Specification and Implementation of Replicated List
— The Jupiter Protocol Revisited

(Brief Announcement at PODC'2018)

Hengfeng Wei, Yu Huang, Jian Lu

Nanjing University

July 24, 2018

The Jupiter protocol [Nichols et al., 1995][1] for replicated list satisfies the weak list specification [Attiya et al., 2016][2].

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

[1] 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.

[2] 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.

Background for the Conjecture

Collaborative Text Editing Systems

(a) Google Docs

(b) Apache Wave

(c) Wikipedia

(d) LaTeX Editor

Replication (for availability)

Replication (for availability)

- Replicas respond to user operations immediately
  - Updates are propagated asynchronously

List: to model the core functionality

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

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

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

List: to model the core functionality

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

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

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

To implement a highly available replicated list object.

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

The lists at all replicas are identical *at quiescence*.

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

The lists at the replicas that *have executed the same set of user operations* are identical.

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

The lists at all replicas are identical *at quiescence*.

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

The lists at the replicas that *have executed the same set of user operations* are identical.

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

Strong/weak list specification [Attiya et al., 2016]

Specify global properties on all (intermediate) states at all 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

Strong/weak list specification [Attiya et al., 2016]

Specify global properties on all (intermediate) states at all replicas.

Proved: RGA [Roh et al., 2011] satisfies the strong list spec.

Conjecture: *Jupiter* [Nichols et al., 1995] satisfies the weak list spec.

Does Jupiter satisfy the weak list specification?

Yes, it does.

Weak List Specification

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.

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.

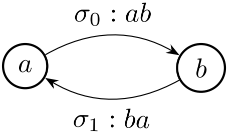
Pairwise state compatibility property:

$$\forall \sigma, \sigma' : a, b \in \sigma \cap \sigma' \implies (a \prec_{\sigma} b \iff a \prec_{\sigma'} b)$$

$$(\sigma, \sigma' : \text{ list}; \quad a, b : \text{ element}; \quad \prec_{\sigma}: \text{ precedes})$$

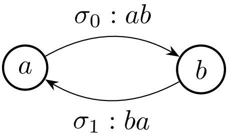
$$\forall \sigma, \sigma' : a, b \in \sigma \cap \sigma' \implies (a \prec_{\sigma} b \iff a \prec_{\sigma'} b)$$

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

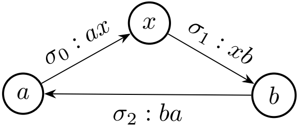


$$\forall \sigma, \sigma' : a, b \in \sigma \cap \sigma' \implies (a \prec_{\sigma} b \iff a \prec_{\sigma'} b)$$

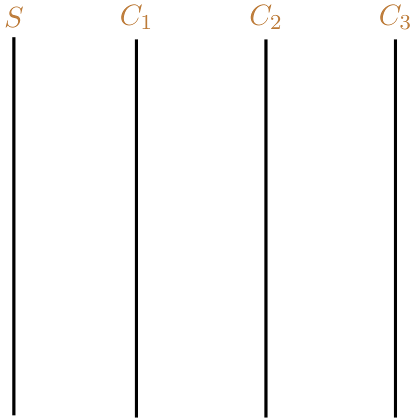
$\boxed{\sigma_0 : ab}$ $\boxed{\sigma_1 : ba}$



$\boxed{\sigma_0 : ax}$ $\boxed{\sigma_1 : xb}$ $\boxed{\sigma_2 : ba}$

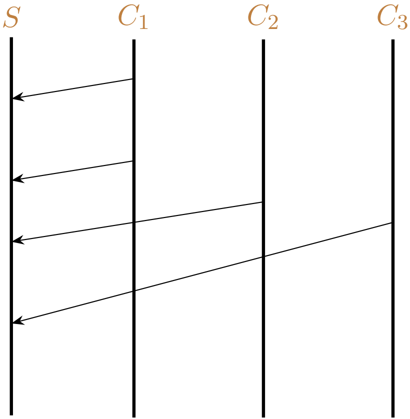


Jupiter



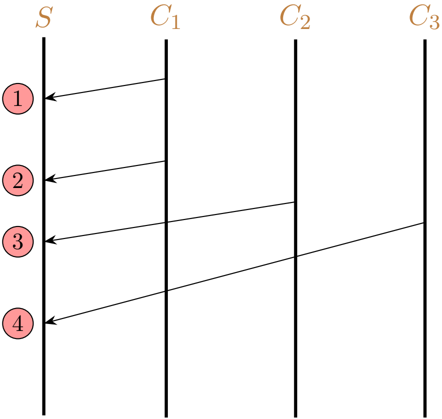
System model of Jupiter:

- client-server architecture



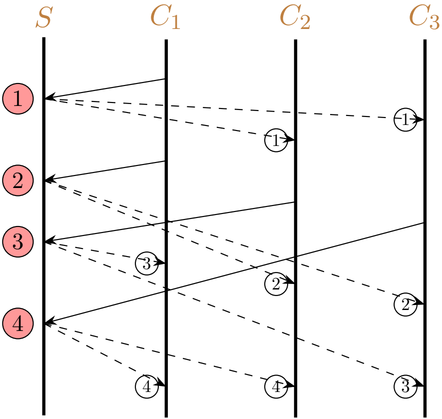
System model of Jupiter:

- client-server architecture
- client $\xrightarrow{\text{FIFO}}$ server



System model of Jupiter:

- client-server architecture
- client $\xrightarrow{\text{FIFO}}$ server
- totally ordered at the server

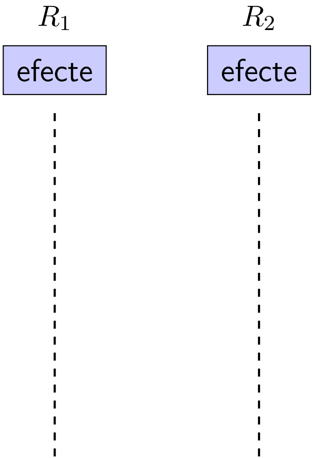


System model of Jupiter:

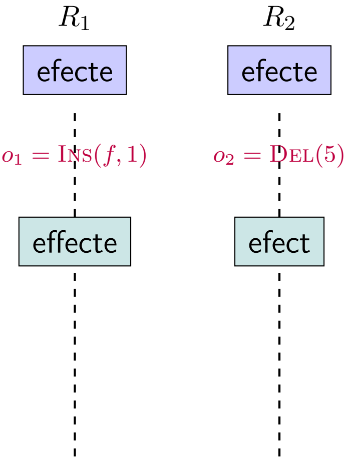
- client-server architecture
- client $\xrightarrow{\text{FIFO}}$ server
- totally ordered at the server
- server $\xrightarrow{\text{FIFO}}$ client

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

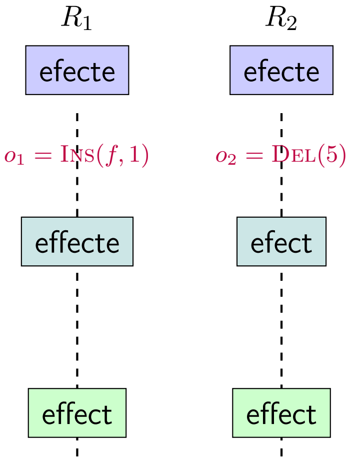
OT (Operational Transformation) [Ellis and Gibbs, 1989]



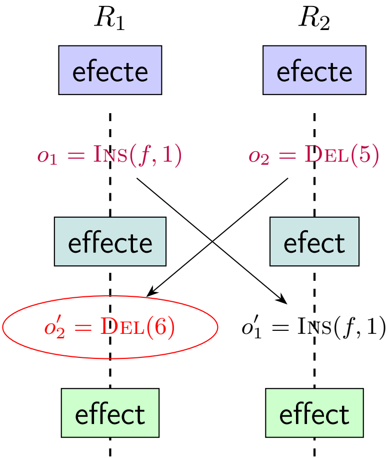
OT (Operational Transformation) [Ellis and Gibbs, 1989]



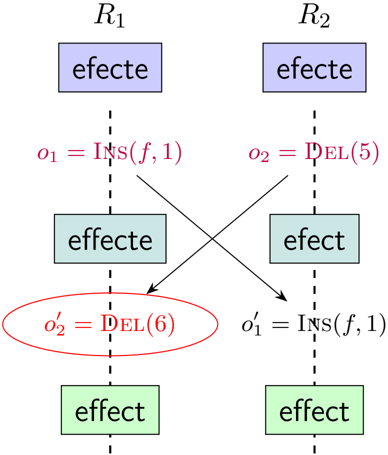
OT (Operational Transformation) [Ellis and Gibbs, 1989]

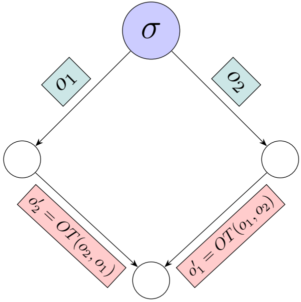


OT (Operational Transformation) [Ellis and Gibbs, 1989]



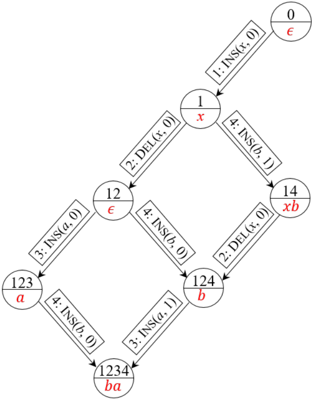
OT (Operational Transformation) [Ellis and Gibbs, 1989]



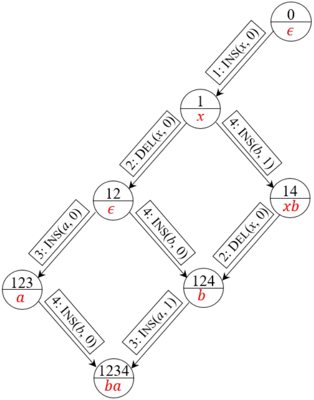


$$\sigma; o_1; o'_2 \equiv \sigma; o_2; o'_1$$

Jupiter uses $2D$ state spaces [Sun, Xu, and Agustina, 2014] to manage how and when to perform OTs.

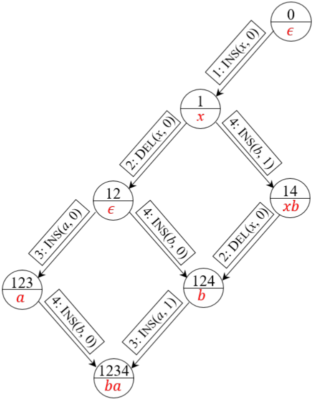


Jupiter uses $2D$ state spaces [Sun, Xu, and Agustina, 2014] to manage how and when to perform OTs.



Nodes represent states. Edges are labeled with operations.

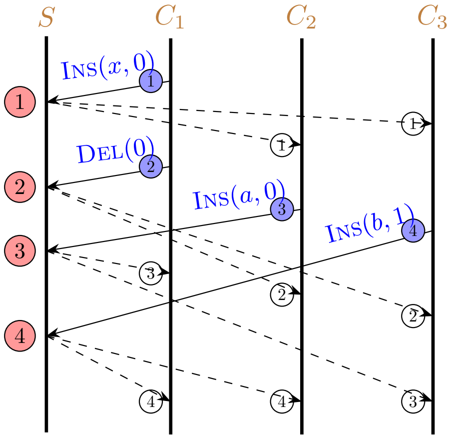
Jupiter uses $2D$ state spaces [Sun, Xu, and Agustina, 2014]
to manage how and when to perform OTs.



Nodes represent states. Edges are labeled with operations.

$2D$: An operation from the same node is either LOCAL or GLOBAL.

Consider a replicated system with n $(= 3)$ clients.



Each client maintains a $2D$ state space.

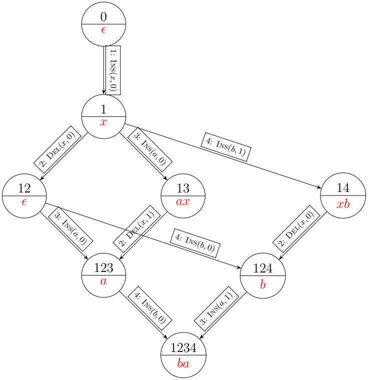
The server maintains $n\ (= 3)\ 2D$ state spaces, one for each client.

Global property on all replica states
specified by the weak list specification

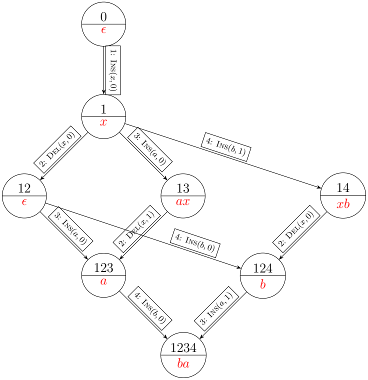
Local view each replica maintains in Jupiter

CJupiter (Compact Jupiter)

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

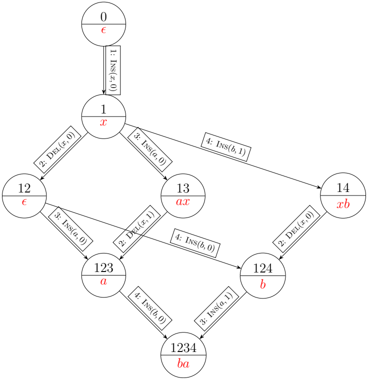


CJupiter maintains an n-ary ordered state space for each replica.



Nodes represent states. Edges are labeled with operations.

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

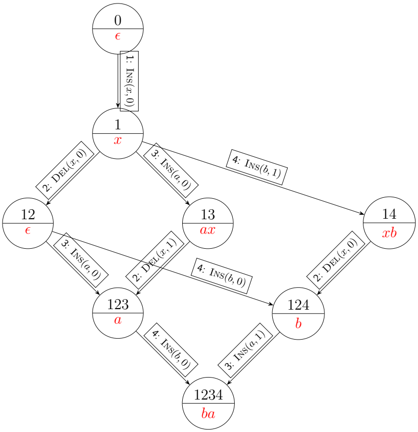


Nodes represent states. Edges are labeled with operations.

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

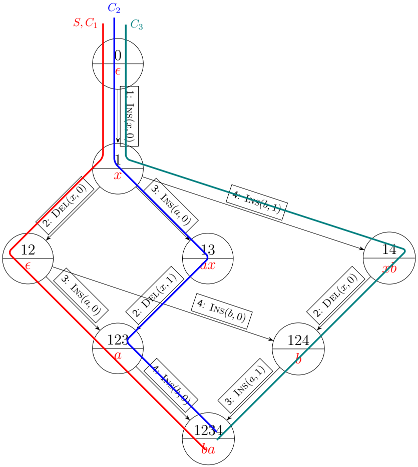
Proposition ($n + 1 \to 1$ (Informal))

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



Proposition ($n+1 \to 1$ (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)

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

Theorem (Equivalence)

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

At the server side:

Proposition ($n \leftrightarrow 1$ (Informal))

The single n-ary ordered state space at the server side in CJupiter is a compact representation of n $2D$ state spaces at the server side in Jupiter.

Theorem (Equivalence)

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

At the server side:

Proposition ($n \leftrightarrow 1$ (Informal))

The single n-ary ordered state space at the server side in CJupiter is a compact representation of n $2D$ state spaces at the server side in Jupiter.

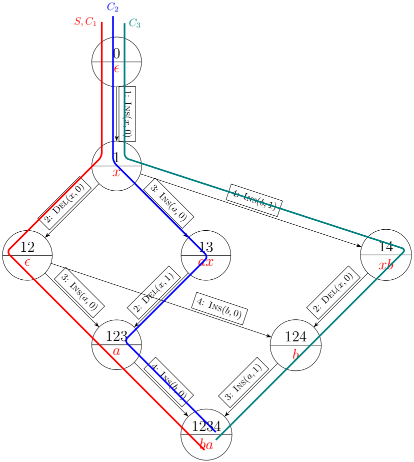
At the client side:

Proposition ($1 \leftrightarrow 1$ (Informal))

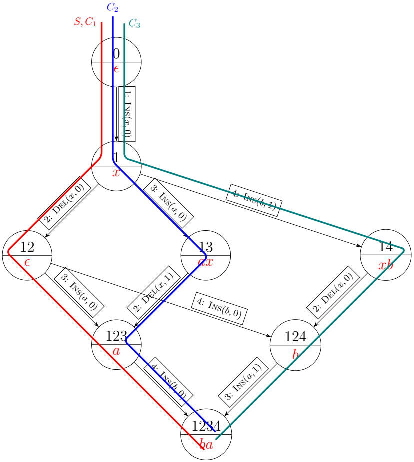
Jupiter is slightly optimized in implementation at clients by eliminating redundant OTs than CJupiter.

CJupiter satisfies the weak list specification.

We study a single n-ary ordered state space CSS_s at the server which provides a global view of all possible replica states.



We study a single n-ary ordered state space CSS_s at the server which provides a global view of all possible replica states.

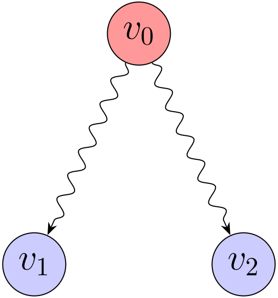


To show the pairwise state compatibility property in three steps.

1. Take any two nodes/states v_1 and v_2.

Lemma (LCA (Lowest Common Ancestor))

Each pair of states in the n-ary ordered state space has a unique LCA.

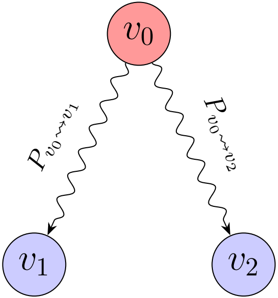


$$v_0 = \mathsf{LCA}(v_1, v_2)$$

2 Consider the paths to v_1 and v_2 from their LCA v_0.

Lemma (Disjoint Paths)

The set of operations $O_{v_0 \rightsquigarrow v_1}$ along $P_{v_0 \rightsquigarrow v_1}$ is disjoint from the set of operations $O_{v_0 \rightsquigarrow v_2}$ along $P_{v_0 \rightsquigarrow v_2}$.

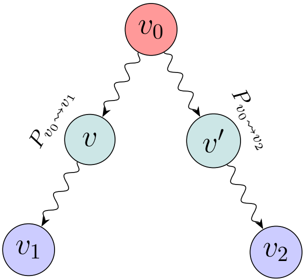


$v_0 = \mathsf{LCA}(v_1, v_2)$

③ Consider the states in these two paths.

Lemma (Compatible Paths)

Each pair of states consisting of one state v in $P_{v_0 \rightsquigarrow v_1}$ and the other v' in $P_{v_0 \rightsquigarrow v_2}$ are compatible.



$$v_0 = \mathsf{LCA}(v_1, v_2)$$

In particular,
v_1 and v_2 are compatible.


Thank You!

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.

Imine, Abdessamad et al. (2006). “Formal Design and Verification of Operational Transformation Algorithms for Copies Convergence”. In: *Theor. Comput. Sci.* 351.2, pp. 167–183.

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.

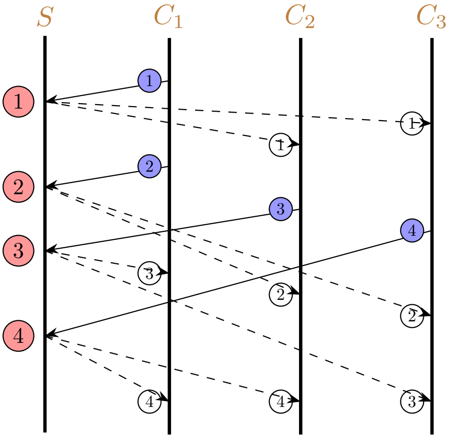
Roh, Hyun-Gul et al. (2011). “Replicated Abstract Data Types: Building Blocks for Collaborative Applications”. In: *J. Parallel Distrib. Comput.* 71.3, pp. 354–368.

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.

Sun, Chengzheng, Yi Xu, and Agustina Agustina (2014). "Exhaustive Search of Puzzles in Operational Transformation". In: *Proceedings of the 17th ACM Conference on Computer Supported Cooperative Work*. CSCW '14. Baltimore, Maryland, USA: ACM, pp. 519–529.

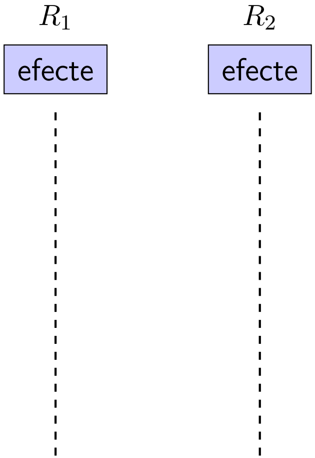
Backup

It is still challenging to achieve convergence despite the server.

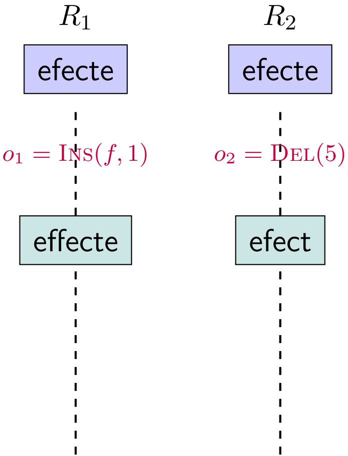


OT (Operational Transformation)

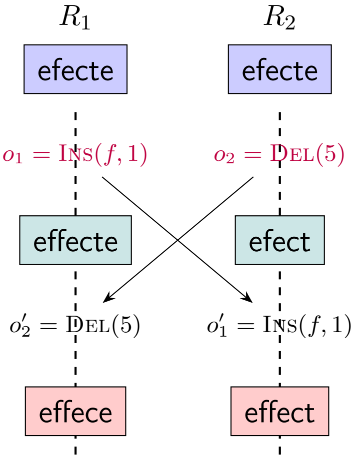
OT (Operational Transformation)



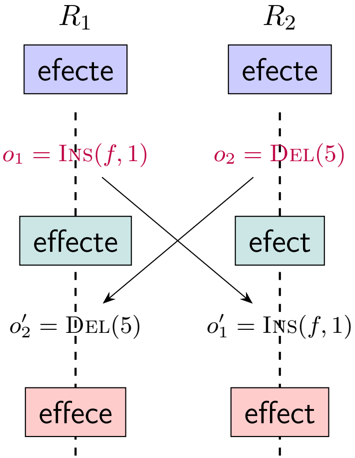
OT (Operational Transformation)

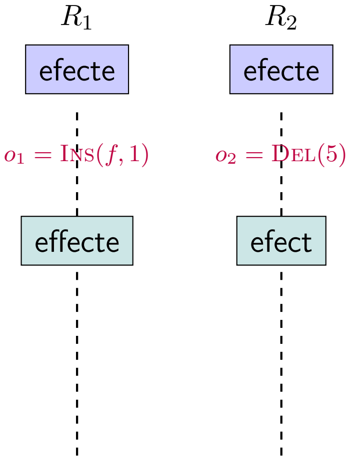


OT (Operational Transformation)

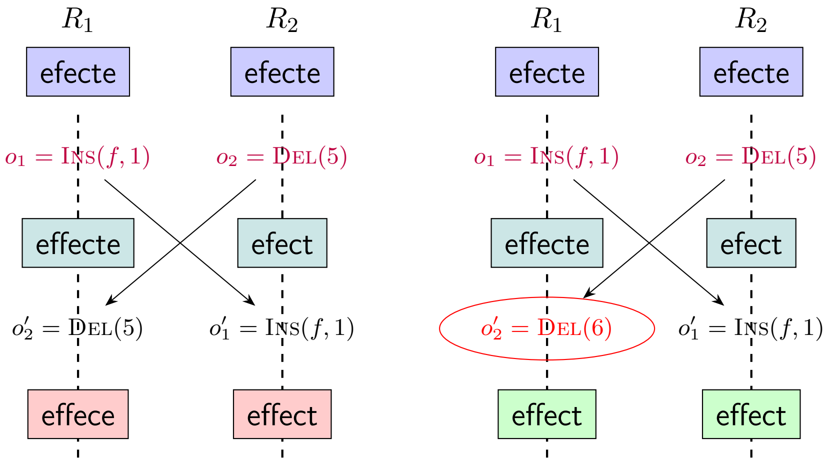


OT (Operational Transformation)





OT (Operational Transformation)



OT functions for a replicated list object [Ellis and Gibbs, 1989; Imine et al., 2006]

$$OT\Big(\textsc{Ins}(a_1, p_1, pr_1), \textsc{Ins}(a_2, p_2, pr_2)\Big) = \begin{cases} \textsc{Ins}(a_1, p_1, pr_1) & p_1 < p_2 \\ \textsc{Ins}(a_1, p_1 + 1, pr_1) & p_1 > p_2 \\ \text{NOP} & p_1 = p_2 \land a_1 = a_2 \\ \textsc{Ins}(a_1, p_1 + 1, pr_1) & p_1 = p_2 \land a_1 \neq a_2 \land pr_1 > pr_2 \\ \textsc{Ins}(a_1, p_1, pr_1) & p_1 = p_2 \land a_1 \neq a_2 \land pr_1 \leq pr_2 \end{cases}$$

$$OT\Big(\textsc{Ins}(a_1, p_1, pr_1), \textsc{Del}(\_, p_2, pr_2)\Big) = \begin{cases} \textsc{Ins}(a_1, p_1, pr_1) & p_1 \leq p_2 \\ \textsc{Ins}(a_1, p_1 - 1, pr_1) & p_1 > p_2 \end{cases}$$

$$OT\Big(\textsc{Del}(\_, p_1, pr_1), \textsc{Ins}(a_2, p_2, pr_2)\Big) = \begin{cases} \textsc{Del}(\_, p_1, pr_1) & p_1 < p_2 \\ \textsc{Del}(\_, p_1 + 1, pr_1) & p_1 \geq p_2 \end{cases}$$

$$OT\Big(\textsc{Del}(\_, p_1, pr_1), \textsc{Del}(\_, p_2, pr_2)\Big) = \begin{cases} \textsc{Del}(\_, p_1, pr_1) & p_1 < p_2 \\ \textsc{Del}(\_, p_1 - 1, pr_1) & p_1 > p_2 \\ \text{NOP} & p_1 = p_2 \end{cases}$$