Specification and Implementation of Replicated List

— The Jupiter Protocol Revisited

(OPODIS'2018)

Hengfeng Wei, Yu Huang, Jian Lu

Nanjing University

December 17, 2018

The Main Contribution

The Jupiter protocol [Nichols et al., 1995][a] for replicated list satisfies the weak list specification [Attiya et al., 2016][b].

[a] David A. Nichols et al. (1995). "High-latency, Low-bandwidth Windowing in the Jupiter Collaboration System". In: *Proceedings of the 8th Annual ACM Symposium on User Interface and Software Technology*. UIST '95. ACM, pp. 111–120.

[b] Hagit Attiya et al. (2016). "Specification and complexity of collaborative text editing". In: *Proceedings of the 2016 ACM Symposium on Principles of Distributed Computing*. PODC '16. ACM, pp. 259–268.

The Main Contribution

The Jupiter protocol [Nichols et al., 1995][a] for replicated list satisfies the weak list specification [Attiya et al., 2016][b].

[a] David A. Nichols et al. (1995). “High-latency, Low-bandwidth Windowing in the Jupiter Collaboration System”. In: *Proceedings of the 8th Annual ACM Symposium on User Interface and Software Technology*. UIST '95. ACM, pp. 111–120.

[b] Hagit Attiya et al. (2016). “Specification and complexity of collaborative text editing”. In: *Proceedings of the 2016 ACM Symposium on Principles of Distributed Computing*. PODC '16. ACM, pp. 259–268.

This was proposed as a *conjecture* in a PODC paper [Attiya et al., 2016].

1. Why do we care about replicated list?

1. Why do we care about replicated list?
2. What is the weak list specification?

1. Why do we care about replicated list?
2. What is the weak list specification?
3. How does the Jupiter protocol work?

1. Why do we care about replicated list?
2. What is the weak list specification?
3. How does the Jupiter protocol work?
4. How to prove that Jupiter satisfies the weak list specification?

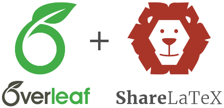
Replicated List

Replicated Collaborative Text Editing Systems

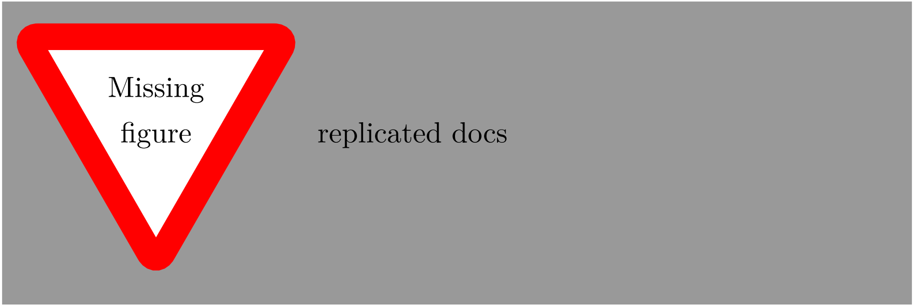
(a) Google Docs

(b) Apache Wave

(c) Wikipedia



(d) LaTeX Editor



Missing figure

replicated docs

Replicas are required to respond to user operations immediately.

Updates are propagated to other replicas asynchronously.

Replicated list object: to model the core functionality

$\textsc{Ins}(a, p)$: Insert a at position p.

$\textsc{Del}(p)$: Delete the element at position p.

$\textsc{Read}$: Return the list.

Weak List Specification

Definition (Eventual Convergence [Ellis and Gibbs, 1989])

The lists are identical at all replicas at quiescence,
i.e., all update operations have been executed at all replicas.

Definition (Eventual Convergence [Ellis and Gibbs, 1989])

The lists are identical at all replicas at quiescence,
i.e., all update operations have been executed at all replicas.

Definition (Strong Eventual Consistency [Shapiro et al., 2011])

The lists are identical at the replicas whenever they have executed the same set of update operations.

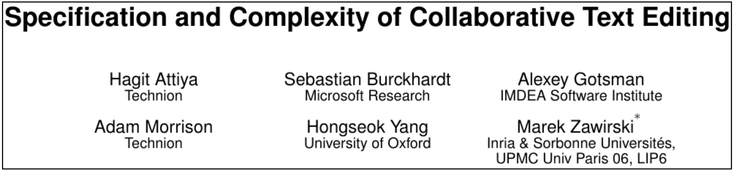
Definition (Eventual Convergence [Ellis and Gibbs, 1989])

The lists are identical at all replicas at quiescence,
i.e., all update operations have been executed at all replicas.

Definition (Strong Eventual Consistency [Shapiro et al., 2011])

The lists are identical at the replicas whenever they have executed the same set of update operations.

Specify little on *intermediate states* going through by replicas.



Specification and Complexity of Collaborative Text Editing

Hagit Attiya
Technion

Sebastian Burckhardt
Microsoft Research

Alexey Gotsman
IMDEA Software Institute

Adam Morrison
Technion

Hongseok Yang
University of Oxford

Marek Zawirski*
Inria & Sorbonne Universités,
UPMC Univ Paris 06, LIP6

Definition (Weak List Specification $\mathcal{A}_{\text{weak}}$ [Attiya et al., 2016])

Informally, $\mathcal{A}_{\text{weak}}$ requires the ordering between elements that are not deleted to be consistent across the system.

Specify a global property *on all states* across the system.

We show that $\mathcal{A}_{\text{weak}}$ can be rephrased as

Definition (Pairwise State Compatibility Property)

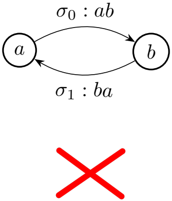
For any pair of list states, there cannot be two elements a and b such that a precedes b in one state but b precedes a in the other.

We show that $\mathcal{A}_{\text{weak}}$ can be rephrased as

Definition (Pairwise State Compatibility Property)

For any pair of list states, there cannot be two elements a and b such that a precedes b in one state but b precedes a in the other.

$\boxed{\sigma_0 : ab}$ $\boxed{\sigma_1 : ba}$

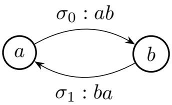


We show that $\mathcal{A}_{\text{weak}}$ can be rephrased as

Definition (Pairwise State Compatibility Property)

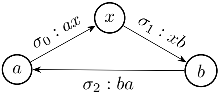
For any pair of list states, there cannot be two elements a and b such that a precedes b in one state but b precedes a in the other.

$\sigma_0 : ab$ $\sigma_1 : ba$



✗

$\sigma_0 : ax$ $\sigma_1 : xb$ $\sigma_2 : ba$

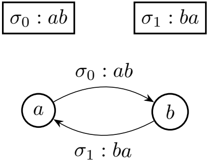


✓

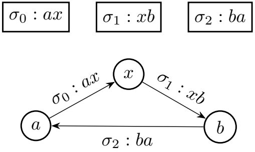
We show that $\mathcal{A}_{\text{weak}}$ can be rephrased as

Definition (Pairwise State Compatibility Property)

For any pair of list states, there cannot be two elements a and b such that a precedes b in one state but b precedes a in the other.



✗



✓

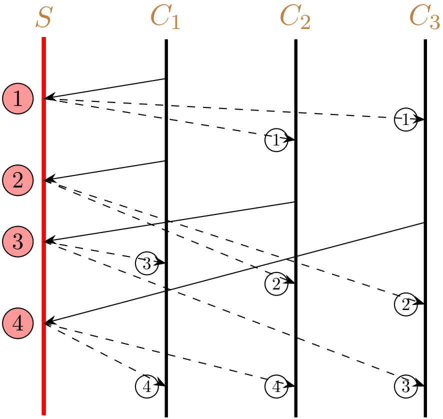
Prohibited by "Strong List Specification"

Jupiter

Jupiter adopts the client-server architecture [Nichols et al., 1995]:

Jupiter adopts the client-server architecture [Nichols et al., 1995]:

$(n+1)$ replicas $\triangleq$ (n) Client + (1) Server



$$\text{User} \xrightarrow[\text{Respond Immed.}]{\text{Issue Ops}} \text{Client} \xrightarrow[\text{Updates}]{\text{FIFO}} \text{Server} \xrightarrow[\text{Updates}]{\text{FIFO}} \text{Other Clients}$$

Challenge: Conflicts caused by concurrent operations

Challenge: Conflicts caused by concurrent operations

Missing figure

conflicts

A simple yet infeasible solution:

A simple yet infeasible solution:

(1) Replicas respond to local user operations immediately as required.

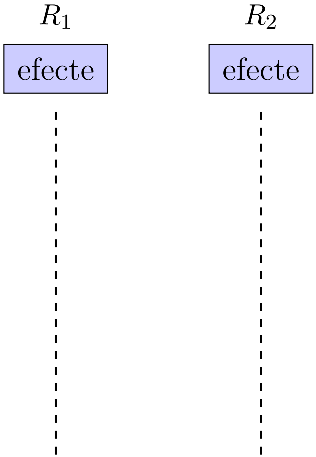
(2) Updates are propogated to the server.

(3) All updates are totally ordered at the server.

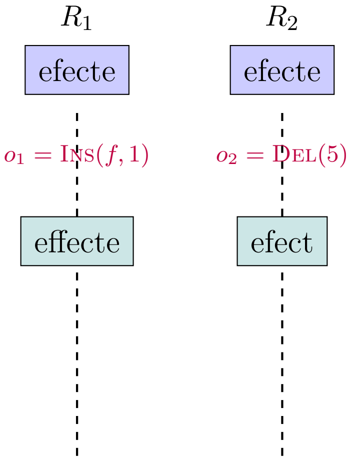
(4) Replicas are notified to synchronize with the server.

Operational Transformation (OT) [Ellis and Gibbs, 1989]

Operational Transformation (OT) [Ellis and Gibbs, 1989]



Operational Transformation (OT) [Ellis and Gibbs, 1989]



R_1

efecte

$o_1 = \textsc{Ins}(f, 1)$

effecte

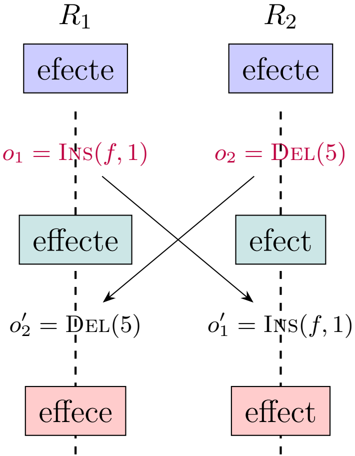
R_2

efecte

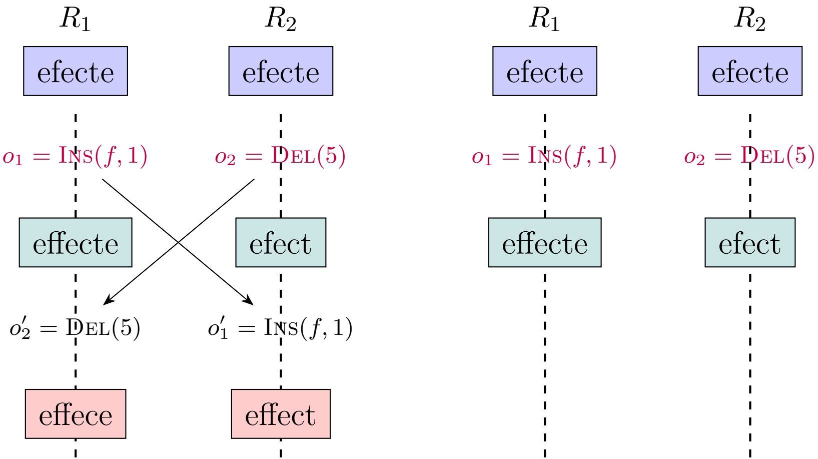
$o_2 = \textsc{Del}(5)$

efect

Operational Transformation (OT) [Ellis and Gibbs, 1989]



Operational Transformation (OT) [Ellis and Gibbs, 1989]



R_1 R_2 R_1 R_2

efecte efecte efecte efecte

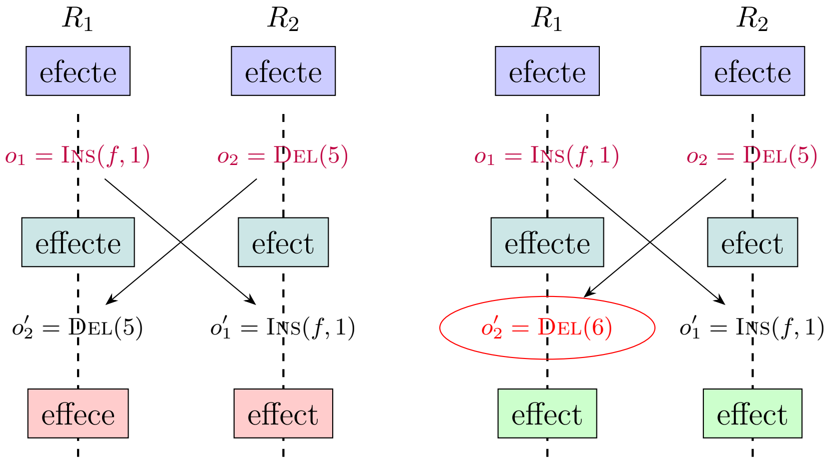
$o_1 = \textsc{Ins}(f, 1)$ $o_2 = \textsc{Del}(5)$ $o_1 = \textsc{Ins}(f, 1)$ $o_2 = \textsc{Del}(5)$

effecte efect effecte efect

$o_2' = \textsc{Del}(5)$ $o_1' = \textsc{Ins}(f, 1)$

effece effect

Operational Transformation (OT) [Ellis and Gibbs, 1989]



R_1 R_2 R_1 R_2

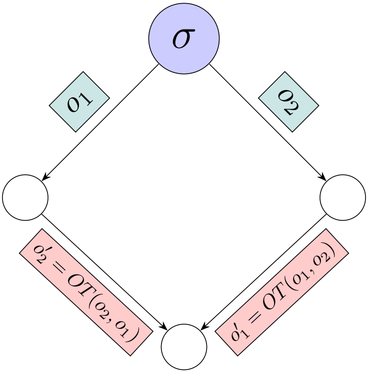
efecte efecte efecte efecte

$o_1 = \textsc{Ins}(f, 1)$ $o_2 = \textsc{Del}(5)$ $o_1 = \textsc{Ins}(f, 1)$ $o_2 = \textsc{Del}(5)$

effecte efect effecte efect

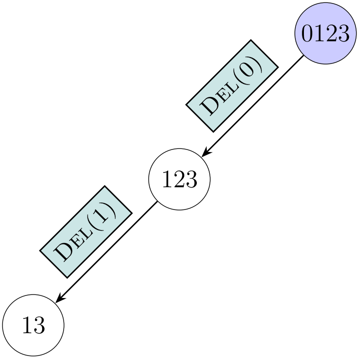
$o_2' = \textsc{Del}(5)$ $o_1' = \textsc{Ins}(f, 1)$ $o_2' = \textsc{Del}(6)$ $o_1' = \textsc{Ins}(f, 1)$

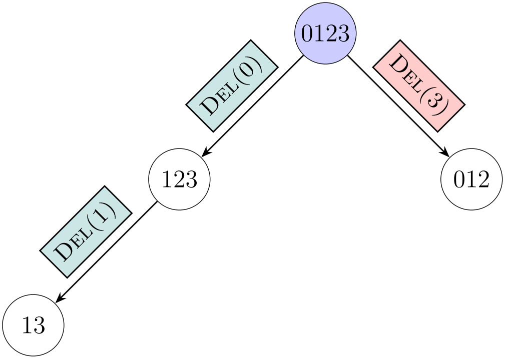
effece effect effect effect

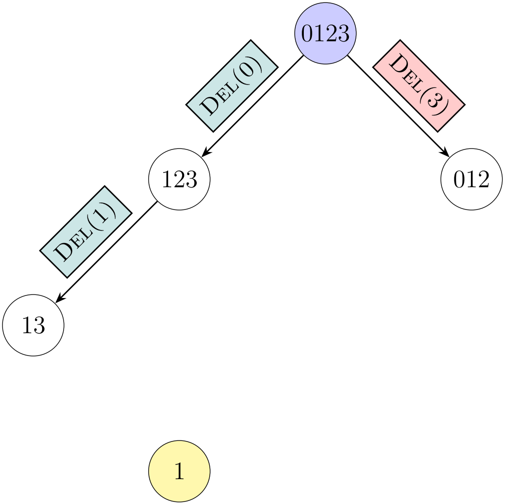


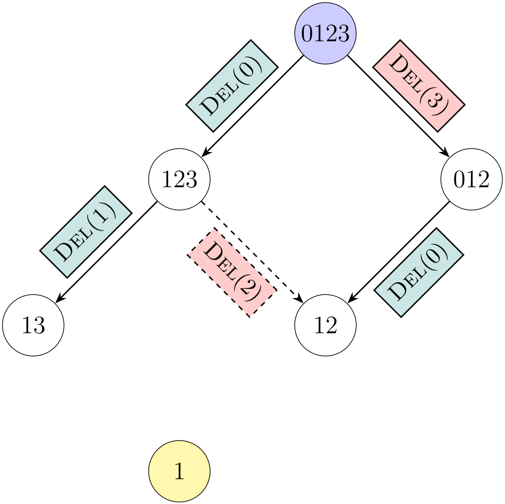
Commutative $\sigma; o_1; o_2' \equiv \sigma; o_2; o_1'$

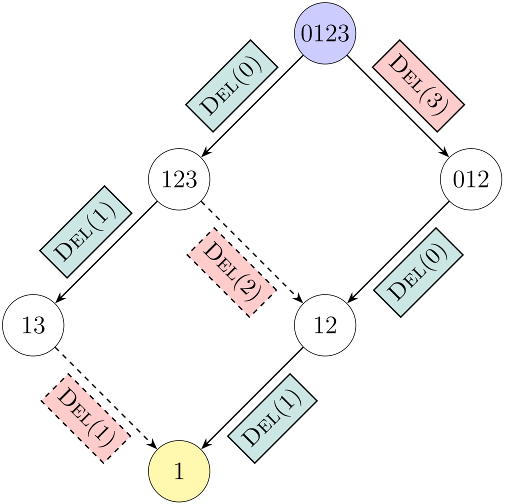
[Ellis and Gibbs, 1989]

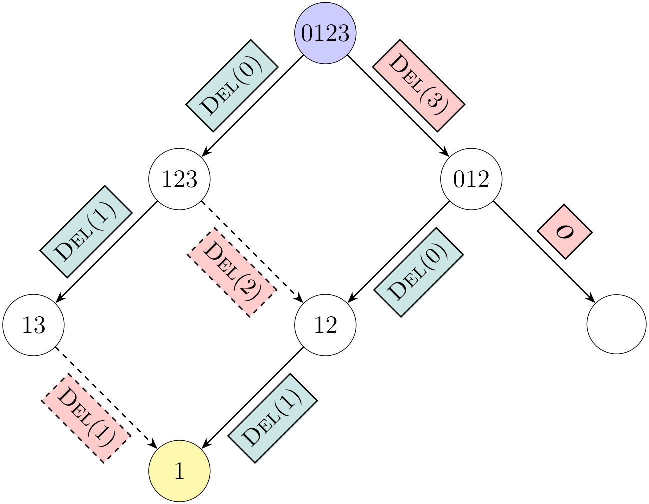

0123
DEL(0)
123
DEL(1)
13


0123
Del(0)
Del(3)
123
012
Del(1)
13


0123
DEL(0)
DEL(3)
123
012
DEL(1)
13
1

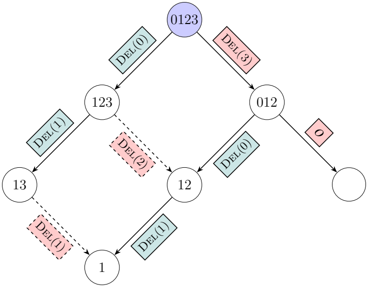

0123
Del(0)
Del(3)
123
012
Del(1)
Del(2)
Del(0)
13
12
1


0123
Del(0)
Del(3)
123
012
Del(1)
Del(2)
Del(0)
13
12
Del(1)
Del(1)
1


0123
Del(0)
Del(3)
123
012
Del(1)
Del(2)
Del(0)
0
13
12
Del(1)
Del(1)
1

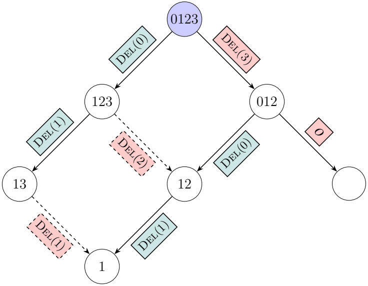
Jupiter uses 2D state spaces [Xu, Sun, and Li, 2014]

to manage how and when to perform OTs [Ellis and Gibbs, 1989].



Jupiter uses 2D state spaces [Xu, Sun, and Li, 2014]

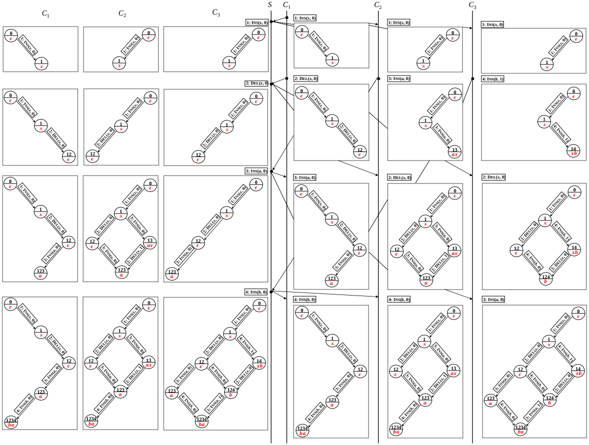
to manage how and when to perform OTs [Ellis and Gibbs, 1989].



Local Dimension: For operations generated by the client

Global Dimension: For operations generated by others

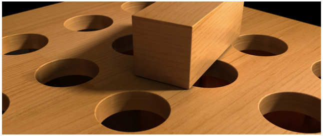
Each client maintains a $2D$ state space.



The server maintains n $(= 3)$ $2D$ state spaces, one for each client.

Mismatch!

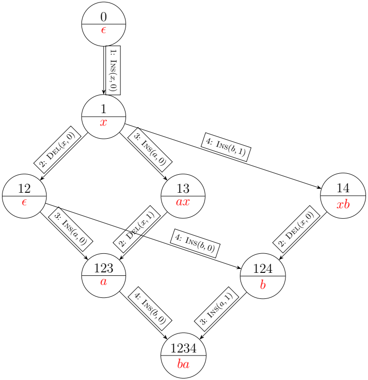
Global property on all replica states specified by $\mathcal{A}_{\text{weak}}$



Local view each replica maintains in Jupiter

CJupiter (Compact Jupiter)

CJupiter maintains an n-ary ordered state space for each replica.

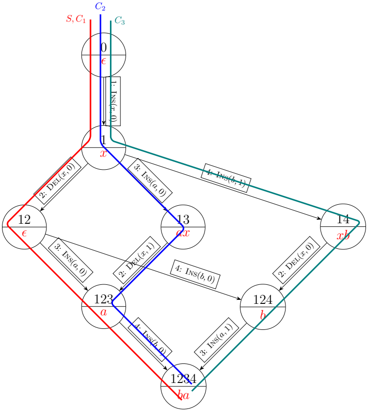


There can be more than two edges coming from the same node.

Edges from the same node are totally ordered by associated operations.

Proposition (Compactness of CJupiter (Informal))

At a high level, CJupiter maintains only one n-ary ordered state space.



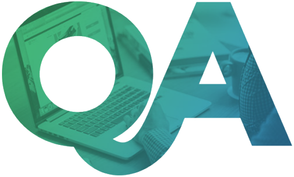
Each replica behavior corresponds to a path going through this state space.

Theorem (Equivalence of CJupiter and Jupiter)

Under the same schedule, the behaviors of corresponding replicas in CJupiter and Jupiter are the same.

From the perspectives of both the server and the clients.

Thank
You!


QA

Attiya, Hagit et al. (2016). “Specification and complexity of collaborative text editing”. In: *Proceedings of the 2016 ACM Symposium on Principles of Distributed Computing*. PODC ’16. ACM, pp. 259–268.

Ellis, C. A. and S. J. Gibbs (1989). “Concurrency Control in Groupware Systems”. In: *Proceedings of the 1989 ACM SIGMOD International Conference on Management of Data*. SIGMOD ’89. ACM, pp. 399–407.

Nichols, David A. et al. (1995). “High-latency, Low-bandwidth Windowing in the Jupiter Collaboration System”. In: *Proceedings of the 8th Annual ACM Symposium on User Interface and Software Technology*. UIST ’95. ACM, pp. 111–120.

Shapiro, Marc et al. (2011). “Conflict-free Replicated Data Types”. In: *Proceedings of the 13th International Conference on Stabilization, Safety, and Security of Distributed Systems*. SSS’11. Springer-Verlag, pp. 386–400.

Xu, Yi, Chengzheng Sun, and Mo Li (2014). “Achieving Convergence in Operational Transformation: Conditions, Mechanisms and Systems”. In: *Proceedings of the 17th ACM Conference on Computer Supported Cooperative Work*. CSCW ’14. ACM, pp. 505–518.