach, transformation functions are application-specific and cannot be reused in different applications.

## OT functions

Various OT functions have been designed for OT systems with different capabilities and used for different applications. OT functions used in different OT systems may be named differently, but they can be classified into two categories:

- Inclusion transformation (or forward transformation): IT(Oa, Ob) or $T(op_1, op_2)$, which transforms operation Oa against another operation Ob in such a way that the impact of Ob is effectively included.
- Exclusion transformation (or backward transformation): ET (Oa, Ob) or $T^{-1}(op_1, op_2)$, which transforms operation Oa against another operation Ob in such a way that the impact of Ob is effectively excluded.

For example, suppose a type String with an operation ins(p, c, sid) where *p* is the position of insertion, *c* is the character to insert and *sid* is the identifier of the site that has generated the operation. We can write the following transformation function:

```
T(ins($p_1$, $c_1$, $sid_1$), ins($p_2$, $c_2$, $sid_2$)) :-
 if ($p_1 < p_2$) return ins($p_1$, $c_1$, $sid_1$)
 else if ($p_1 = p_2$ and $sid_1 < sid_2$) return ins($p_1$, $c_1$, $sid_1$)
 else return ins($p_1 + 1$, $c_1$, $sid_1$)
```

```
$T^{-1}$(ins($p_1$, $c_1$, $sid_1$), ins($p_2$, $c_2$, $sid_2$)) :-
 if ($p_1 < p_2$) return ins($p_1$, $c_1$, $sid_1$)
 else if ($p_1 = p_2$ and $sid_1 < sid_2$) return ins($p_1$, $c_1$, $sid_1$)
 else return ins($p_1 - 1$, $c_1$, $sid_1$)
```

Some OT systems use both IT and ET functions, and some use only IT functions. The complexity of OT function design is determined by various factors:

- the functionality of the OT system: whether the OT system supports do (consistency maintenance), undo, locking,[22] awareness, application sharing,[19][23][24][25] etc.;
- the correctness responsibility in the OT system: what transformation properties (CP1/TP1, CP2/TP2, IP2, IP3, RP) to meet; whether ET is used;
- the operation model of the OT system: whether the OT operation model is generic (e.g. primitive insert, delete, update), or application-specific (all operations of the target application); and
- the data model of the OT system: whether the data in each operation is character-wise (an individual object), string-wise (a sequence of objects), hierarchical, or other structures.

## Transformation properties

Various transformation properties for ensuring OT system correctness have been identified. These properties can be maintained by either the transformation control algorithm[4][5][13][20][26][27] or by the transformation functions.[28] Different OT system designs have different division of responsibilities among these components. The specifications of these properties and preconditions of requiring them are given below.

### Convergence properties

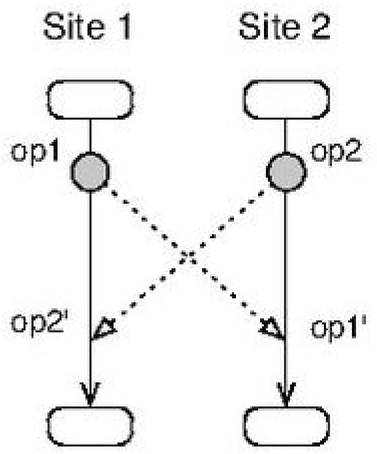


Illustration of the TP1 property

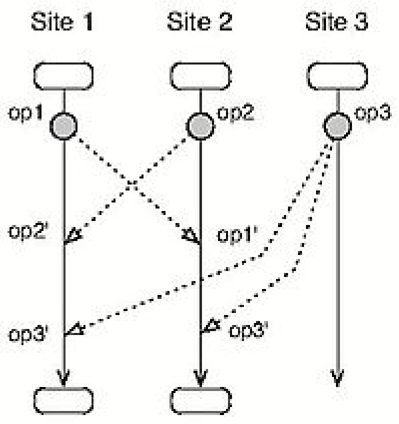


Illustration of the TP2 property

The following two properties are related to achieving convergence.

- **CP1/TP1**: For every pair of concurrent operations $op_1$ and $op_2$ defined on the same state, the transformation function T satisfies CP1/TP1 property if and only if: $op_1 \circ T(op_2, op_1) \equiv op_2 \circ T(op_1, op_2)$ where $op_i \circ op_j$ denotes the sequence of operations containing $op_i$ followed by $op_j$;and where $\equiv$ denotes equivalence of the two sequences of operations. **CP1/TP1 precondition**: CP1/TP1 is required only if the OT system allows any two operations to be executed in different orders.
- **CP2/TP2**: For every three concurrent operations $op_1, op_2$ and $op_3$ defined on the same document state, the transformation function T satisfies CP2/TP2 property if and only if: $T(op_3, op_1 \circ T(op_2, op_1)) = T(op_3, op_2 \circ T(op_1, op_2))$. CP2/TP2 stipulates equality between two operations transformed with regard to two equivalent sequences of operations: the transformation of $op_3$ against the sequence of operation $op_2$ followed by $T(op_1, op_2)$ must give the same operation as the transformation of $op_3$ against the sequence formed by $op_1$ and $T(op_2, op_1)$. **CP2/TP2 precondition**: CP2/TP2 is required only if the OT systems allows two operations $op_1$ and $op_2$ be IT-transformed in two different document states (or contexts).

## Inverse properties

The following three properties are related to achieving the desired group undo effect. They are:

- **IP1**: Given any document state S and the sequence $op \circ \overline{op}$, we have $S \circ op \circ \overline{op} = S$, which means the sequence $op \circ \overline{op}$ is equivalent to a single identity operation I with respect to the effect on the document state. This property is required in an OT system for achieving the correct undo effect, but is not related to IT functions.
- **IP2**: The property IP2 expresses that the sequence $op \circ \overline{op}$ has no effect on the transformation of other operations. The transformation functions satisfy IP2 if and only if: $T(op_x, op \circ \overline{op}) = op_x$, which means that the outcome of transforming $op_x$ against the sequence $op \circ \overline{op}$ is equivalent to the outcome of transforming $op_x$ against the identity operation I. **IP2 precondition**: IP2 is required only if the OT systems allows an operation $op_x$ to be transformed against a pair of do and undo operations $op \circ \overline{op}$, one-by-one.
- **IP3**: Given two concurrent operations $op_1$ and $op_2$ defined on the same document state (or context), if $\overline{op_1}' = T(\overline{op_1}, T(op_2, op_1))$ and $\overline{op_1'} = \overline{T(op_1, op_2)}$. The transformation functions satisfy the property IP3 if and only if $\overline{op_1}' = \overline{op_1'}$, which means that the transformed inverse operation $\overline{op_1}'$ is equal to the inverse of the transformed operation $\overline{op_1'}$. **IP3 precondition**: IP3 is required only if the OT system allows an inverse operation $\overline{op_1}$ to be transformed against an operation $op_2$ that is concurrent and defined on the same document state as (or context-equivalent to) $op_1$.

## OT control (integration) algorithms

Various OT control algorithms have been designed for OT systems with different capabilities and for different applications. The complexity of OT control algorithm design is determined by multiple factors. A key differentiating factor is whether an algorithm is capable of supporting concurrency control (do) and/or group undo.[3][8][12][27][29] In addition, different OT control algorithm designs make different tradeoffs in:

- assigning correctness responsibilities among the control algorithm and transformation functions, and
- time-space complexity of the OT system.

Most existing OT control algorithms for concurrency control adopt the theory of causality/concurrency as the theoretical basis: causally related operations must be executed in their causal order; concurrent operations must be transformed before their execution. However, it was well known that concurrency condition alone cannot capture all OT transformation conditions.[3][4][5][8][30] In a recent work, the theory of operation context has been proposed to explicitly represent the notion of a document state, which can be used to formally express OT transformation conditions for supporting the design and verification of OT control algorithms.[27]

The following table gives an overview of some existing OT control/integration algorithms

| OT control/integration algorithms (systems) | Required transformation function types | Support OT-based Do? | Support OT-based Undo? | Transformation properties supported by control algorithm | Transform propert supporte transform functio |
|---|---|---|---|---|---|
| dOPT[2] (GROVE) | T (IT) | Yes | No | None | CP1/TP1, CP2/TP2 |
| selective-undo[12] (DistEdit) | Transpose (IT and ET) | No | Selective Undo | NA | CP1/TP1, CP2/TP2, I IP1, IP2, IP |
| adOPTed[3][29] (JOINT EMACS) | LTransformation (IT) | Yes | Chronological Undo | IP2, IP3 | CP1/TP1, CP2/TP2, I |
| Jupiter[5] | xform (IT) | Yes | No | CP2/TP2 | CP1/TP1 |
| Google Wave OT[20] | transform and composition(IT) | Yes | No | CP2/TP2 | CP1/TP1 |
| GOT[4] (REDUCE) | IT and ET | Yes | No | CP1/TP1, CP2/TP2 | None |
| GOTO[6] (REDUCE, CoWord, CoPPT, CoMaya) | IT and ET | Yes | No | None | CP1/TP1, CP2/TP2 |
| AnyUndo[8] (REDUCE, CoWord, CoPPT, CoMaya) | IT and ET | No | Undo any operation | IP2, IP3, RP | IP1, CP1/T CP2/TP2 |
| SCOP[26] (NICE) | IT | Yes | No | CP2/TP2 | CP1/TP1 |
| COT [27] (REDUCE, | IT | Yes | Undo any | CP2/TP2, IP2, | CP1/TP1, ( |

| | | | | | |
|---|---|---|---|---|---|
| CoWord, CoPPT, CoMaya) | | | operation | IP3 | therefore r necessary) |
| TIBOT [31] | IT | Yes | No | CP2/TP2 | CP1/TP1 |
| SOCT4[13] | Forward transformation (IT) | Yes | No | CP2/TP2 | CP1/TP1 |
| SOCT2[30] | Forward Transformation(IT) and Backward Transformation(ET) | Yes | No | None | CP1/TP1, CP2/TP2, I |
| MOT2[32] | Forward transformation (IT) | Yes | No | ?? | CP1/TP1, CP2/TP2 |

> A continuous total order is a strict total order where it is possible to detect a missing element i.e. 1,2,3,4,... is a continuous total order, 1,2,3,5,... is not a continuous total order.

The transformation-based algorithms proposed in [10][11] are based on the alternative consistency models "CSM" and "CA" as described above. Their approaches differ from those listed in the table. They use vector timestamps for causality preservation. The other correctness conditions are "single-"/"multi-" operation effects relation preservation or "admissibility" preservation. Those conditions are ensured by the control procedure and transformation functions synergistically. There is no need to discuss TP1/TP2 in their work. Hence they are not listed in the above table.

There exist some other optimistic consistency control algorithms that seek alternative ways to design transformation algorithms, but do not fit well with the above taxonomy and characterization. For example, Mark and Retrace[33]

The correctness problems of OT led to introduction of transformationless post-OT schemes, such as WOOT,[34] Logoot[35] and Causal Trees (CT).[36] "Post-OT" schemes decompose the document into atomic operations, but they workaround the need to transform operations by employing a combination of unique symbol identifiers, vector timestamps and/or tombstones.

## Critique of OT

While the classic OT approach of defining operations through their offsets in the text seems to be simple and natural, real-world distributed systems raise serious issues. Namely, that operations propagate with finite speed, states of participants are often different, thus the resulting combinations of states and operations are extremely hard to foresee and understand. As Li and Li put it, "Due to the need to consider complicated case coverage, formal proofs are very complicated and error-prone, even for OT algorithms that only treat two characterwise primitives (insert and delete)".[37]

Similarly, Joseph Gentle who is a former Google Wave engineer and an author of the Share.JS library wrote, "Unfortunately, implementing OT sucks. There's a million algorithms with different tradeoffs, mostly trapped in academic papers. [...] Wave took 2 years to write and if we rewrote it today, it would take almost as long to write a second time."[38] But later he amends his comment with "I no longer believe that wave would take 2 years to implement now - mostly because of advances in web frameworks and web browsers." [39]

For OT to work, every single change to the data needs to be captured: "Obtaining a snapshot of the state is usually trivial, but capturing edits is a different matter altogether. [...] The richness of modern user interfaces can make this problematic, especially within a browser-based environment." An alternative to OT is differential synchronization.[40]

Another alternative to OT is using sequence types of conflict-free replicated data type.

## See also

- Data synchronization
- Collaborative real-time editors
- Conflict-free replicated data types
- Consistency models
- Optimistic replication

## References

1. Sun, Chengzheng. "OT FAQ" .
2. Ellis, C.A.; Gibbs, S.J. (1989). "Concurrency control in groupware systems". *ACM SIGMOD Record*. **18** (2): 399–407. CiteSeerX 10.1.1.465.2026 . doi:10.1145/67544.66963 .
3. Ressel, Matthias and Nitsche-Ruhland, Doris and Gunzenhäuser, Rul (1996). "An integrating, transformation-oriented approach to concurrency control and undo in group editors". *CSCW '96: Proceedings of the 1996 ACM conference on Computer supported cooperative work*. pp. 288–297. doi:10.1145/240080.240305 .
4. Chengzheng Sun; Xiaohua Jia; Yanchun Zhang; Yun Yang; David Chen (1998). "Achieving convergence, causality preservation, and intention preservation in real-time cooperative editing systems". *ACM Trans. Comput.-Hum. Interact*. **5** (1): 63–108. CiteSeerX 10.1.1.56.1251 . doi:10.1145/274444.274447 .

5. Nichols, D.A.; Curtis, P.; Dixon, M.; Lamping, J. (1995). "High-latency, low-bandwidth windowing in the Jupiter collaboration system" . *Proceedings of the 8th Annual ACM Symposium on User Interface and Software Technology*: 111–120. Archived from the original on 2015-11-30. Retrieved 2009-09-27.

6. Sun, C.; Ellis, C. (1998). "Operational transformation in real-time group editors: issues, algorithms, and achievements" . *Proceedings of the 1998 ACM conference on Computer supported cooperative work*. ACM Press New York, NY, USA. pp. 59–68.

7. "SIGCE - An International Special Interest Group on Collaborative Editing" . *cooffice.ntu.edu.sg*. Archived from the original on 2012-12-24. Retrieved 2020-01-10.

8. C. Sun (2002). "Undo as concurrent inverse in group editors" . *ACM Trans. Comput.-Hum. Interact*. **9** (4): 309–361. doi:10.1145/586081.586085 .

9. Du Li; Rui Li (2004). "Preserving Operation Effects Relation in Group Editors". *Proceedings of the ACM CSCW'04 Conference on Computer-Supported Cooperative Work*. ACM Press New York, NY, USA. pp. 457–466.

10. Rui Li; Du Li (2007). "A New Operational Transformation Framework for Real-Time Group Editors" . *IEEE Transactions on Parallel and Distributed Systems*. **18** (3): 307–319. doi:10.1109/TPDS.2007.35 .

11. Rui Li; Du Li (2005). "Commutativity-Based Concurrency Control in Groupware". *Proceedings of the First IEEE Conference on Collaborative Computing: Networking, Applications and Worksharing (CollaborateCom'05)*.

12. Prakash, Atul & Knister, Michael J. (1994). "A framework for undoing actions in collaborative systems". *ACM Trans. Comput.-Hum. Interact*. **1** (4): 295–330. CiteSeerX 10.1.1.51.4793 . doi:10.1145/198425.198427 .

13. Vidot, N.; Cart, M.; Ferrie, J.; Suleiman, M. (2000). "Copies convergence in a distributed real-time collaborative environment" (PDF). *Proceedings of the 2000 ACM conference on Computer supported cooperative work*. ACM Press New York, NY, USA. pp. 171–180. Archived from the original (PDF) on 2004-10-12.

14. D. Sun and S. Xia and C. Sun and D. Chen (2004). "Operational transformation for collaborative word processing". *Proc. of the ACM Conf. on Computer-Supported Cooperative Work*. pp. 437–446.

15. Agustina and F. Liu and S. Xia and H. Shen and C. Sun (November 2008). "CoMaya: Incorporating advanced collaboration capabilities into {3D} digital media design tools". *Proc. of ACM Conf. on Computer-Supported Cooperative Work*. pp. 5–8.

16. Davis, Aguido Horatio and Sun, Chengzheng and Lu, Junwei (2002). "Generalizing operational transformation to the standard general markup language". *CSCW '02: Proceedings of the 2002 ACM conference on Computer supported cooperative work*. New Orleans, Louisiana, USA. pp. 58–67. doi:10.1145/587078.587088 .

17. Claudia-Lavinia Ignat; Moira C. Norrie (2003). "Customizable collaborative editor relying on treeOPT algorithm". *ECSCW'03: Proceedings of the eighth conference on European Conference on Computer Supported Cooperative Work*. Kluwer Academic Publishers. pp. 315–334. doi:10.1007/978-94-010-0068-0_17 .

18. Claudia-Lavinia Ignat; Moira C. Norrie (2008). "Multi-level Editing of Hierarchical Documents". *Computer Supported Cooperative Work (CSCW)*. **17** (5–6): 423–468. doi:10.1007/s10606-007-9071-2 .

19. C.Sun, S.Xia, D.Sun, D.Chen, H.Shen, and W.Cai (2006). "Transparent adaptation of single-user applications for multi-user real-time collaboration" . *ACM Trans. Comput.-Hum. Interact*. **13** (4): 531–582. doi:10.1145/1188816.1188821 .

20. "Google Wave Operational Transform" . Archived from the original on 2009-05-31. Retrieved 2009-05-29.

21. Christopher R. Palmer; Gordon V. Cormack (1998). "Operation transforms for a distributed shared spreadsheet". *CSCW '98: Proceedings of the 1998 ACM conference on Computer supported cooperative work*. ACM Press. pp. 69–78. doi:10.1145/289444.289474 .

22. C. Sun & R. Sosic (1999). "Optional Locking Integrated with Operational Transformation in Distributed Real-Time Group Editors". *In Proc. of the 18th ACM Symposium on Principles of Distributed Computing*. pp. 43–52.

23. Begole, James and Rosson, Mary Beth and Shaffer, Clifford A. (1999). "Flexible collaboration transparency: supporting worker independence in replicated application-sharing systems". *ACM Trans. Comput.-Hum. Interact*. **6** (2): 95–132. CiteSeerX 10.1.1.23.1185 . doi:10.1145/319091.319096 .

24. Li, Du & Li, Rui (2002). "Transparent sharing and interoperation of heterogeneous single-user applications". *CSCW '02: Proceedings of the 2002 ACM conference on Computer supported cooperative work*. New Orleans, USA. pp. 246–255.

25. Li, Du & Lu, Jiajun (2006). "A lightweight approach to transparent sharing of familiar single-user editors". *CSCW '06: Proceedings of the 2006 20th anniversary conference on Computer supported cooperative work*. Banff, Alberta, Canada. pp. 139–148. doi:10.1145/1180875.1180896 .

26. Shen, Haifeng & Sun, Chengzheng (2002). "Flexible notification for collaborative systems". *CSCW '02: Proceedings of the 2002 ACM conference on Computer supported cooperative work*. pp. 77–86. doi:10.1145/587078.587090 .

27. D. Sun & C. Sun (2009). "Context-based Operational Transformation for Distributed Collaborative Editing Systems". *IEEE Transactions on Parallel and Distributed Systems*. **20** (10): 1454–1470. doi:10.1109/TPDS.2008.240 .

28. Gerald Oster; Pascal Molli; Pascal Urso; Abdessamad Imine (2006). "Tombstone Transformation Functions for Ensuring Consistency in Collaborative Editing Systems" (PDF). *Procs. 2nd Intl. Conf. On Collaborative Computing: Networking, Appln. And Worksharing*. Retrieved 2007-07-26.

29. M. Ressel & R. Gunzenhauser (1999). "Reducing the Problems of Group Undo". *Proc. of the ACM Conf. on Supporting Group Work*. pp. 131–139.

30. Suleiman, M.; Cart, M.; Ferrié, J. (1998). "Concurrent Operations in a Distributed and Mobile Collaborative Environment". *Proceedings of the Fourteenth International Conference on Data Engineering, February*. pp. 23–27. doi:10.1109/ICDE.1998.655755 .

31. R. Li, D. Li & C. Sun (2004). "A Time Interval Based Consistency Control Algorithm for Interactive Groupware Applications". *ICPADS '04: Proceedings of the Parallel and Distributed Systems, Tenth International Conference*. p. 429. doi:10.1109/ICPADS.2004.12 .

32. M. Cart, Jean Ferrié (2007). "Synchronizer Based on Operational Transformation for P2P Environments" (PDF). *Proceedings of the 3rd International Conference on Collaborative Computing: Networking, Applications and Worksharing*. pp. 127–138. Retrieved 2007-07-26.

33. Gu, Ning and Yang, Jiangming and Zhang, Qiwei (2005). "Consistency maintenance based on the mark \& retrace technique in groupware systems". *GROUP '05: Proceedings of the 2005 international ACM SIGGROUP conference on Supporting group work*. pp. 264–273. doi:10.1145/1099203.1099250 .

34. Imine, Abdessamad and Molli, Pascal and Oster, Gerald and Urso, Pascal (2005). "Real time group editors without Operational transformation" : 24.

35. Stephane Weiss and Pascal Urso and Pascal Molli (2010). "Logoot-Undo: Distributed Collaborative Editing System on P2P Networks" . *IEEE Transactions on Parallel and Distributed Systems*. **21** (8): 1162. doi:10.1109/TPDS.2009.173 .

36. Victor Grishchenko (2010). "Deep Hypertext with embedded revision control implemented in regular expressions" (PDF). *The Proceedings of the 6th International Symposium on Wikis and Open Collaboration (WikiSym '10)*. WikiSym 2010 . Archived from the original (PDF) on 2012-03-09. Retrieved 2010-06-30.

37. Du Li & Rui Li (2010). "An Admissibility-Based Operational Transformation Framework for Collaborative Editing Systems" . *Computer Supported Cooperative Work (CSCW)*. **19** (1): 1–43. doi:10.1007/s10606-009-9103-1 .

38. "ShareJS" . 2011-11-06. Archived from the original on 2012-05-11. Retrieved 2013-08-16.

39. "Yes thats me! For what its worth, I no longer believe that wave would take 2 yea... | Hacker News" . *news.ycombinator.com*. Retrieved 2019-02-13.

40. Neil Fraser (January 2009). "Differential Synchronization" .

## External links

- OT FAQ: Operational Transformation Frequently Asked Questions and Answers
- SIGCE: Special Interest Group of Collaborative Editing
- International Workshop on Collaborative Editing Systems
- Distributed System Online - Collaborative editing
- Simple explanation of OT in Google Docs

- Basics of OT in the Open Coweb Framework

### Relevant online talks

- Google Wave: Live collaborative editing
- Google Tech Talk: Issues and Experiences in Designing Real-time Collaborative Editing Systems
- Microsoft Research talk: Consistency maintenance in real-time collaborative editing systems