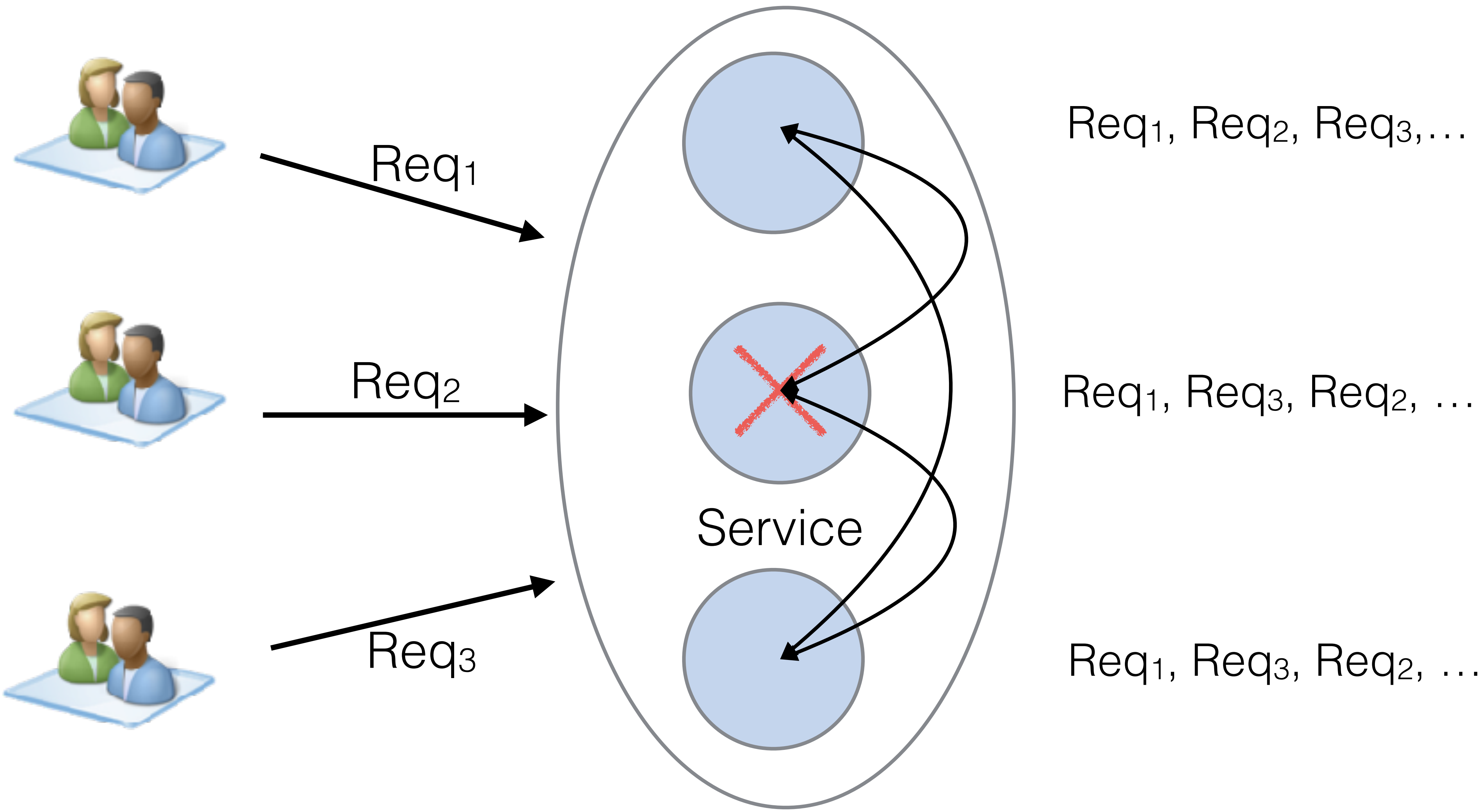
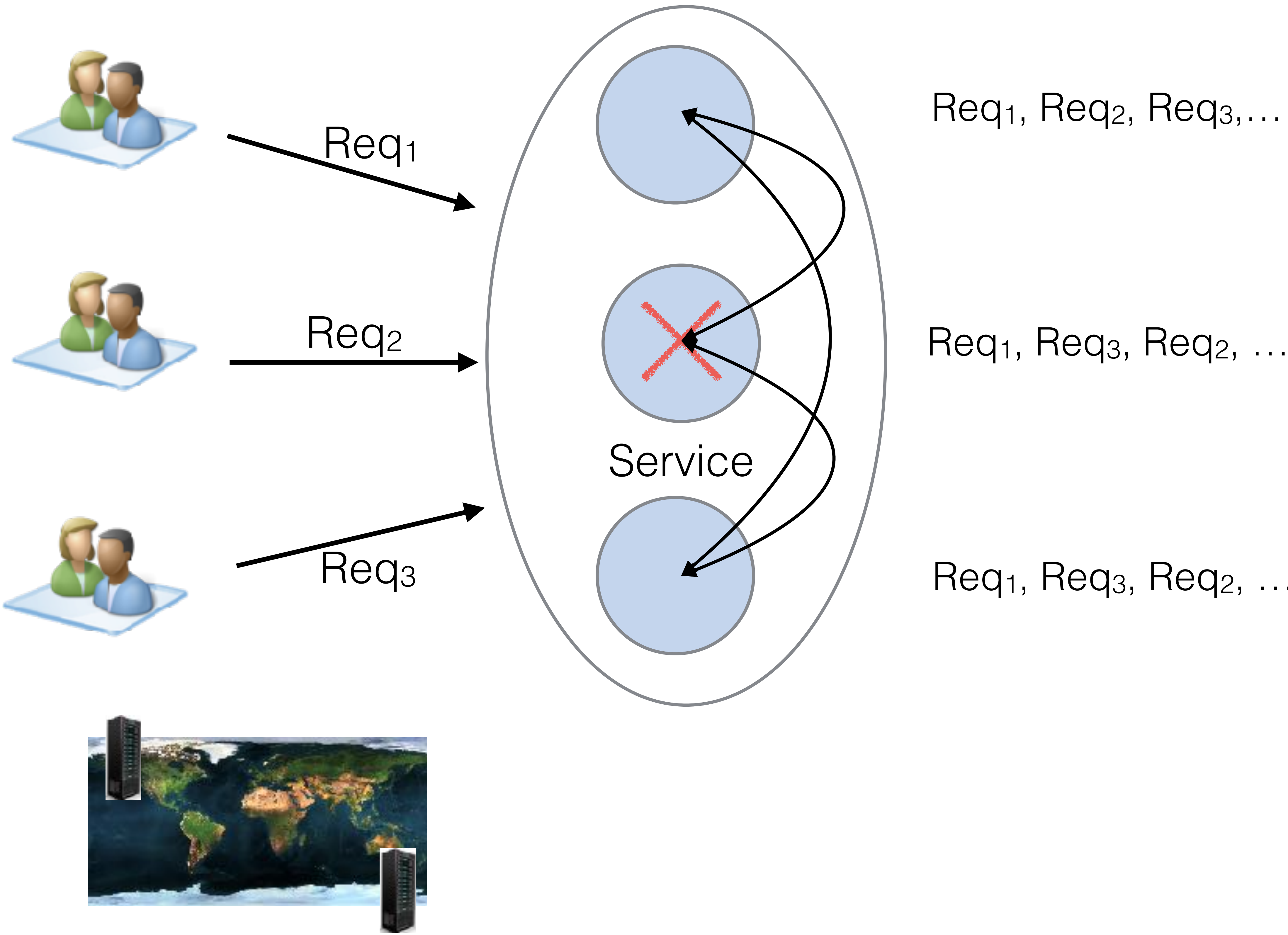
Commutativity Reasoning for Automated Distributed Coordination

Mohsen Lesani
University of California, Riverside

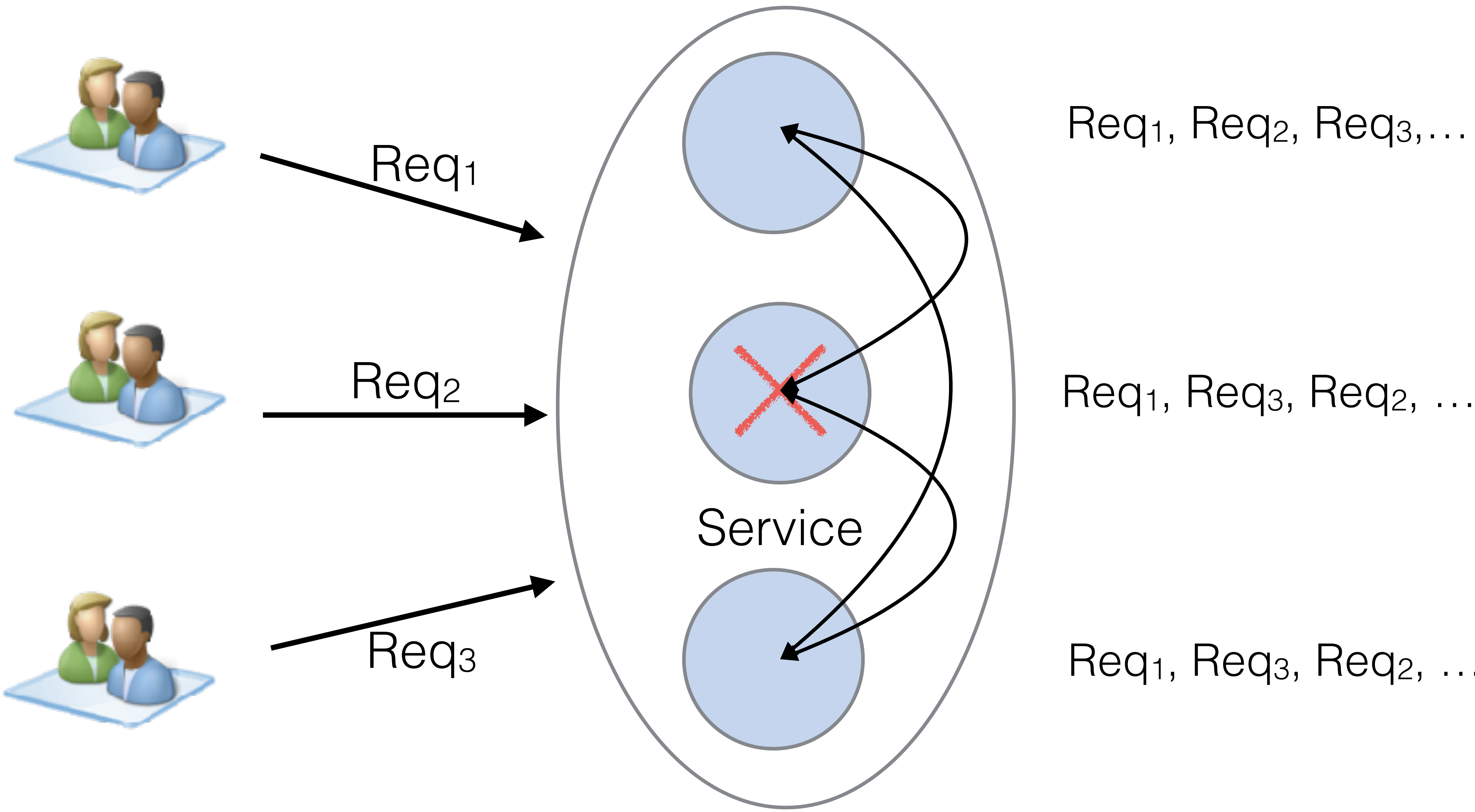
Replication



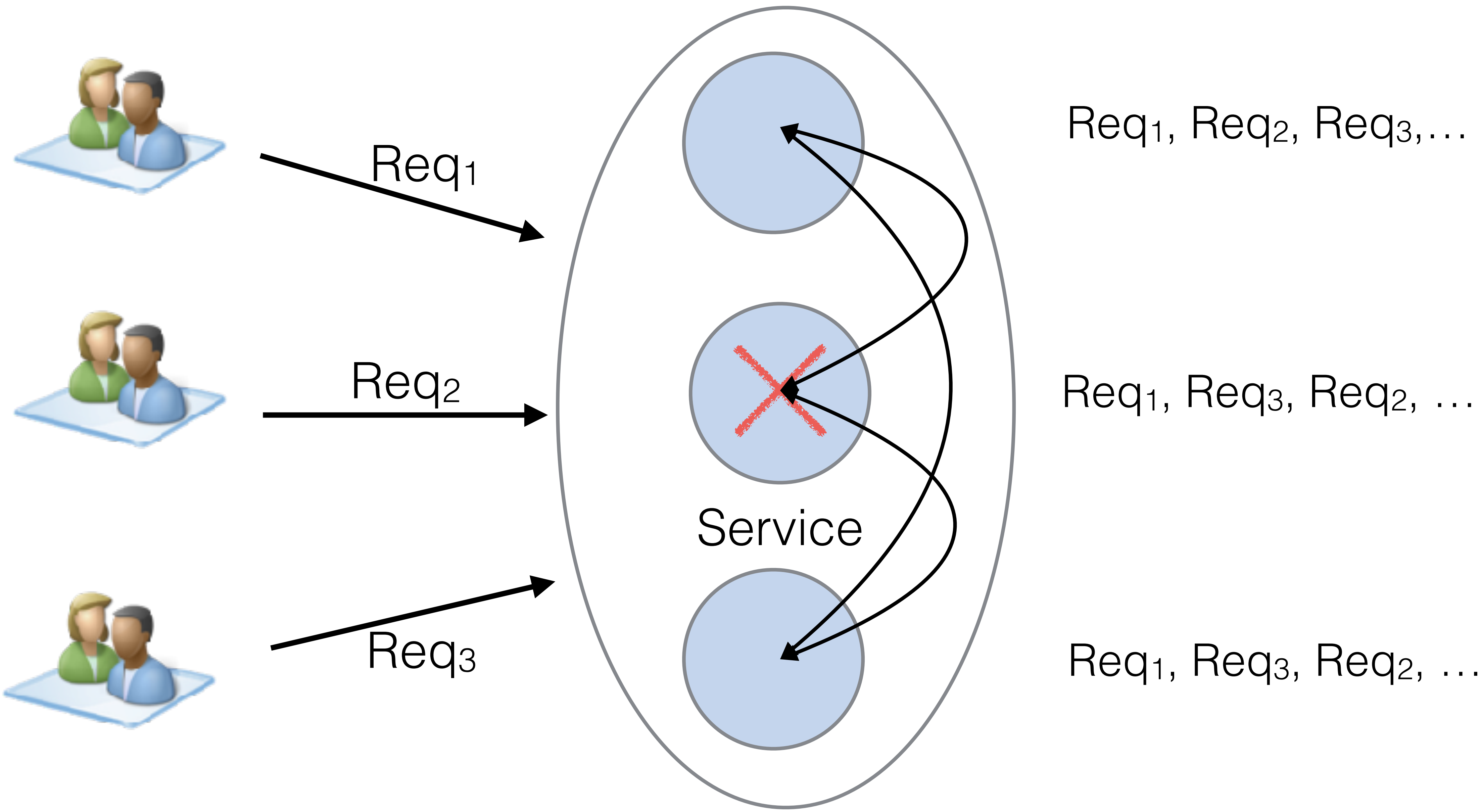
Replication



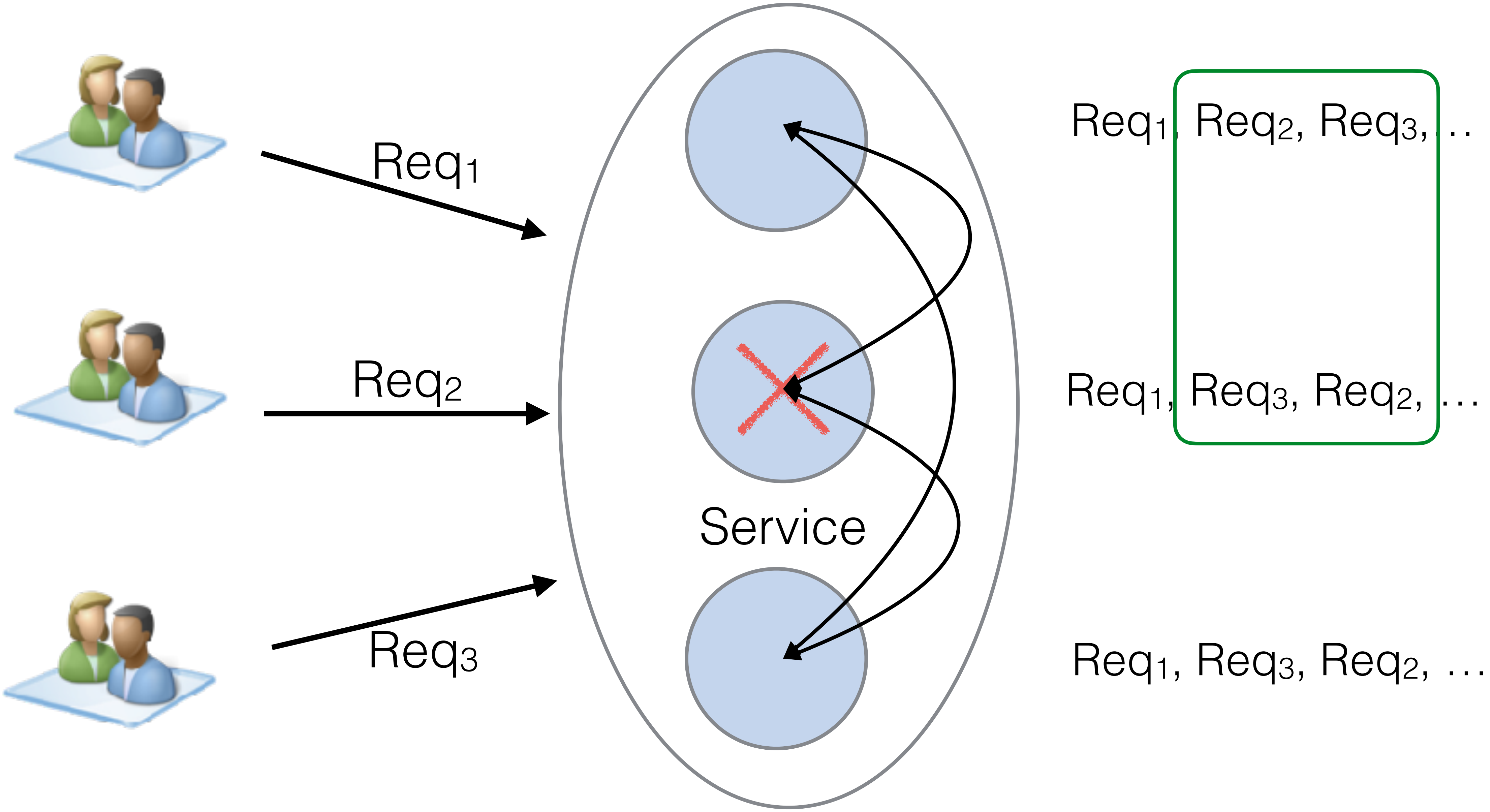
Replication

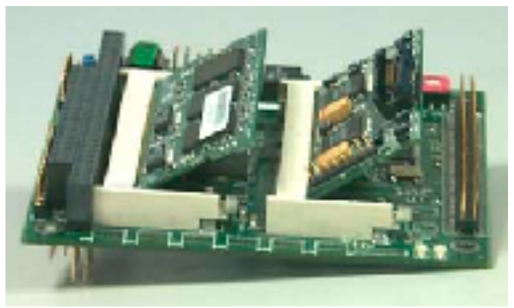


Replication



Replication





Consistency vs. Responsiveness and Availability

Consistency

Viewstamp [PODC'88]
Paxos [98]
Raft [USENIX'14]

Sequential Consistency

Eventual Consistency

amazon DynamoDB
SOSP'07

cassandra
OSR'10

Responsiveness
Availability

Consistency vs. Responsiveness and Availability

Consistency

Viewstamp [PODC'88]
Paxos [98]
Raft [USENIX'14]

Sequential Consistency

COPS [SOSP'11]
Eiger [NSDI'13]
BoltOn [SIGMOD'13]
GentleRain [SOCC'14]

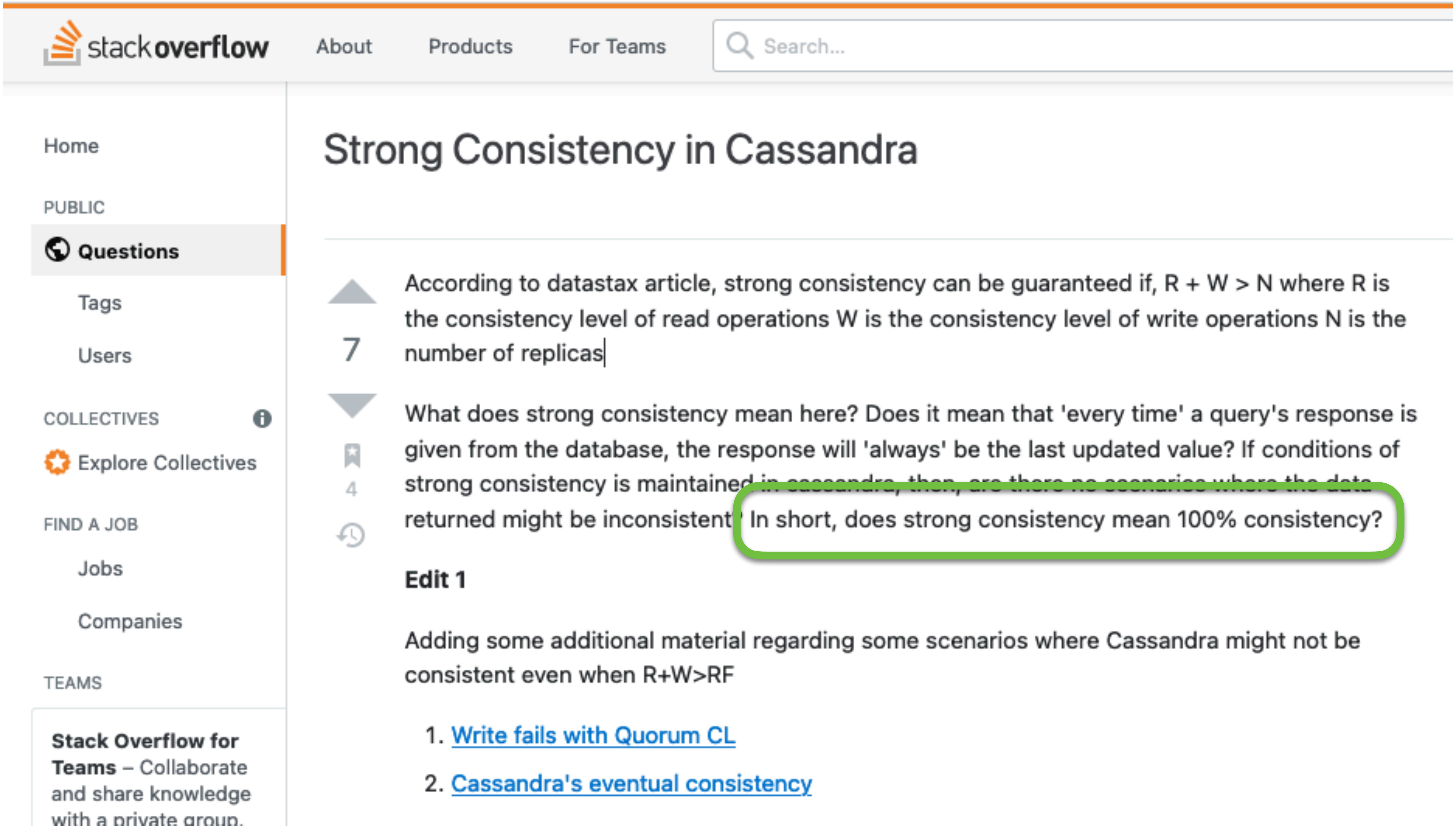
Causal Consistency

amazon DynamoDB — SOSP'07
cassandra — OSR'10

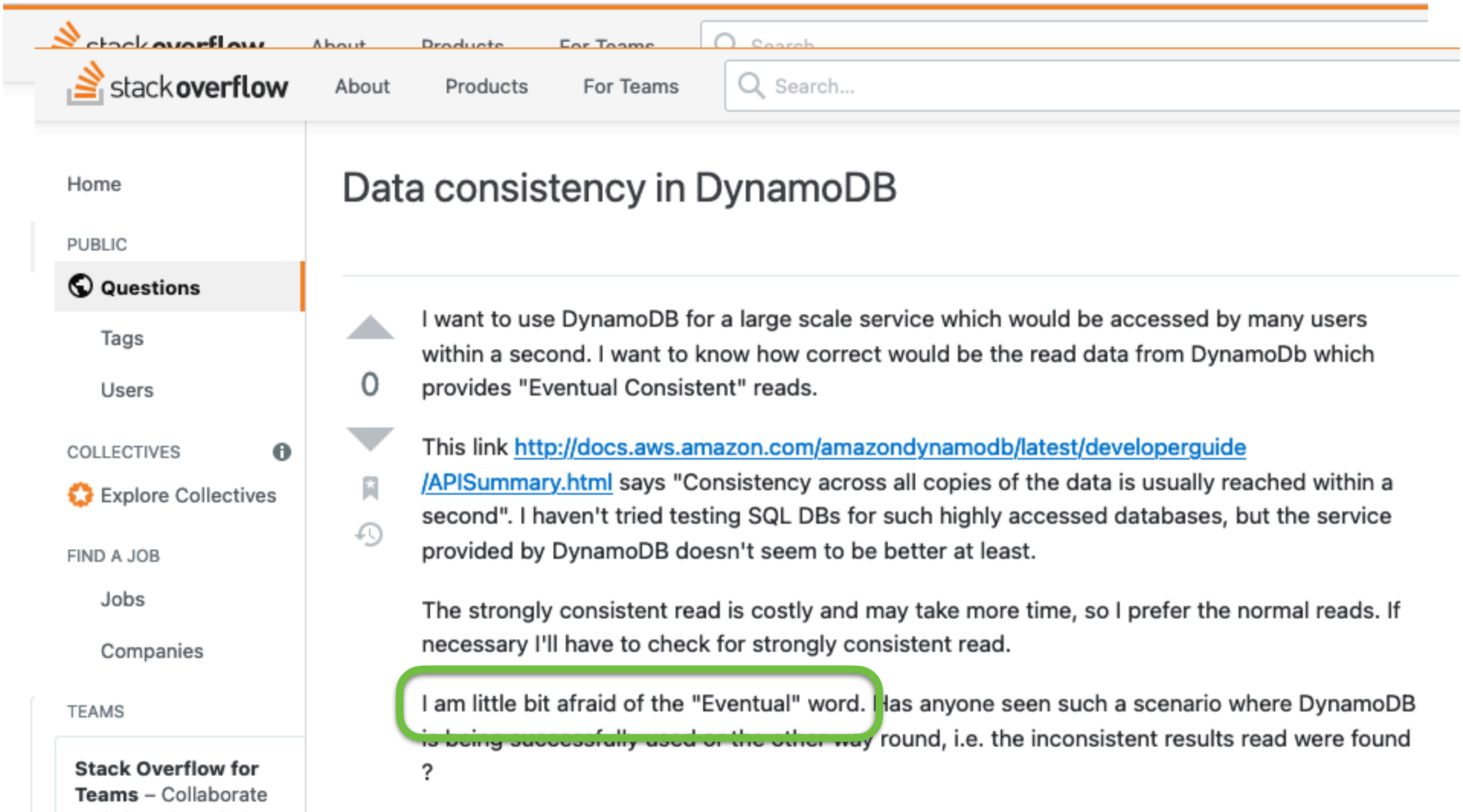
Eventual Consistency

Responsiveness
Availability

Confusing Weak Consistency Notions



Confusing Weak Consistency Notions



stack overflow About Products For Teams Search...

Home

PUBLIC

Questions

Tags

Users

COLLECTIVES

Explore Collectives

FIND A JOB

Jobs

Companies

TEAMS

Stack Overflow for Teams – Collaborate

Data consistency in DynamoDB

0

I want to use DynamoDB for a large scale service which would be accessed by many users within a second. I want to know how correct would be the read data from DynamoDb which provides "Eventual Consistent" reads.

This link http://docs.aws.amazon.com/amazondynamodb/latest/developerguide/APISummary.html says "Consistency across all copies of the data is usually reached within a second". I haven't tried testing SQL DBs for such highly accessed databases, but the service provided by DynamoDB doesn't seem to be better at least.

The strongly consistent read is costly and may take more time, so I prefer the normal reads. If necessary I'll have to check for strongly consistent read.

I am little bit afraid of the "Eventual" word. Has anyone seen such a scenario where DynamoDB is being successfully used or the other way round, i.e. the inconsistent results read were found ?

Consistency and Integrity

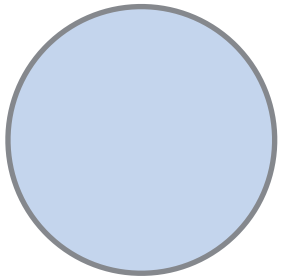
Consistency and Integrity

- Bank Account. Integrity: Non-negative balance.

Consistency and Integrity

- Bank Account. Integrity: Non-negative balance.
- What users need is integrity and **Consistency** is just a means to **Integrity**.

Hamsaz: Coordination-avoiding Replicated Object Synthesis

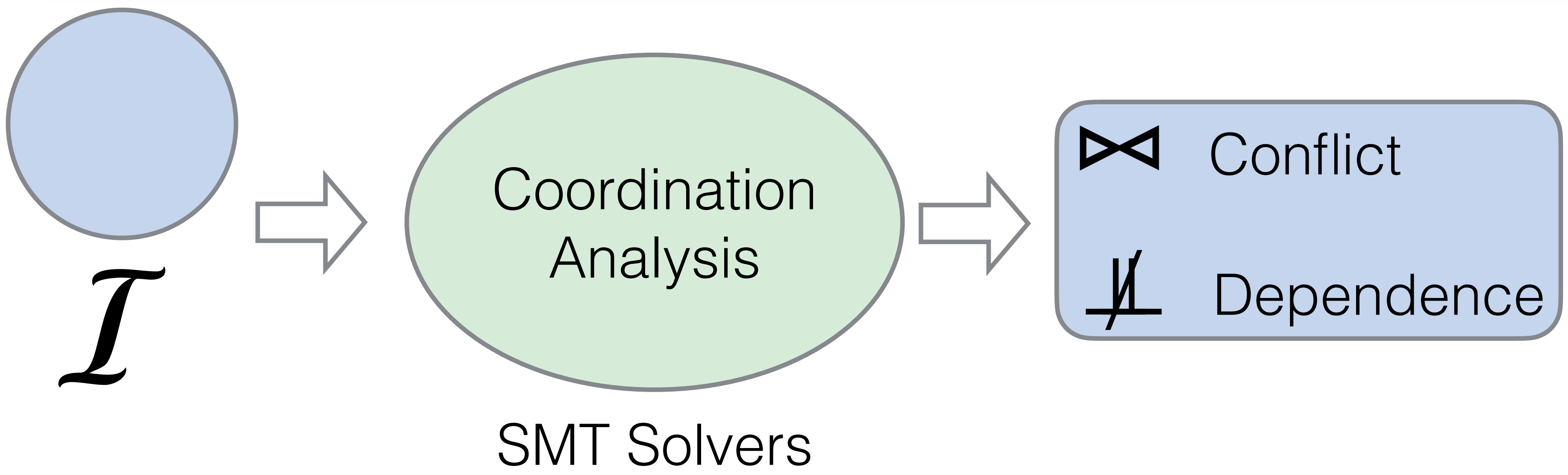


Class

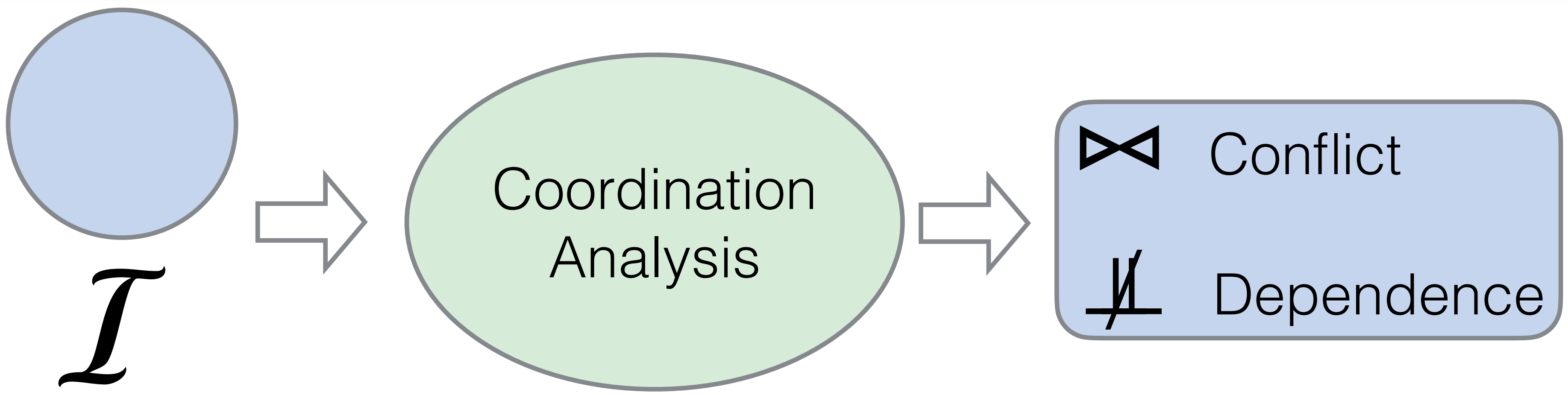
$\mathcal{I}$ Integrity Property

Synthesis of replicated objects that preserve integrity and convergence and minimize coordination

Hamsaz: Coordination-avoiding Replicated Object Synthesis



Hamsaz: Coordination-avoiding Replicated Object Synthesis



Well-coordination:

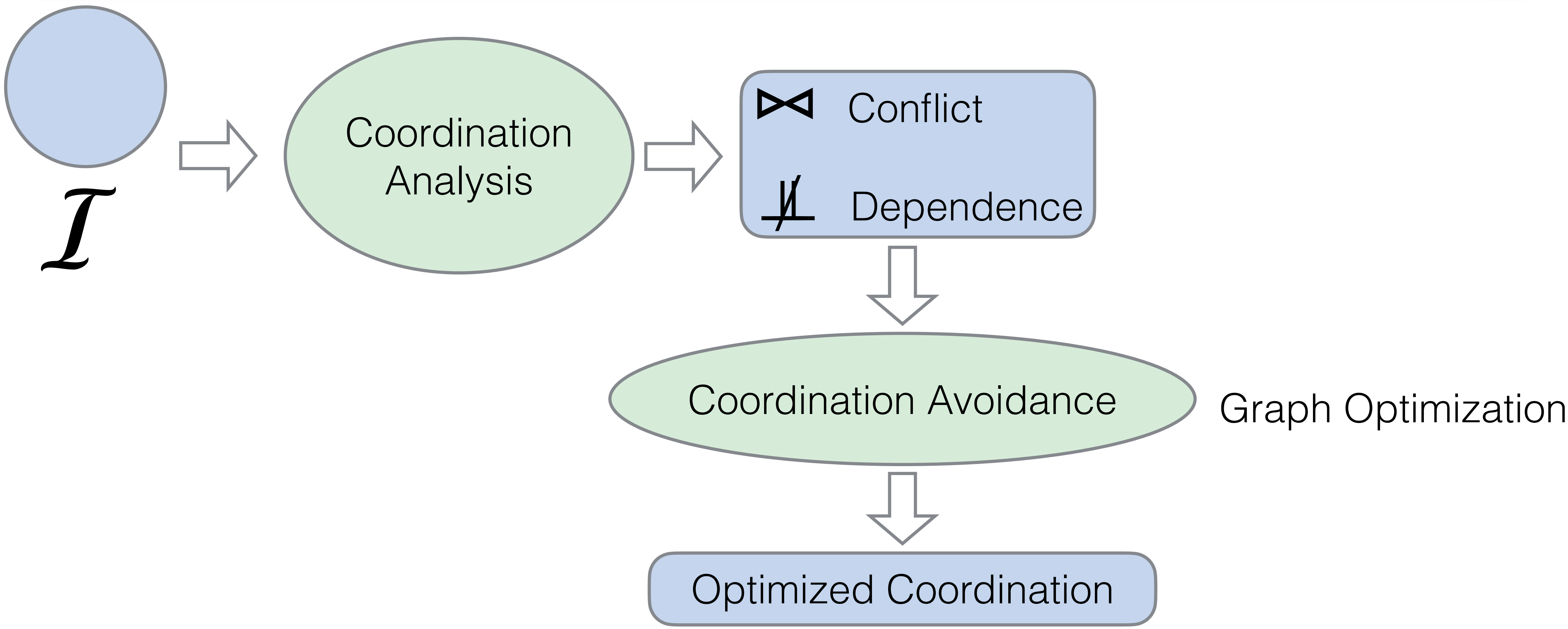
Synchronization between conflicting

Causality between dependent

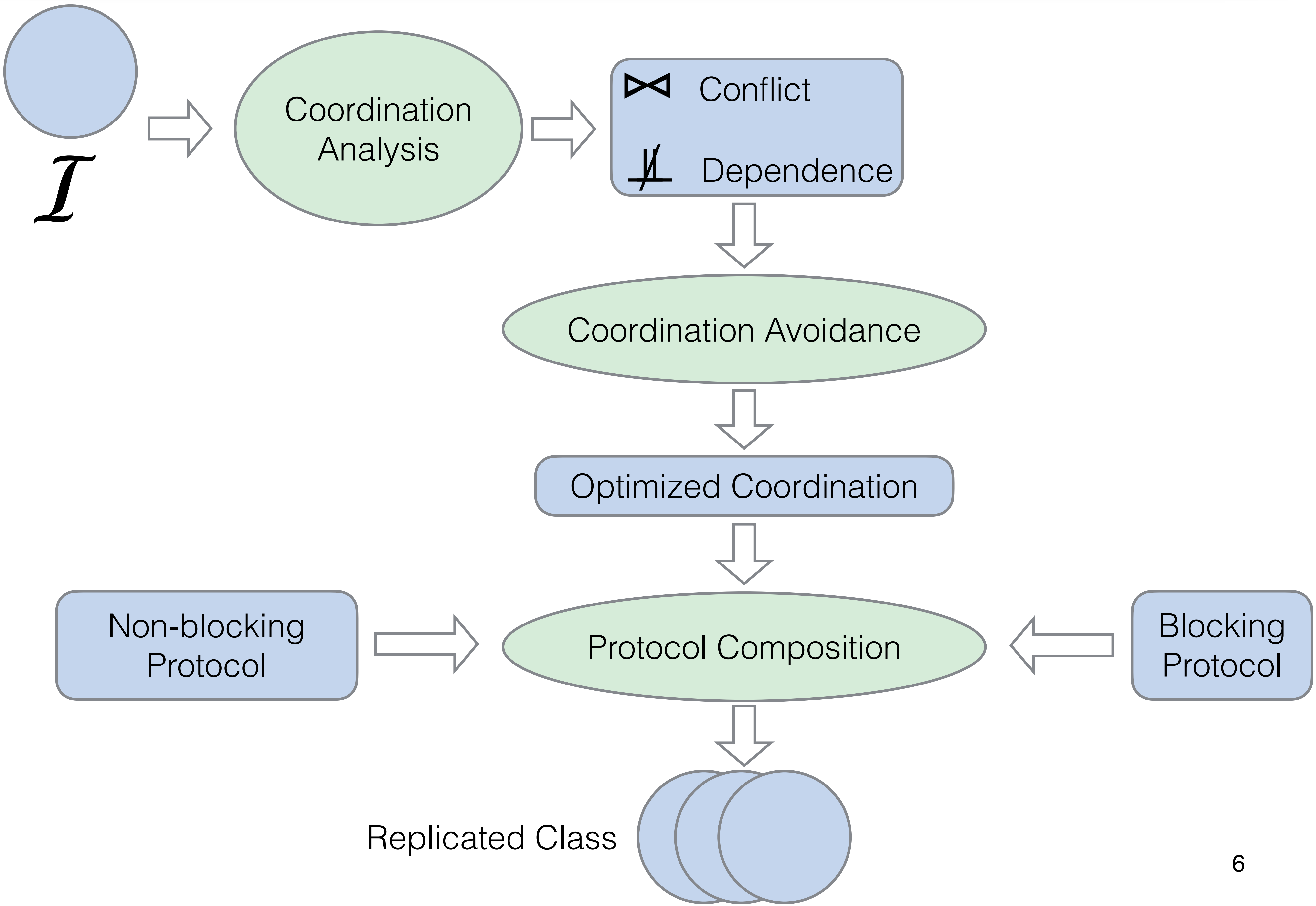
Theorem:

Well-coordination is sufficient for integrity and convergence

Hamsaz: Coordination-avoiding Replicated Object Synthesis



Hamsaz: Coordination-avoiding Replicated Object Synthesis



Example Class

$$
\begin{array}{l}
\text{Class Courseware} \\
\quad \text{let Student} := \text{Set}\ \langle \text{sid}\colon \text{SId}\rangle\ \text{in} \\
\quad \text{let Course} := \text{Set}\ \langle \text{cid}\colon \text{CId}\rangle\ \text{in} \\
\quad \text{let Enrolment} := \\
\qquad \text{Set}\ \langle \text{esid}\colon \text{SId}, \text{ecid}\colon \text{CId}\rangle\ \text{in} \\
\quad \Sigma := \text{Student} \times \text{Course} \times \text{Enrolment} \\
\quad \mathcal{I} := \lambda\ \langle ss, cs, es\rangle. \\
\qquad \text{refIntegrity}(es, \text{esid}, ss, \text{sid}) \wedge \\
\qquad \text{refIntegrity}(es, \text{ecid}, cs, \text{cid}) \\
\quad \text{register}(s) := \lambda\ \langle ss, cs, es\rangle. \\
\qquad \langle \mathbb{T}, \quad \langle ss \cup \{s\}, cs, es\rangle, \quad \bot\rangle \\
\quad \text{addCourse}(c) := \lambda\ \langle ss, cs, es\rangle. \\
\qquad \langle \mathbb{T}, \quad \langle ss, cs \cup \{c\}, es\rangle, \quad \bot\rangle \\
\quad \text{enroll}(s, c) := \lambda\ \langle ss, cs, es\rangle. \\
\qquad \langle \mathbb{T}, \quad \langle ss, cs, es \cup \{(s, c)\}\rangle, \quad \bot\rangle \\
\quad \text{deleteCourse}(c) := \lambda\ \langle ss, cs, es\rangle. \\
\qquad \langle \mathbb{T}, \quad \langle ss, cs \setminus \{c\}, es\rangle, \quad \bot\rangle \\
\quad \text{query} := \lambda\ \sigma.\ \langle \mathbb{T}, \quad \sigma, \quad \sigma\rangle
\end{array}
$$

$$
\text{refIntegrity}(R, f, R', f') := \forall r.\ \ r \in R \rightarrow \exists r'.\ r' \in R' \wedge f(r) = f'(r')
$$

Example Class

$$
\begin{aligned}
&\text{Class Courseware}\\
&\quad \text{let Student} := \text{Set}\,\langle \text{sid}\colon \text{SId}\rangle \text{ in}\\
&\quad \text{let Course} := \text{Set}\,\langle \text{cid}\colon \text{CId}\rangle \text{ in}\\
&\quad \text{let Enrolment} :=\\
&\qquad \text{Set}\,\langle \text{esid}\colon \text{SId}, \text{ecid}\colon \text{CId}\rangle \text{ in}\\
&\quad \Sigma := \text{Student} \times \text{Course} \times \text{Enrolment}\\
&\quad \mathcal{I} := \lambda\,\langle ss, cs, es\rangle.\\
&\qquad \text{refIntegrity}(es, \text{esid}, ss, \text{sid}) \wedge\\
&\qquad \text{refIntegrity}(es, \text{ecid}, cs, \text{cid})\\
&\quad \text{register}(s) := \lambda\,\langle ss, cs, es\rangle.\\
&\qquad \langle \mathbb{T},\quad \langle ss \cup \{s\}, cs, es\rangle,\quad \bot\rangle\\
&\quad \text{addCourse}(c) := \lambda\,\langle ss, cs, es\rangle.\\
&\qquad \langle \mathbb{T},\quad \langle ss, cs \cup \{c\}, es\rangle,\quad \bot\rangle\\
&\quad \text{enroll}(s, c) := \lambda\,\langle ss, cs, es\rangle.\\
&\qquad \langle \mathbb{T},\quad \langle ss, cs, es \cup \{(s, c)\}\rangle,\quad \bot\rangle\\
&\quad \text{deleteCourse}(c) := \lambda\,\langle ss, cs, es\rangle.\\
&\qquad \langle \mathbb{T},\quad \langle ss, cs \setminus \{c\}, es\rangle,\quad \bot\rangle\\
&\quad \text{query} := \lambda\,\sigma.\,\langle \mathbb{T},\quad \sigma,\quad \sigma\rangle
\end{aligned}
$$

$$\text{refIntegrity}(R, f, R', f') := \forall r.\ \ r \in R \rightarrow \exists r'.\ r' \in R' \wedge f(r) = f'(r')$$

Example Class

$$
\begin{array}{l}
\text{Class Courseware} \\
\quad \text{let Student} := \text{Set}\,\langle \text{sid}\colon \text{SId}\rangle\ \text{in} \\
\quad \text{let Course} := \text{Set}\,\langle \text{cid}\colon \text{CId}\rangle\ \text{in} \\
\quad \text{let Enrolment} := \\
\qquad \text{Set}\,\langle \text{esid}\colon \text{SId}, \text{ecid}\colon \text{CId}\rangle\ \text{in} \\
\quad \Sigma := \text{Student} \times \text{Course} \times \text{Enrolment} \\
\quad \mathcal{I} := \lambda\,\langle ss, cs, es\rangle. \\
\qquad \text{refIntegrity}(es, \text{esid}, ss, \text{sid}) \wedge \\
\qquad \text{refIntegrity}(es, \text{ecid}, cs, \text{cid}) \\
\quad \text{register}(s) := \lambda\,\langle ss, cs, es\rangle. \\
\qquad \langle \mathbb{T}, \quad \langle ss \cup \{s\}, cs, es\rangle, \quad \bot\rangle \\
\quad \text{addCourse}(c) := \lambda\,\langle ss, cs, es\rangle. \\
\qquad \langle \mathbb{T}, \quad \langle ss, cs \cup \{c\}, es\rangle, \quad \bot\rangle \\
\quad \text{enroll}(s, c) := \lambda\,\langle ss, cs, es\rangle. \\
\qquad \langle \mathbb{T}, \quad \langle ss, cs, es \cup \{(s, c)\}\rangle, \quad \bot\rangle \\
\quad \text{deleteCourse}(c) := \lambda\,\langle ss, cs, es\rangle. \\
\qquad \langle \mathbb{T}, \quad \langle ss, cs \setminus \{c\}, es\rangle, \quad \bot\rangle \\
\quad \text{query} := \lambda\,\sigma.\ \langle \mathbb{T}, \quad \sigma, \quad \sigma\rangle
\end{array}
$$

$$\text{refIntegrity}(R, f, R', f') := \forall r.\ \ r \in R \rightarrow \exists r'.\ r' \in R' \wedge f(r) = f'(r')$$

Example Class

$$
\begin{array}{l}
\text{Class Courseware} \\
\quad \text{let Student} := \text{Set}\,\langle \text{sid} \colon \text{SId} \rangle \text{ in} \\
\quad \text{let Course} := \text{Set}\,\langle \text{cid} \colon \text{CId} \rangle \text{ in} \\
\quad \text{let Enrolment} := \\
\quad\quad \text{Set}\,\langle \text{esid} \colon \text{SId}, \text{ecid} \colon \text{CId} \rangle \text{ in} \\
\quad \Sigma := \text{Student} \times \text{Course} \times \text{Enrolment} \\
\quad \mathcal{I} := \lambda\,\langle ss, cs, es \rangle. \\
\quad\quad \text{refIntegrity}(es, \text{esid}, ss, \text{sid}) \wedge \\
\quad\quad \text{refIntegrity}(es, \text{ecid}, cs, \text{cid}) \\
\quad \text{register}(s) := \lambda\,\langle ss, cs, es \rangle. \\
\quad\quad \langle \mathbb{T}, \quad \langle ss \cup \{s\}, cs, es \rangle, \quad \bot \rangle \\
\quad \text{addCourse}(c) := \lambda\,\langle ss, cs, es \rangle. \\
\quad\quad \langle \mathbb{T}, \quad \langle ss, cs \cup \{c\}, es \rangle, \quad \bot \rangle \\
\quad \text{enroll}(s, c) := \lambda\,\langle ss, cs, es \rangle. \\
\quad\quad \langle \mathbb{T}, \quad \langle ss, cs, es \cup \{(s, c)\} \rangle, \quad \bot \rangle \\
\quad \text{deleteCourse}(c) := \lambda\,\langle ss, cs, es \rangle. \\
\quad\quad \langle \mathbb{T}, \quad \langle ss, cs \setminus \{c\}, es \rangle, \quad \bot \rangle \\
\quad \text{query} := \lambda\,\sigma.\,\langle \mathbb{T}, \quad \sigma, \quad \sigma \rangle
\end{array}
$$

$$
\text{refIntegrity}(R, f, R', f') := \forall r.\;\; r \in R \rightarrow \exists r'.\; r' \in R' \wedge f(r) = f'(r')
$$

Example Class

$$
\begin{array}{l}
\text{Class Courseware} \\
\quad \text{let Student} := \text{Set}\,\langle \text{sid}\colon \text{SId}\rangle \text{ in} \\
\quad \text{let Course} := \text{Set}\,\langle \text{cid}\colon \text{CId}\rangle \text{ in} \\
\quad \text{let Enrolment} := \\
\qquad \text{Set}\,\langle \text{esid}\colon \text{SId}, \text{ecid}\colon \text{CId}\rangle \text{ in} \\
\quad \Sigma := \text{Student} \times \text{Course} \times \text{Enrolment} \\
\quad \mathcal{I} := \lambda\,\langle ss, cs, es\rangle. \\
\qquad \text{refIntegrity}(es, \text{esid}, ss, \text{sid}) \wedge \\
\qquad \text{refIntegrity}(es, \text{ecid}, cs, \text{cid}) \\
\quad \text{register}(s) := \lambda\,\langle ss, cs, es\rangle. \\
\qquad \langle \mathbb{T}, \quad \langle ss \cup \{s\}, cs, es\rangle, \quad \bot\rangle \\
\quad \text{addCourse}(c) := \lambda\,\langle ss, cs, es\rangle. \\
\qquad \langle \mathbb{T}, \quad \langle ss, cs \cup \{c\}, es\rangle, \quad \bot\rangle \\
\quad \text{enroll}(s, c) := \lambda\,\langle ss, cs, es\rangle. \\
\qquad \langle \mathbb{T}, \quad \langle ss, cs, es \cup \{(s, c)\}\rangle, \quad \bot\rangle \\
\quad \text{deleteCourse}(c) := \lambda\,\langle ss, cs, es\rangle. \\
\qquad \langle \mathbb{T}, \quad \langle ss, cs \setminus \{c\}, es\rangle, \quad \bot\rangle \\
\quad \text{query} := \lambda\,\sigma.\,\langle \mathbb{T}, \quad \sigma, \quad \sigma\rangle
\end{array}
$$

$$
\text{refIntegrity}(R, f, R', f') := \forall r.\;\; r \in R \rightarrow \exists r'.\, r' \in R' \wedge f(r) = f'(r')
$$

Example Class

Class Courseware

$$
\begin{aligned}
&\text{let Student} := \text{Set}\,\langle \text{sid}\colon \text{SId}\rangle \text{ in}\\
&\text{let Course} := \text{Set}\,\langle \text{cid}\colon \text{CId}\rangle \text{ in}\\
&\text{let Enrolment} :=\\
&\quad \text{Set}\,\langle \text{esid}\colon \text{SId}, \text{ecid}\colon \text{CId}\rangle \text{ in}\\
&\Sigma := \text{Student} \times \text{Course} \times \text{Enrolment}\\
&\mathcal{I} := \lambda\,\langle ss, cs, es\rangle.\\
&\quad \text{refIntegrity}(es, \text{esid}, ss, \text{sid}) \land\\
&\quad \text{refIntegrity}(es, \text{ecid}, cs, \text{cid})\\
&\text{register}(s) := \lambda\,\langle ss, cs, es\rangle.\\
&\quad \langle \mathbb{T}, \quad \langle ss \cup \{s\}, cs, es\rangle, \quad \bot\rangle\\
&\text{addCourse}(c) := \lambda\,\langle ss, cs, es\rangle.\\
&\quad \langle \mathbb{T}, \quad \langle ss, cs \cup \{c\}, es\rangle, \quad \bot\rangle\\
&\text{enroll}(s, c) := \lambda\,\langle ss, cs, es\rangle.\\
&\quad \langle \mathbb{T}, \quad \langle ss, cs, es \cup \{(s, c)\}\rangle, \quad \bot\rangle\\
&\text{deleteCourse}(c) := \lambda\,\langle ss, cs, es\rangle.\\
&\quad \langle \mathbb{T}, \quad \langle ss, cs \setminus \{c\}, es\rangle, \quad \bot\rangle\\
&\text{query} := \lambda\,\sigma.\,\langle \mathbb{T}, \quad \sigma, \quad \sigma\rangle
\end{aligned}
$$

$\langle \text{guard}, \text{update}, \text{retv}\rangle$

$$\text{refIntegrity}(R, f, R', f') := \forall r.\;\; r \in R \rightarrow \exists r'.\; r' \in R' \land f(r) = f'(r')$$

Example Class

$$
\begin{array}{l}
\text{Class Courseware} \\
\quad \text{let Student} := \text{Set}\,\langle \text{sid}\colon \text{SId}\rangle \text{ in} \\
\quad \text{let Course} := \text{Set}\,\langle \text{cid}\colon \text{CId}\rangle \text{ in} \\
\quad \text{let Enrolment} := \\
\quad\quad \text{Set}\,\langle \text{esid}\colon \text{SId}, \text{ecid}\colon \text{CId}\rangle \text{ in} \\
\quad \Sigma := \text{Student} \times \text{Course} \times \text{Enrolment} \\
\quad \mathcal{I} := \lambda\,\langle ss, cs, es\rangle. \\
\quad\quad \text{refIntegrity}(es, \text{esid}, ss, \text{sid}) \wedge \\
\quad\quad \text{refIntegrity}(es, \text{ecid}, cs, \text{cid}) \\
\quad \text{register}(s) := \lambda\,\langle ss, cs, es\rangle. \\
\quad\quad \langle \mathbb{T},\quad \boxed{\langle ss \cup \{s\}, cs, es\rangle,}\quad \bot\rangle \\
\quad \text{addCourse}(c) := \lambda\,\langle ss, cs, es\rangle. \\
\quad\quad \langle \mathbb{T},\quad \langle ss, cs \cup \{c\}, es\rangle,\quad \bot\rangle \\
\quad \text{enroll}(s, c) := \lambda\,\langle ss, cs, es\rangle. \\
\quad\quad \langle \mathbb{T},\quad \langle ss, cs, es \cup \{(s, c)\}\rangle,\quad \bot\rangle \\
\quad \text{deleteCourse}(c) := \lambda\,\langle ss, cs, es\rangle. \\
\quad\quad \langle \mathbb{T},\quad \langle ss, cs \setminus \{c\}, es\rangle,\quad \bot\rangle \\
\quad \text{query} := \lambda\,\sigma.\,\langle \mathbb{T},\quad \sigma,\quad \sigma\rangle
\end{array}
$$

$\langle \text{guard}, \boxed{\text{update},} \text{retv}\rangle$

$$\text{refIntegrity}(R, f, R', f') := \forall r.\;\; r \in R \rightarrow \exists r'.\; r' \in R' \wedge f(r) = f'(r')$$

Example Class

$$
\begin{array}{l}
\text{Class Courseware} \\
\quad \text{let Student} := \text{Set}\ \langle \text{sid}\colon \text{SId}\rangle \text{ in} \\
\quad \text{let Course} := \text{Set}\ \langle \text{cid}\colon \text{CId}\rangle \text{ in} \\
\quad \text{let Enrolment} := \\
\qquad \text{Set}\ \langle \text{esid}\colon \text{SId}, \text{ecid}\colon \text{CId}\rangle \text{ in} \\
\quad \Sigma := \text{Student} \times \text{Course} \times \text{Enrolment} \\
\quad \mathcal{I} := \lambda\, \langle ss, cs, es\rangle. \\
\qquad \text{refIntegrity}(es, \text{esid}, ss, \text{sid}) \wedge \\
\qquad \text{refIntegrity}(es, \text{ecid}, cs, \text{cid}) \\
\quad \text{register}(s) := \lambda\, \langle ss, cs, es\rangle. \\
\qquad \langle \mathbb{T}, \quad \langle ss \cup \{s\}, cs, es\rangle, \quad \boxed{\bot\rangle} \\
\quad \text{addCourse}(c) := \lambda\, \langle ss, cs, es\rangle. \\
\qquad \langle \mathbb{T}, \quad \langle ss, cs \cup \{c\}, es\rangle, \quad \bot\rangle \\
\quad \text{enroll}(s, c) := \lambda\, \langle ss, cs, es\rangle. \\
\qquad \langle \mathbb{T}, \quad \langle ss, cs, es \cup \{(s, c)\}\rangle, \quad \bot\rangle \\
\quad \text{deleteCourse}(c) := \lambda\, \langle ss, cs, es\rangle. \\
\qquad \langle \mathbb{T}, \quad \langle ss, cs \setminus \{c\}, es\rangle, \quad \bot\rangle \\
\quad \text{query} := \lambda\, \sigma.\ \langle \mathbb{T}, \quad \sigma, \quad \sigma\rangle
\end{array}
$$

$\langle \text{guard}, \text{update}, \boxed{\text{retv}\rangle}$

$$\text{refIntegrity}(R, f, R', f') := \forall r.\ \ r \in R \rightarrow \exists r'.\ r' \in R' \wedge f(r) = f'(r')$$

Example Class

Class Courseware

$$\begin{aligned}
&\text{let Student} := \text{Set}\,\langle \text{sid}\colon \text{SId}\rangle \text{ in}\\
&\text{let Course} := \text{Set}\,\langle \text{cid}\colon \text{CId}\rangle \text{ in}\\
&\text{let Enrolment} :=\\
&\quad \text{Set}\,\langle \text{esid}\colon \text{SId}, \text{ecid}\colon \text{CId}\rangle \text{ in}\\
&\Sigma := \text{Student} \times \text{Course} \times \text{Enrolment}\\
&\mathcal{I} := \lambda\,\langle ss, cs, es\rangle.\\
&\quad \text{refIntegrity}(es, \text{esid}, ss, \text{sid}) \wedge\\
&\quad \text{refIntegrity}(es, \text{ecid}, cs, \text{cid})\\
&\text{register}(s) := \lambda\,\langle ss, cs, es\rangle.\\
&\quad \langle \mathbb{T},\ \ \langle ss \cup \{s\}, cs, es\rangle,\ \ \bot\rangle\\
&\text{addCourse}(c) := \lambda\,\langle ss, cs, es\rangle.\\
&\quad \langle \mathbb{T},\ \ \langle ss, cs \cup \{c\}, es\rangle,\ \ \bot\rangle\\
&\text{enroll}(s, c) := \lambda\,\langle ss, cs, es\rangle.\\
&\quad \langle \mathbb{T},\ \ \langle ss, cs, es \cup \{(s, c)\}\rangle,\ \ \bot\rangle\\
&\text{deleteCourse}(c) := \lambda\,\langle ss, cs, es\rangle.\\
&\quad \langle \mathbb{T},\ \ \langle ss, cs \setminus \{c\}, es\rangle,\ \ \bot\rangle\\
&\text{query} := \lambda\,\sigma.\,\langle \mathbb{T},\ \ \sigma,\ \ \sigma\rangle
\end{aligned}$$

$\langle \text{guard}, \text{update}, \text{retv}\rangle$

$$\text{refIntegrity}(R, f, R', f') := \forall r.\ \ r \in R \rightarrow \exists r'.\ r' \in R' \wedge f(r) = f'(r')$$

Example Class

Class Courseware

let Student := Set ⟨sid : SId⟩ in

let Course := Set ⟨cid : CId⟩ in

let Enrolment :=

Set ⟨esid : SId, ecid : CId⟩ in

Σ := Student × Course × Enrolment

$\mathcal{I} := \lambda\, \langle ss, cs, es \rangle.$

refIntegrity(es, esid, ss, sid) $\wedge$

refIntegrity(es, ecid, cs, cid)

register(s) $:= \lambda\, \langle ss, cs, es \rangle.$

$\langle \mathbb{T}, \quad \langle ss \cup \{s\}, cs, es \rangle, \quad \bot \rangle$

addCourse(c) $:= \lambda\, \langle ss, cs, es \rangle.$

$\langle \mathbb{T}, \quad \langle ss, cs \cup \{c\}, es \rangle, \quad \bot \rangle$

enroll(s, c) $:= \lambda\, \langle ss, cs, es \rangle.$

$\langle \mathbb{T}, \quad \langle ss, cs, es \cup \{(s, c)\} \rangle, \quad \bot \rangle$

deleteCourse(c) $:= \lambda\, \langle ss, cs, es \rangle.$

$\langle \mathbb{T}, \quad \langle ss, cs \setminus \{c\}, es \rangle, \quad \bot \rangle$

query $:= \lambda\, \sigma.\, \langle \mathbb{T}, \quad \sigma, \quad \sigma \rangle$

⟨guard, update, retv⟩

refIntegrity(R, f, R', f') $:= \forall r.\;\; r \in R \rightarrow \exists r'.\; r' \in R' \wedge f(r) = f'(r')$

Example Class

$$
\begin{array}{l}
\text{Class Courseware} \\
\quad \text{let Student} := \text{Set}\ \langle \text{sid}\colon \text{SId}\rangle\ \text{in} \\
\quad \text{let Course} := \text{Set}\ \langle \text{cid}\colon \text{CId}\rangle\ \text{in} \\
\quad \text{let Enrolment} := \\
\quad\quad \text{Set}\ \langle \text{esid}\colon \text{SId}, \text{ecid}\colon \text{CId}\rangle\ \text{in} \\
\quad \Sigma := \text{Student} \times \text{Course} \times \text{Enrolment} \\
\quad \mathcal{I} := \lambda\ \langle ss, cs, es\rangle. \\
\quad\quad \text{refIntegrity}(es, \text{esid}, ss, \text{sid}) \wedge \\
\quad\quad \text{refIntegrity}(es, \text{ecid}, cs, \text{cid}) \\
\quad \text{register}(s) := \lambda\ \langle ss, cs, es\rangle. \\
\quad\quad \langle \mathbb{T},\quad \langle ss \cup \{s\}, cs, es\rangle,\quad \bot\rangle \\
\quad \text{addCourse}(c) := \lambda\ \langle ss, cs, es\rangle. \\
\quad\quad \langle \mathbb{T},\quad \langle ss, cs \cup \{c\}, es\rangle,\quad \bot\rangle \\
\quad \text{enroll}(s, c) := \lambda\ \langle ss, cs, es\rangle. \\
\quad\quad \langle \mathbb{T},\quad \langle ss, cs, es \cup \{(s, c)\}\rangle,\quad \bot\rangle \\
\quad \text{deleteCourse}(c) := \lambda\ \langle ss, cs, es\rangle. \\
\quad\quad \langle \mathbb{T},\quad \langle ss, cs \setminus \{c\}, es\rangle,\quad \bot\rangle \\
\quad \text{query} := \lambda\ \sigma.\ \langle \mathbb{T},\quad \sigma,\quad \sigma\rangle
\end{array}
$$

$\langle \text{guard}, \text{update}, \text{retv}\rangle$

$$\text{refIntegrity}(R, f, R', f') := \forall r.\ \ r \in R \rightarrow \exists r'.\ r' \in R' \wedge f(r) = f'(r')$$

Example Class

$$
\begin{array}{l}
\text{Class Courseware} \\
\quad \text{let Student} := \text{Set}\,\langle \text{sid}\colon \text{SId}\rangle \text{ in} \\
\quad \text{let Course} := \text{Set}\,\langle \text{cid}\colon \text{CId}\rangle \text{ in} \\
\quad \text{let Enrolment} := \\
\qquad \text{Set}\,\langle \text{esid}\colon \text{SId}, \text{ecid}\colon \text{CId}\rangle \text{ in} \\
\quad \Sigma := \text{Student} \times \text{Course} \times \text{Enrolment} \\
\quad \mathcal{I} := \lambda\,\langle ss, cs, es\rangle. \\
\qquad \text{refIntegrity}(es, \text{esid}, ss, \text{sid}) \wedge \\
\qquad \text{refIntegrity}(es, \text{ecid}, cs, \text{cid}) \\
\quad \text{register}(s) := \lambda\,\langle ss, cs, es\rangle. \\
\qquad \langle \mathbb{T}, \quad \langle ss \cup \{s\}, cs, es\rangle, \quad \bot\rangle \\
\quad \text{addCourse}(c) := \lambda\,\langle ss, cs, es\rangle. \\
\qquad \langle \mathbb{T}, \quad \langle ss, cs \cup \{c\}, es\rangle, \quad \bot\rangle \\
\quad \text{enroll}(s, c) := \lambda\,\langle ss, cs, es\rangle. \\
\qquad \langle \mathbb{T}, \quad \langle ss, cs, es \cup \{(s, c)\}\rangle, \quad \bot\rangle \\
\quad \text{deleteCourse}(c) := \lambda\,\langle ss, cs, es\rangle. \\
\qquad \langle \mathbb{T}, \quad \langle ss, cs \setminus \{c\}, es\rangle, \quad \bot\rangle \\
\quad \text{query} := \lambda\,\sigma.\,\langle \mathbb{T}, \quad \sigma, \quad \sigma\rangle
\end{array}
$$

$\langle \text{guard}, \text{update}, \text{retv}\rangle$

$$\text{refIntegrity}(R, f, R', f') := \forall r.\ r \in R \rightarrow \exists r'.\ r' \in R' \wedge f(r) = f'(r')$$

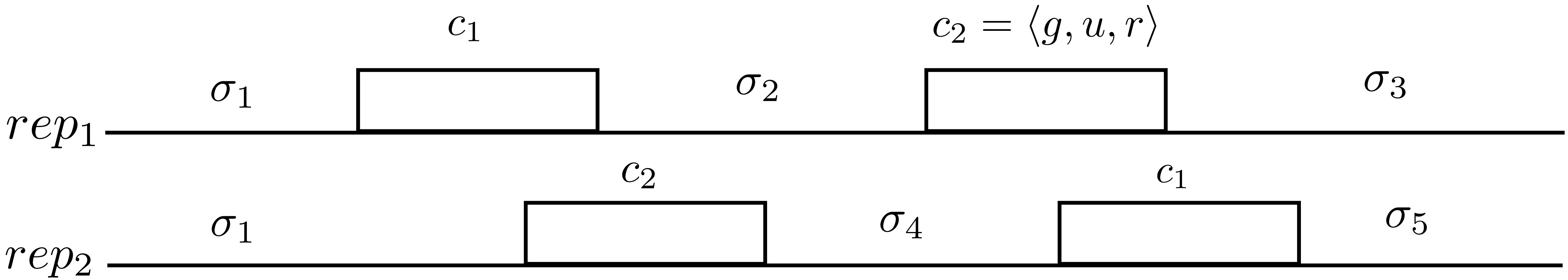
Example Class

$$
\begin{array}{l}
\text{Class Courseware} \\
\quad \text{let Student} := \text{Set}\,\langle \text{sid}\colon \text{SId}\rangle \text{ in} \\
\quad \text{let Course} := \text{Set}\,\langle \text{cid}\colon \text{CId}\rangle \text{ in} \\
\quad \text{let Enrolment} := \\
\qquad \text{Set}\,\langle \text{esid}\colon \text{SId}, \text{ecid}\colon \text{CId}\rangle \text{ in} \\
\quad \Sigma := \text{Student} \times \text{Course} \times \text{Enrolment} \\
\quad \mathcal{I} := \lambda\,\langle ss, cs, es\rangle. \\
\qquad \text{refIntegrity}(es, \text{esid}, ss, \text{sid}) \wedge \\
\qquad \text{refIntegrity}(es, \text{ecid}, cs, \text{cid}) \\
\quad \text{register}(s) := \lambda\,\langle ss, cs, es\rangle. \\
\qquad \langle \mathbb{T},\quad \langle ss \cup \{s\}, cs, es\rangle,\quad \bot\rangle \\
\quad \text{addCourse}(c) := \lambda\,\langle ss, cs, es\rangle. \\
\qquad \langle \mathbb{T},\quad \langle ss, cs \cup \{c\}, es\rangle,\quad \bot\rangle \\
\quad \text{enroll}(s, c) := \lambda\,\langle ss, cs, es\rangle. \\
\qquad \langle \mathbb{T},\quad \langle ss, cs, es \cup \{(s, c)\}\rangle,\quad \bot\rangle \\
\quad \text{deleteCourse}(c) := \lambda\,\langle ss, cs, es\rangle. \\
\qquad \langle \mathbb{T},\quad \langle ss, cs \setminus \{c\}, es\rangle,\quad \bot\rangle \\
\quad \text{query} := \lambda\,\sigma.\,\langle \mathbb{T},\quad \sigma,\quad \sigma\rangle
\end{array}
$$

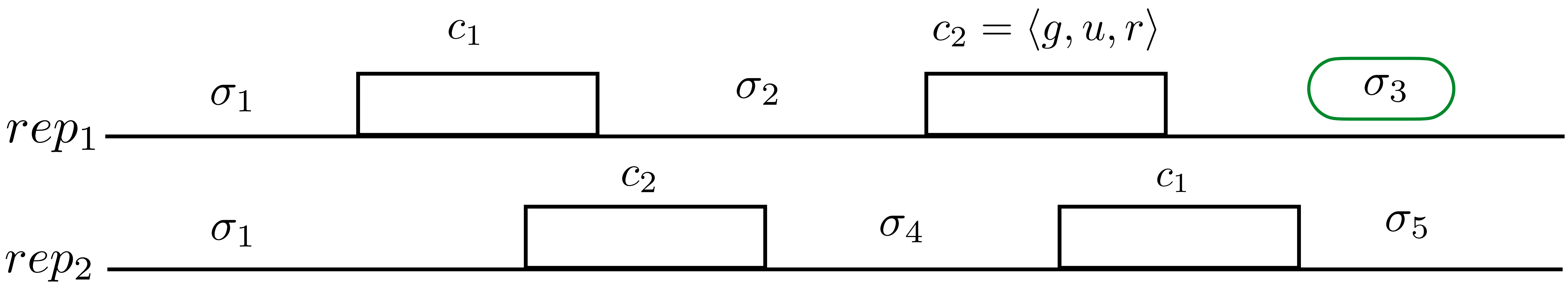
$\langle \text{guard}, \text{update}, \text{retv}\rangle$

$$\text{refIntegrity}(R, f, R', f') := \forall r.\;\; r \in R \rightarrow \exists r'.\; r' \in R' \wedge f(r) = f'(r')$$

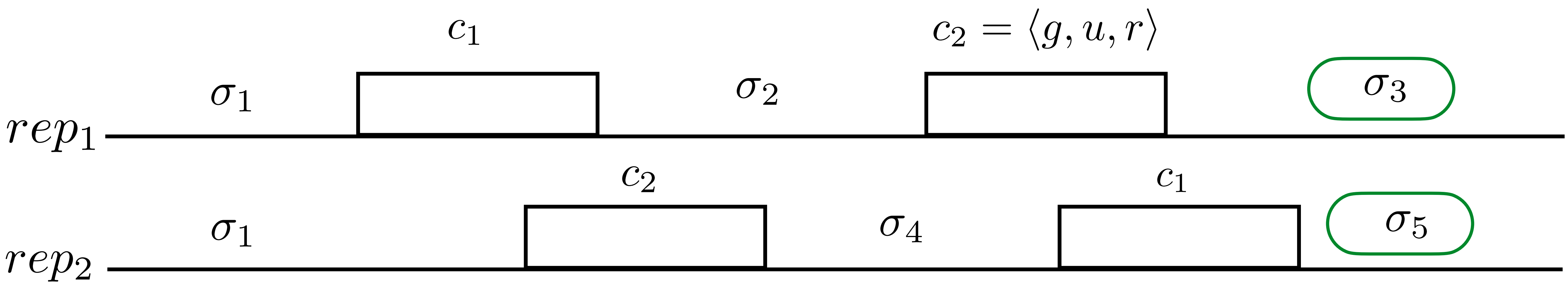
Convergence



Convergence



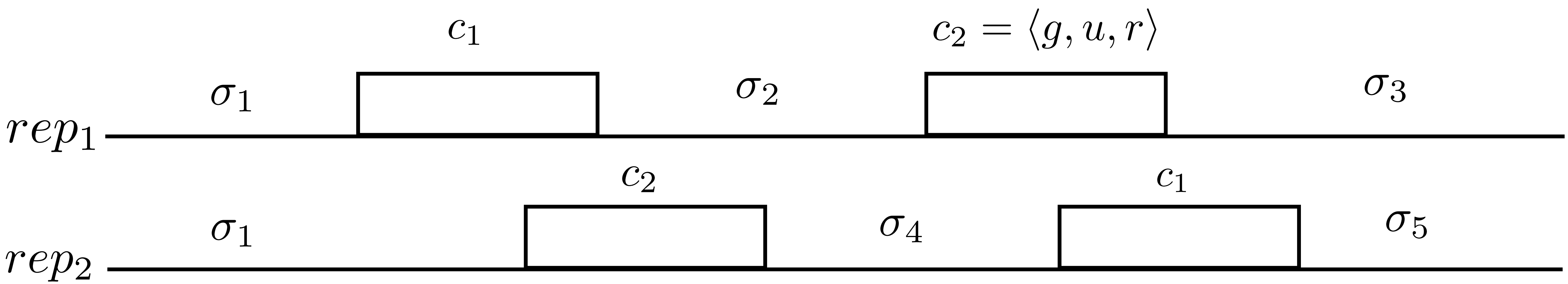
Convergence



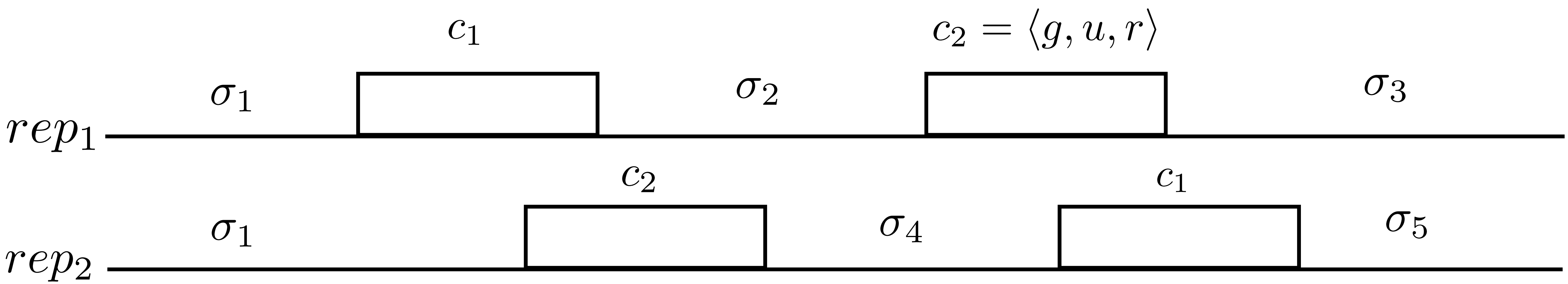
Convergence and Integrity

Convergence

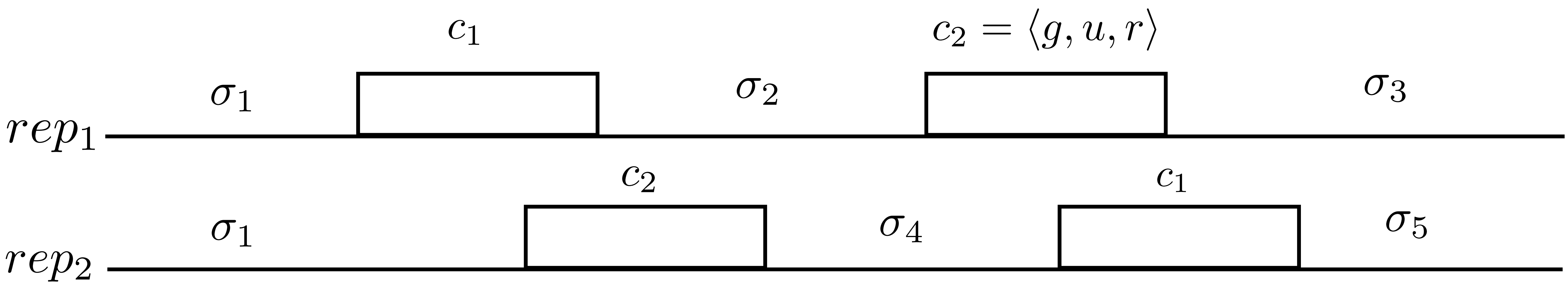
$\sigma_3 = \sigma_5$



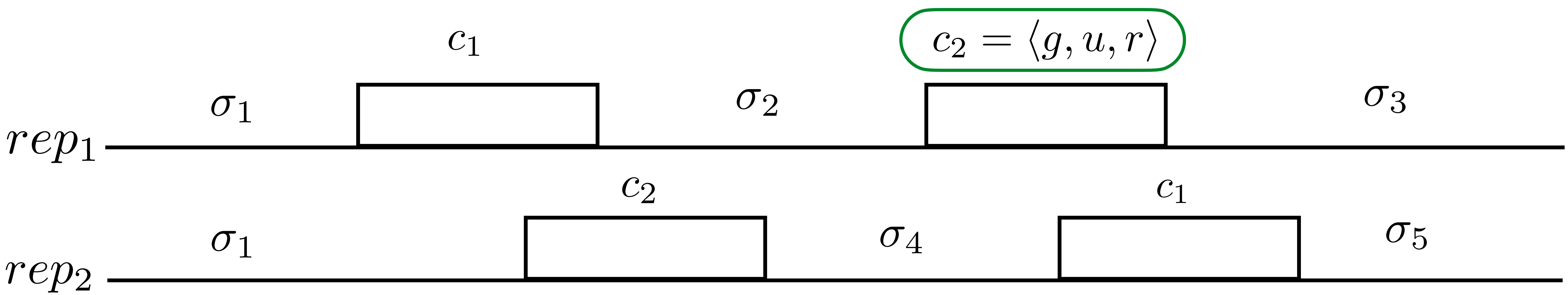
Convergence



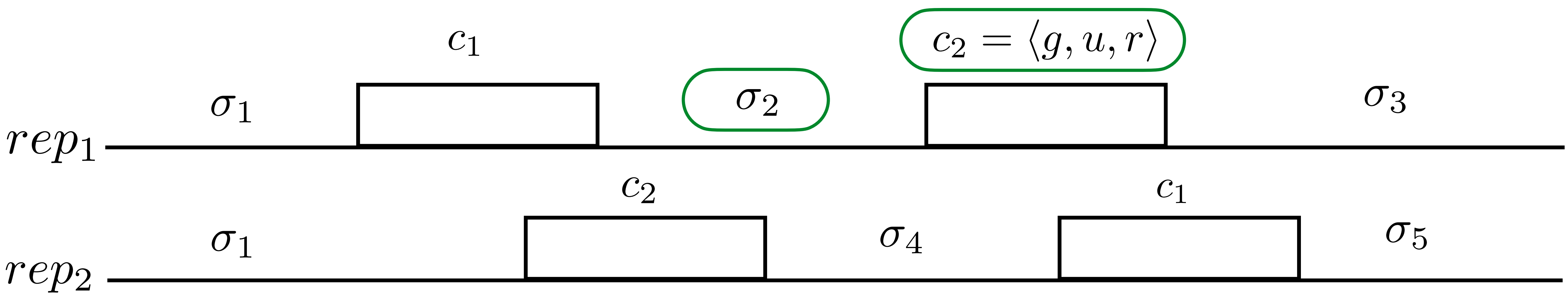
Integrity


c_1
$c_2 = \langle g, u, r \rangle$
σ_1
σ_2
σ_3
rep_1
c_2
c_1
σ_1
σ_4
σ_5
rep_2

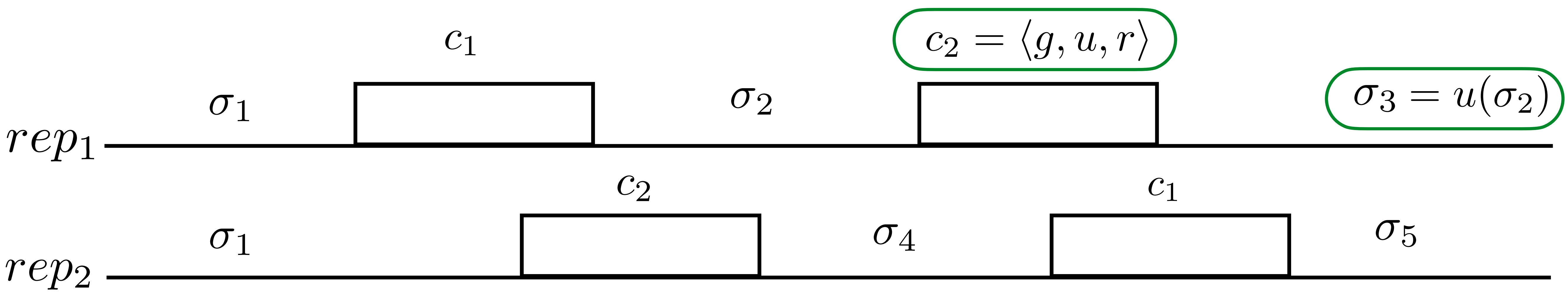
Integrity


c_1
$c_2 = \langle g, u, r \rangle$
σ_1
σ_2
σ_3
rep_1
c_2
c_1
σ_1
σ_4
σ_5
rep_2

Integrity


c_1
$c_2 = \langle g, u, r \rangle$
σ_1
σ_2
σ_3
rep_1
c_2
c_1
σ_1
σ_4
σ_5
rep_2

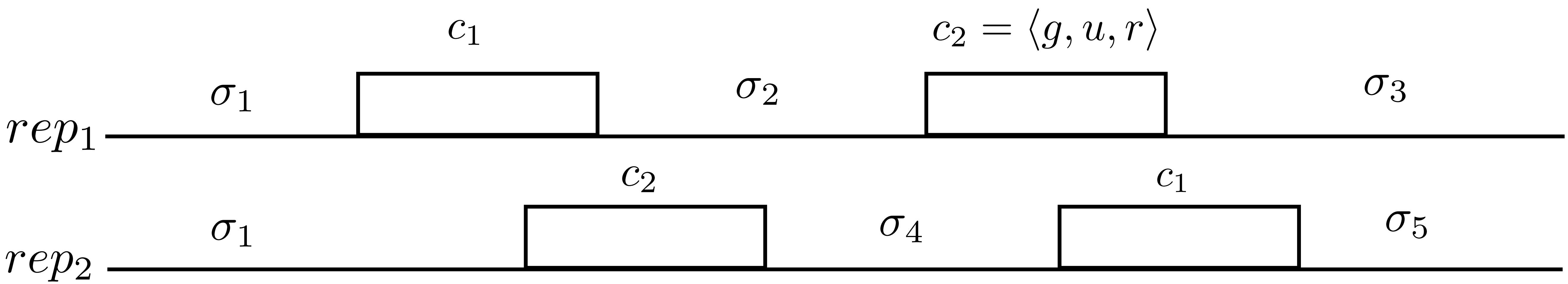
Integrity


c_1
$c_2 = \langle g, u, r \rangle$
σ_1
σ_2
$\sigma_3 = u(\sigma_2)$
rep_1
c_2
c_1
σ_1
σ_4
σ_5
rep_2

Convergence and Integrity

Integrity

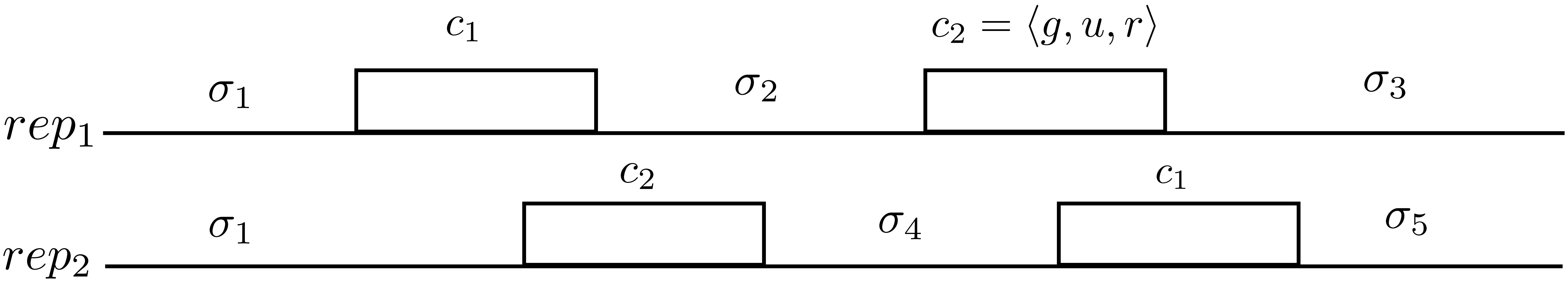
$$\mathcal{C}(\sigma_2, c_2) = g(\sigma_2) \wedge \mathcal{I}(\sigma_2)$$



Convergence and Integrity

Integrity

$$\mathcal{C}(\sigma_2, c_2) = g(\sigma_2) \wedge \mathcal{I}(\sigma_2)$$

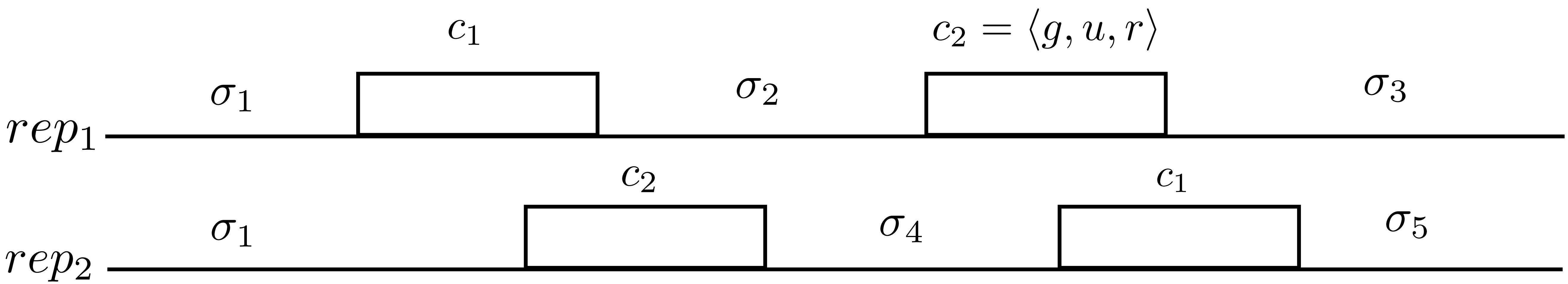


Convergence and Integrity

Integrity

Permissibility

$$\mathcal{C}(\sigma_2, c_2) = g(\sigma_2) \wedge \mathcal{I}(\sigma_2)$$



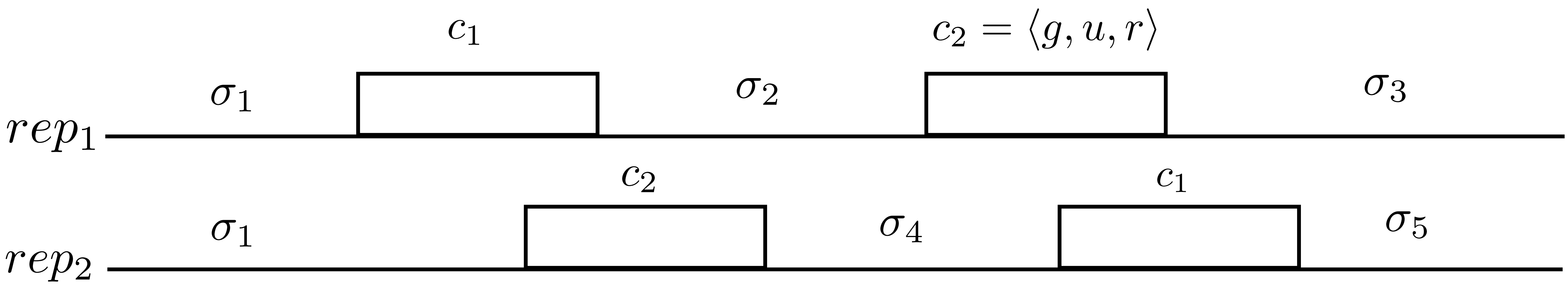
Convergence and Integrity

Integrity

Permissibility

$\mathcal{C}(\sigma_2, c_2) =$
$g(\sigma_2) \wedge$
$\mathcal{I}(\sigma_2)$

$\mathcal{P}(\sigma_2, c_2) =$
$g(\sigma_2) \wedge$
$\mathcal{I}(u(\sigma_2))$



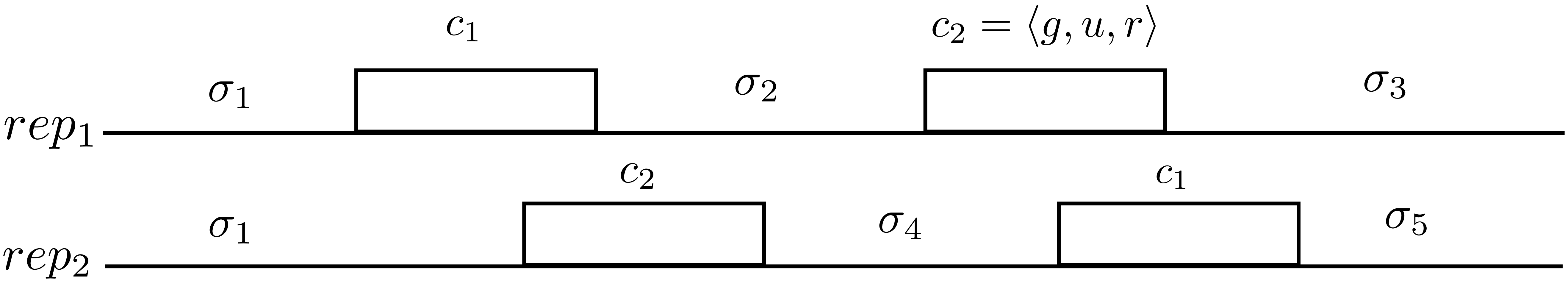
Convergence and Integrity

Integrity

Permissibility

$\mathcal{C}(\sigma_2, c_2) = g(\sigma_2) \wedge \mathcal{I}(\sigma_2)$

$\mathcal{P}(\sigma_2, c_2) = g(\sigma_2) \wedge \mathcal{I}(u(\sigma_2))$

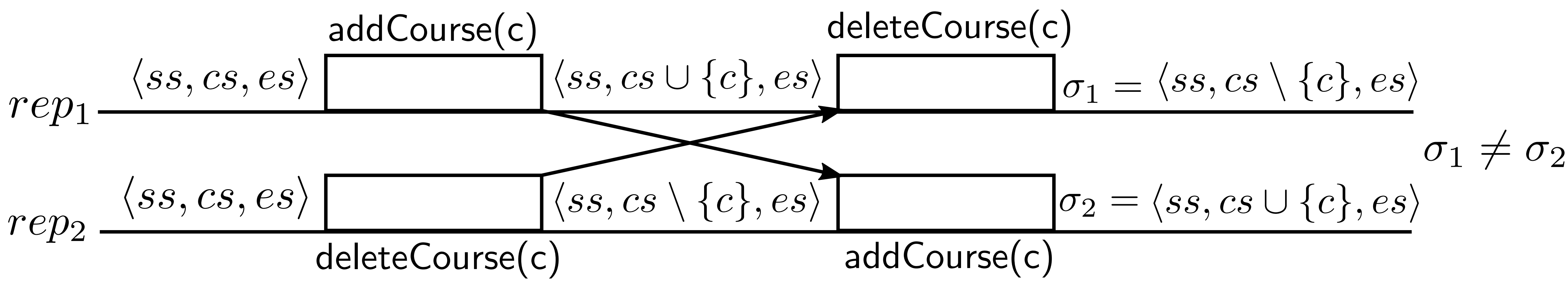


Coordination Conditions as Commutativity Conditions

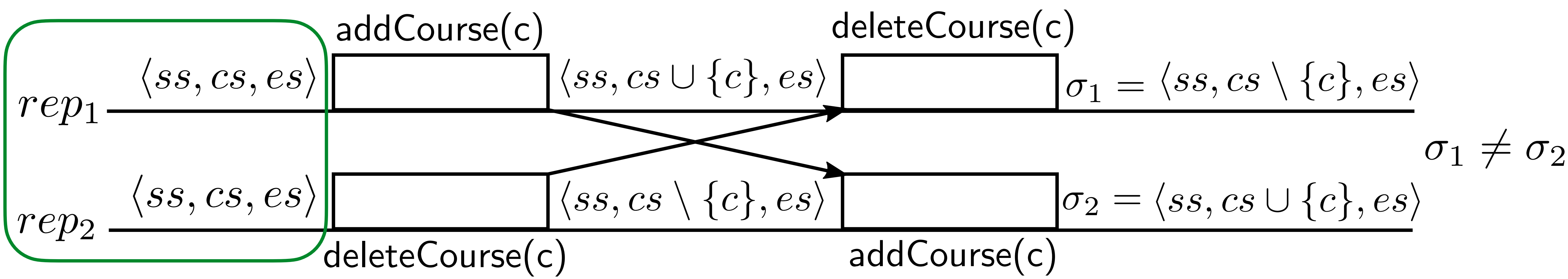
Conflict
Dependency

1 State-Conflict

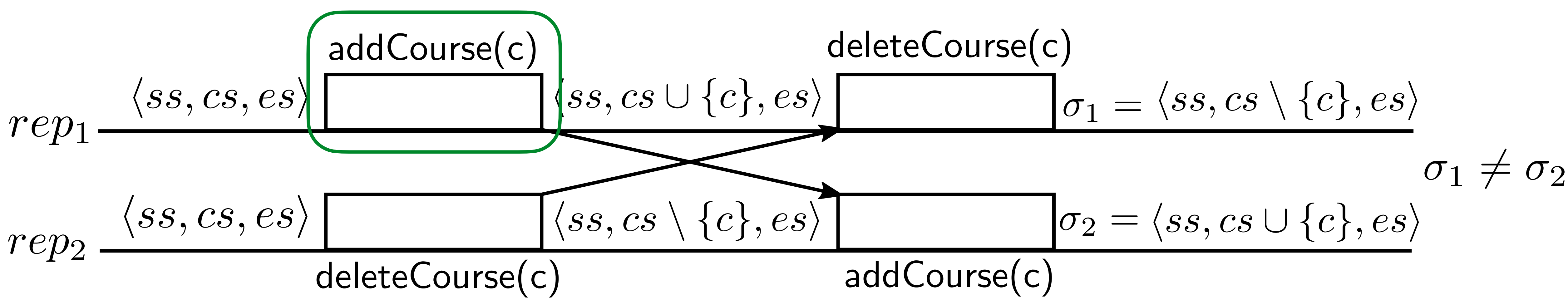
$\mathcal{S}$-conflict



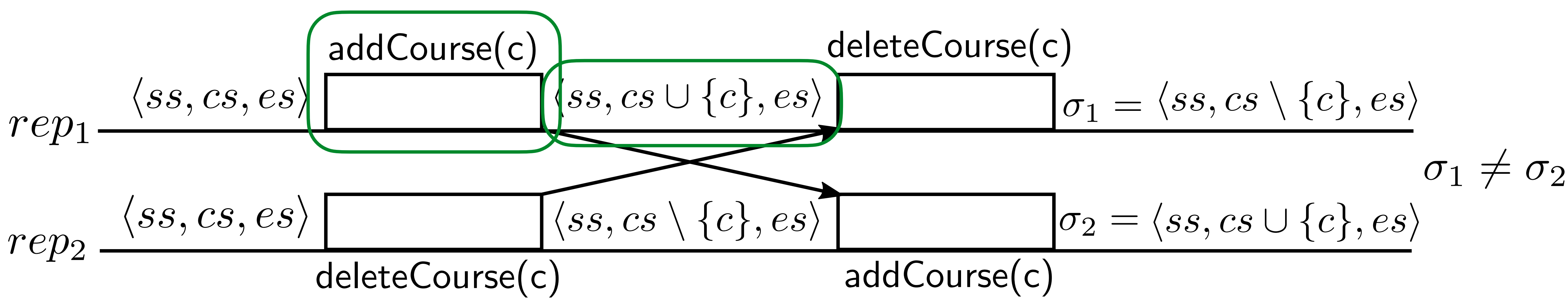
$\mathcal{S}$-conflict



$\mathcal{S}$-conflict

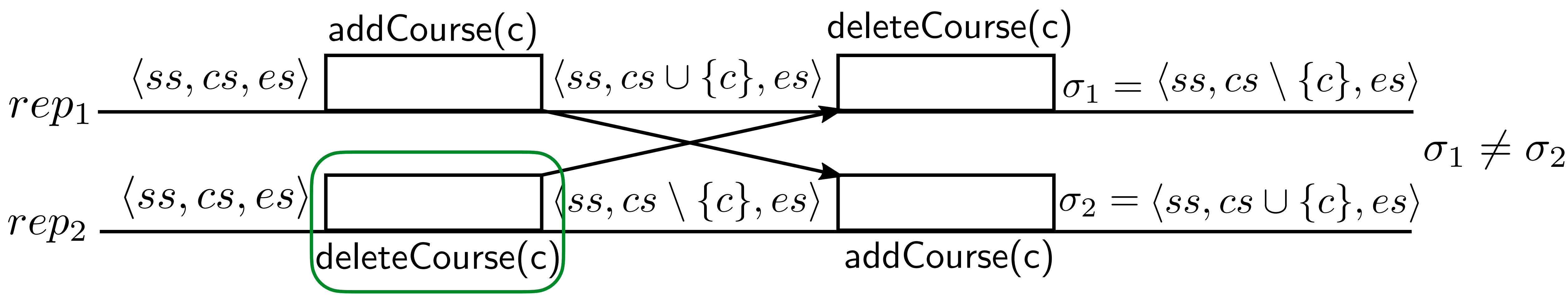


$\mathcal{S}$-conflict

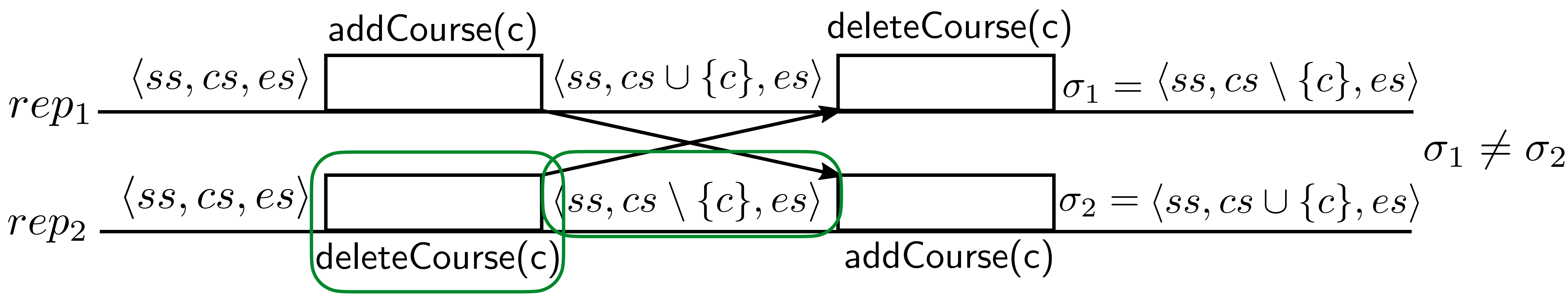


① State-Conflict

$\mathcal{S}$-conflict

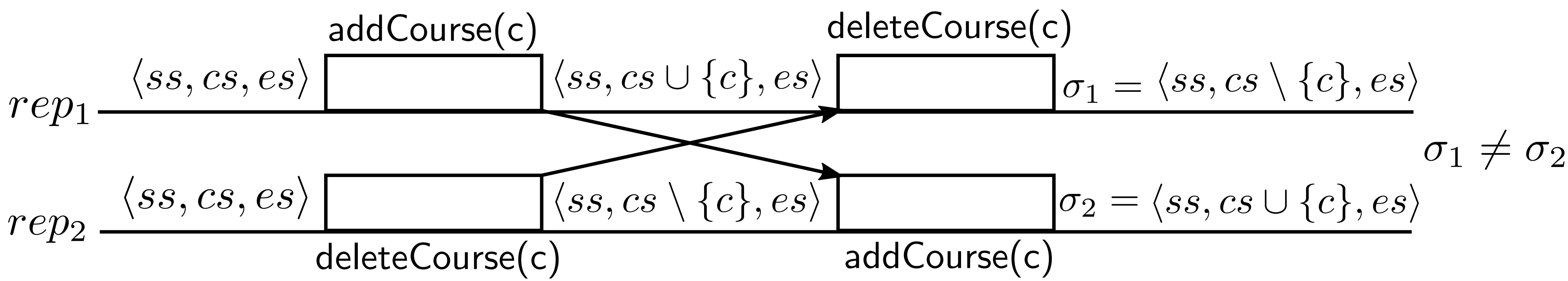


$\mathcal{S}$-conflict

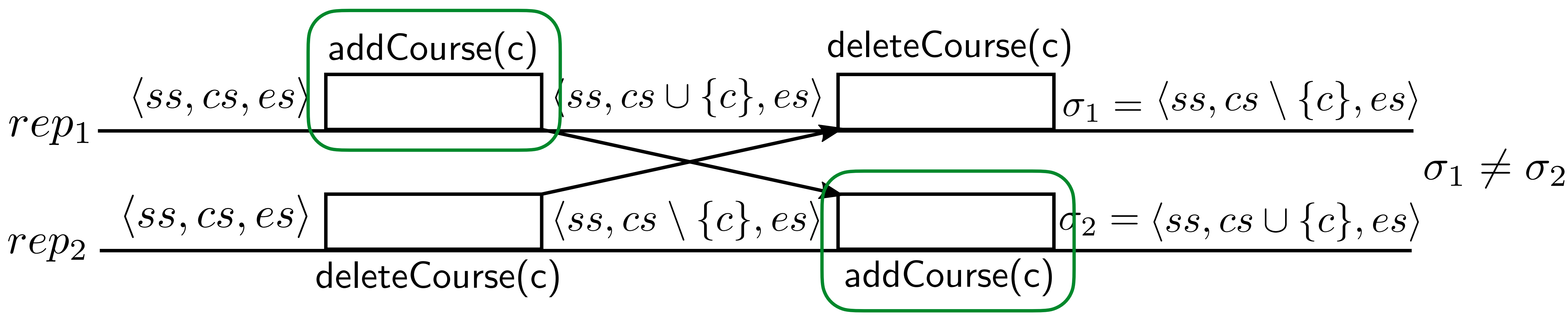


1 State-Conflict

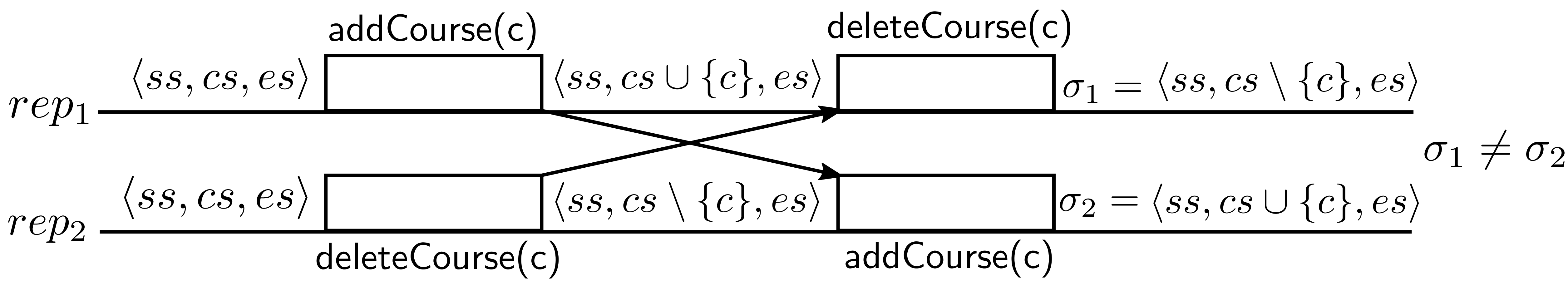
$\mathcal{S}$-conflict



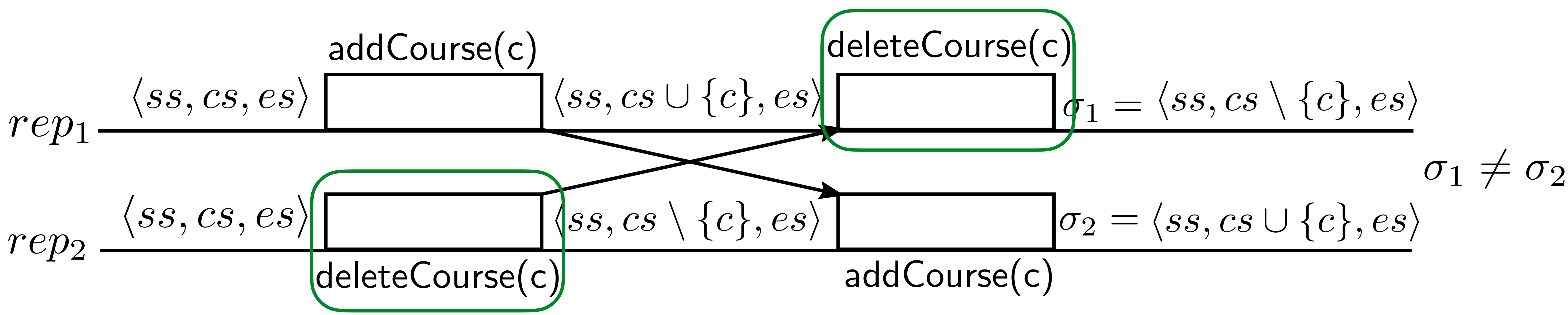
$\mathcal{S}$-conflict



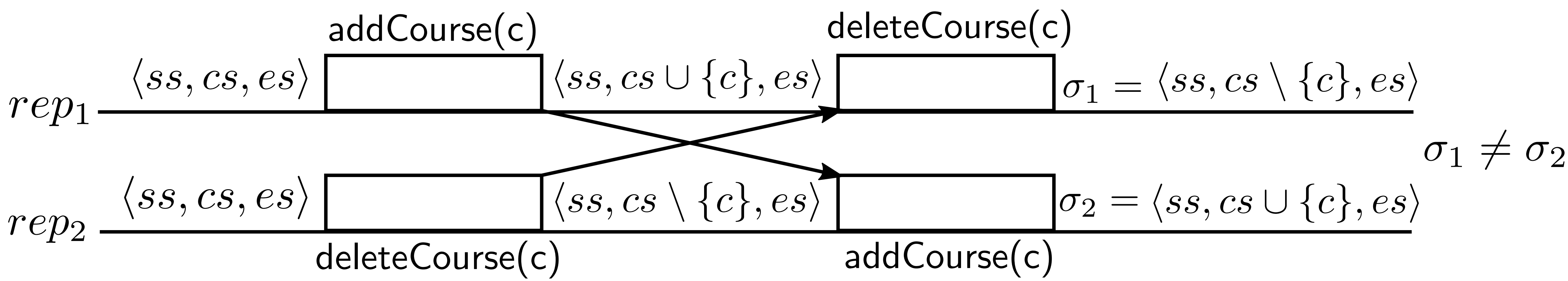
$\mathcal{S}$-conflict



$\mathcal{S}$-conflict

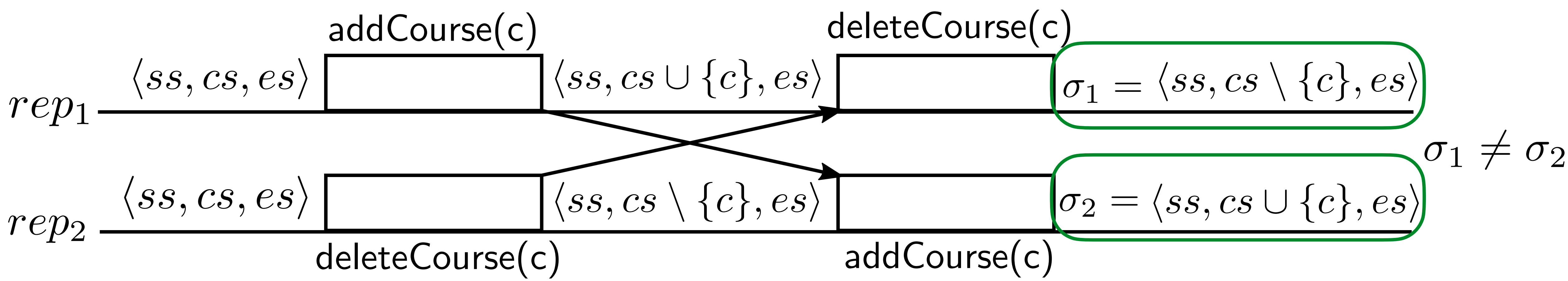


$\mathcal{S}$-conflict

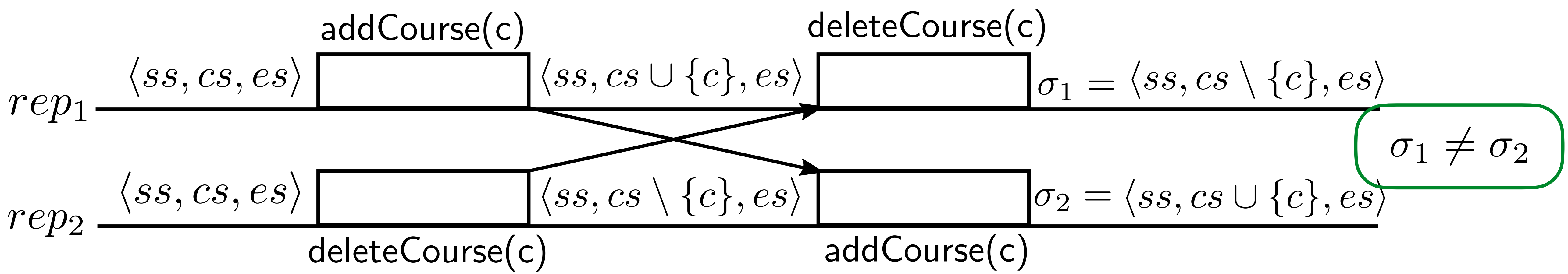


1 State-Conflict

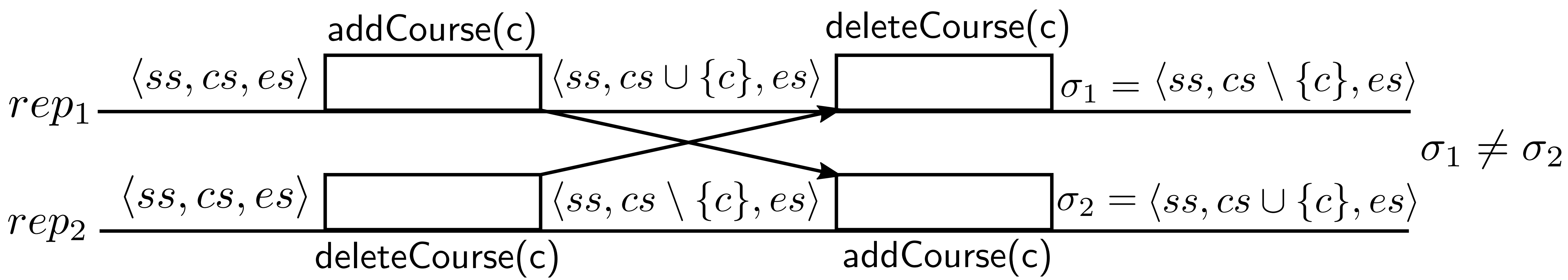
$\mathcal{S}$-conflict



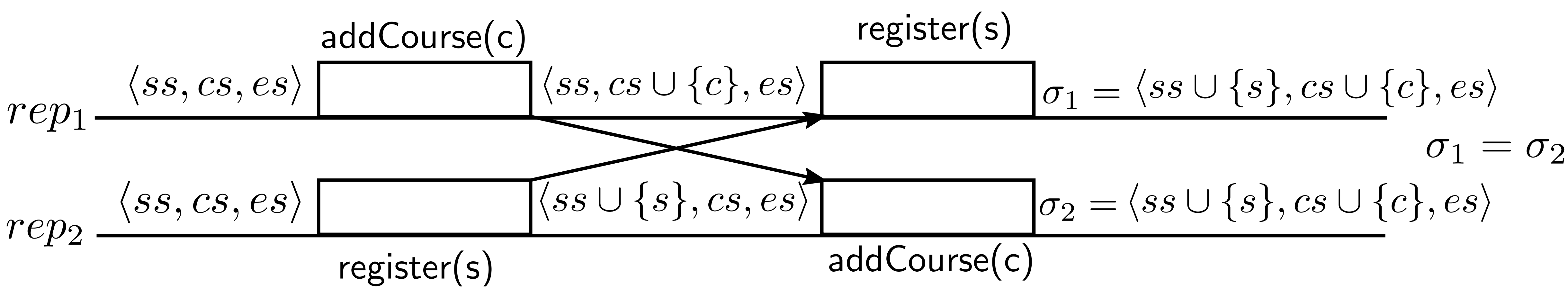
$\mathcal{S}$-conflict



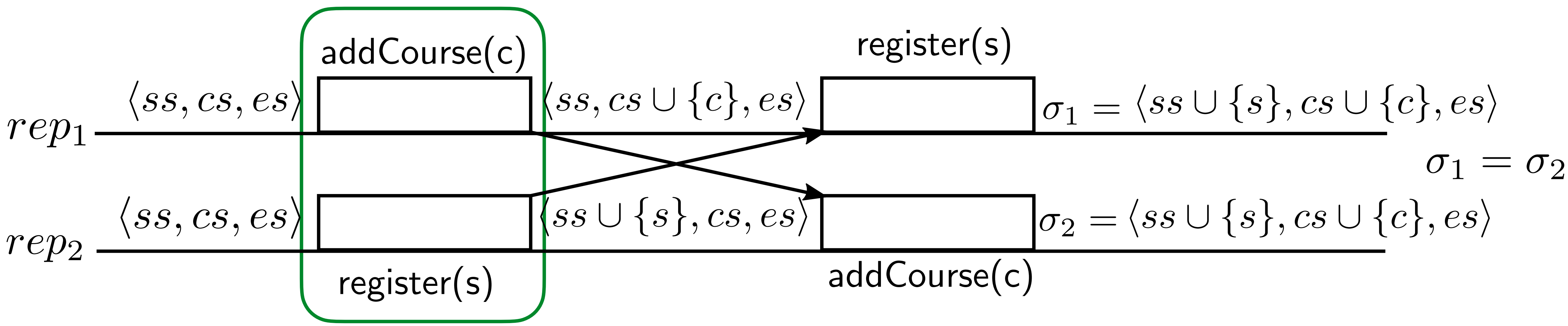
$\mathcal{S}$-conflict



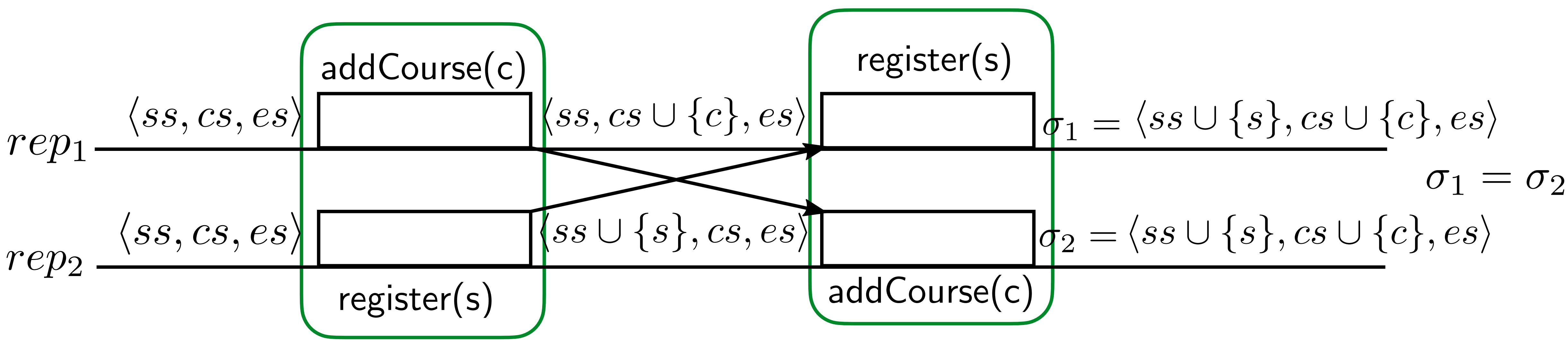
$\mathcal{S}$-commute



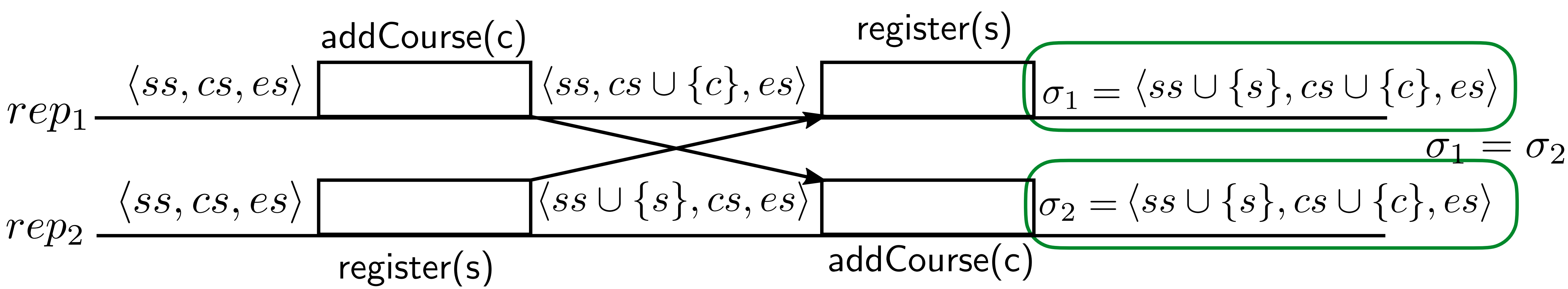
$\mathcal{S}$-commute



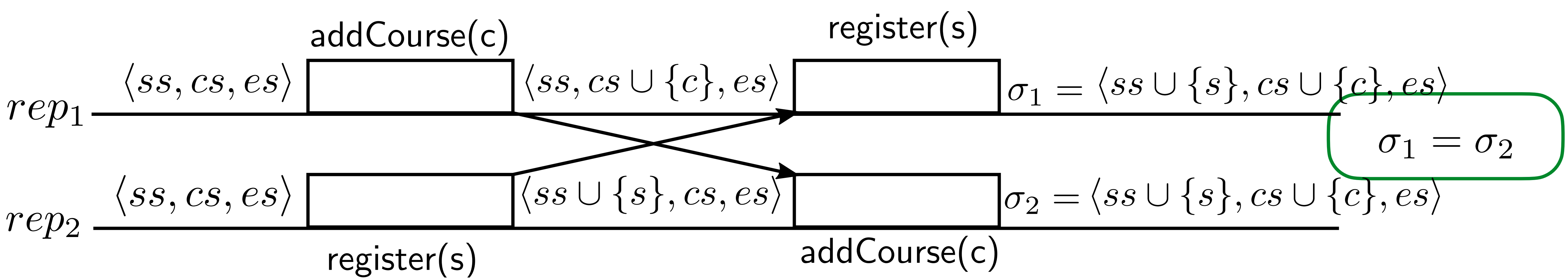
$\mathcal{S}$-commute



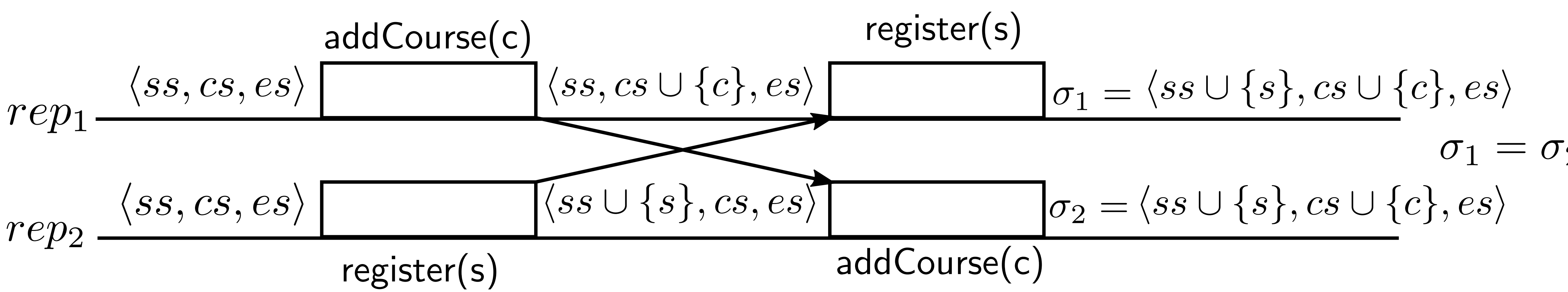
$\mathcal{S}$-commute



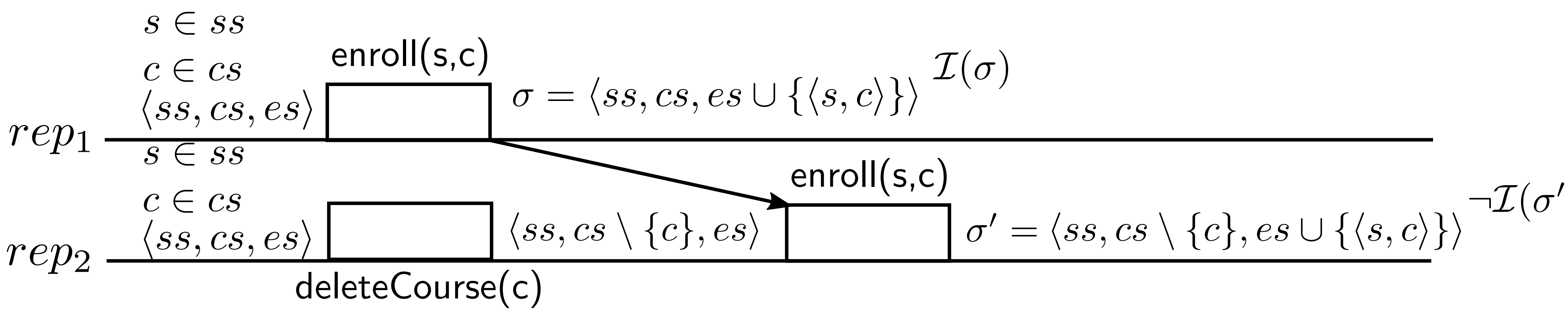
$\mathcal{S}$-commute



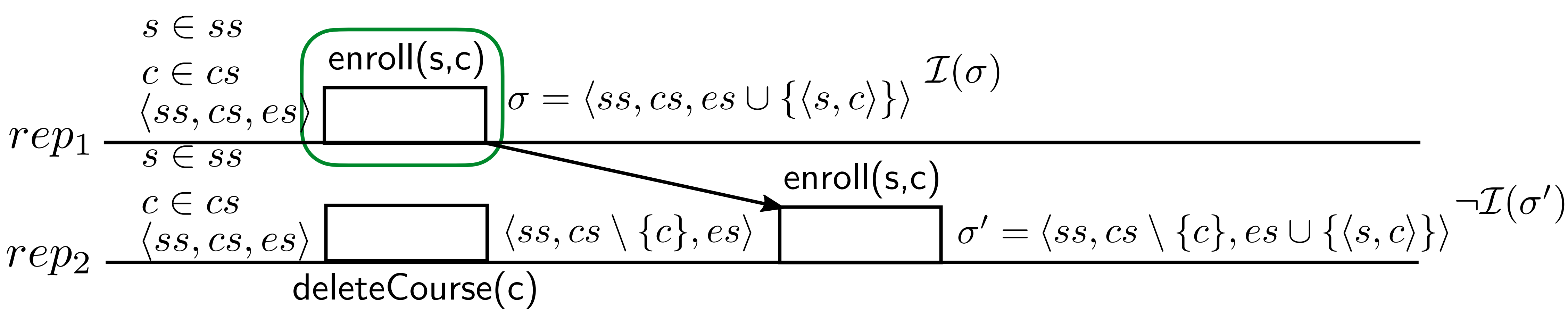
$\mathcal{S}$-commute



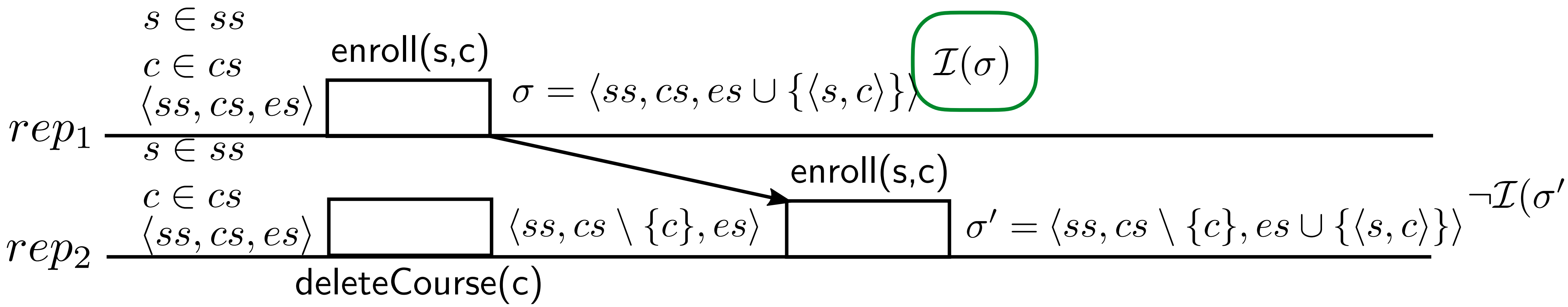
$\mathcal{P}$-conflict



$\mathcal{P}$-conflict

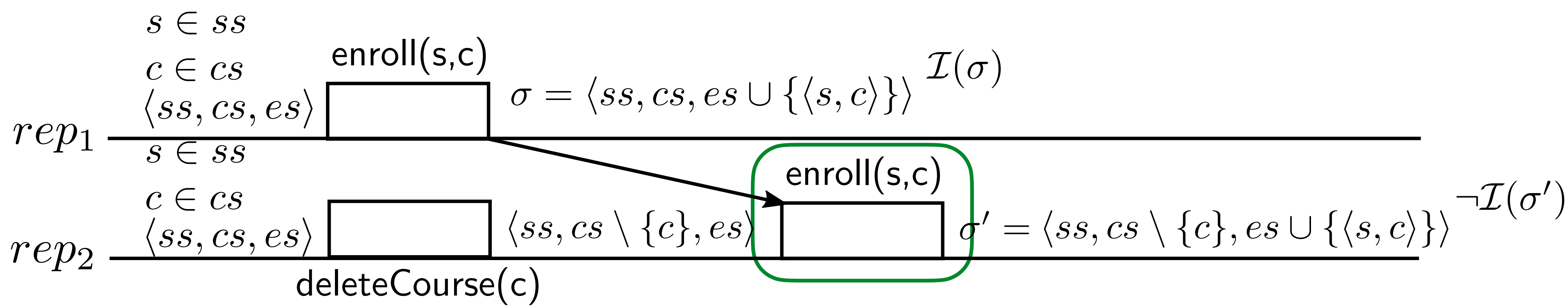


$\mathcal{P}$-conflict

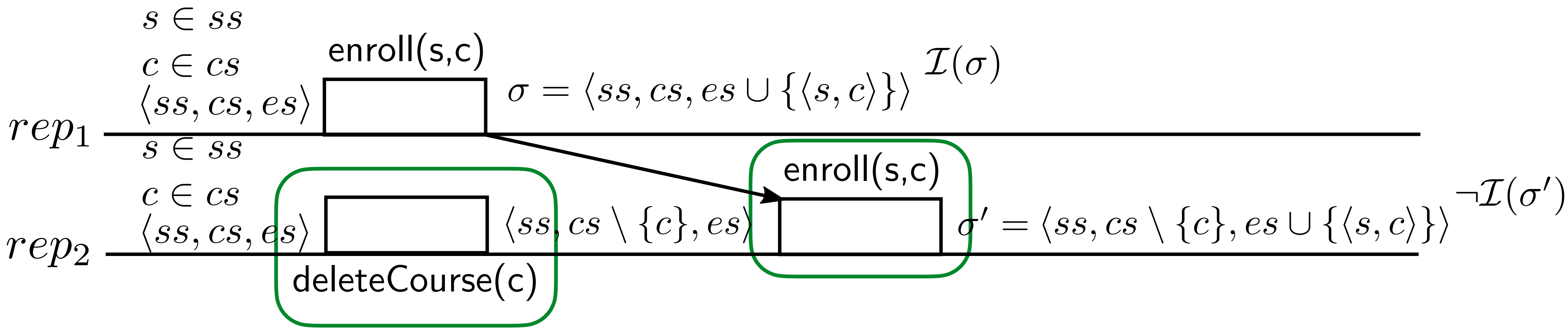


2 Permissible-Conflict

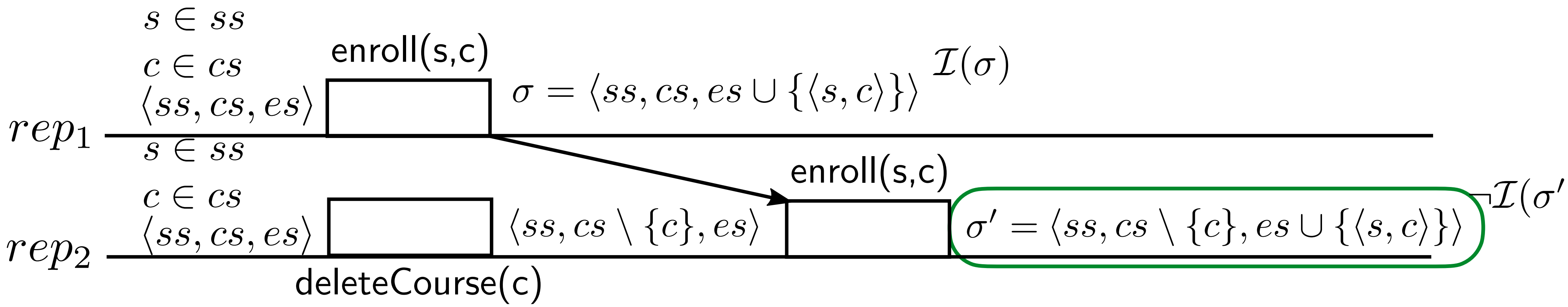
$\mathcal{P}$-conflict



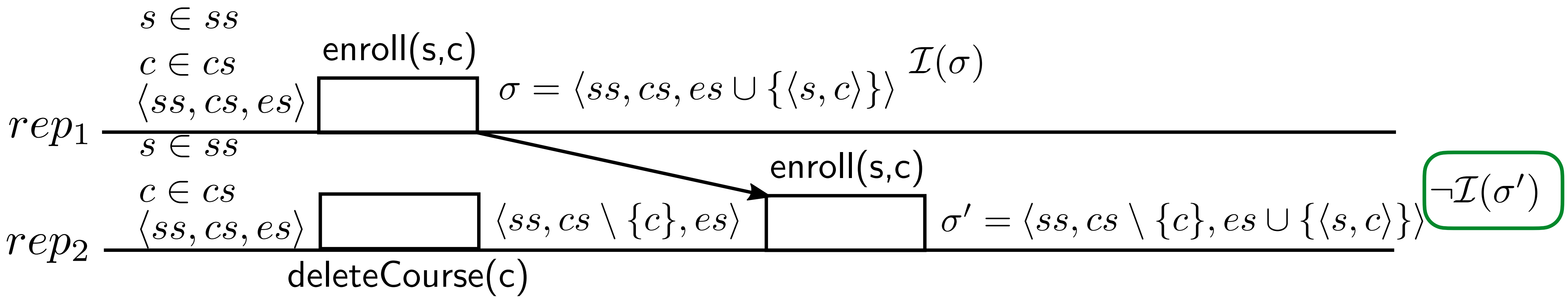
$\mathcal{P}$-conflict



$\mathcal{P}$-conflict

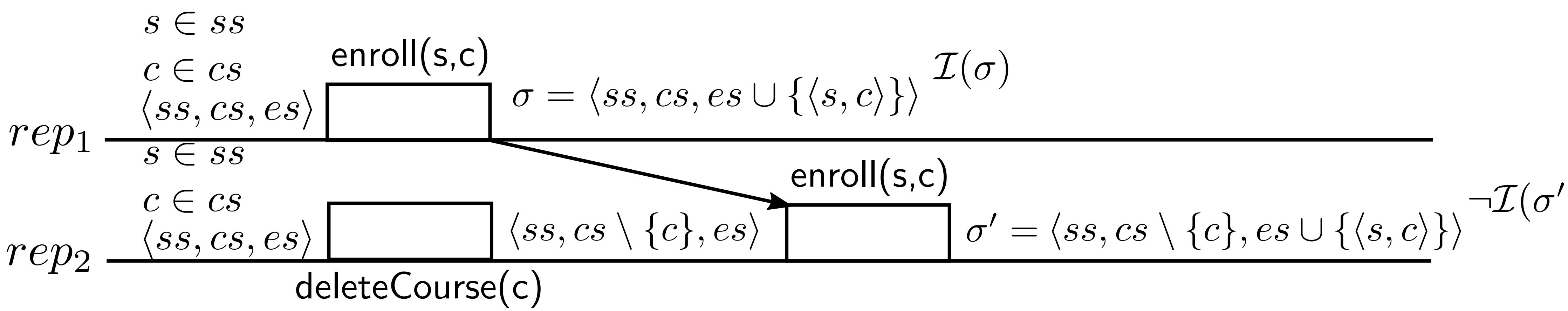


$\mathcal{P}$-conflict

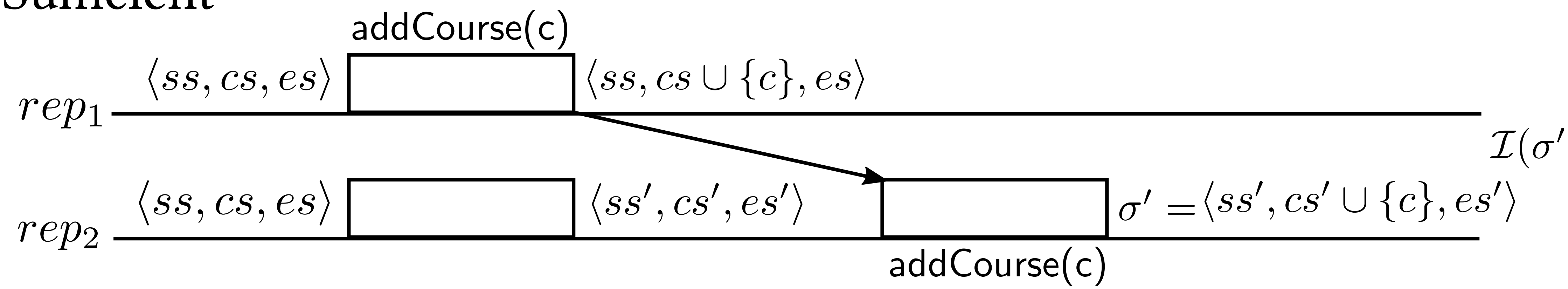


2 Permissible-Conflict

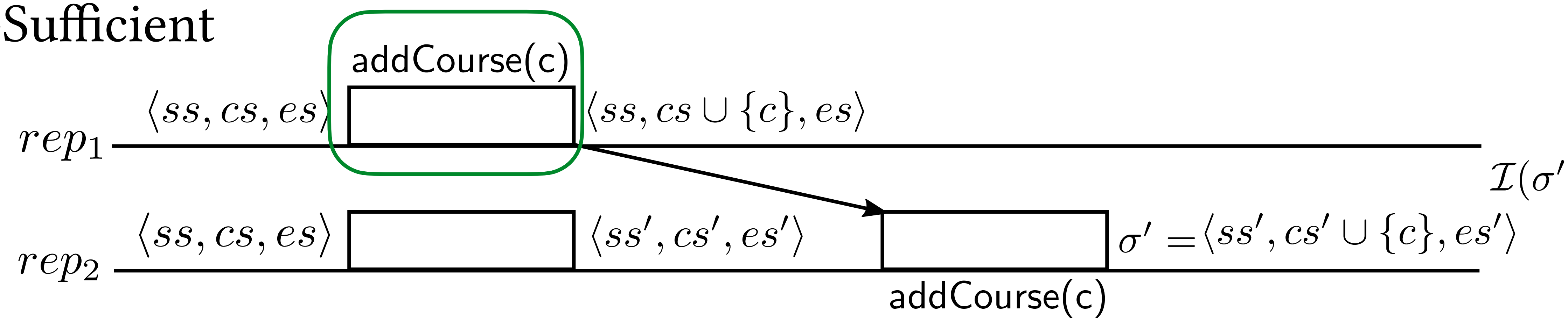
$\mathcal{P}$-conflict



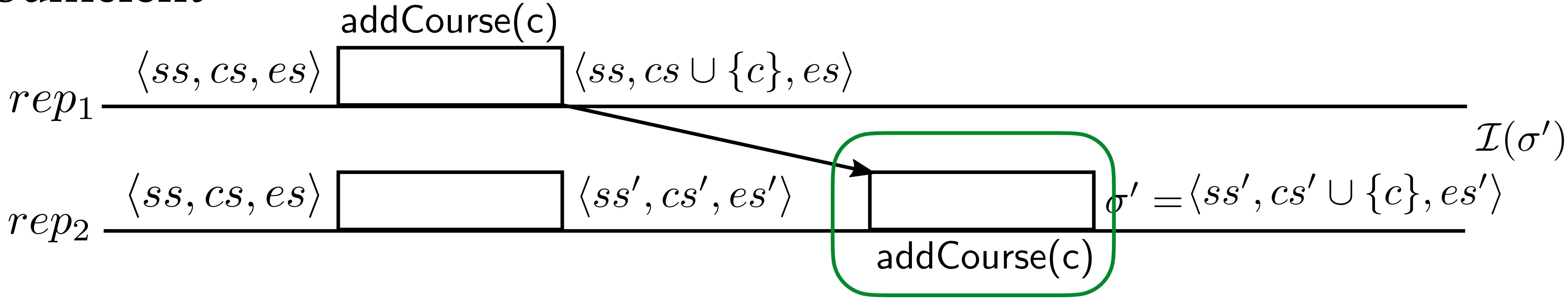
$\mathcal{I}$-Sufficient



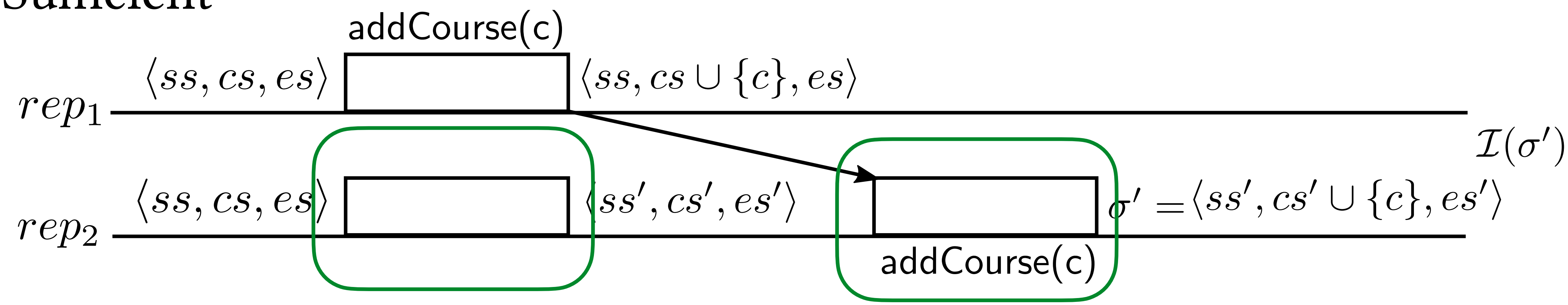
$\mathcal{I}$-Sufficient



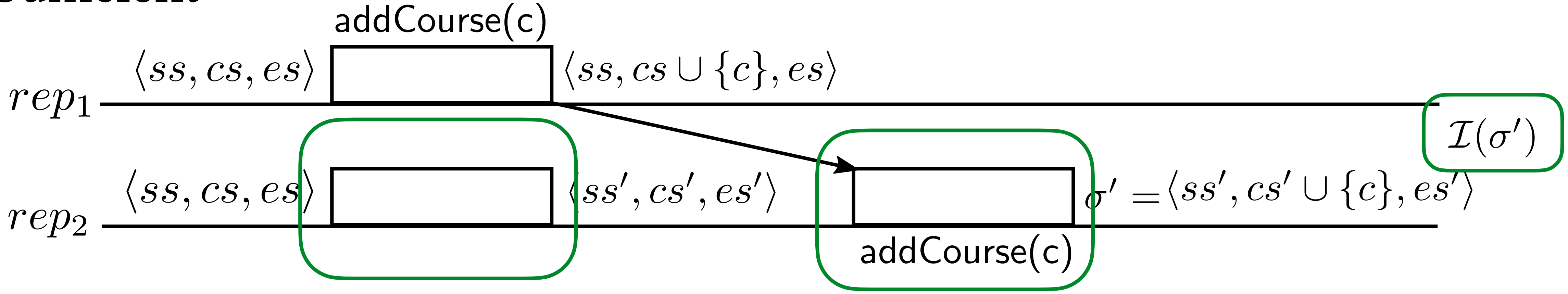
$\mathcal{I}$-Sufficient



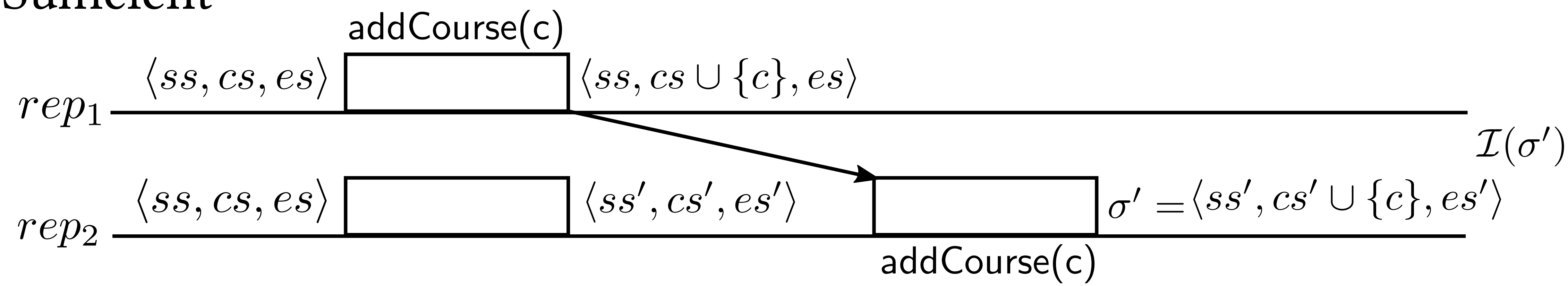
$\mathcal{I}$-Sufficient



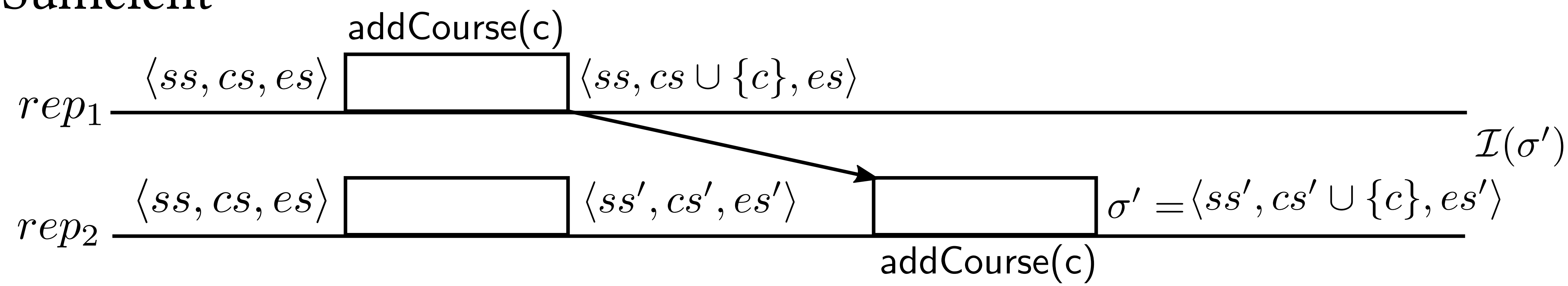
$\mathcal{I}$-Sufficient



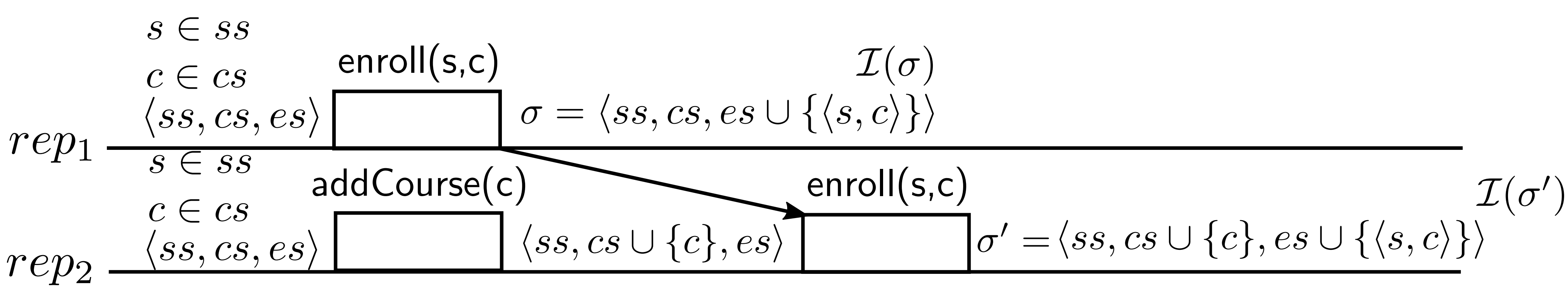
$\mathcal{I}$-Sufficient



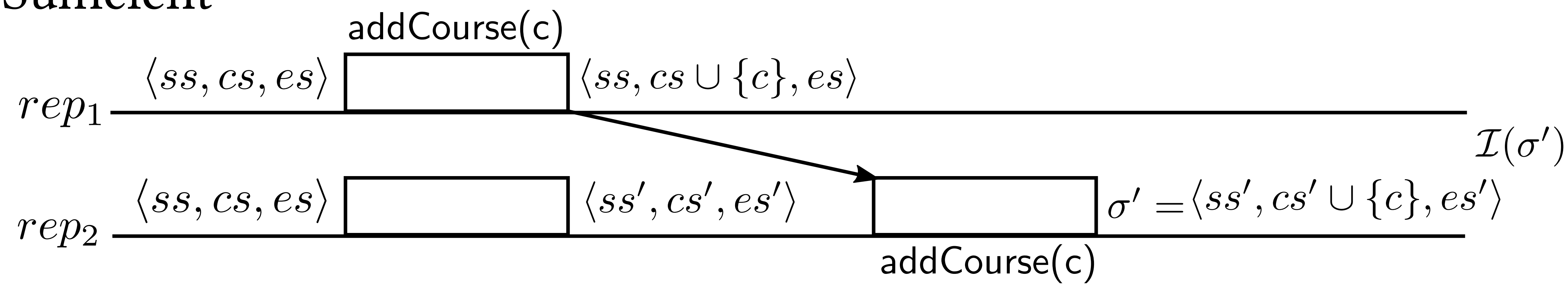
$\mathcal{I}$-Sufficient



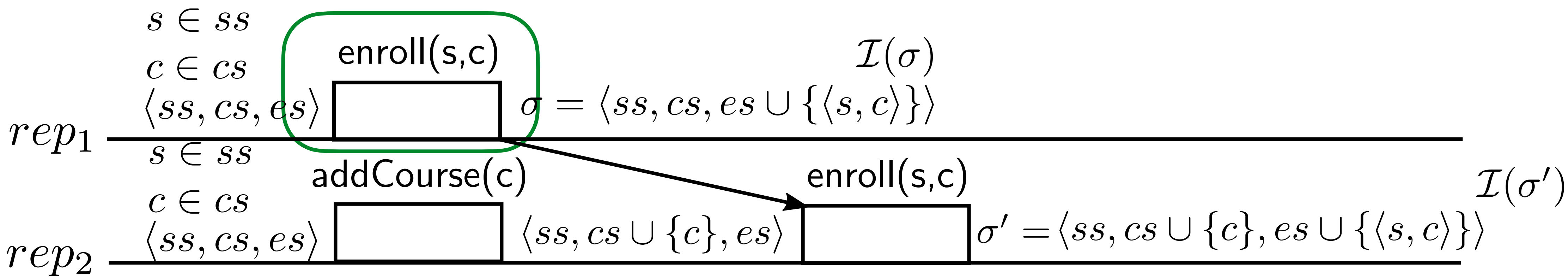
$\mathcal{P}$-R-Commutativity



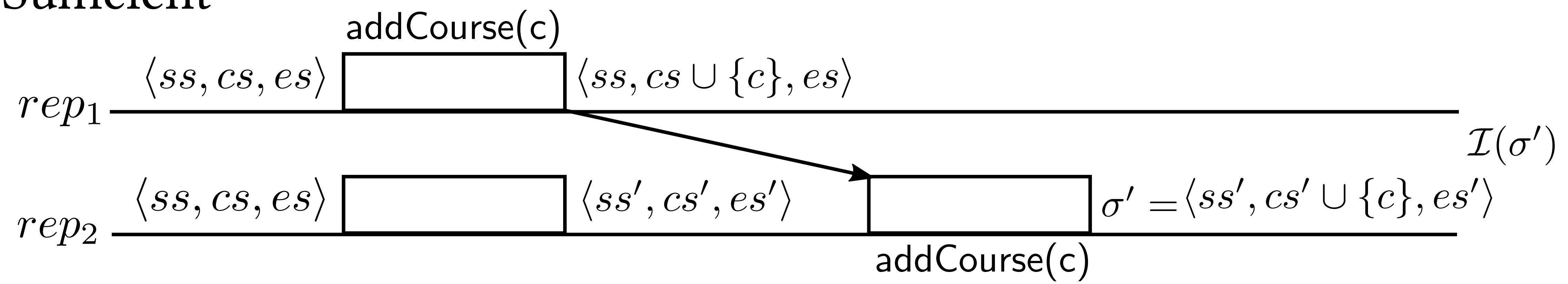
$\mathcal{I}$-Sufficient



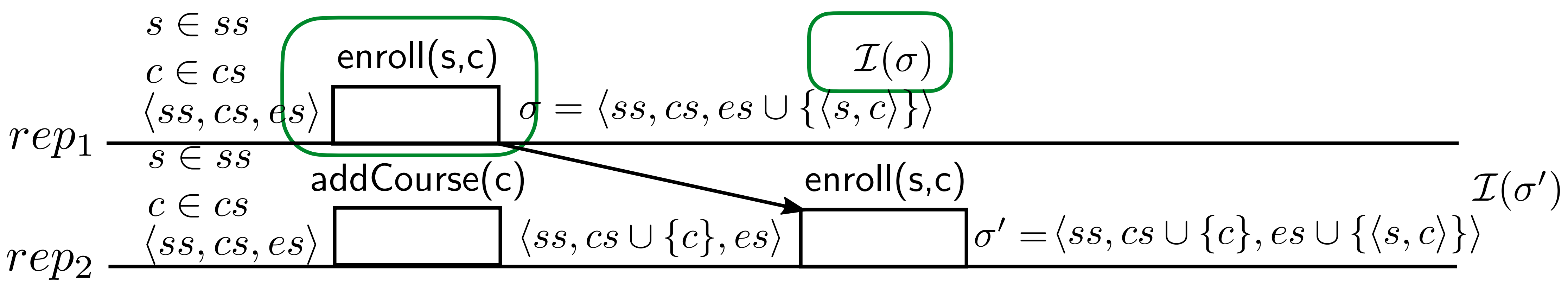
$\mathcal{P}$-R-Commutativity



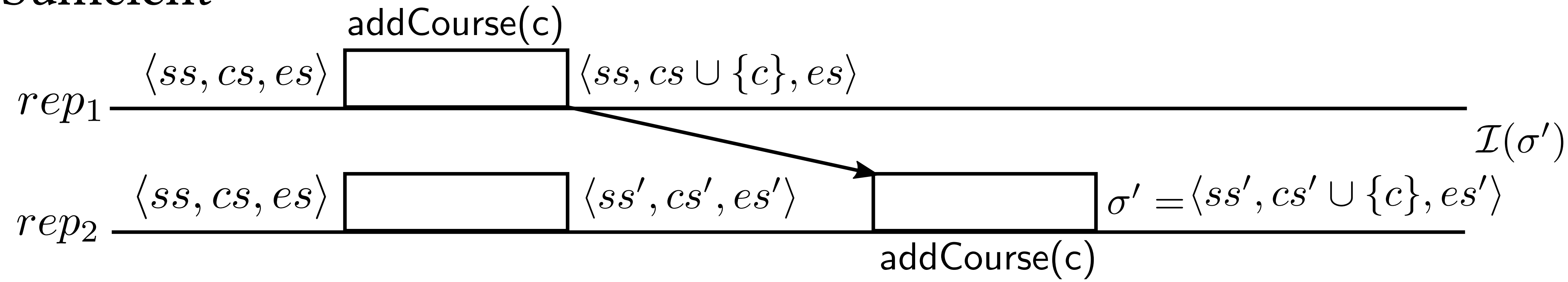
$\mathcal{I}$-Sufficient



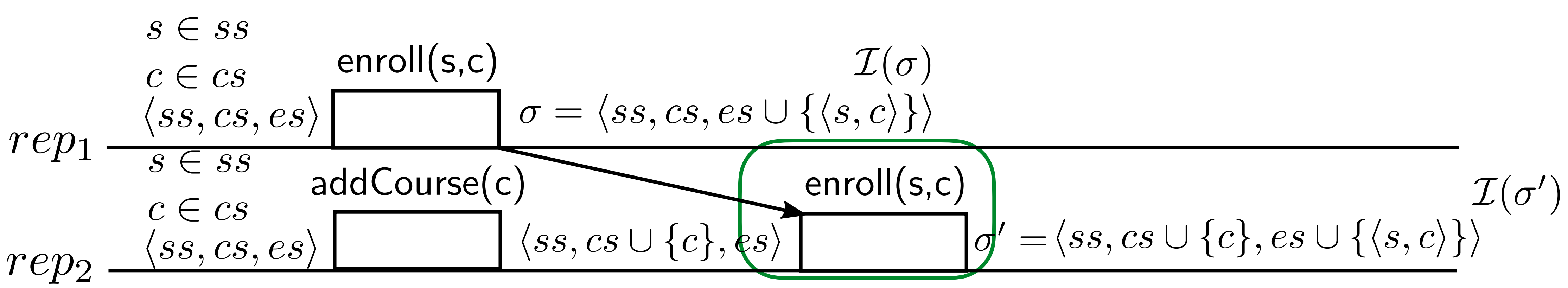
$\mathcal{P}$-R-Commutativity



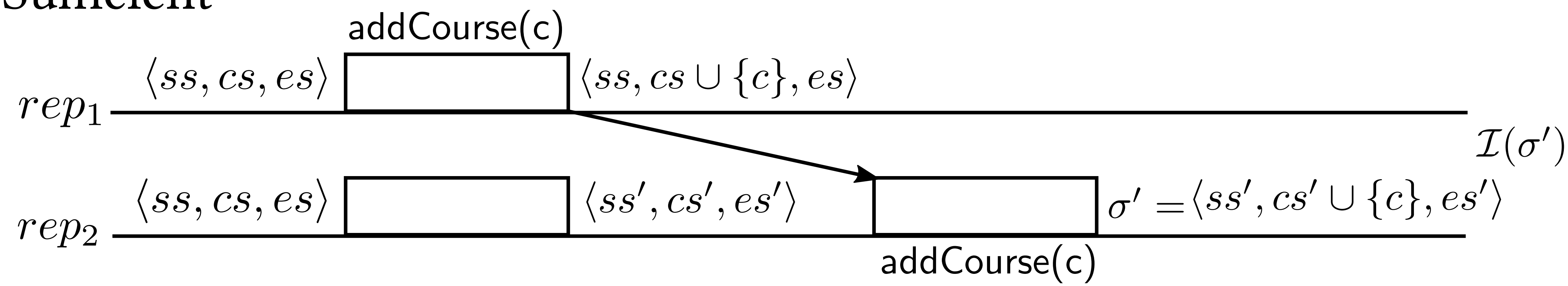
$\mathcal{I}$-Sufficient



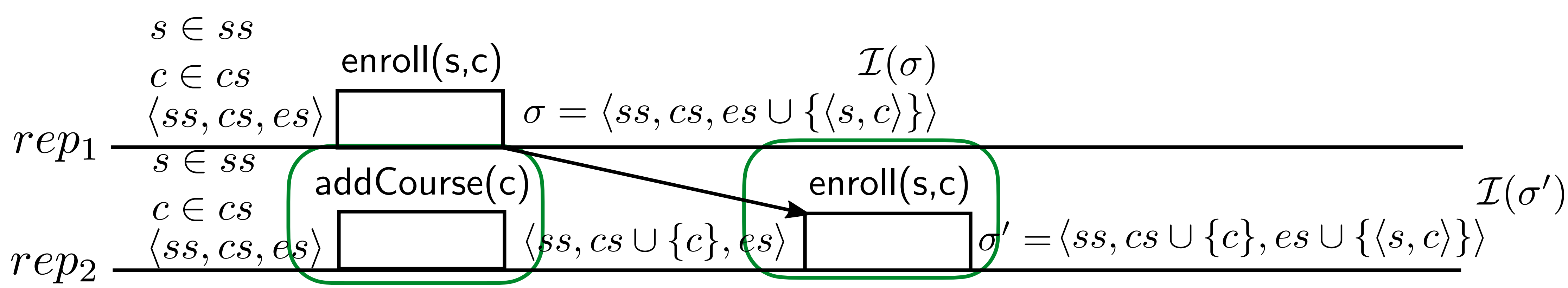
$\mathcal{P}$-R-Commutativity



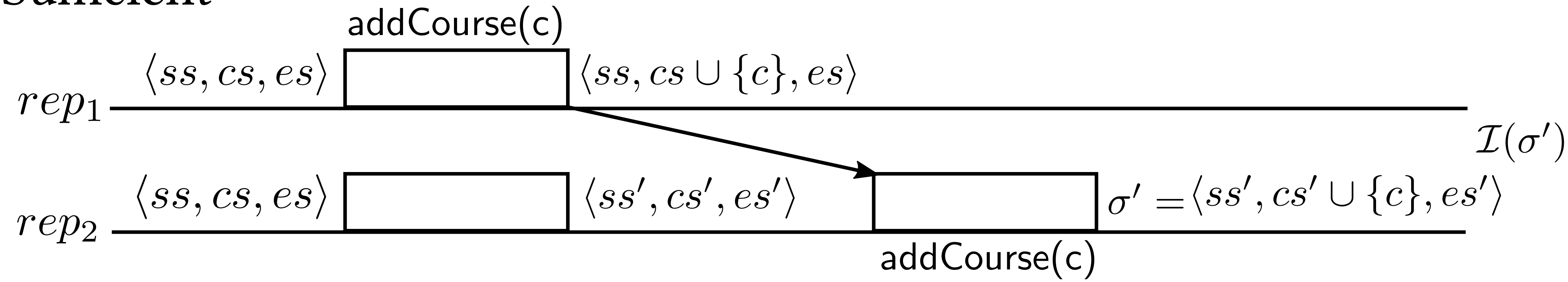
$\mathcal{I}$-Sufficient



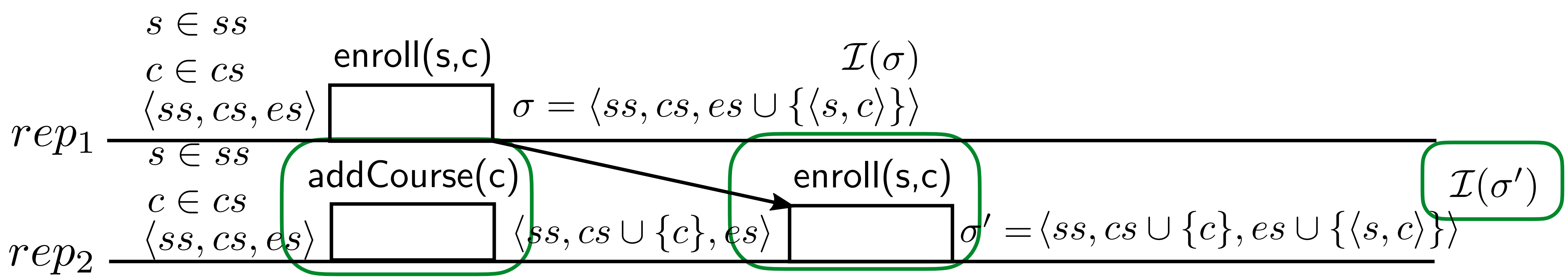
$\mathcal{P}$-R-Commutativity



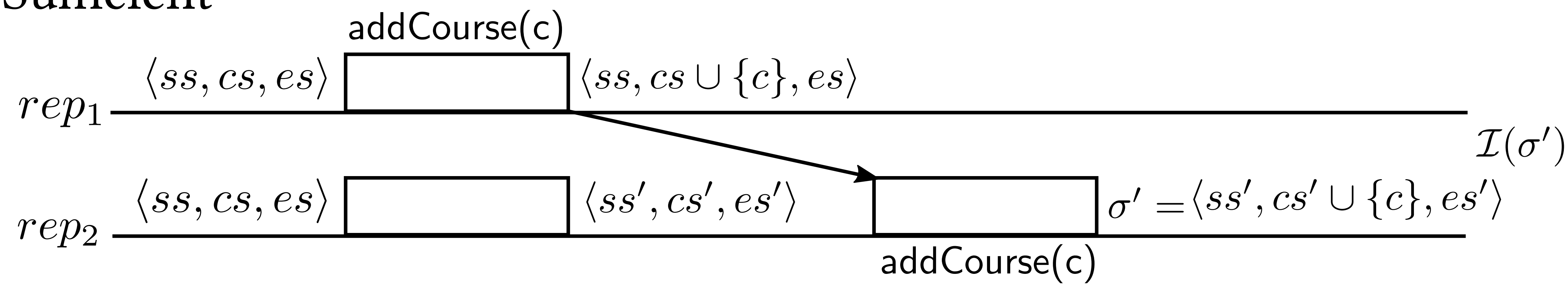
$\mathcal{I}$-Sufficient



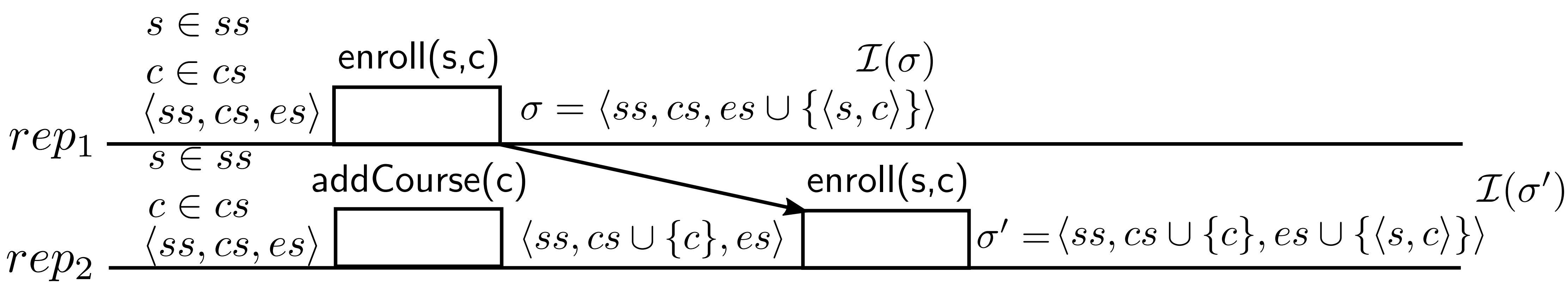
$\mathcal{P}$-R-Commutativity

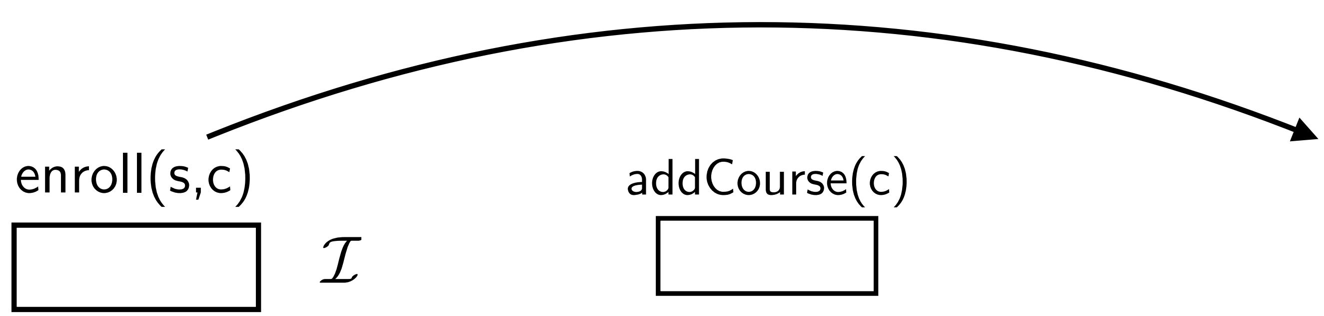


$\mathcal{I}$-Sufficient

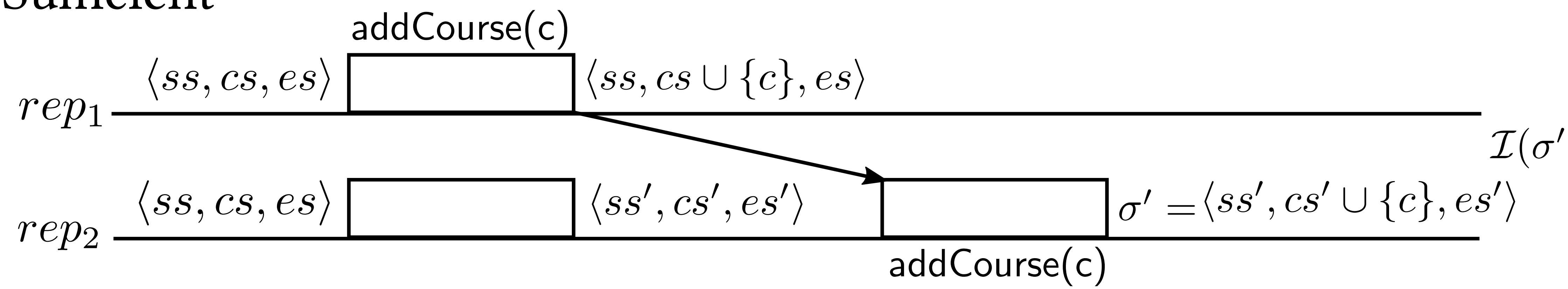


$\mathcal{P}$-R-Commutativity

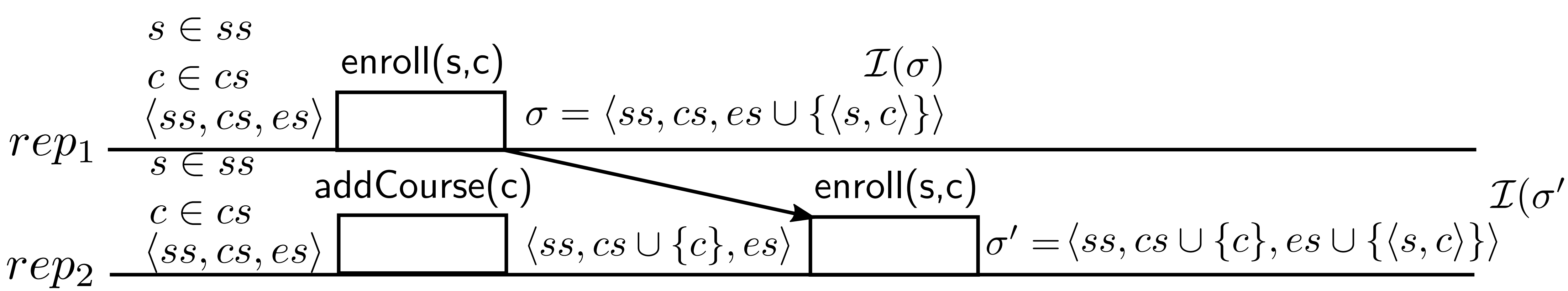


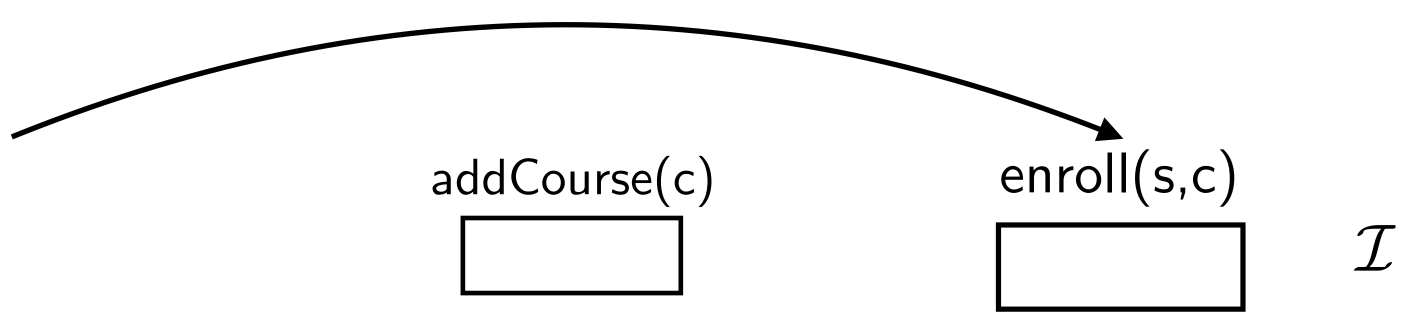


$\mathcal{I}$-Sufficient



$\mathcal{P}$-R-Commutativity





Concur and Conflict

$\mathcal{S}$-commute

Concur and Conflict

$\mathcal{S}$-commute

$\mathcal{P}$-concur

Concur and Conflict

$\mathcal{S}$-commute

$\mathcal{P}$-concur

Concur

$\mathcal{S}$-commute $\wedge$ $\mathcal{P}$-concur

Concur and Conflict

$\mathcal{S}$-commute

$\mathcal{P}$-concur

Concur

$\mathcal{S}$-commute $\wedge$ $\mathcal{P}$-concur

Conflict

$\neg$ Concur

Concur and Conflict

$\mathcal{S}$-commute

	r	a	e	d	q
r	✓	✓	✓	✓	✓
a	✓	✓	✓	×	✓
e	✓	✓	✓	✓	✓
d	✓	×	✓	✓	✓
q	✓	✓	✓	✓	✓

$\mathcal{P}$-concur

	r	a	e	d	q
r	✓	✓	✓	✓	✓
a	✓	✓	✓	✓	✓
e	✓	✓	✓	×	✓
d	✓	✓	×	✓	✓
q	✓	✓	✓	✓	✓

Concur

$\mathcal{S}$-commute $\wedge$ $\mathcal{P}$-concur

	r	a	e	d	q
r	✓	✓	✓	✓	✓
a	✓	✓	✓	×	✓
e	✓	✓	✓	×	✓
d	✓	×	×	✓	✓
q	✓	✓	✓	✓	✓

Conflict

$\neg$ Concur

Concur and Conflict

$\mathcal{S}$-commute

	r	a	e	d	q
r	✓	✓	✓	✓	✓
a	✓	✓	✓	×	✓
e	✓	✓	✓	✓	✓
d	✓	×	✓	✓	✓
q	✓	✓	✓	✓	✓

$\mathcal{P}$-concur

	r	a	e	d	q
r	✓	✓	✓	✓	✓
a	✓	✓	✓	✓	✓
e	✓	✓	✓	×	✓
d	✓	✓	×	✓	✓
q	✓	✓	✓	✓	✓

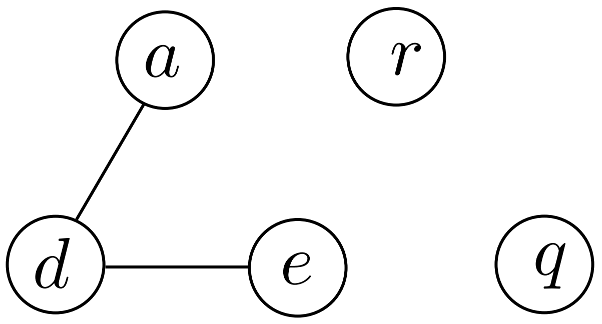
Concur

$\mathcal{S}$-commute $\wedge$ $\mathcal{P}$-concur

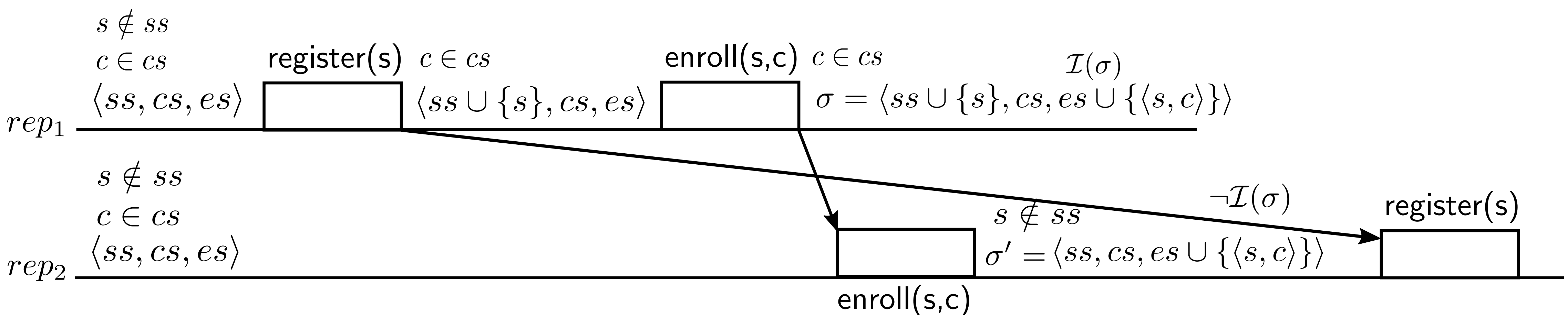
	r	a	e	d	q
r	✓	✓	✓	✓	✓
a	✓	✓	✓	×	✓
e	✓	✓	✓	×	✓
d	✓	×	×	✓	✓
q	✓	✓	✓	✓	✓

Conflict

$\neg$ Concur



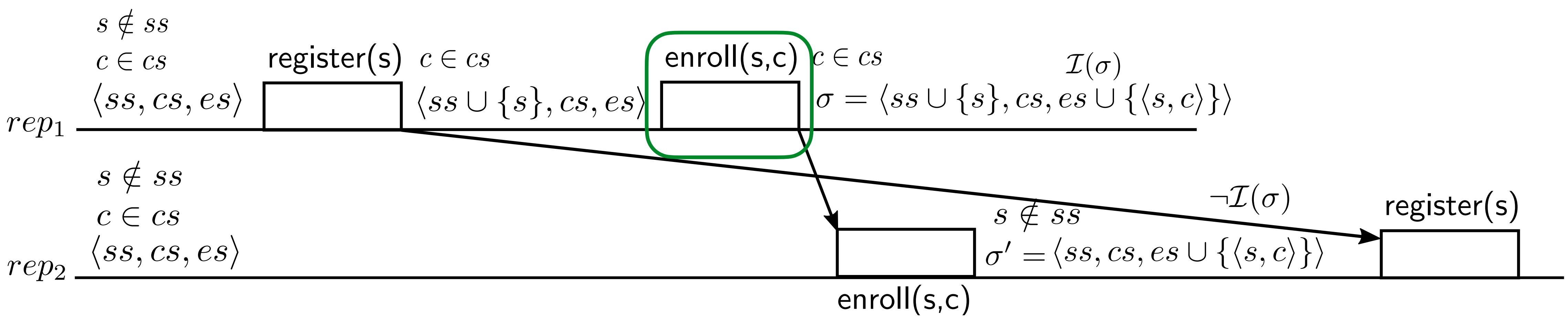
Dependence



$s \notin ss$
$c \in cs$
$\langle ss, cs, es \rangle$
register(s)
$c \in cs$
$\langle ss \cup \{s\}, cs, es \rangle$
enroll(s,c)
$c \in cs$
$\sigma = \langle ss \cup \{s\}, cs, es \cup \{\langle s, c \rangle\}\rangle$
$\mathcal{I}(\sigma)$
rep_1
$s \notin ss$
$c \in cs$
$\langle ss, cs, es \rangle$
rep_2
$\neg\mathcal{I}(\sigma)$
register(s)
$s \notin ss$
$\sigma' = \langle ss, cs, es \cup \{\langle s, c \rangle\}\rangle$
enroll(s,c)

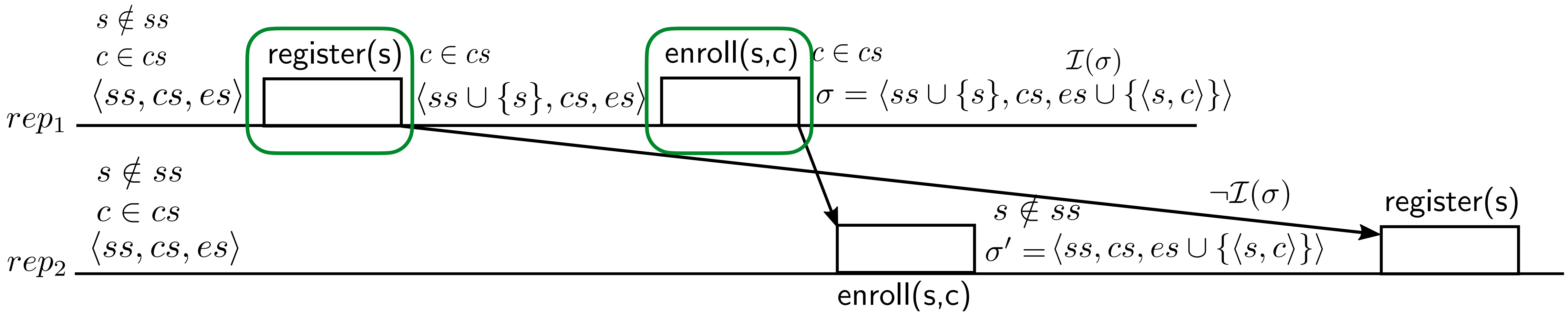
Dependence

Dependence

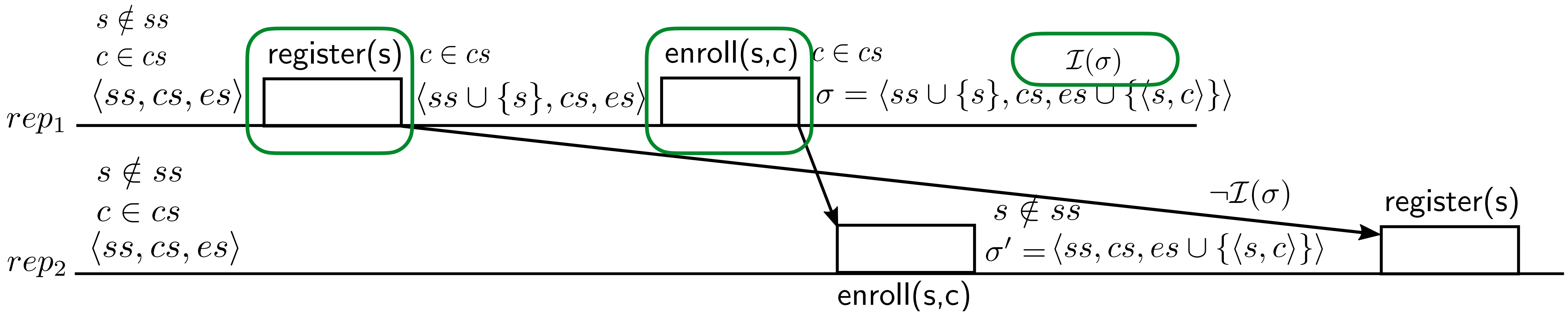


Dependence

Dependence

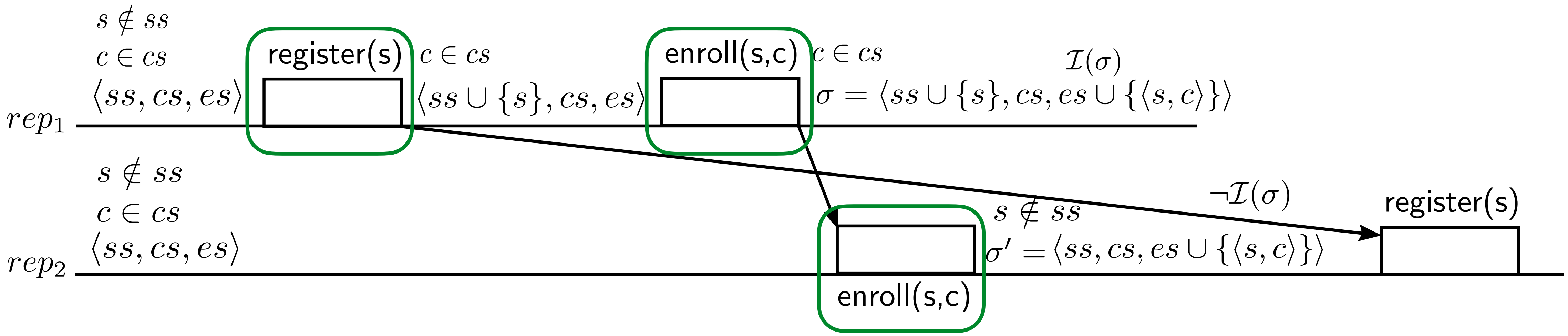


Dependence



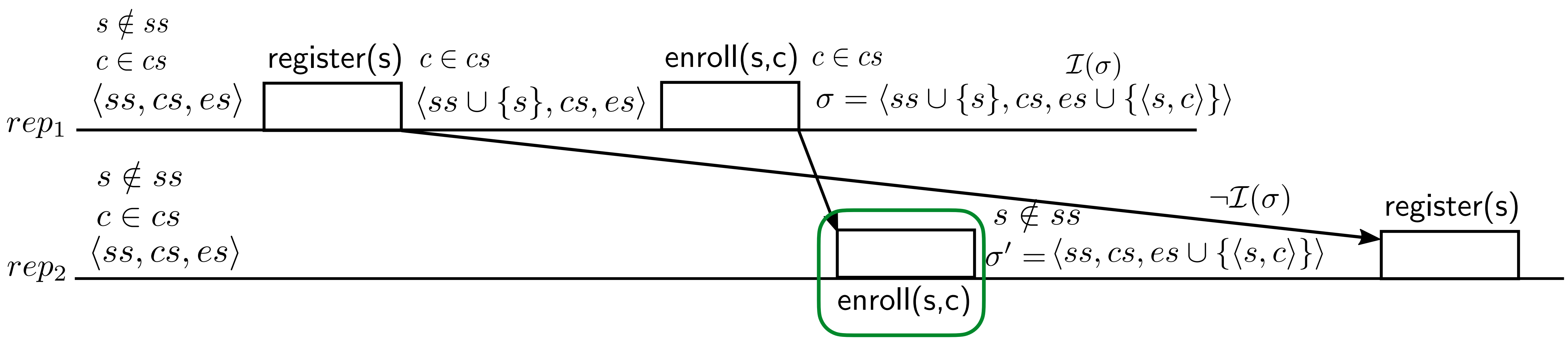
$s \notin ss$
$c \in cs$
$\langle ss, cs, es \rangle$
register(s)
$c \in cs$
$\langle ss \cup \{s\}, cs, es \rangle$
enroll(s,c)
$c \in cs$
$\sigma = \langle ss \cup \{s\}, cs, es \cup \{\langle s, c \rangle\}\rangle$
$\mathcal{I}(\sigma)$
rep_1
$s \notin ss$
$c \in cs$
$\langle ss, cs, es \rangle$
$\neg\mathcal{I}(\sigma)$
register(s)
$s \notin ss$
$\sigma' = \langle ss, cs, es \cup \{\langle s, c \rangle\}\rangle$
rep_2
enroll(s,c)

Dependence


$s \notin ss$
$c \in cs$
$\langle ss, cs, es \rangle$
register(s)
$c \in cs$
$\langle ss \cup \{s\}, cs, es \rangle$
enroll(s,c)
$c \in cs$
$\sigma = \langle ss \cup \{s\}, cs, es \cup \{\langle s, c \rangle\} \rangle$
$\mathcal{I}(\sigma)$
rep_1
$s \notin ss$
$c \in cs$
$\langle ss, cs, es \rangle$
rep_2
$s \notin ss$
$\sigma' = \langle ss, cs, es \cup \{\langle s, c \rangle\} \rangle$
enroll(s,c)
$\neg\mathcal{I}(\sigma)$
register(s)

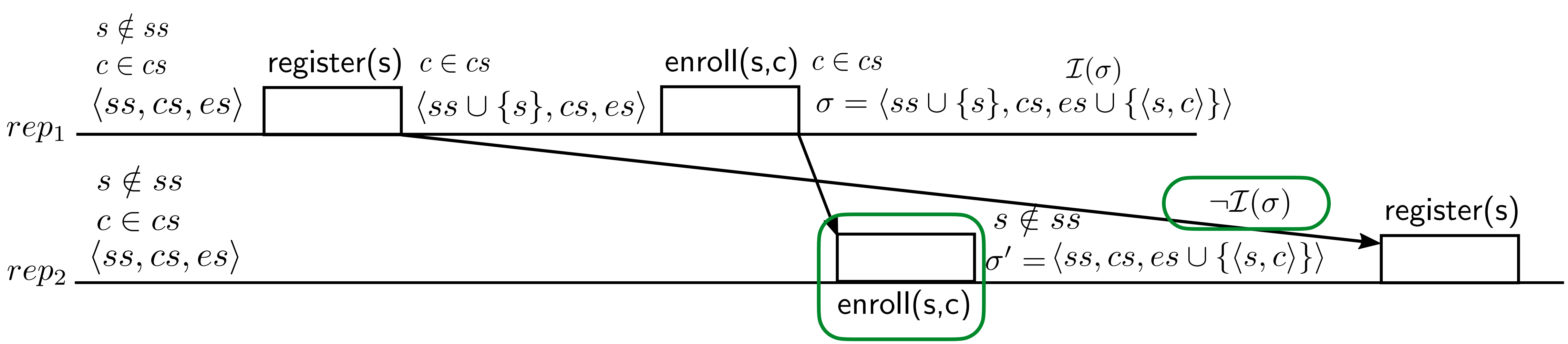
Dependence

Dependence



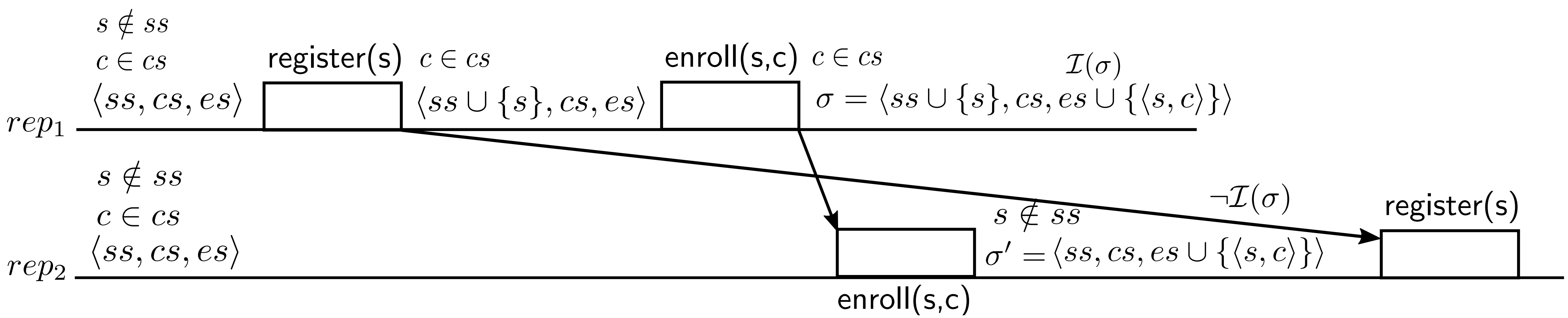
Dependence

Dependence



Dependence

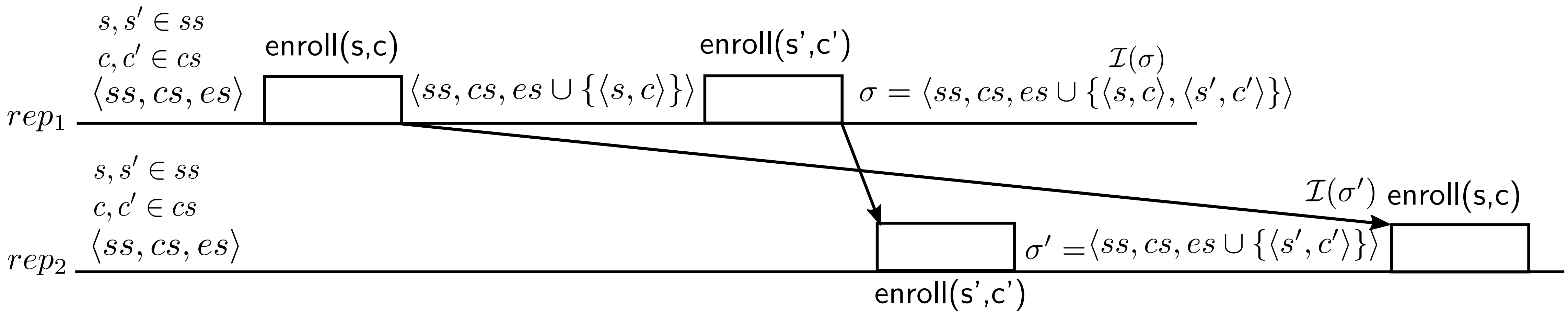
Dependence



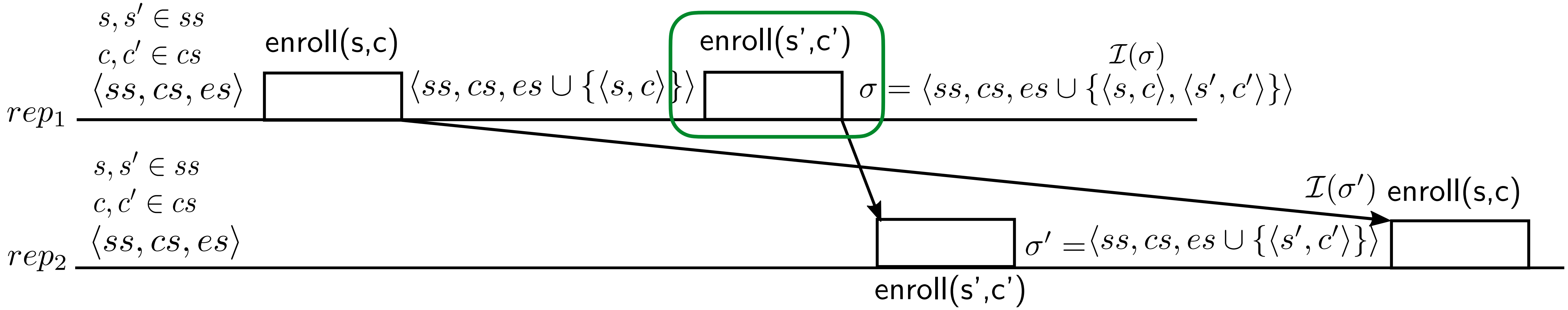
$\mathcal{I}$-Sufficient

Independence

$\mathcal{P}$-L-commute

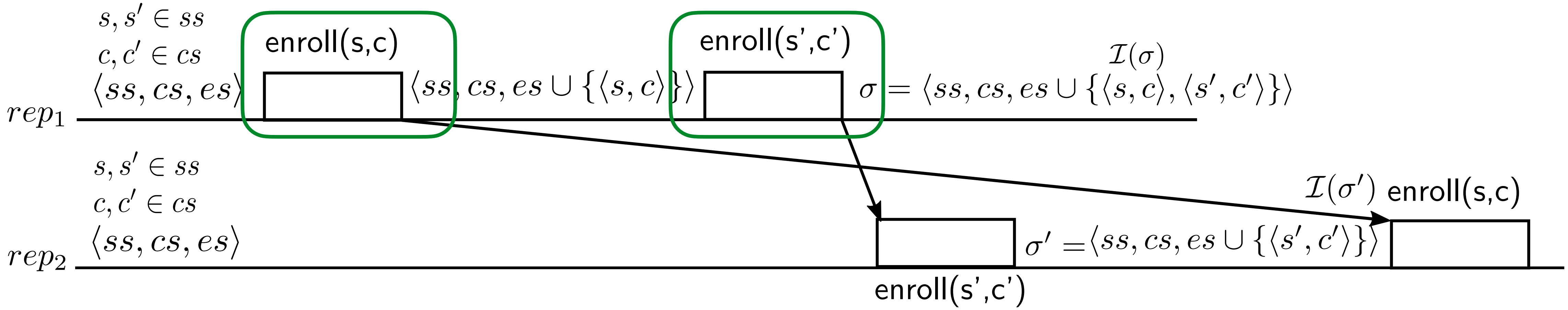


$\mathcal{P}$-L-commute

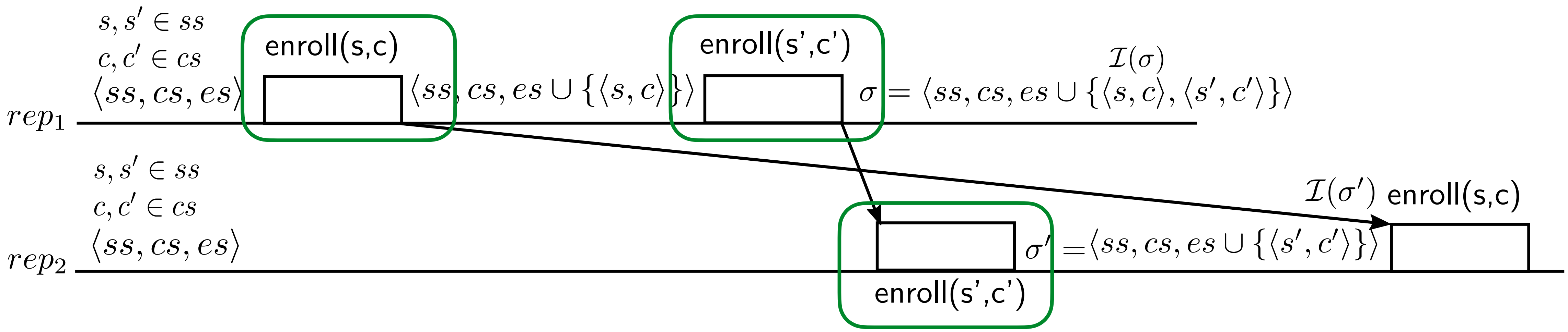


Independence

$\mathcal{P}$-L-commute

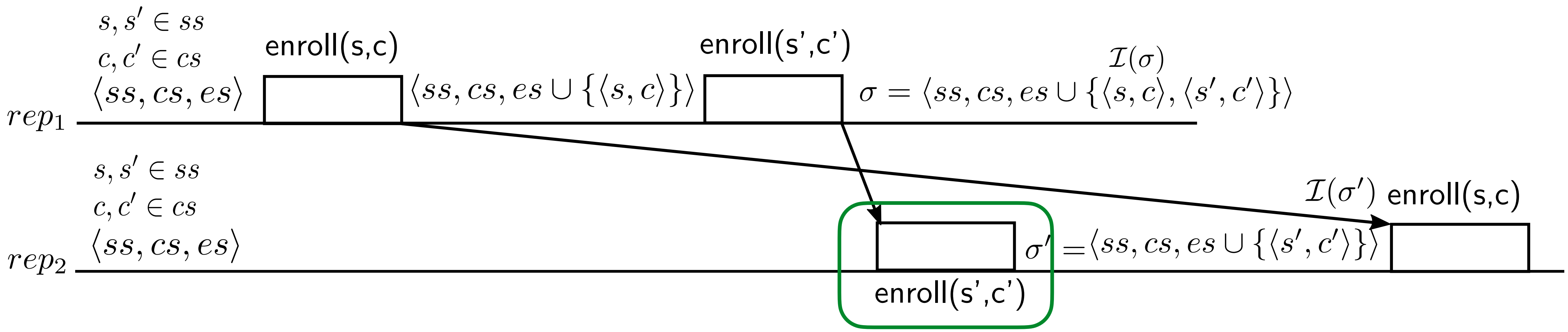


$\mathcal{P}$-L-commute



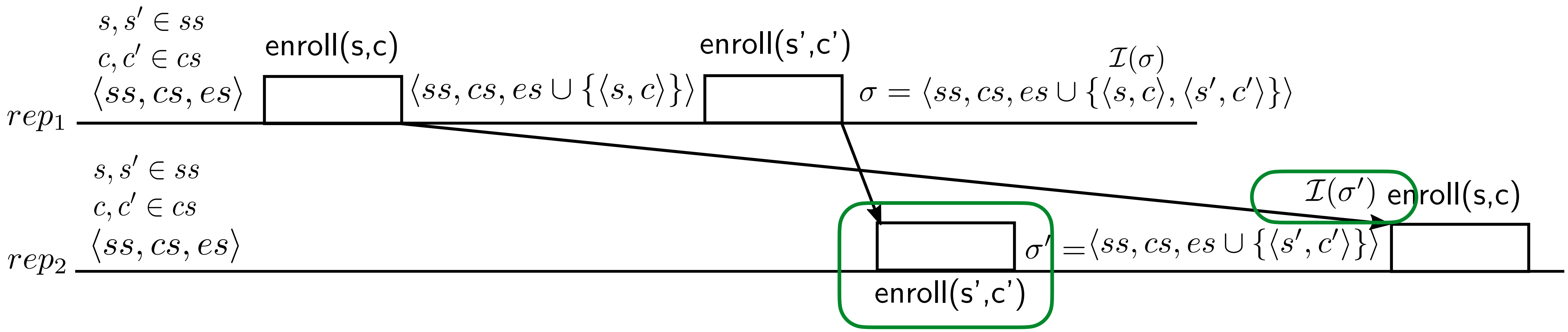
Independence

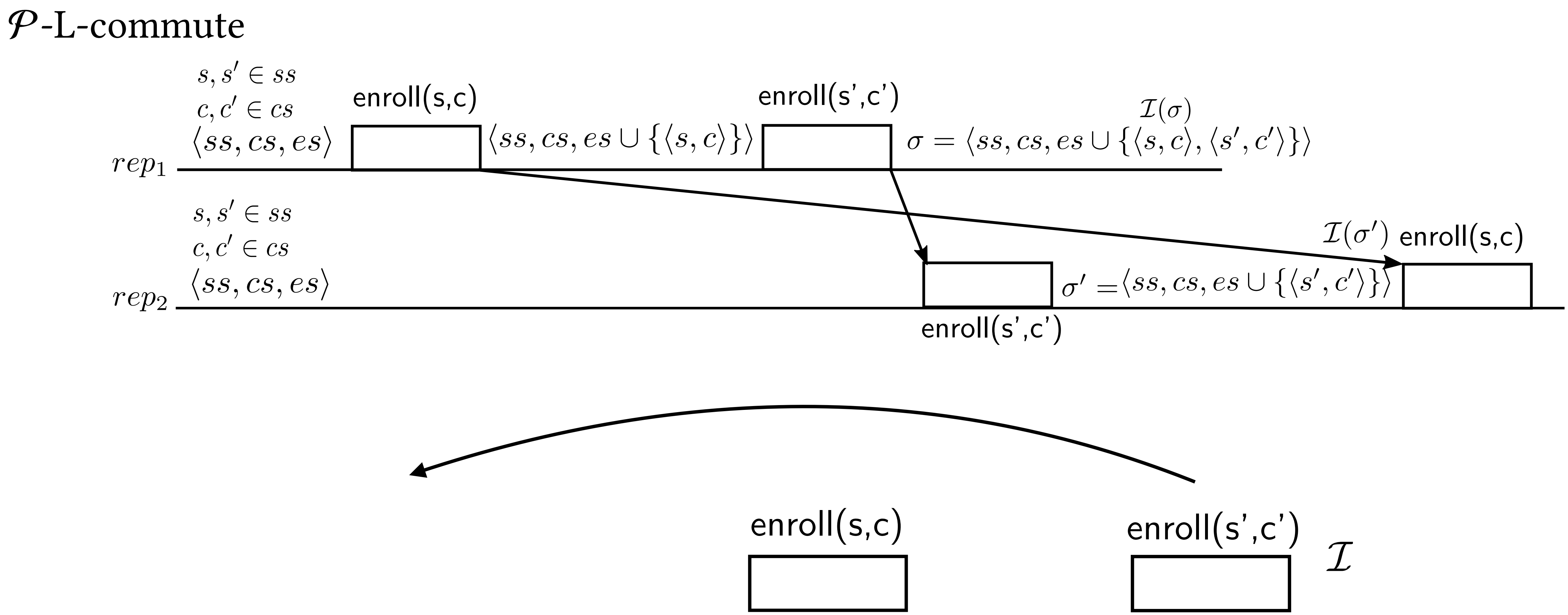
$\mathcal{P}$-L-commute



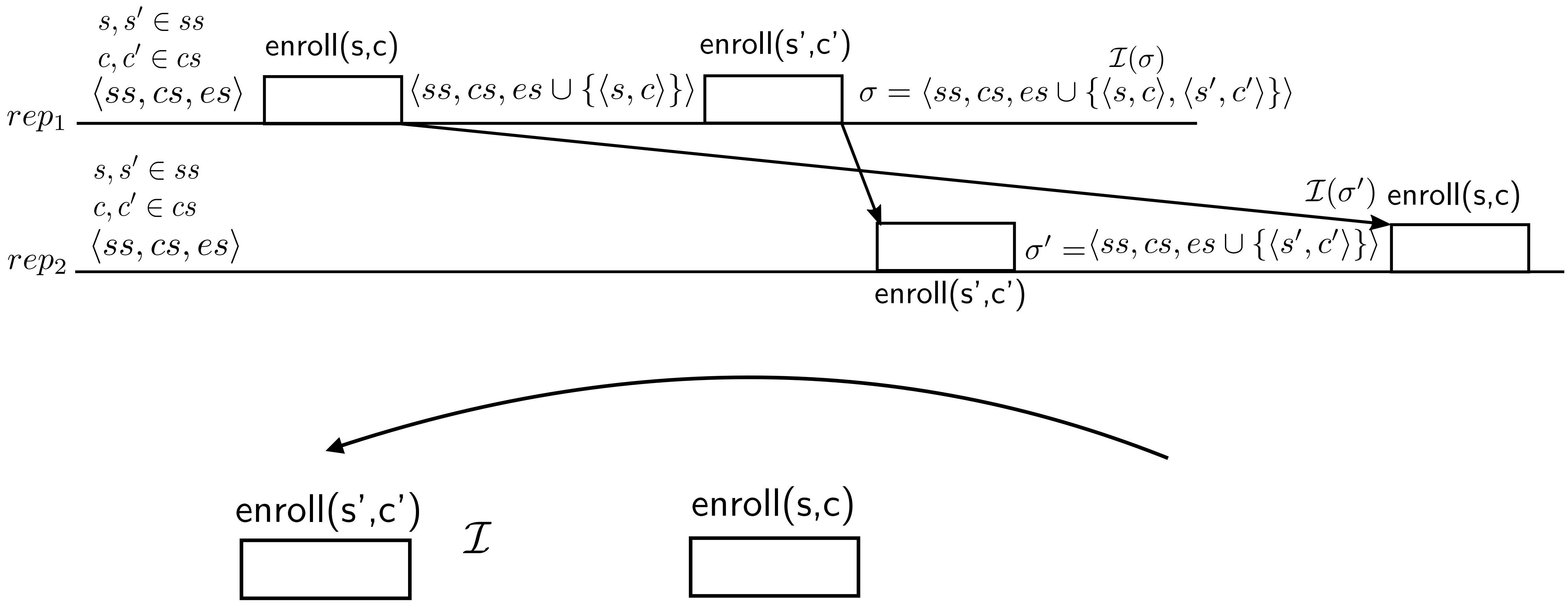
Independence

$\mathcal{P}$-L-commute




$\mathcal{P}$-L-commute
$s, s' \in ss$
$c, c' \in cs$
$\langle ss, cs, es \rangle$
enroll(s,c)
$\langle ss, cs, es \cup \{\langle s, c \rangle\} \rangle$
enroll(s',c')
$\mathcal{I}(\sigma)$
$\sigma = \langle ss, cs, es \cup \{\langle s, c \rangle, \langle s', c' \rangle\} \rangle$
rep_1
$s, s' \in ss$
$c, c' \in cs$
$\langle ss, cs, es \rangle$
rep_2
enroll(s',c')
$\sigma' = \langle ss, cs, es \cup \{\langle s', c' \rangle\} \rangle$
$\mathcal{I}(\sigma')$
enroll(s,c)
enroll(s,c)
enroll(s',c')
$\mathcal{I}$

$\mathcal{P}$-L-commute



Dependence

Dependence

Independent

$\mathcal{I}$-Sufficient $\vee$ $\mathcal{P}$-L-commute

Dependence

Independent

$\mathcal{I}$-Sufficient $\vee$ $\mathcal{P}$-L-commute

Dependent

$\neg$ Independent

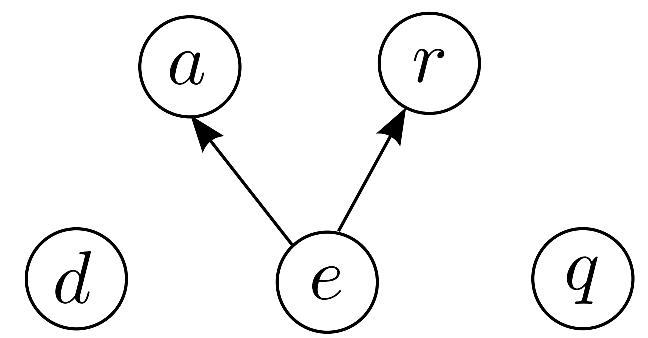
Independent

$\mathcal{I}$-Sufficient $\vee$ $\mathcal{P}$-L-commute

Dependent

$\neg$ Independent

	r	a	e	d	q
r	✓	✓	✓	✓	✓
a	✓	✓	✓	✓	✓
e	×	×	✓	✓	✓
d	✓	✓	✓	✓	✓
q	✓	✓	✓	✓	✓

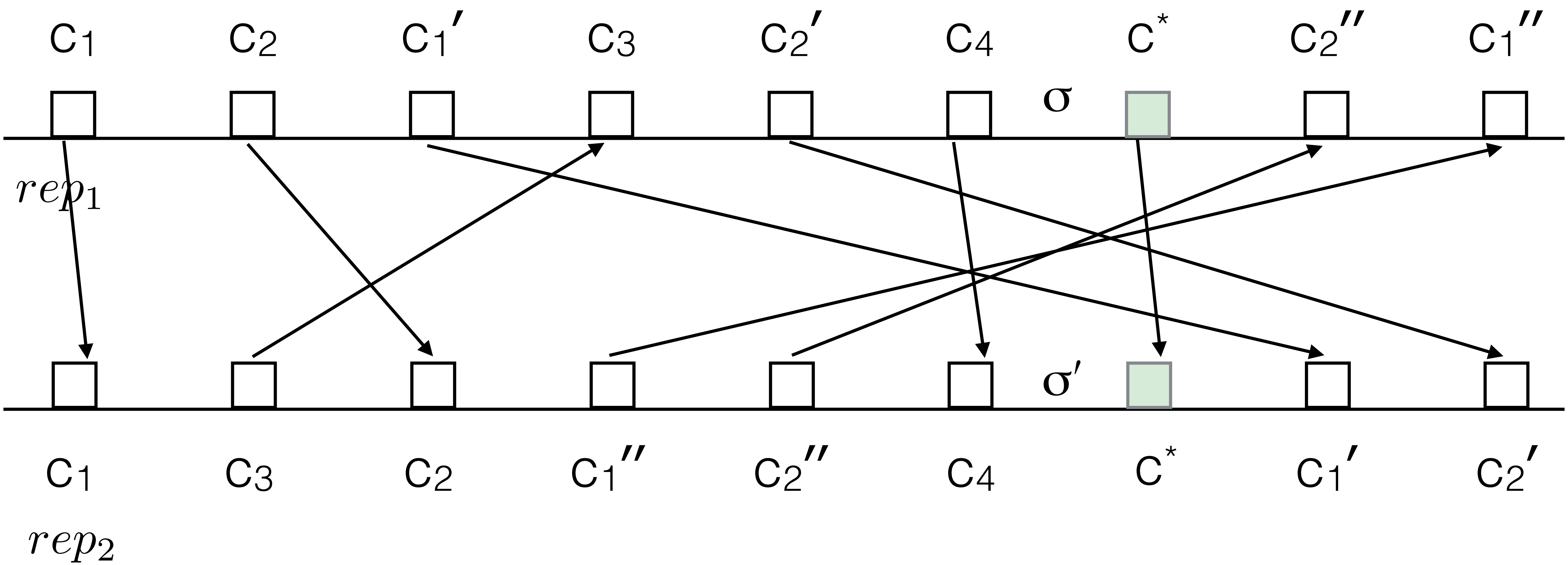


Well-coordination

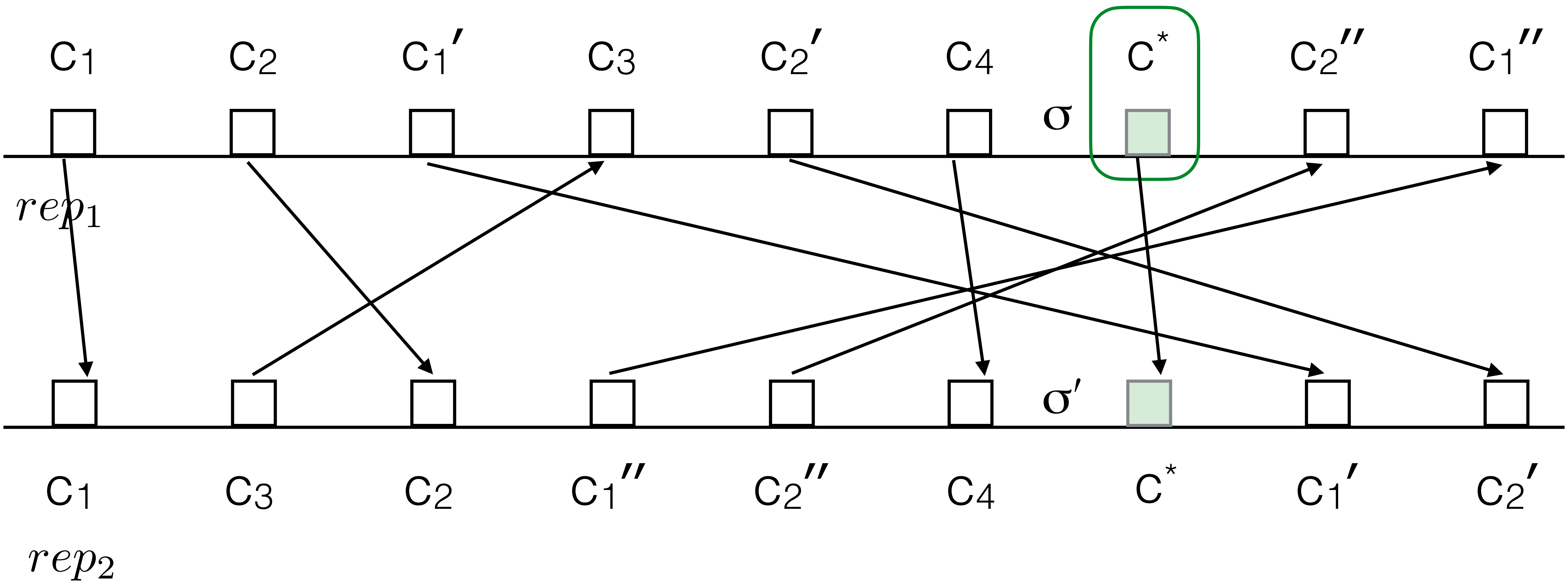
- Well-coordination
  - Locally permissible
  - Conflict-synchronizing
  - Dependency-preserving

- Theorem:
  Well-coordination
  is sufficient for
  integrity and convergence.

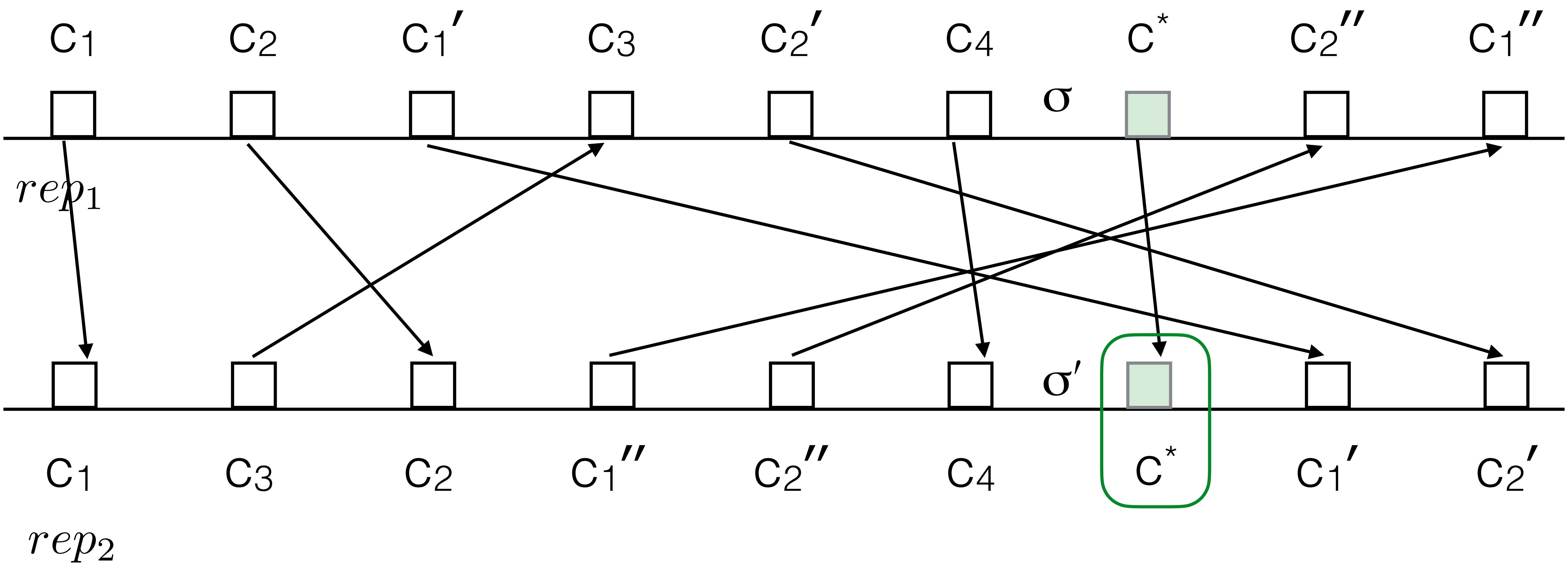
Well-coordination



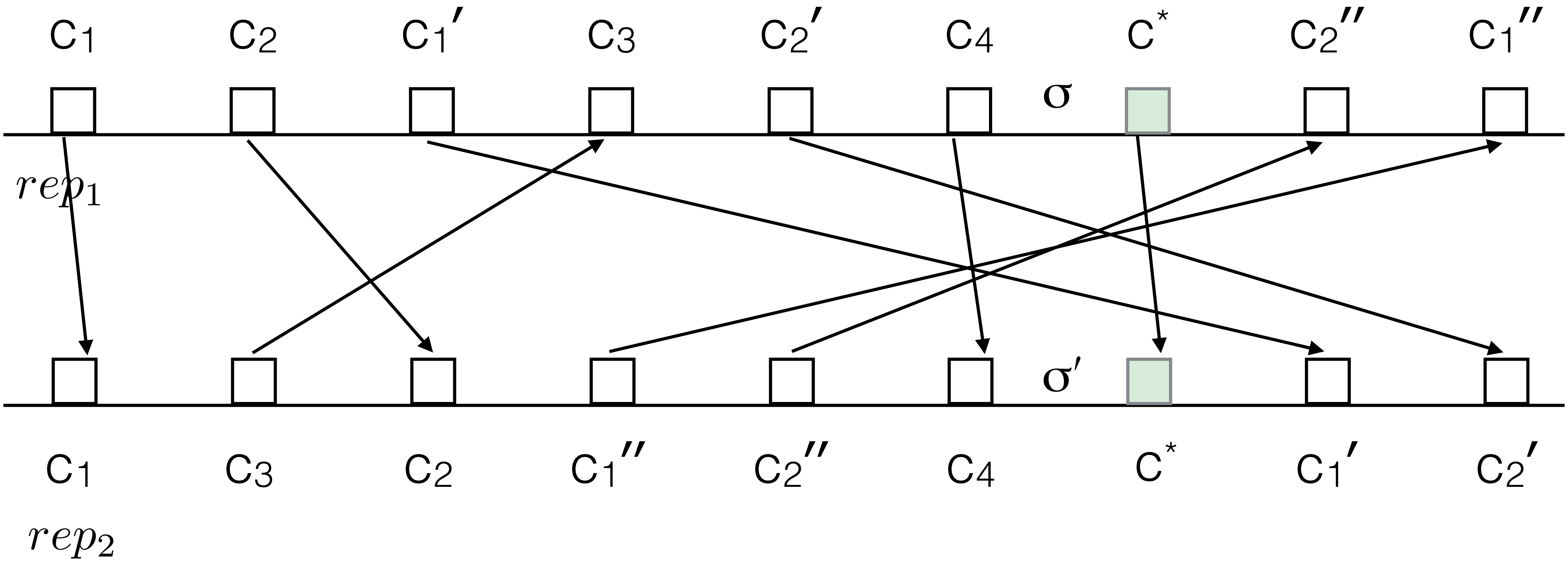
Well-coordination



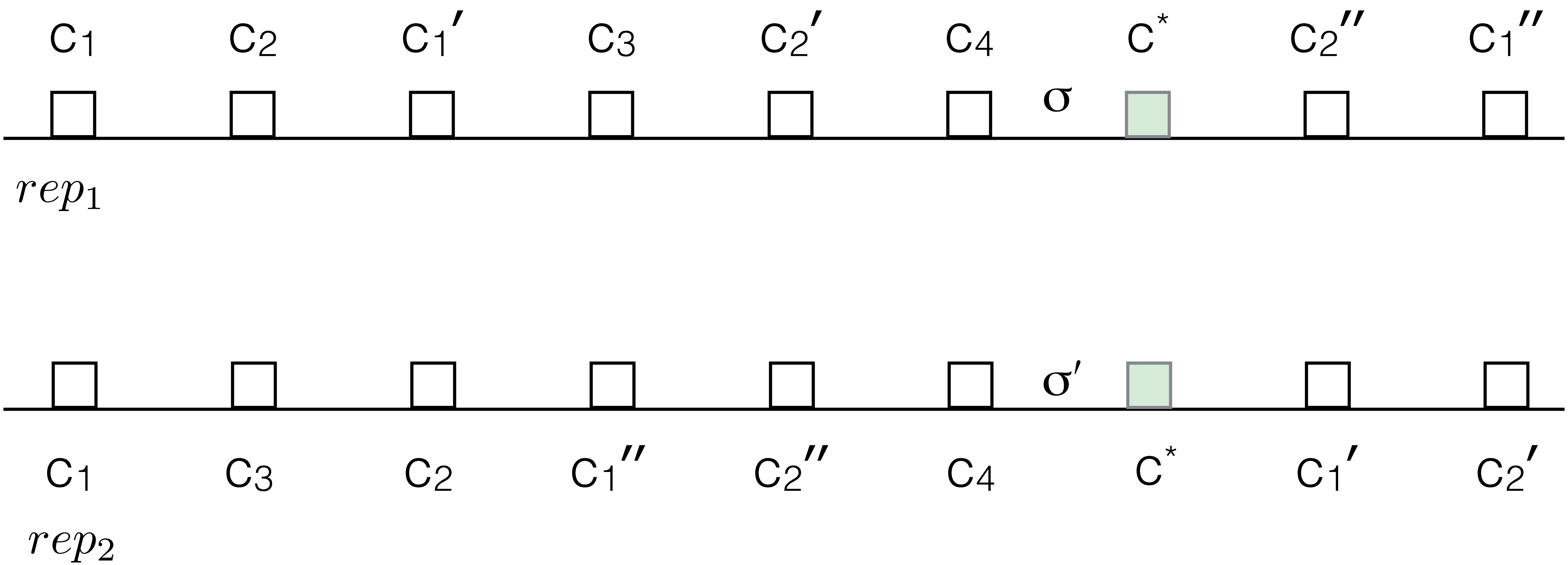
Well-coordination



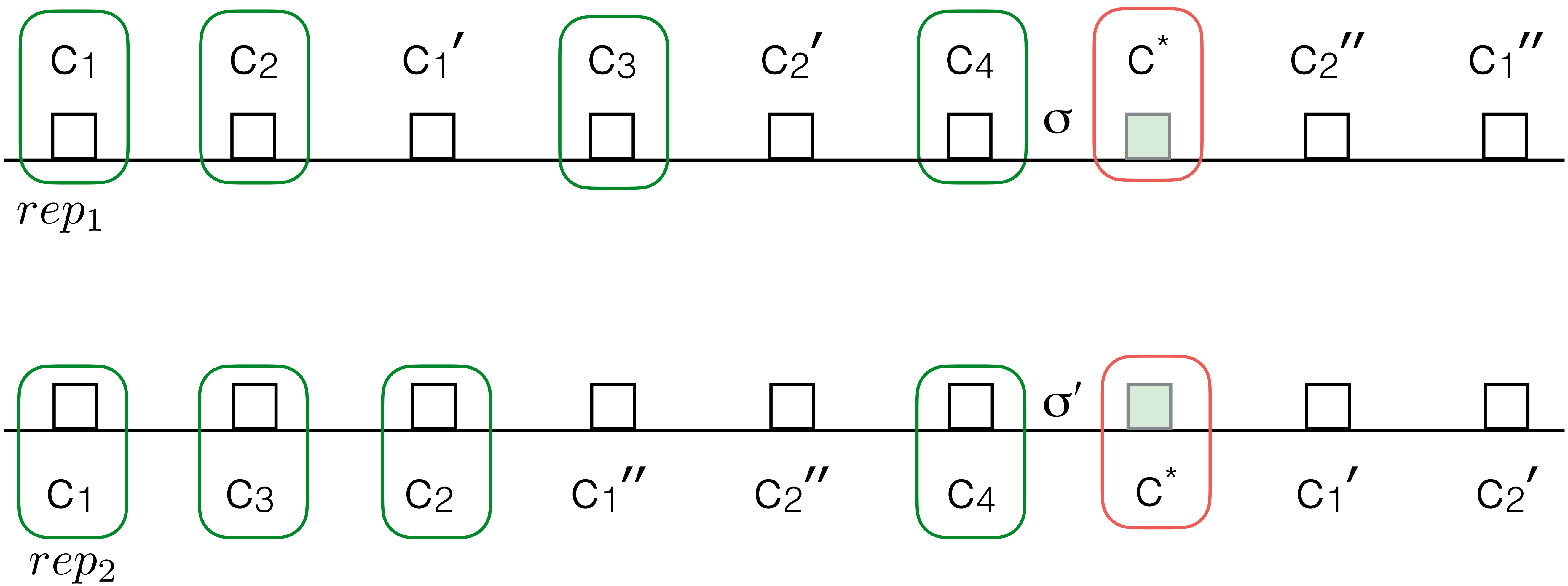
Well-coordination



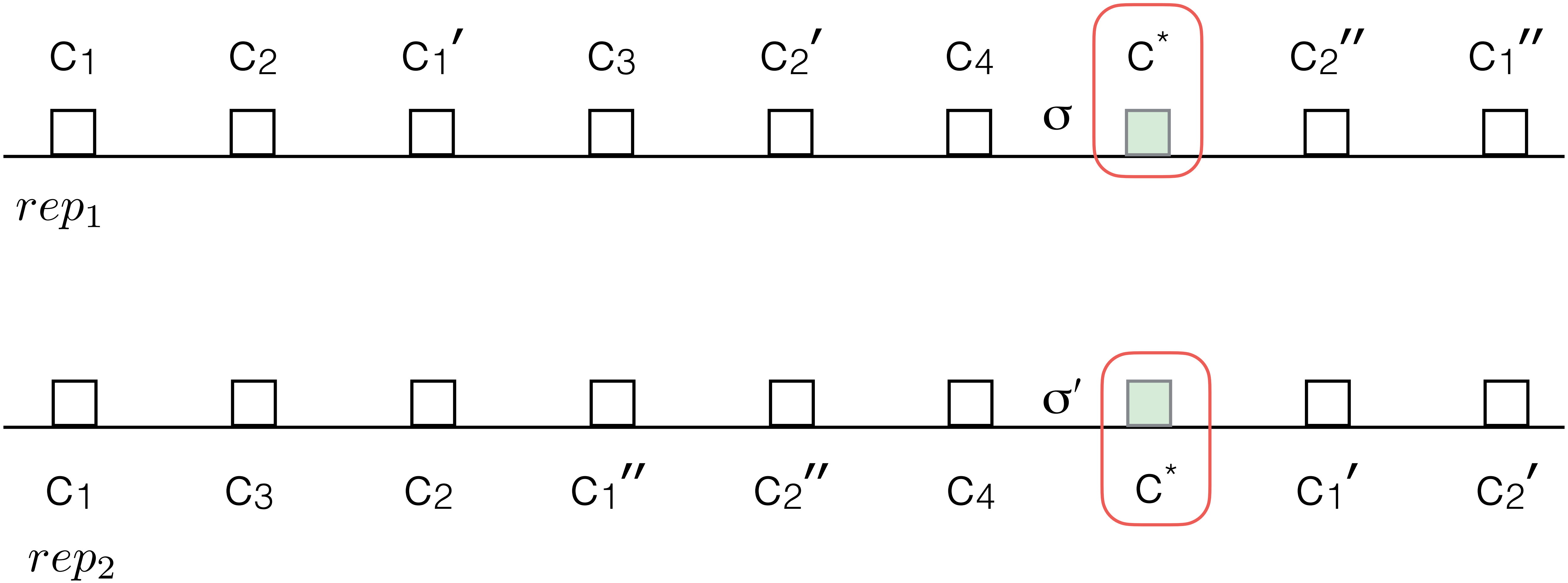
Well-coordination



Well-coordination

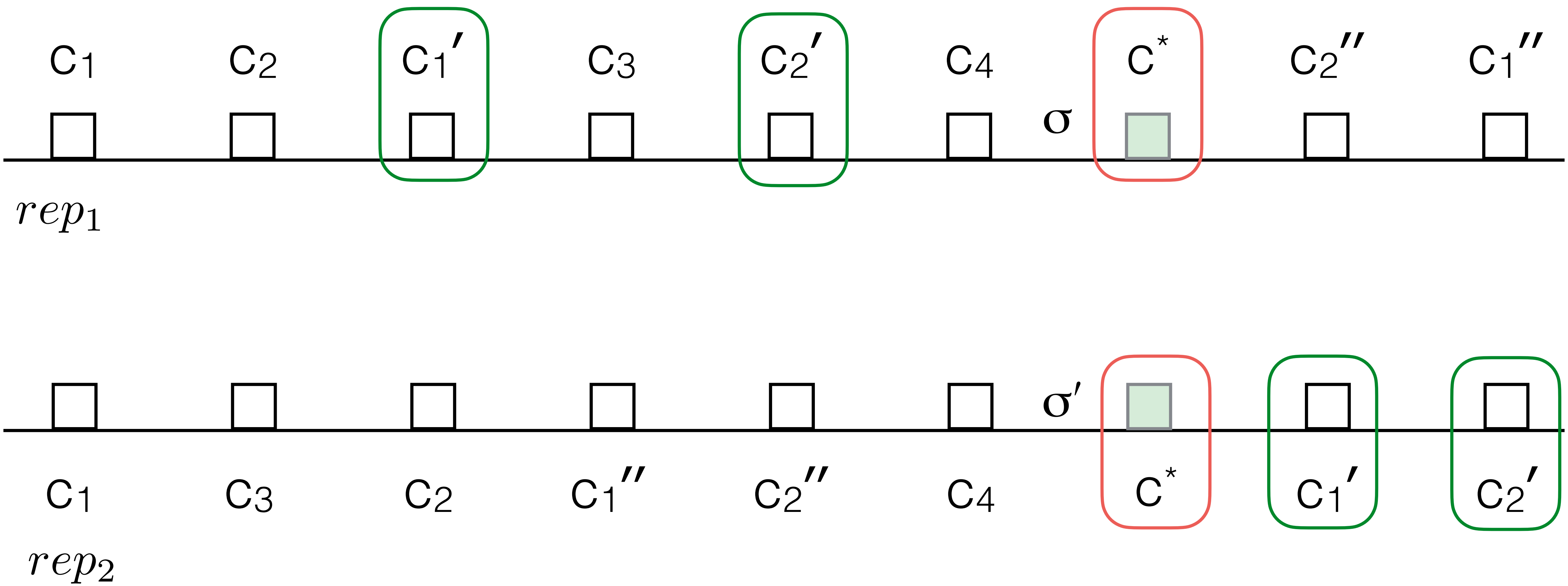


Well-coordination

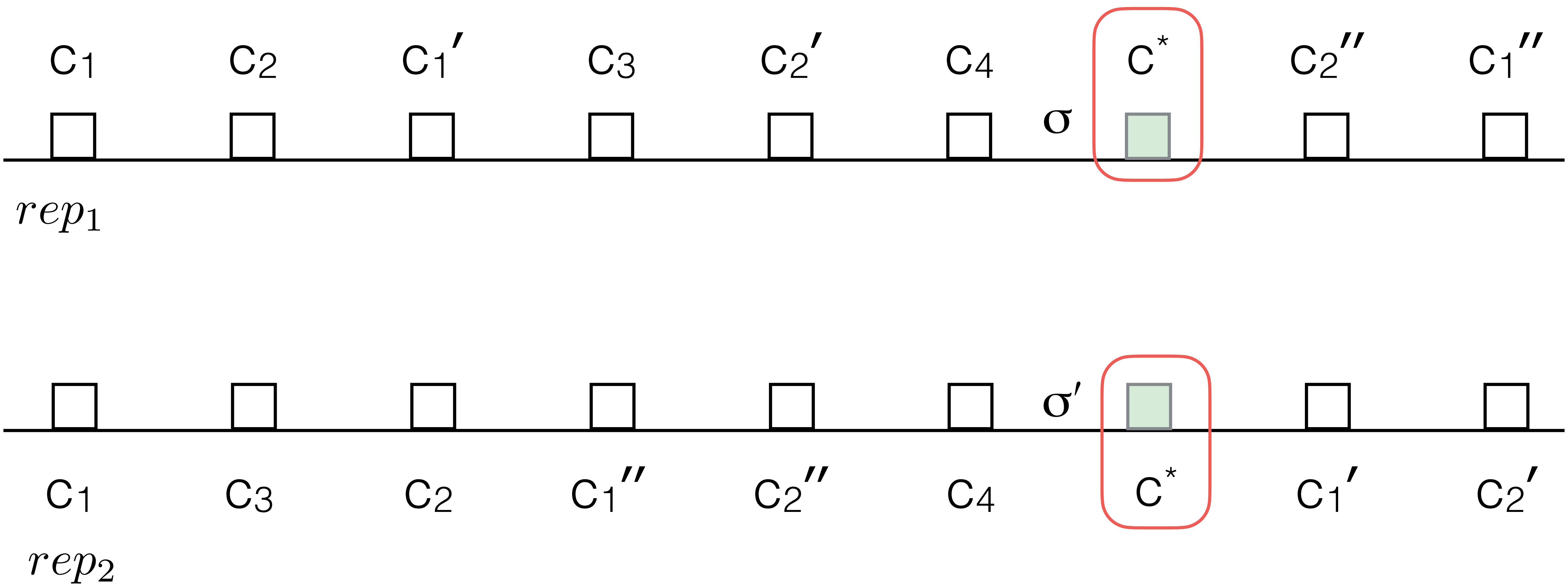


Well-coordination

$\mathcal{P}$-L-Commutativity

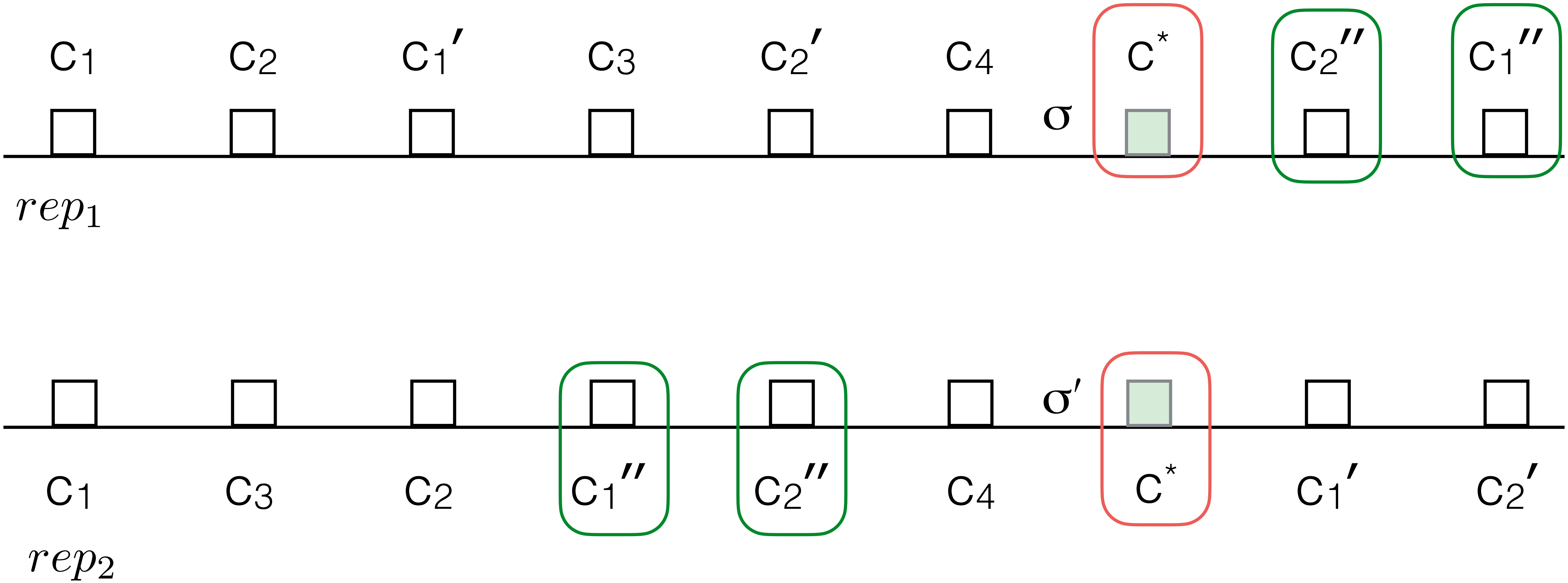


Well-coordination

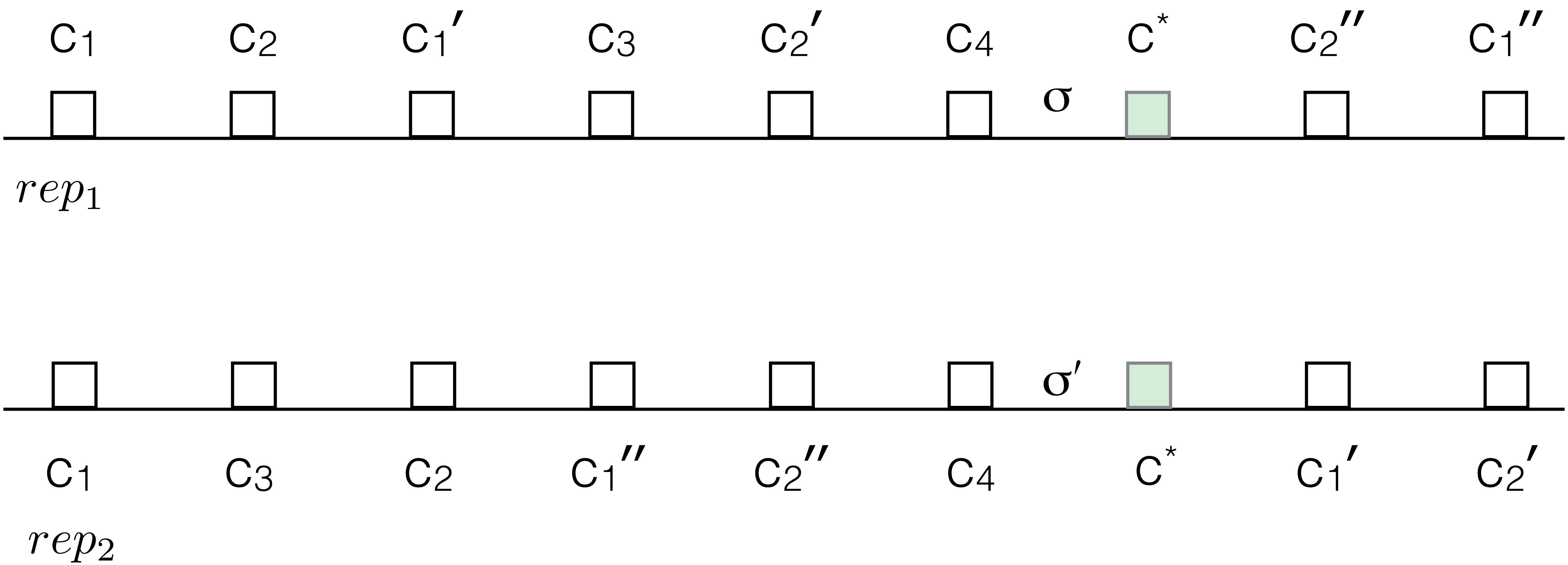


Well-coordination

$\mathcal{P}$-R-Commutativity

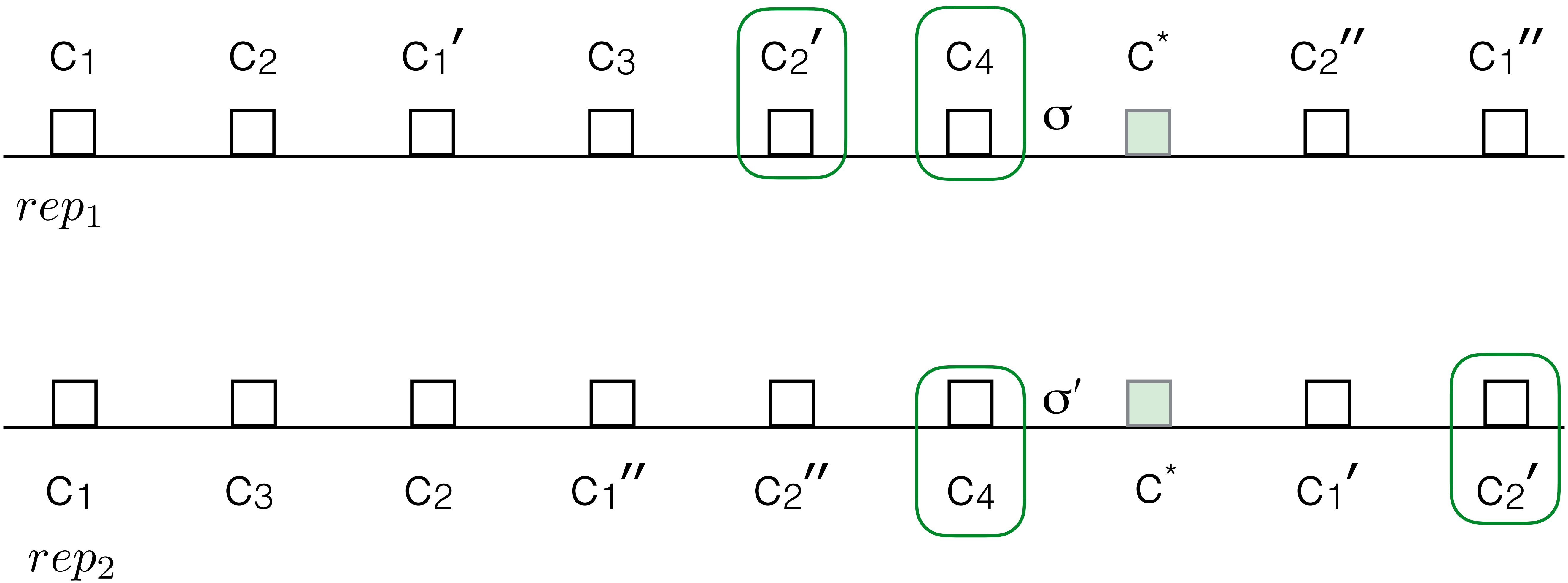


Well-coordination

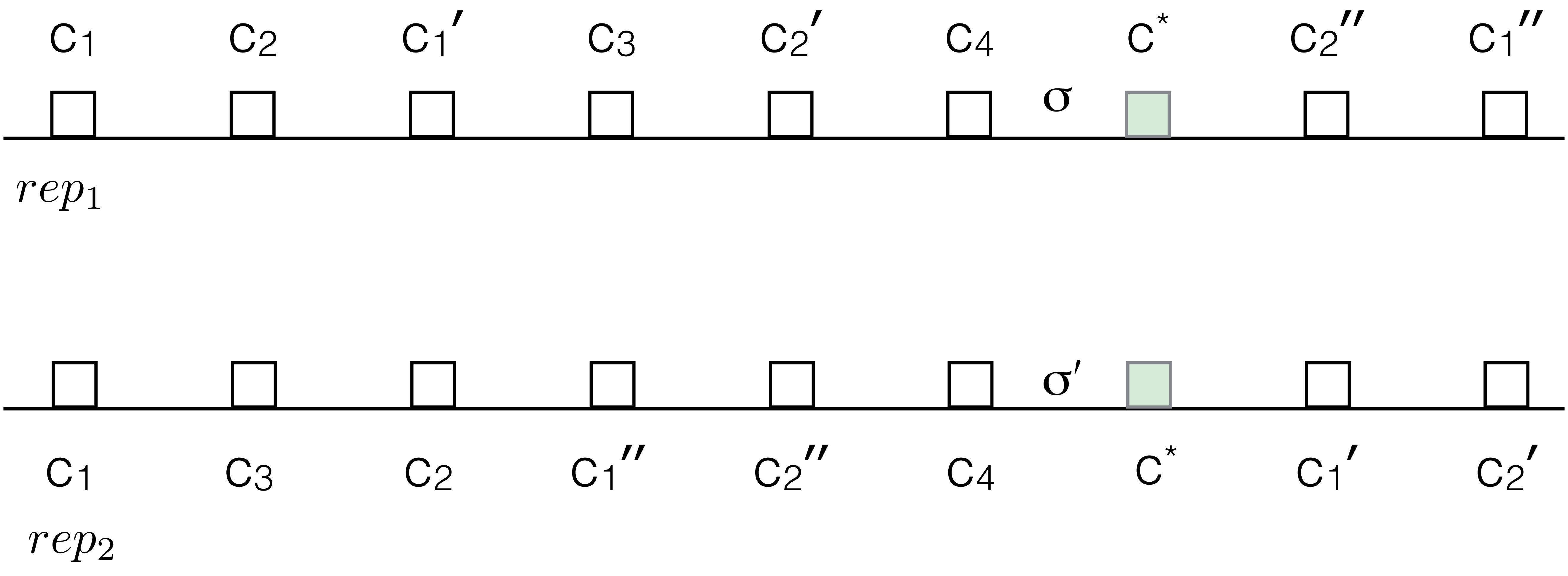


Well-coordination

$\mathcal{S}$-commute

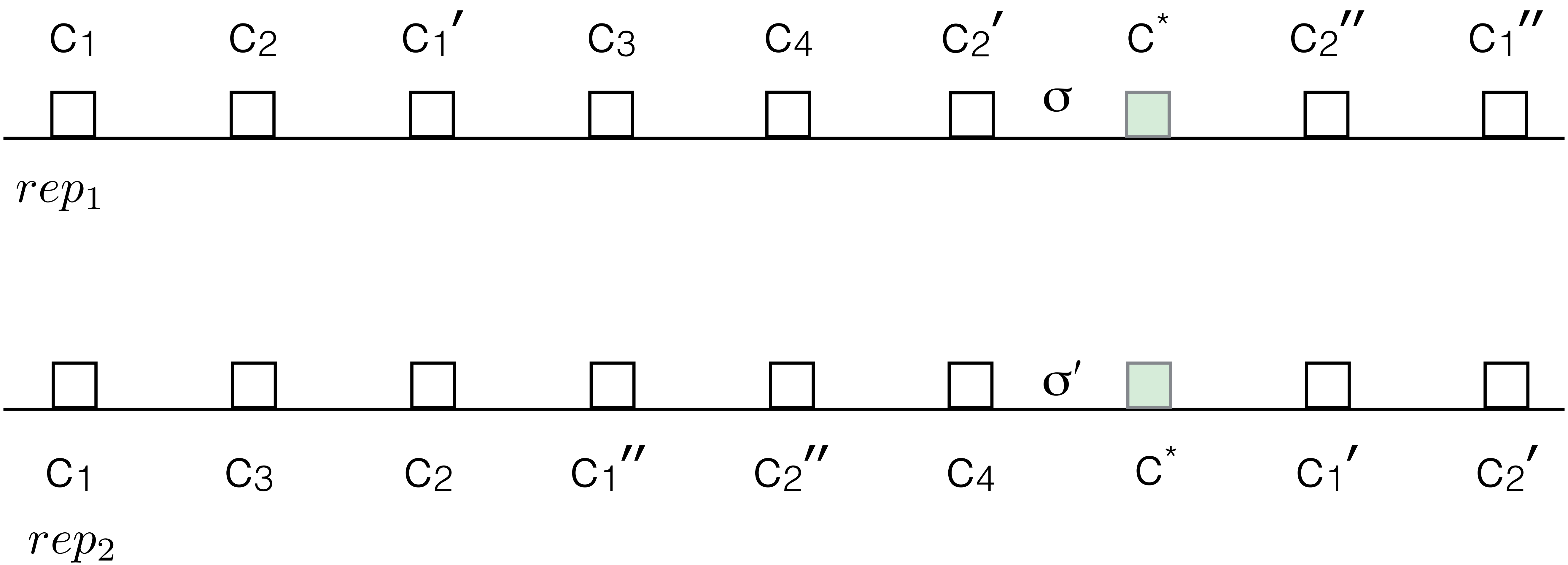


$\mathcal{S}$-commute



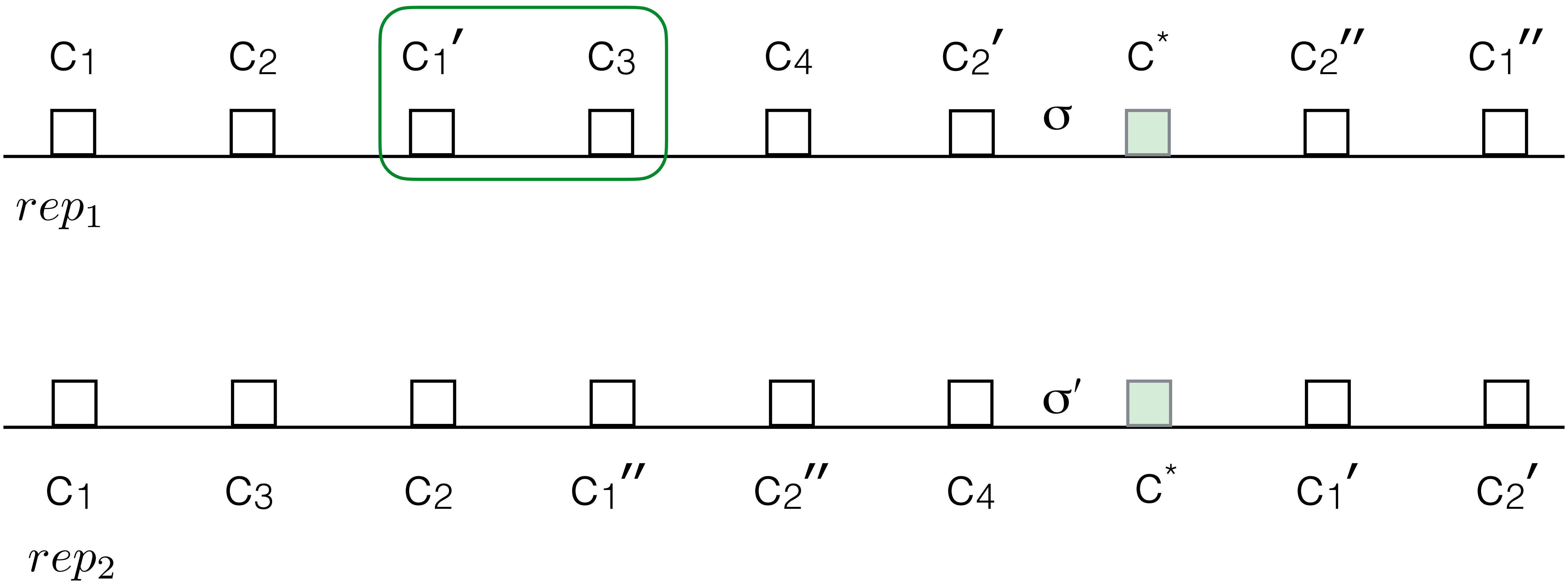
Well-coordination

$\mathcal{S}$-commute



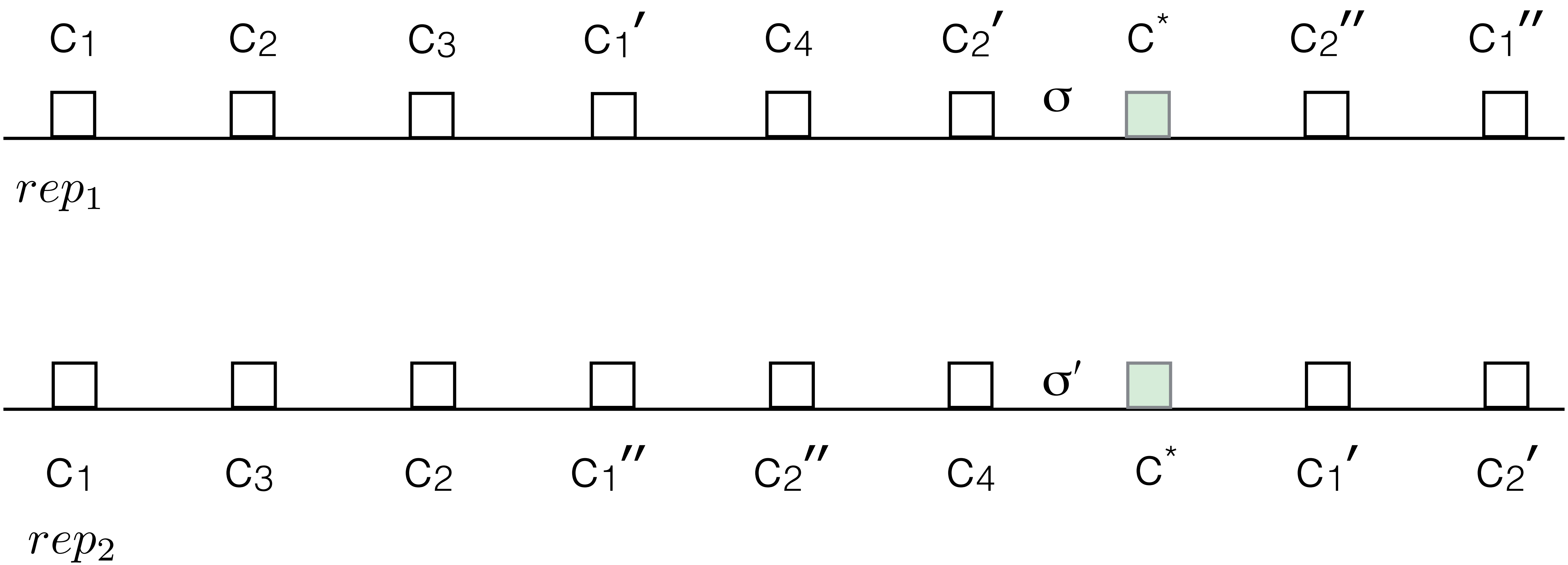
Well-coordination

$\mathcal{S}$-commute



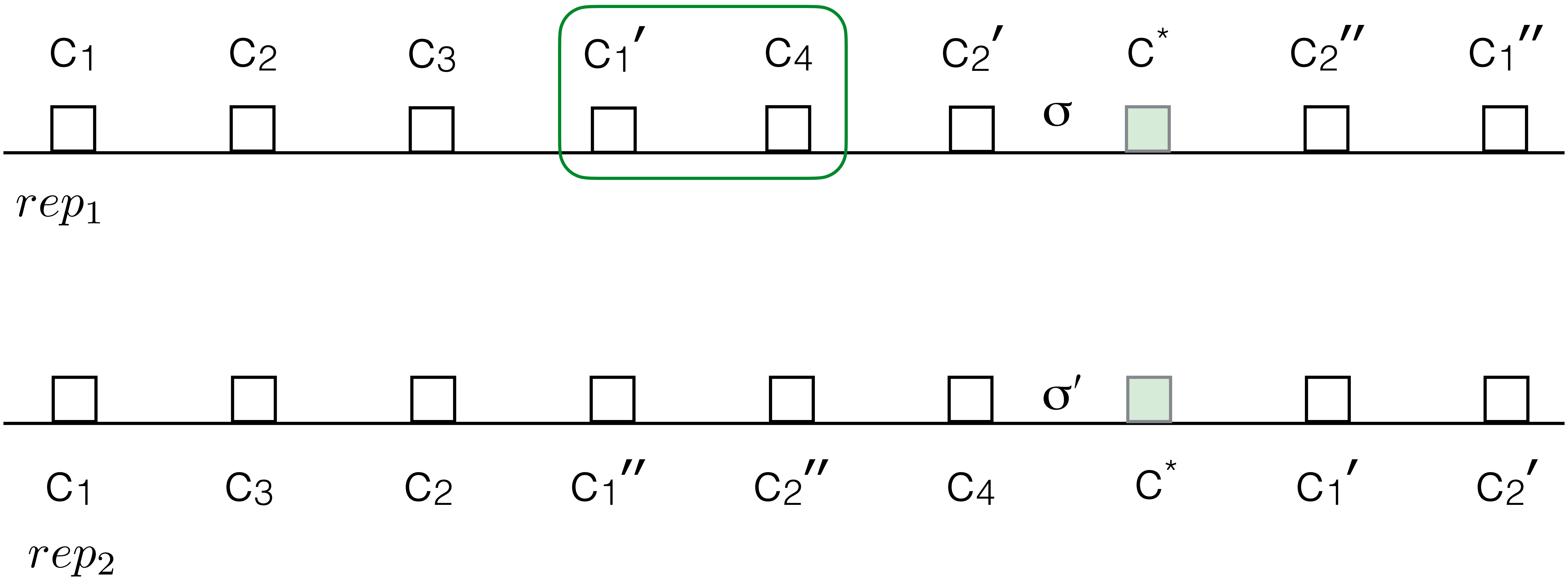
Well-coordination

$\mathcal{S}$-commute



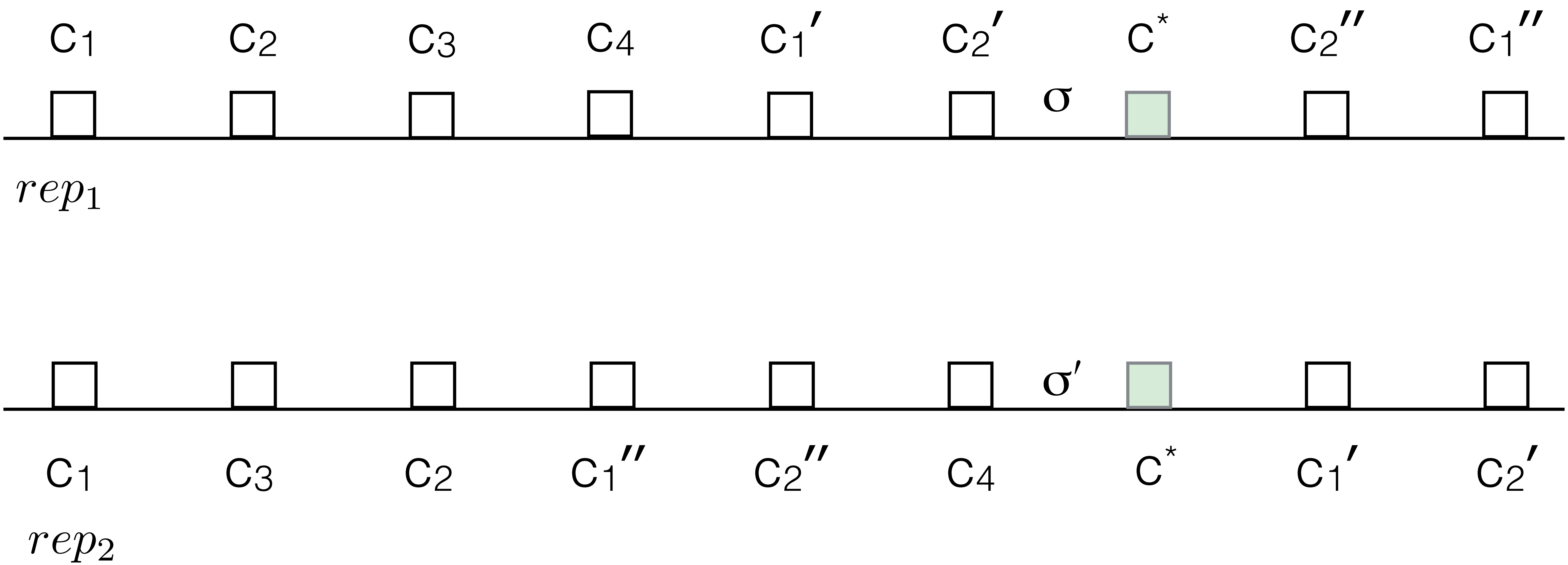
Well-coordination

$\mathcal{S}$-commute



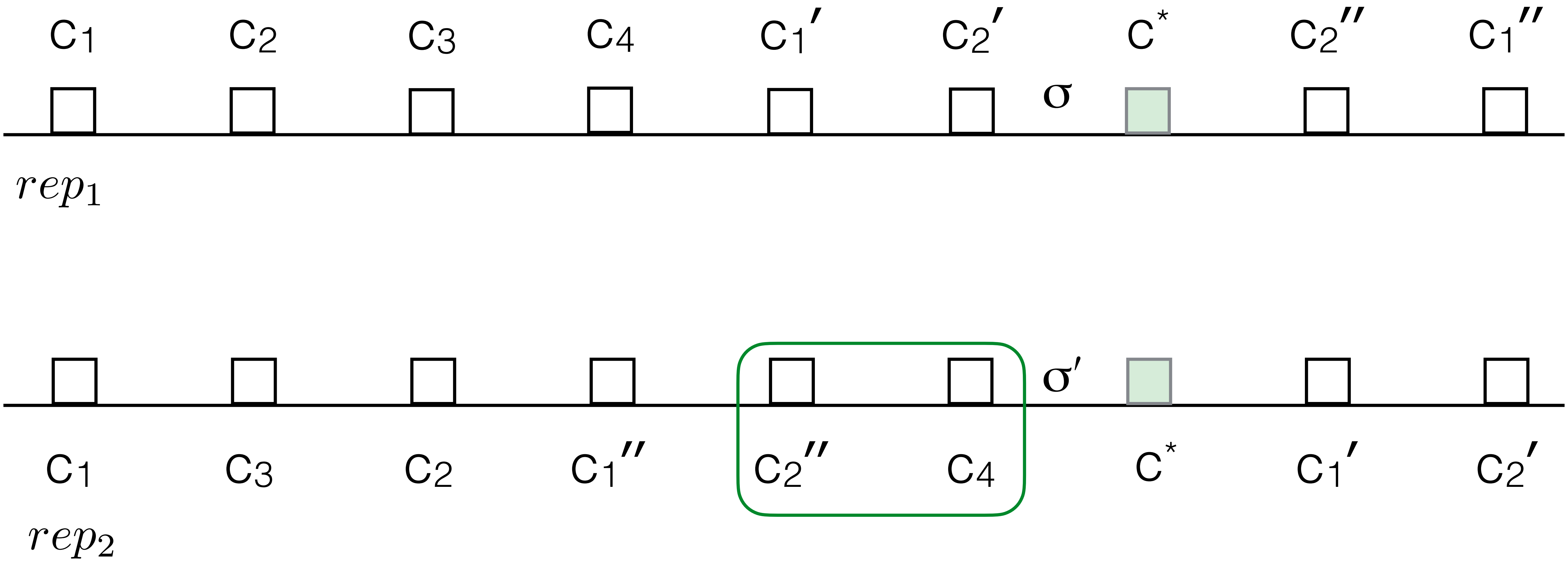
Well-coordination

$\mathcal{S}$-commute



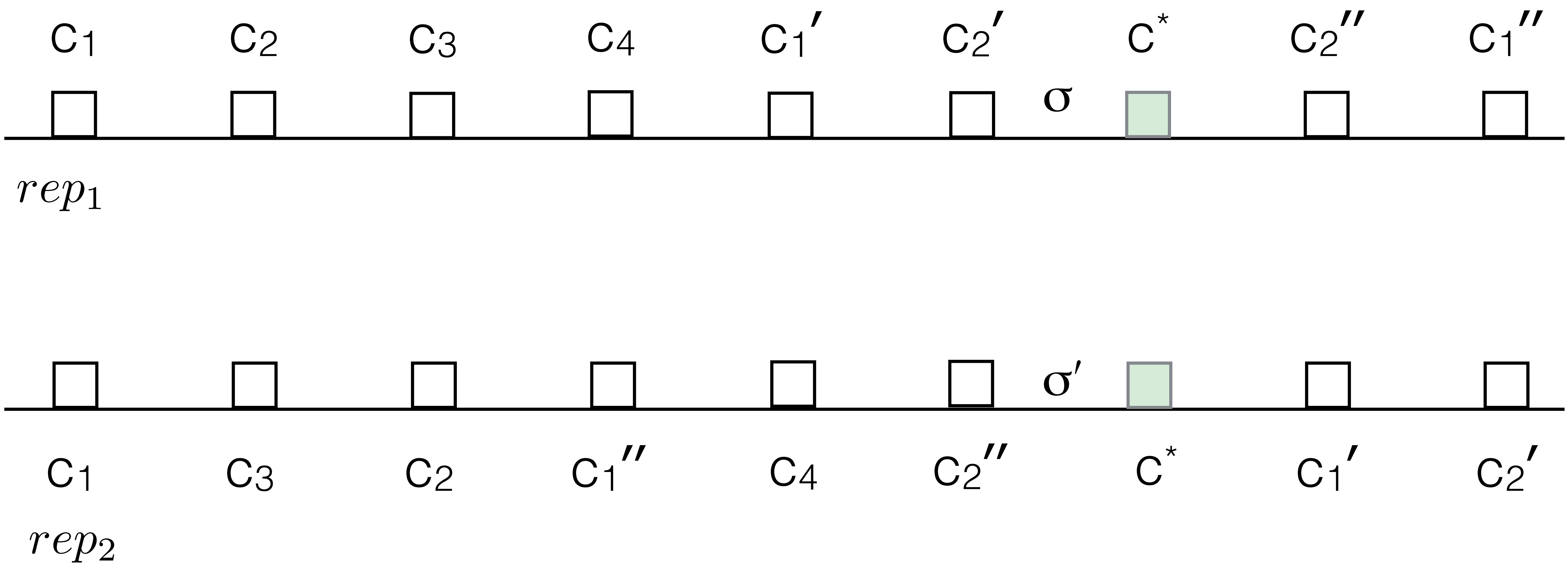
Well-coordination

$\mathcal{S}$-commute



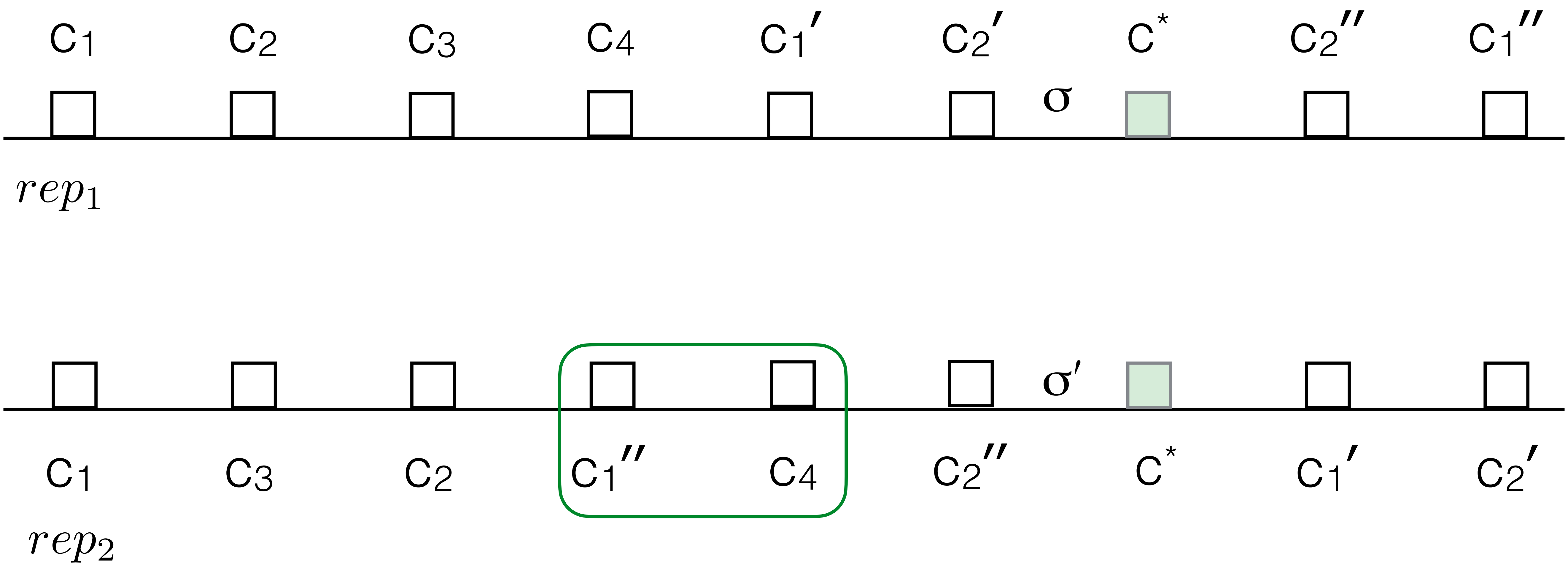
Well-coordination

$\mathcal{S}$-commute



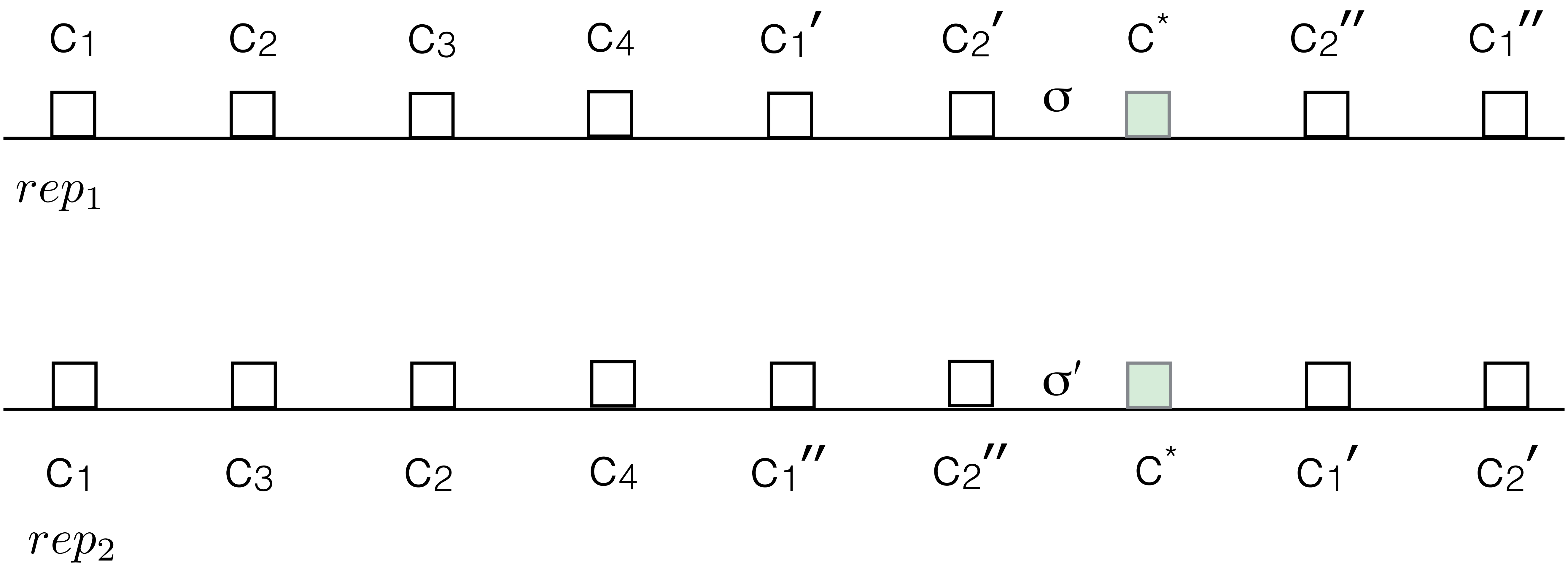
Well-coordination

$\mathcal{S}$-commute



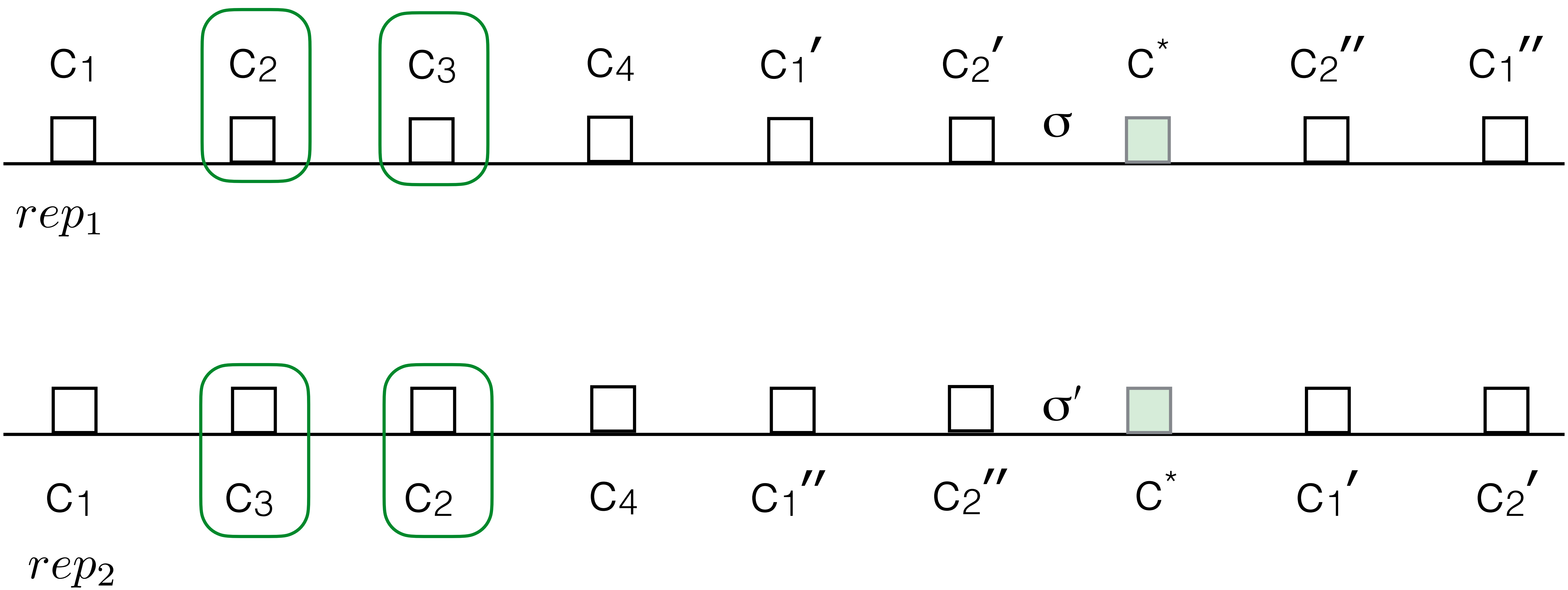
Well-coordination

$\mathcal{S}$-commute



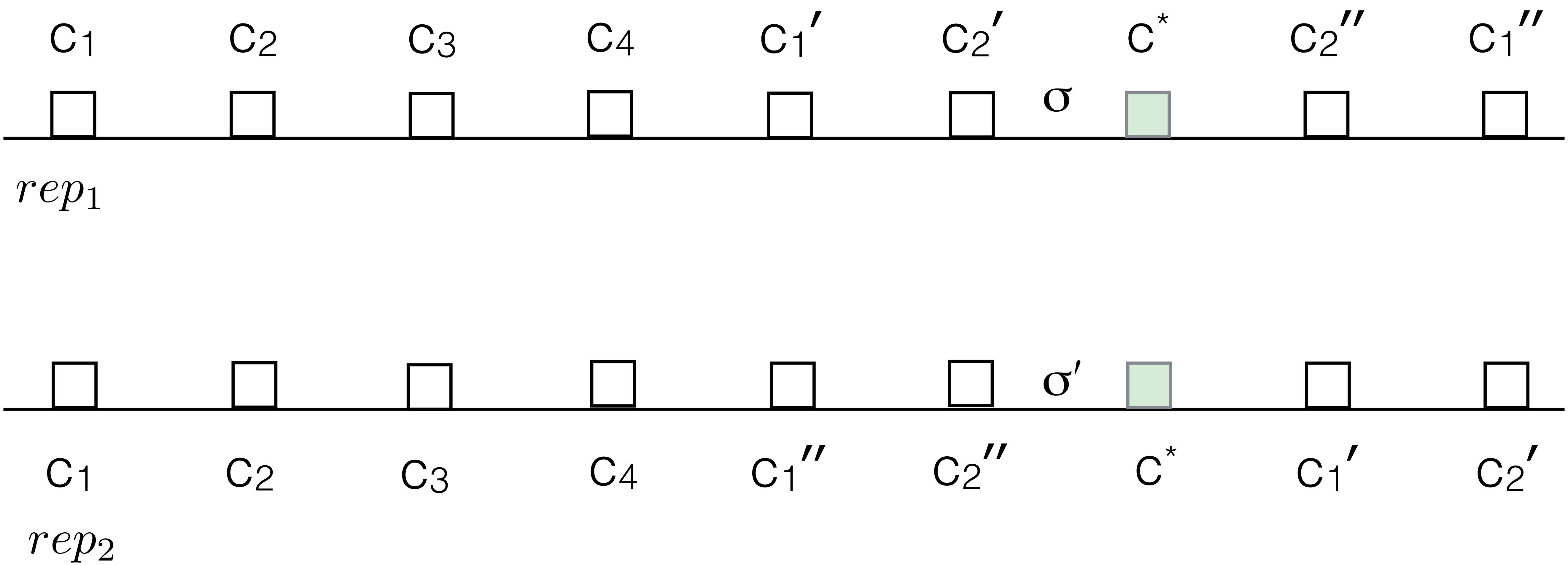
Well-coordination

$\mathcal{S}$-commute

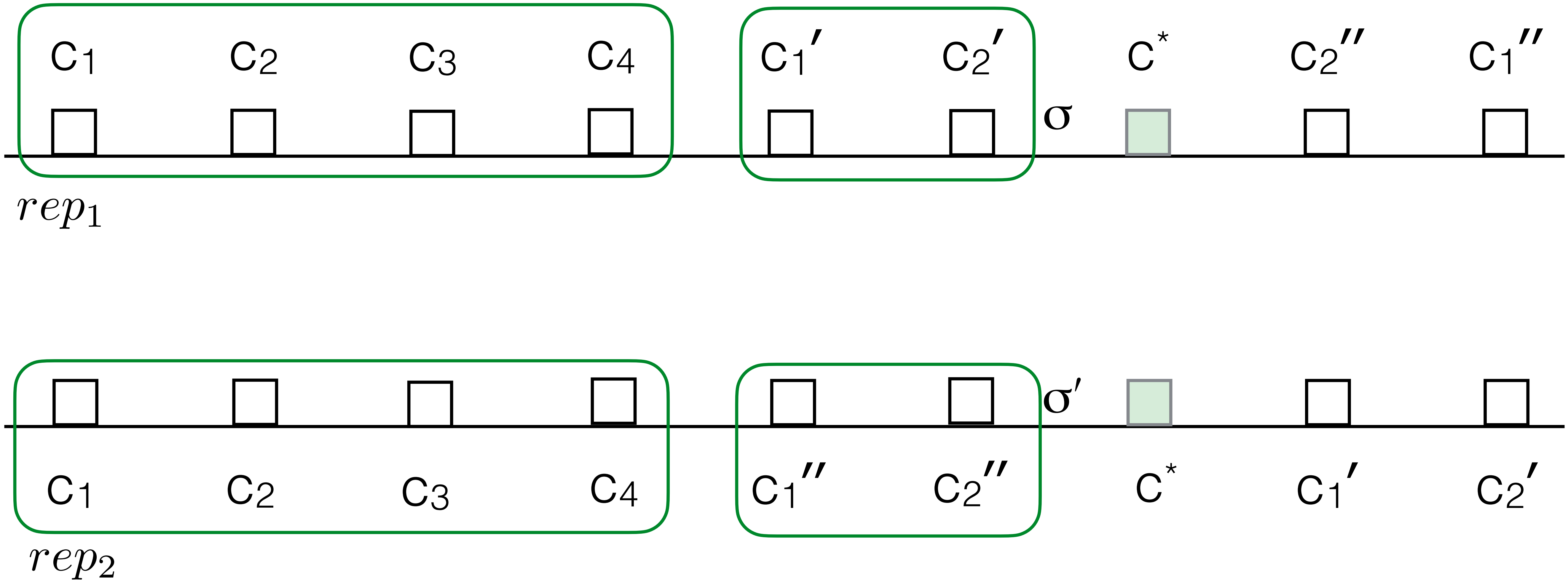


Well-coordination

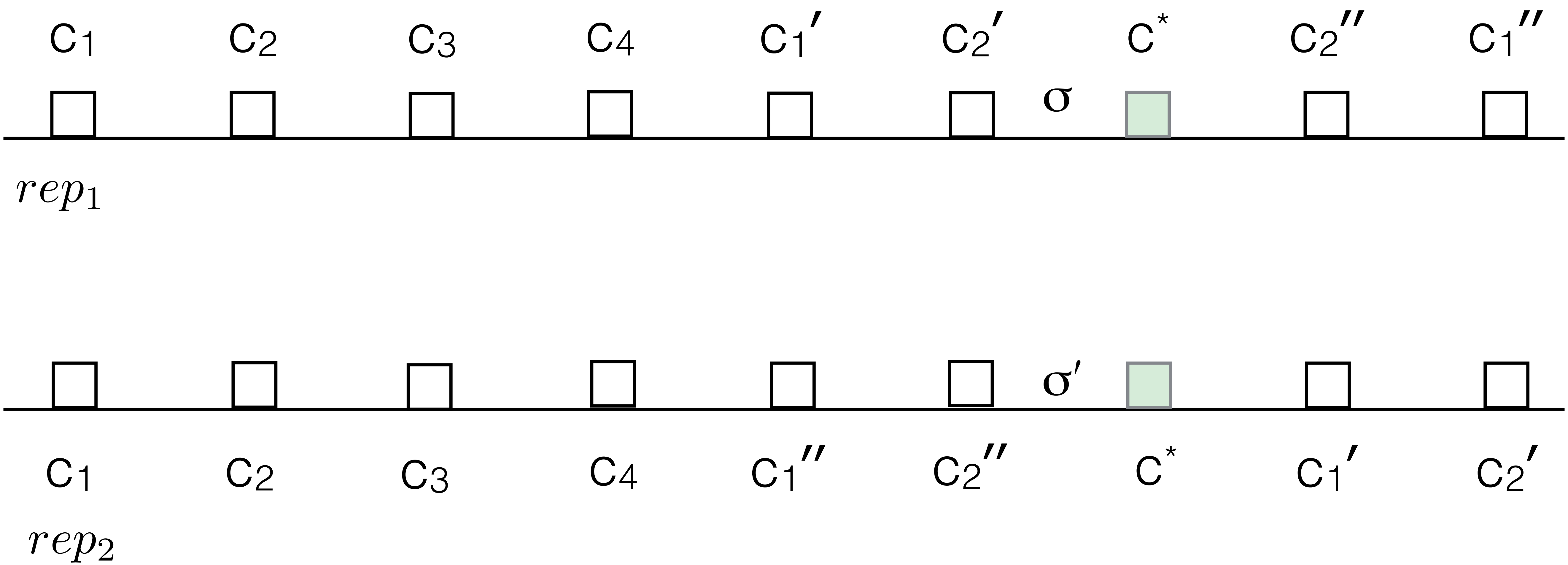
$\mathcal{S}$-commute



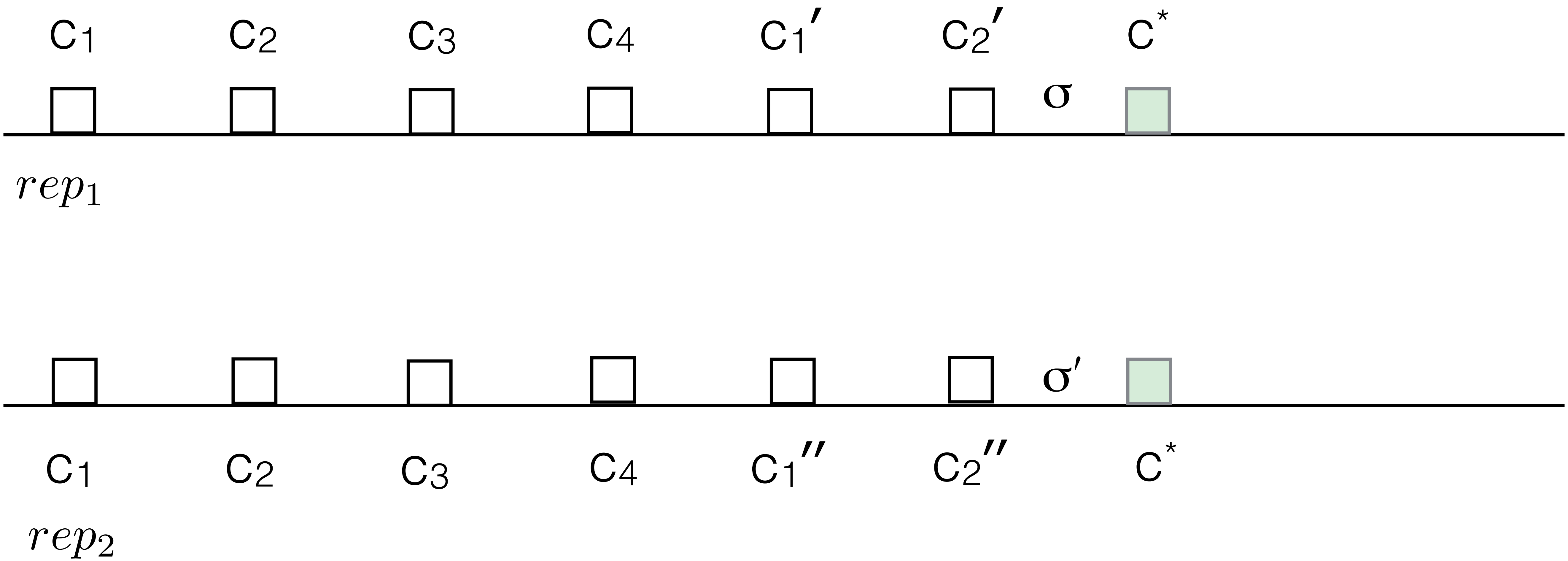
Well-coordination



Well-coordination

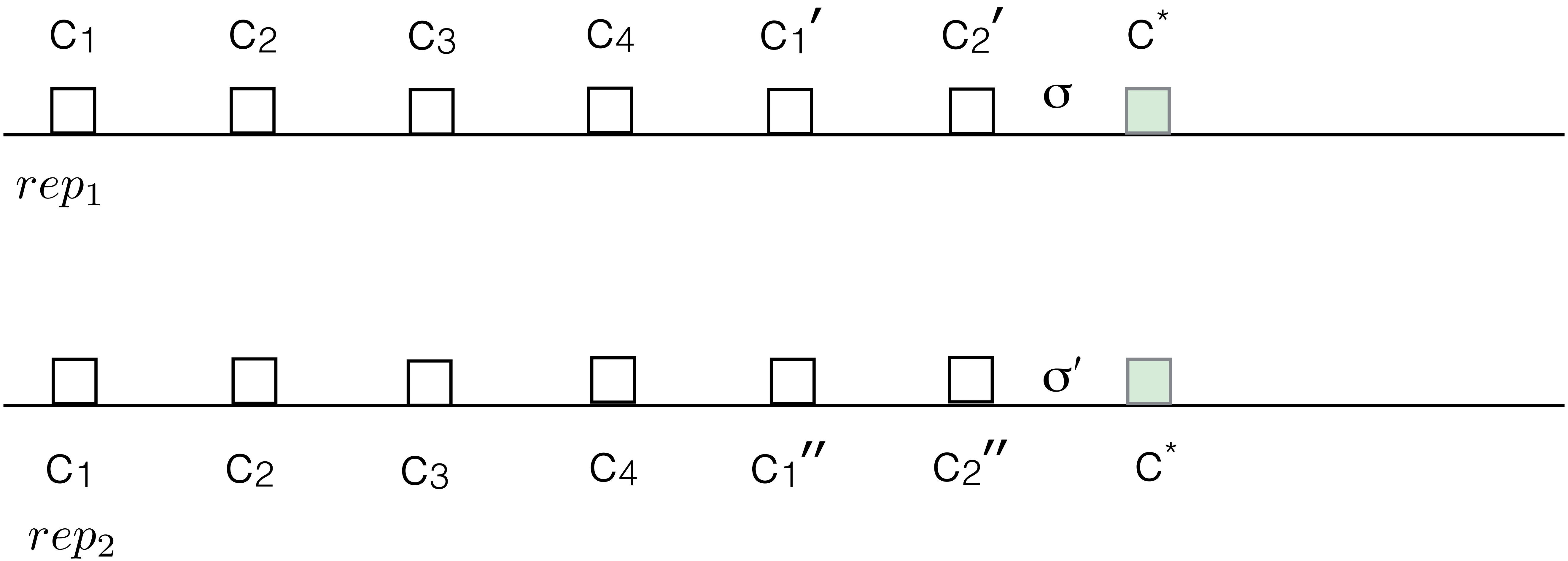


Well-coordination



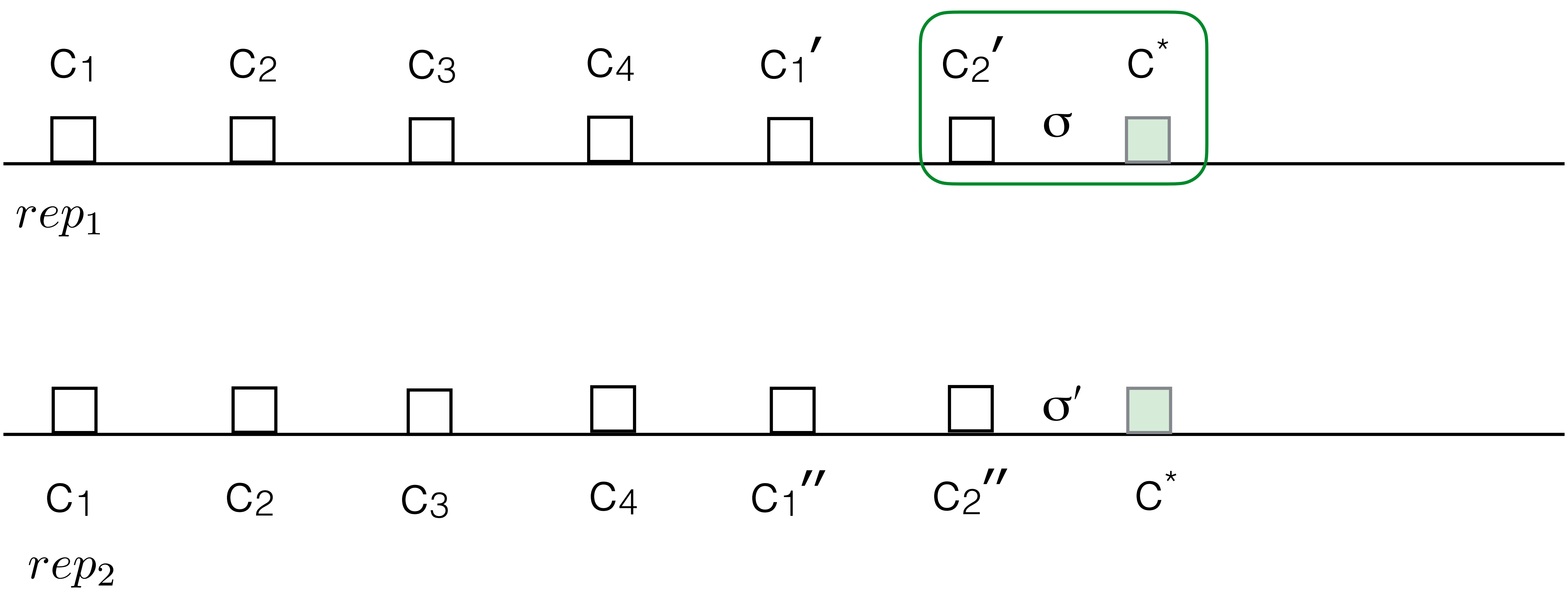
Well-coordination

$\mathcal{P}$-L-Commutativity



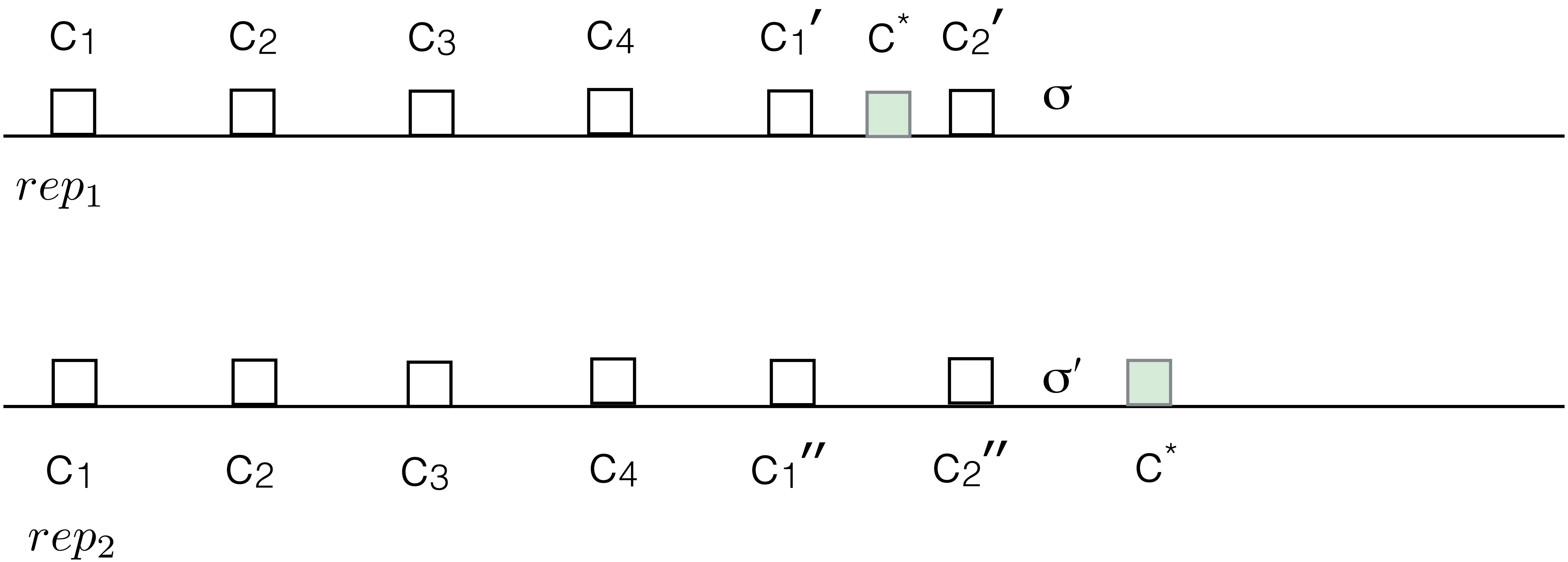
Well-coordination

$\mathcal{P}$-L-Commutativity



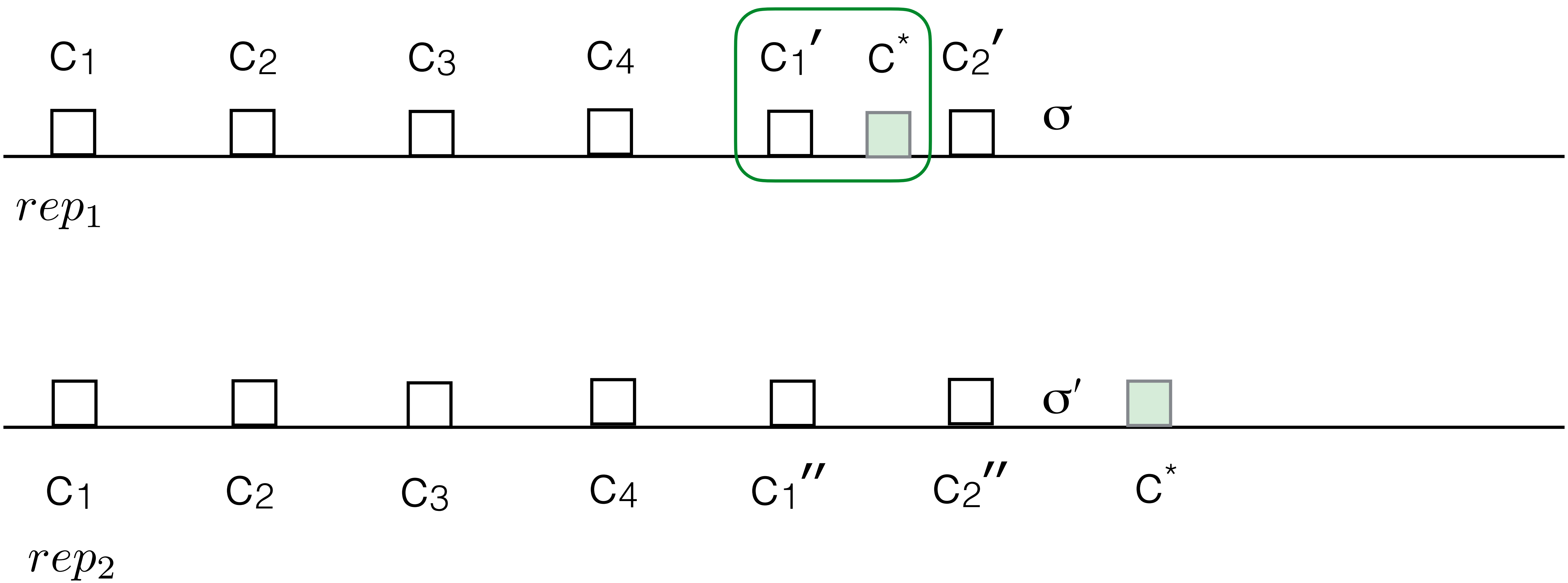
Well-coordination

$\mathcal{P}$-L-Commutativity



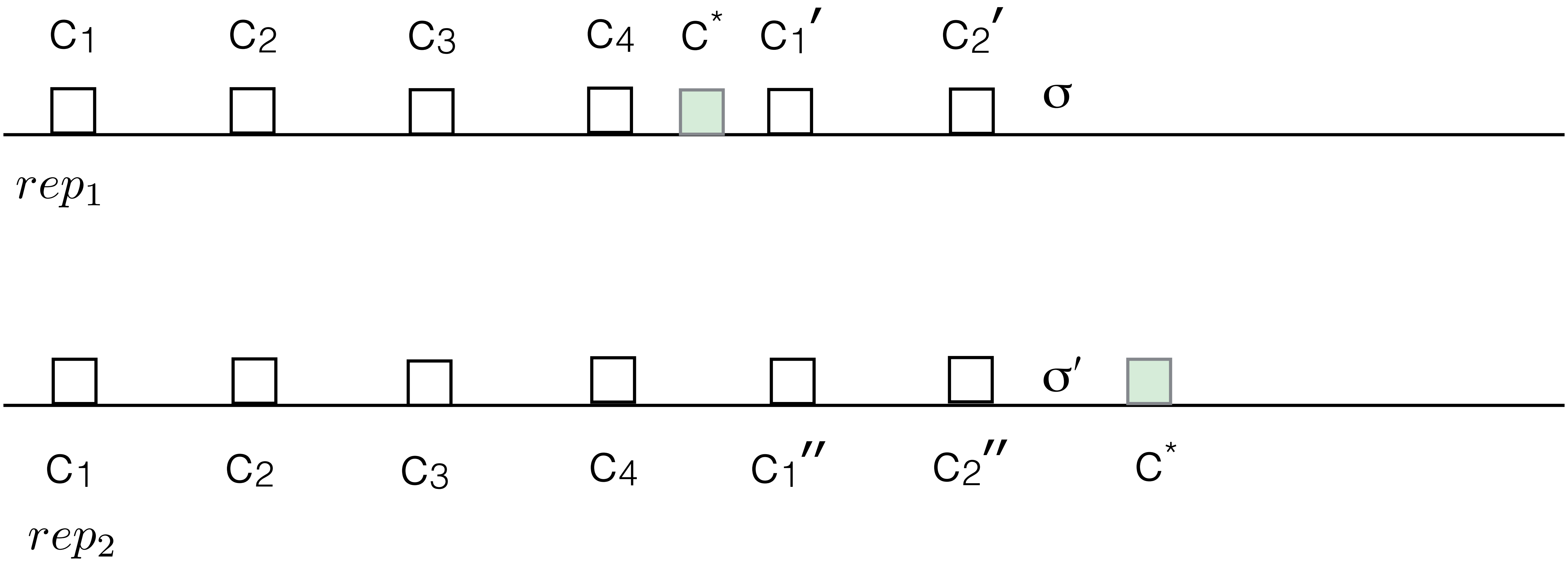
Well-coordination

$\mathcal{P}$-L-Commutativity

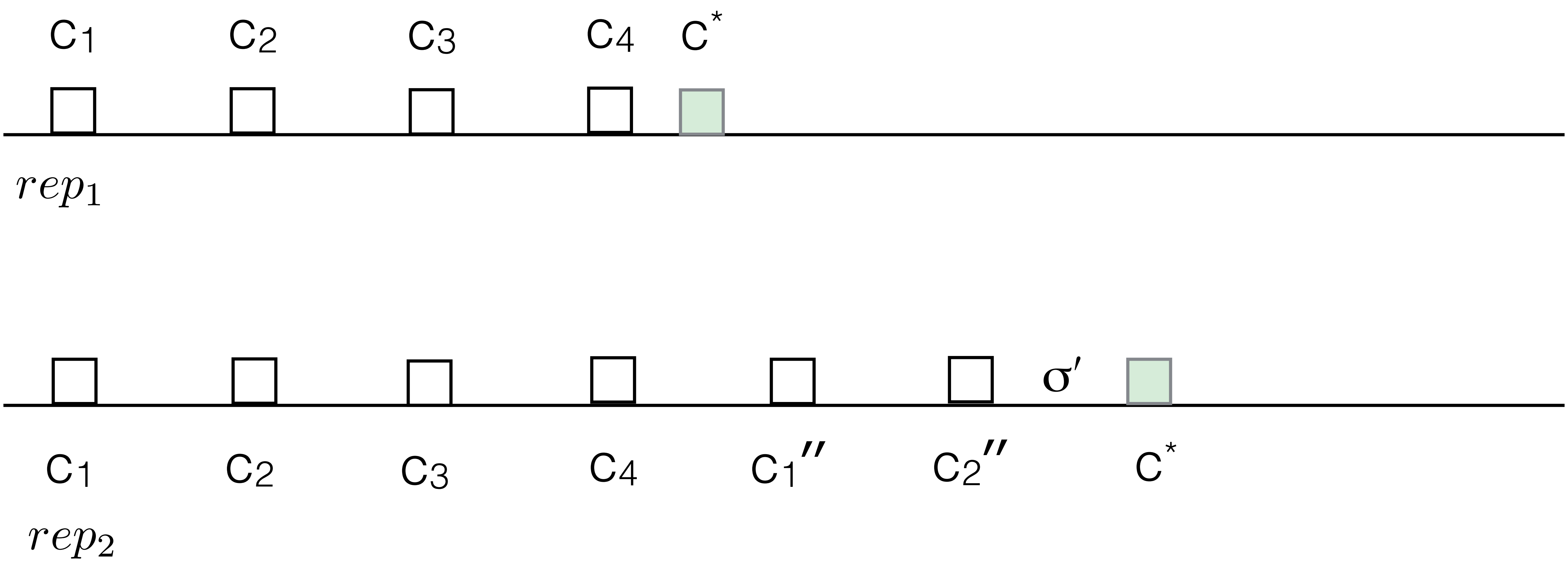


Well-coordination

$\mathcal{P}$-L-Commutativity

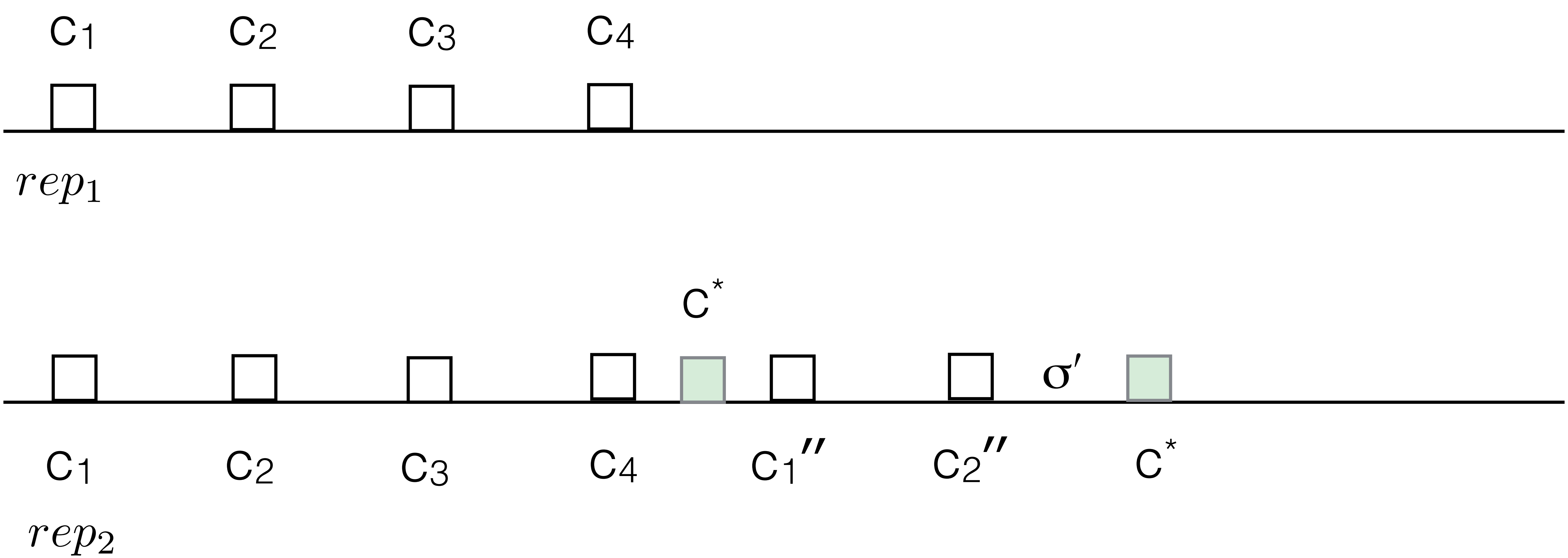


Well-coordination



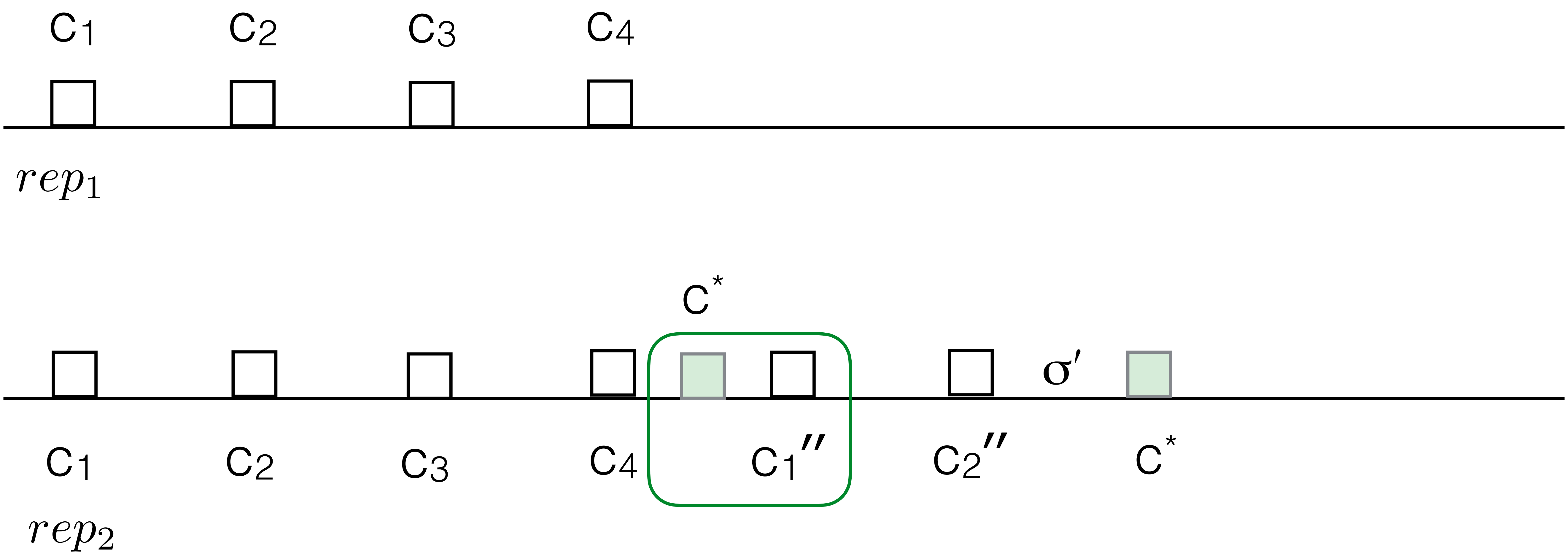
Well-coordination

$\mathcal{P}$-R-Commutativity



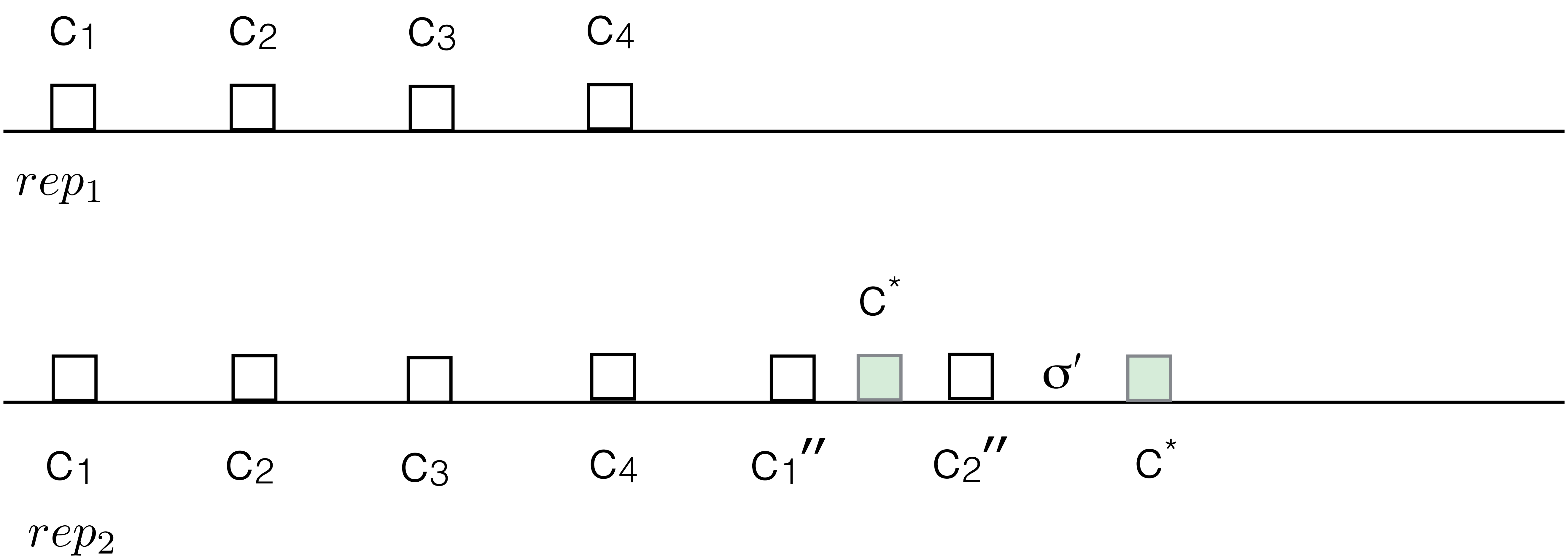
Well-coordination

$\mathcal{P}$-R-Commutativity

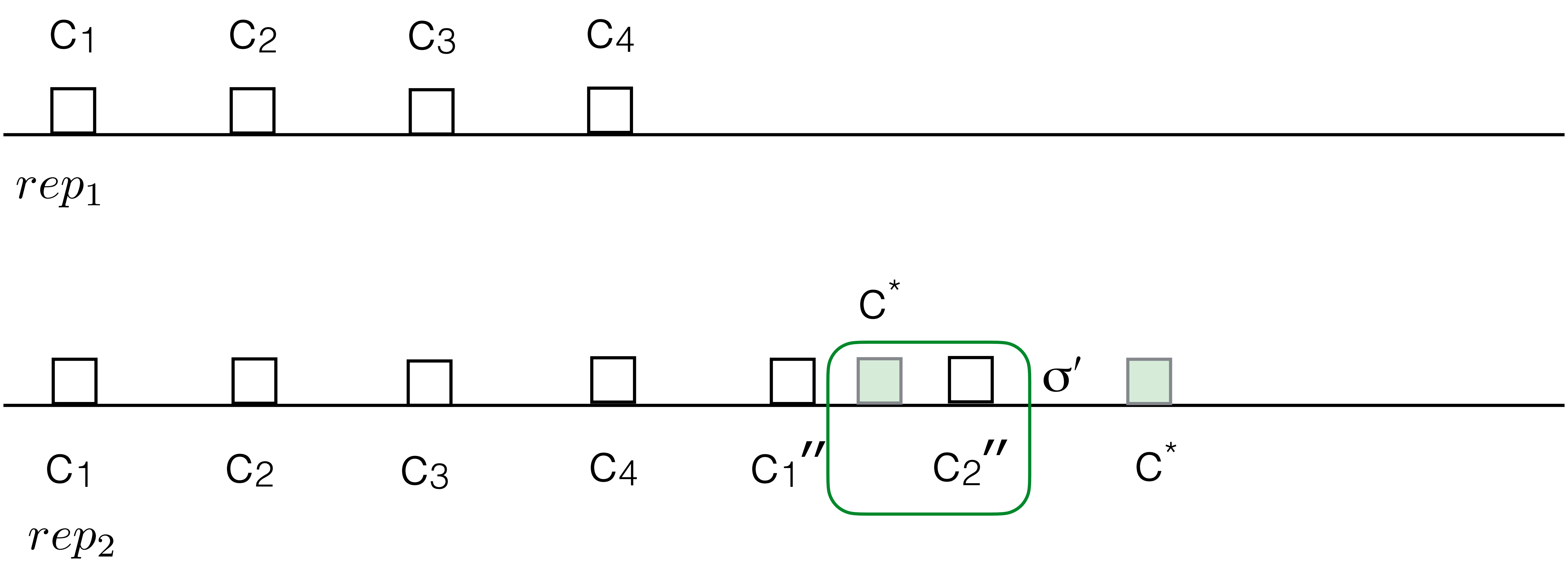


Well-coordination

$\mathcal{P}$-R-Commutativity

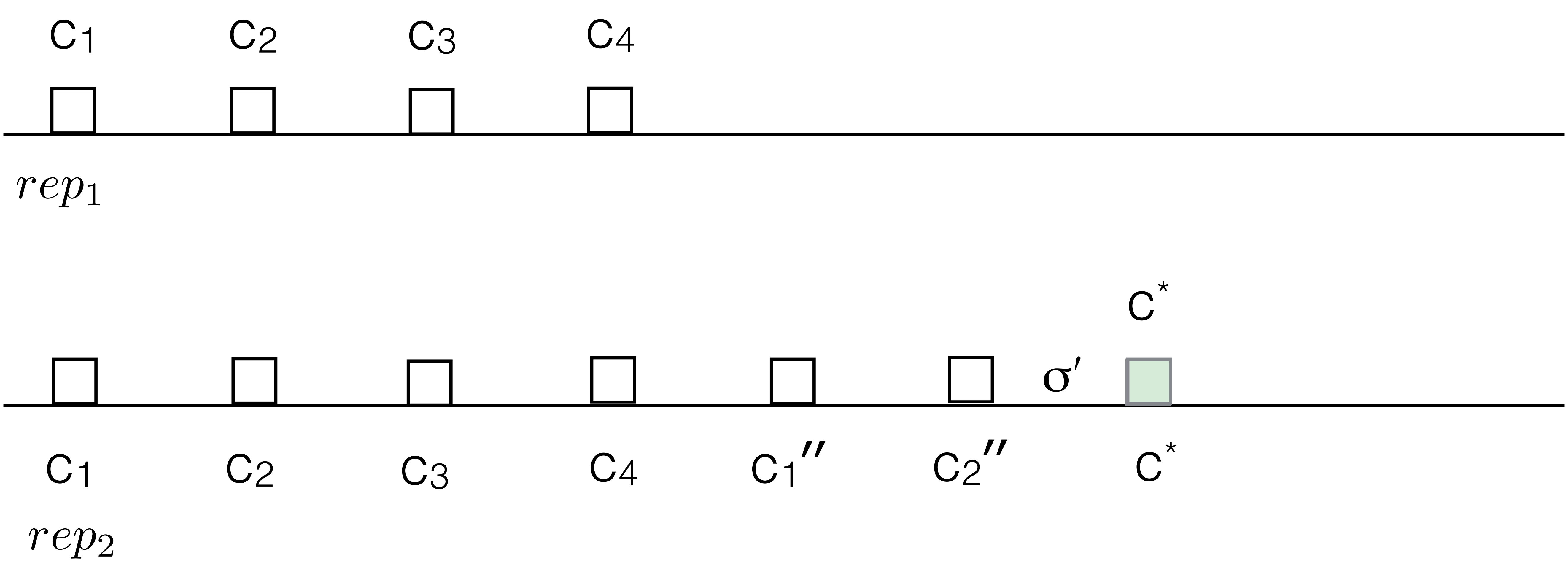


$\mathcal{P}$-R-Commutativity

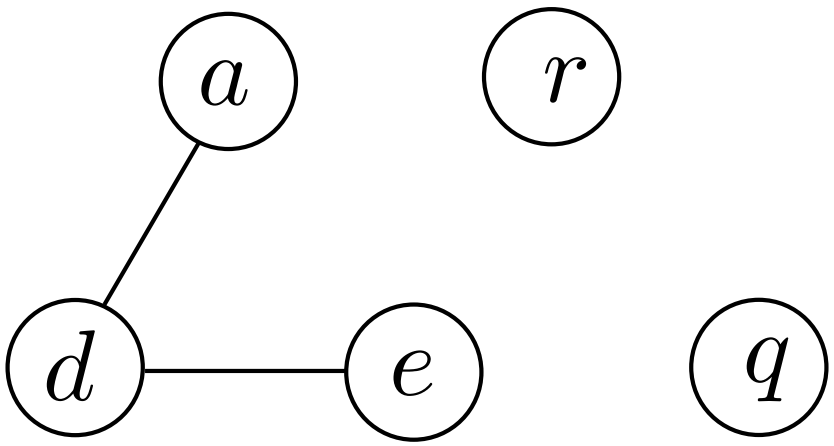


Well-coordination

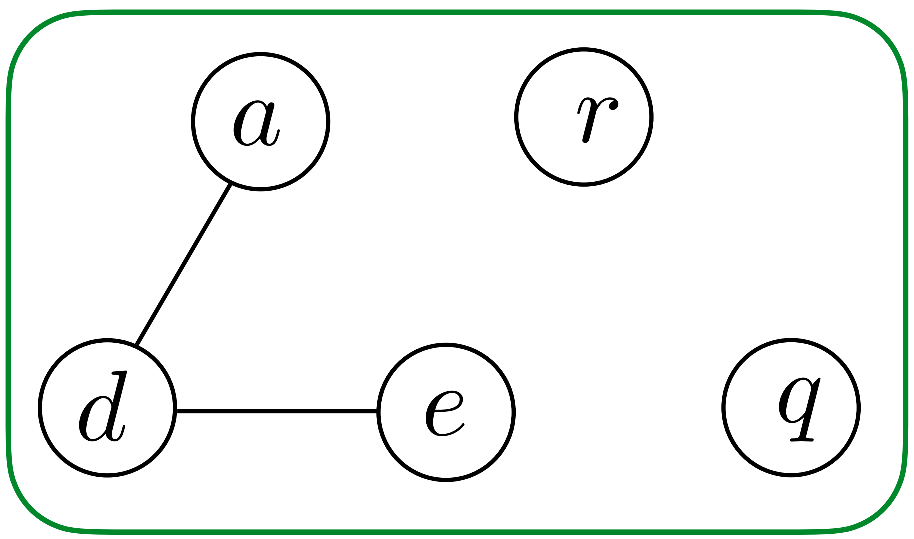
$\mathcal{P}$-R-Commutativity



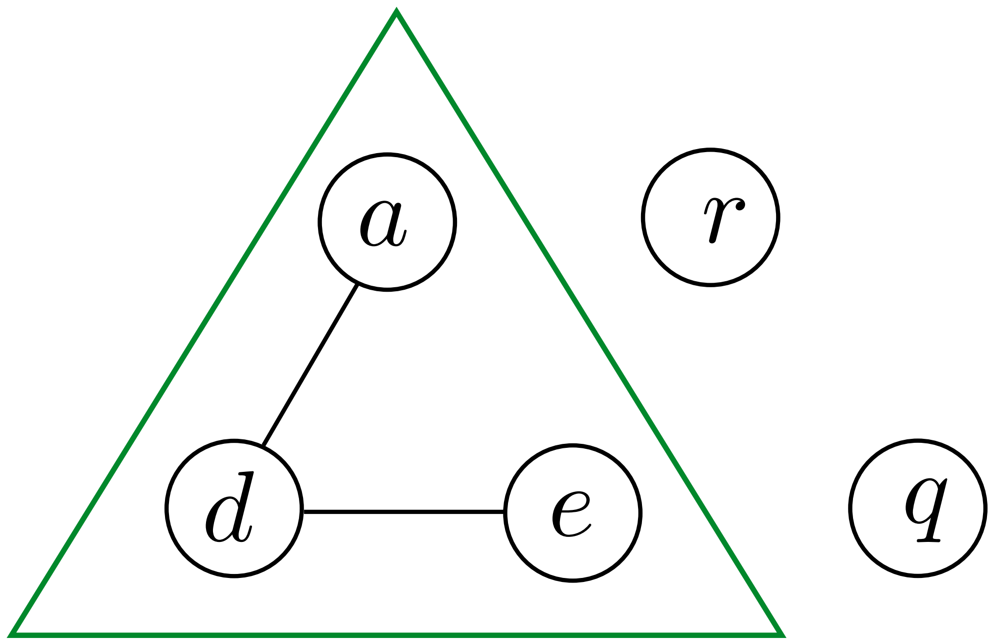
Synchronization

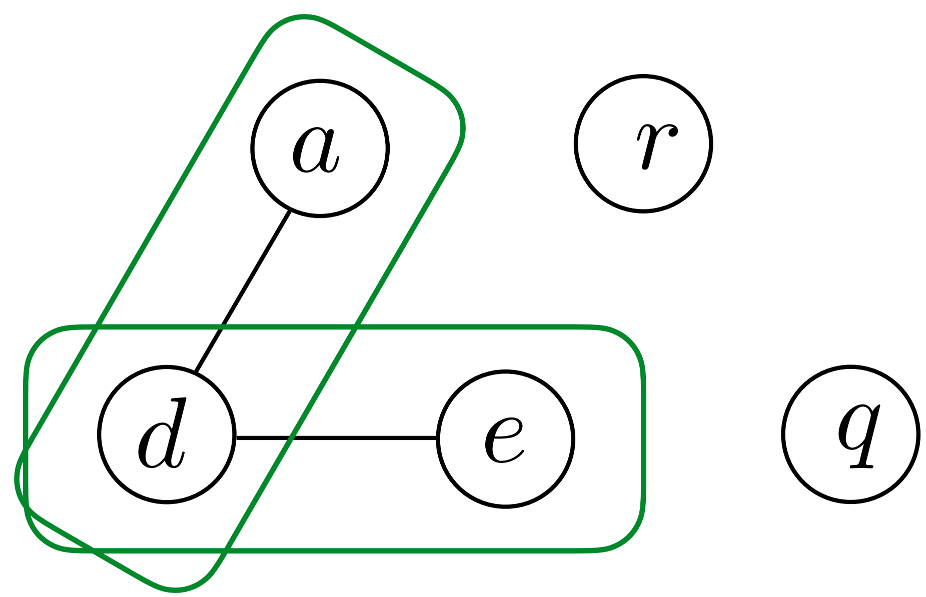


Synchronization

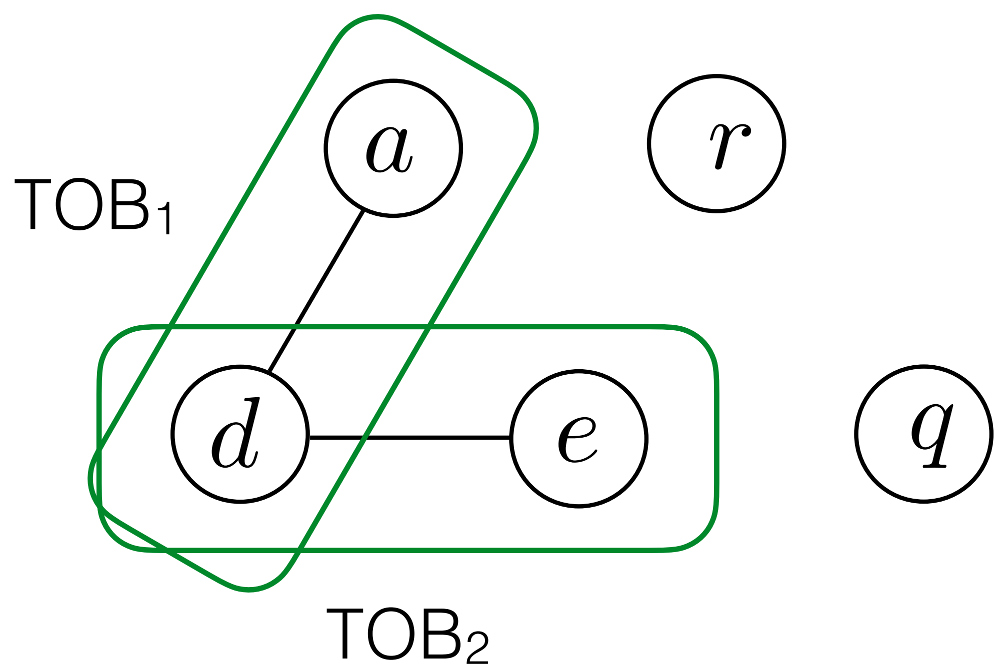


Synchronization



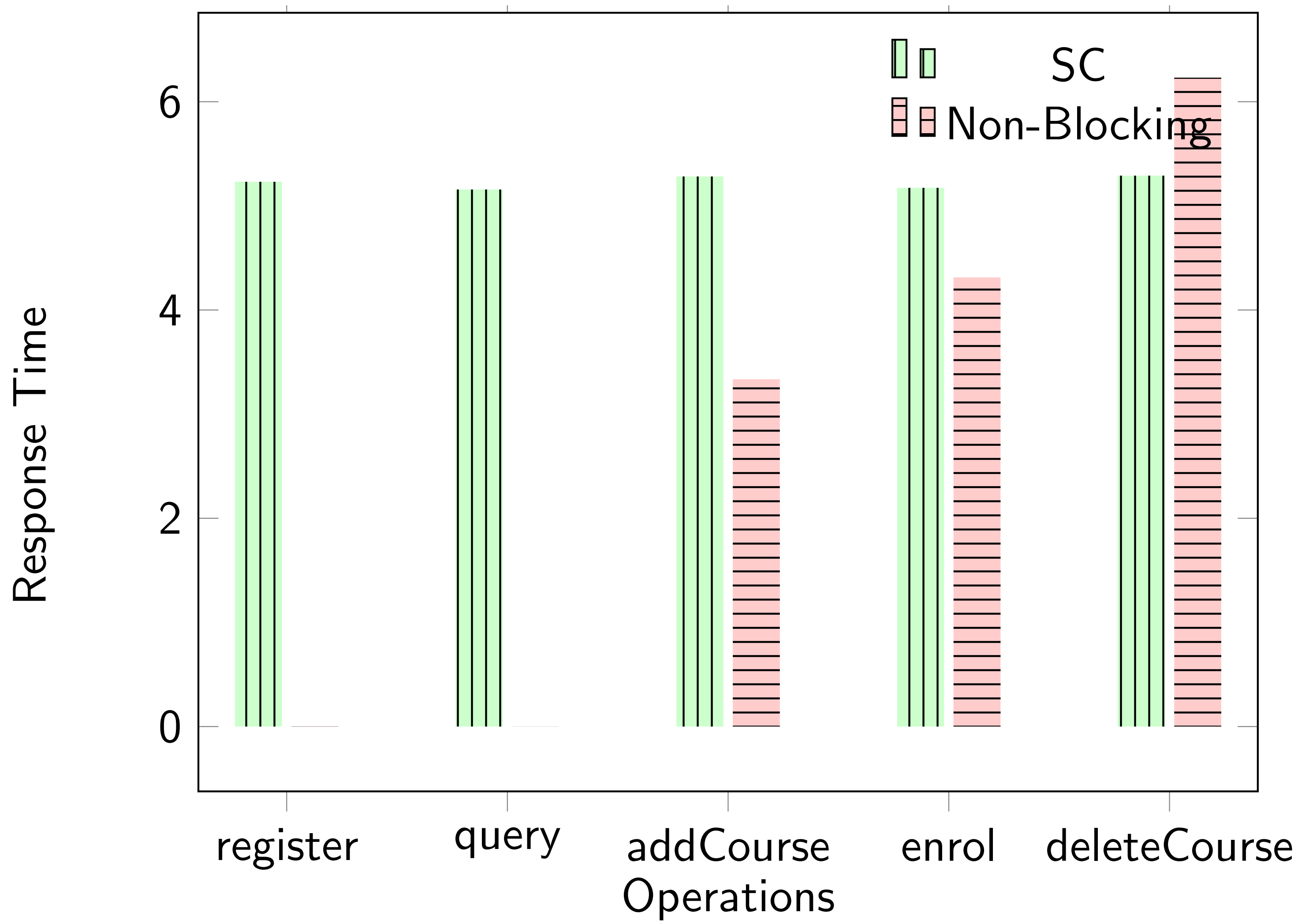


Maximal Cliques



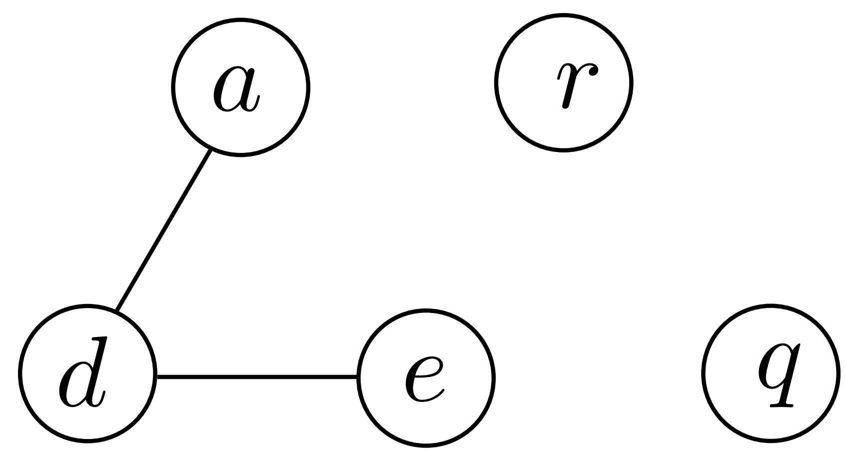
Maximal Cliques

Experimental Results

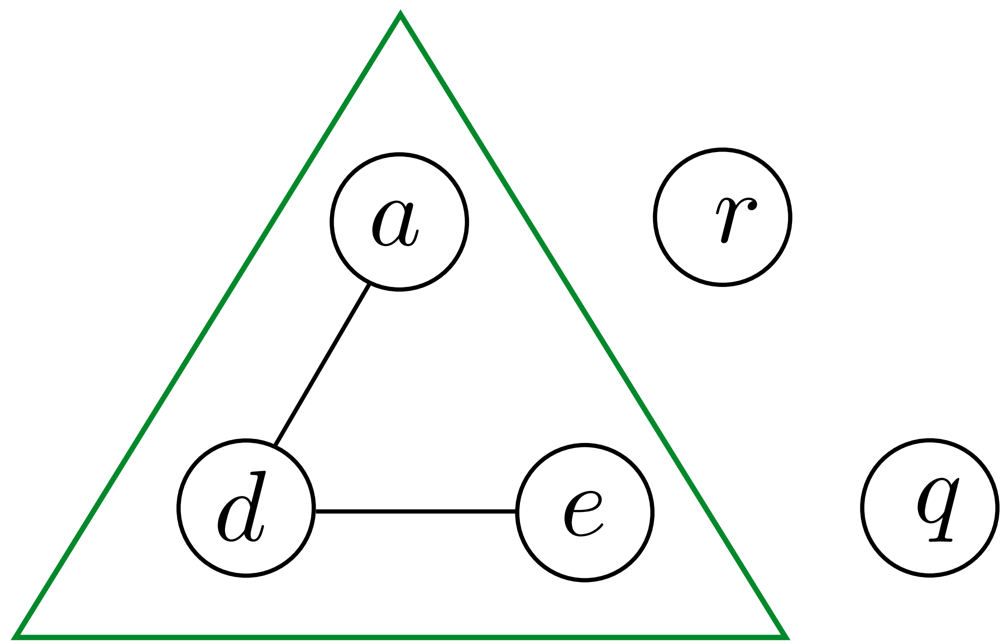


Response time for each method

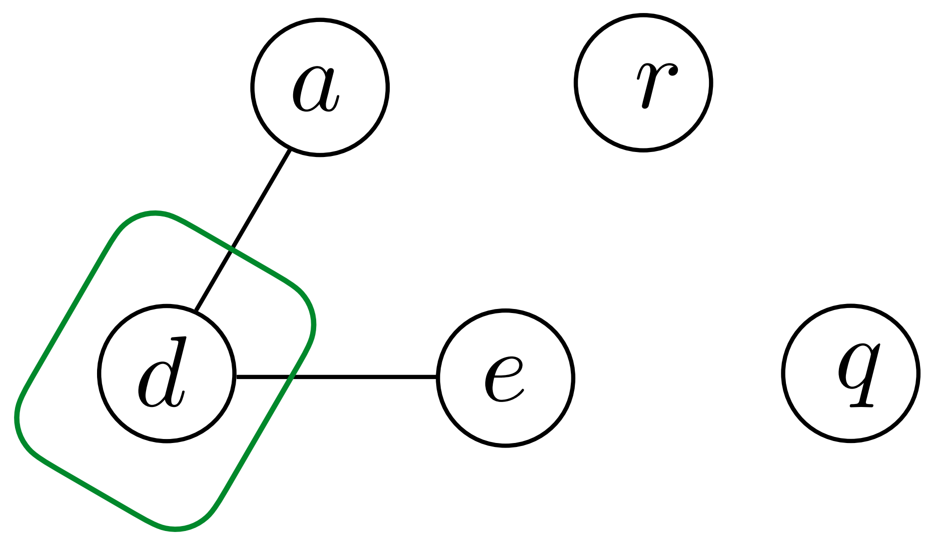
Asymmetric Synchronization



Asymmetric Synchronization



Asymmetric Synchronization

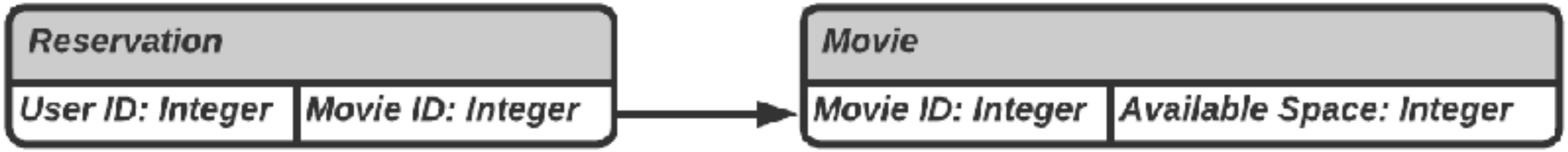


Minimum Vertex Cover

Guarantees

- Convergence
- Integrity
- Recency?

Movie Booking use-case



book($\langle u, m \rangle$)

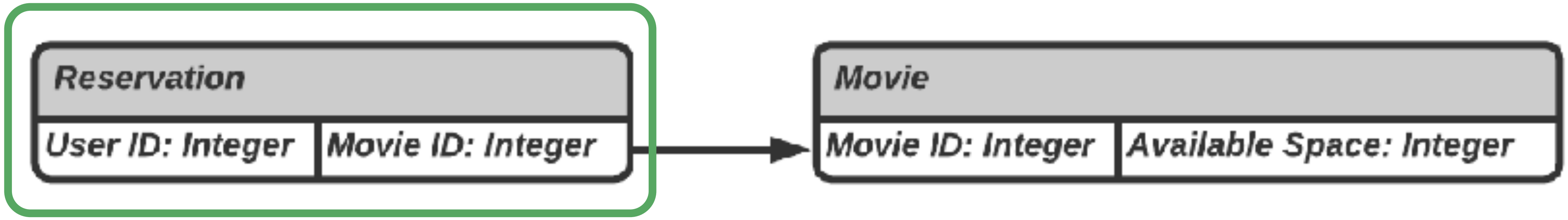
cancelBook($\langle u, m \rangle$)

offScreen(m)

specialReserve($\langle m, n \rangle$)

increaseSpace($\langle m, n \rangle$)

Movie Booking use-case



book$(\langle u, m \rangle)$

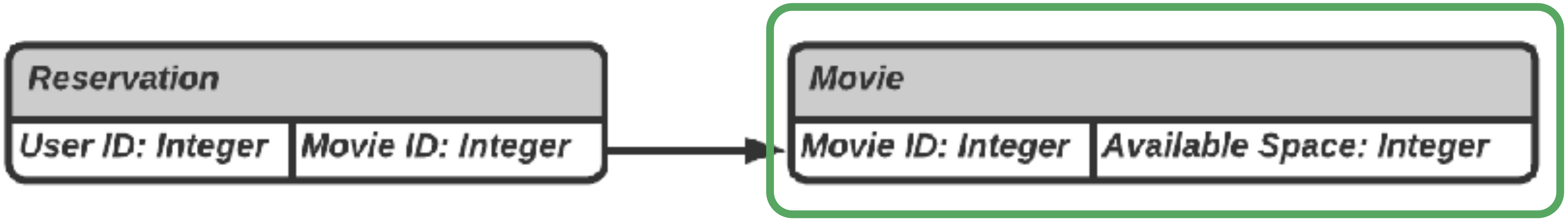
cancelBook$(\langle u, m \rangle)$

offScreen(m)

specialReserve$(\langle m, n \rangle)$

increaseSpace$(\langle m, n \rangle)$

Movie Booking use-case



book($\langle u, m \rangle$)

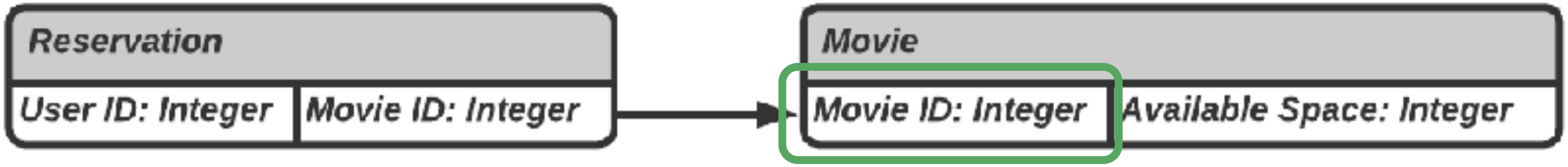
cancelBook($\langle u, m \rangle$)

offScreen(m)

specialReserve($\langle m, n \rangle$)

increaseSpace($\langle m, n \rangle$)

Movie Booking use-case



book$(\langle u, m \rangle)$

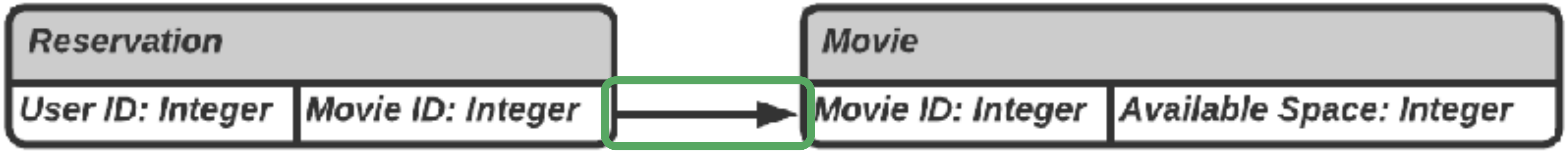
cancelBook$(\langle u, m \rangle)$

offScreen(m)

specialReserve$(\langle m, n \rangle)$

increaseSpace$(\langle m, n \rangle)$

Movie Booking use-case



book$(\langle u, m \rangle)$

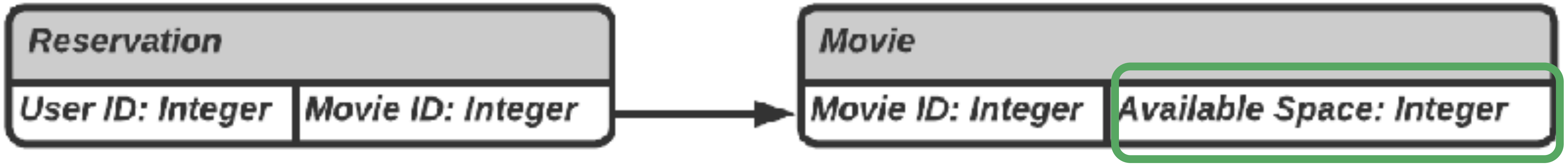
cancelBook$(\langle u, m \rangle)$

offScreen(m)

specialReserve$(\langle m, n \rangle)$

increaseSpace$(\langle m, n \rangle)$

Movie Booking use-case



book($\langle u, m \rangle$)

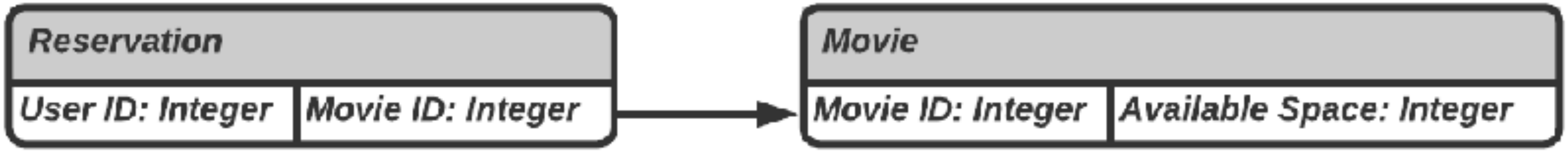
cancelBook($\langle u, m \rangle$)

offScreen(m)

specialReserve($\langle m, n \rangle$)

increaseSpace($\langle m, n \rangle$)

Movie Booking use-case



book($\langle u, m \rangle$)

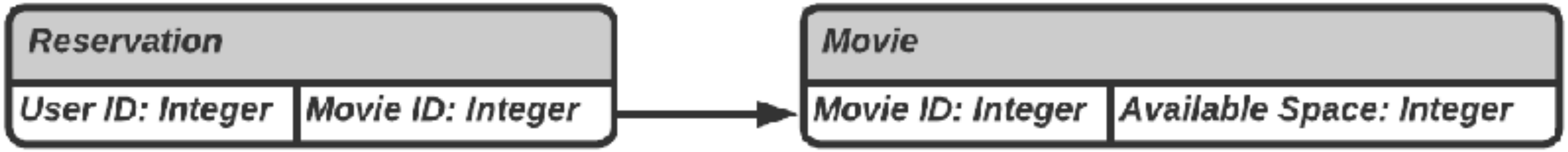
cancelBook($\langle u, m \rangle$)

offScreen(m)

specialReserve($\langle m, n \rangle$)

increaseSpace($\langle m, n \rangle$)

Movie Booking use-case



book($\langle u, m \rangle$)

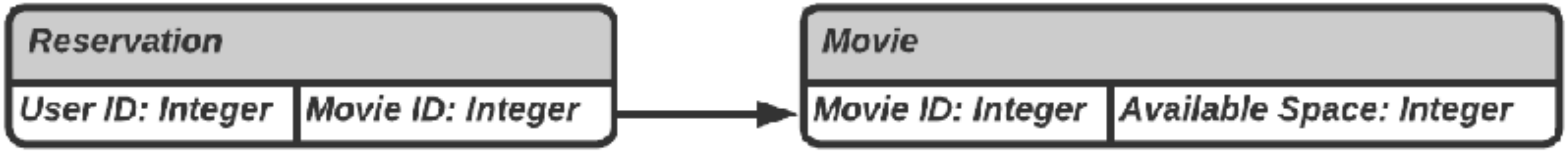
cancelBook($\langle u, m \rangle$)

offScreen(m)

specialReserve($\langle m, n \rangle$)

increaseSpace($\langle m, n \rangle$)

Movie Booking use-case



book($\langle u, m \rangle$)

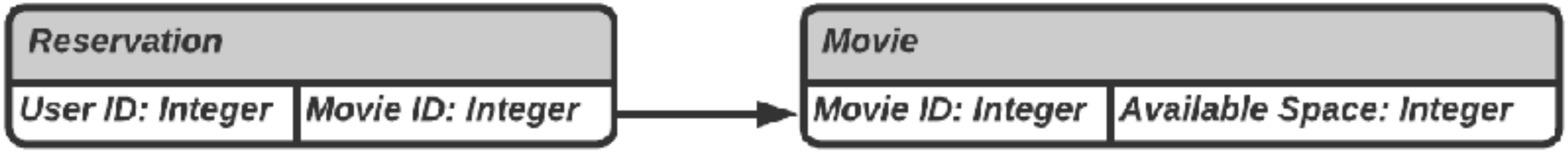
cancelBook($\langle u, m \rangle$)

offScreen(m)

specialReserve($\langle m, n \rangle$)

increaseSpace($\langle m, n \rangle$)

Movie Booking use-case



book$(\langle u, m \rangle)$

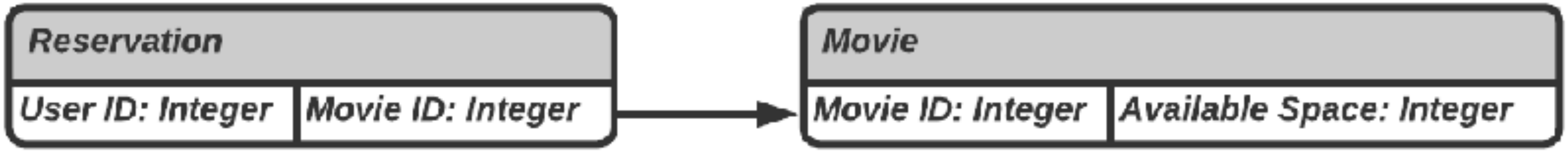
cancelBook$(\langle u, m \rangle)$

offScreen(m)

specialReserve$(\langle m, n \rangle)$

increaseSpace$(\langle m, n \rangle)$

Movie Booking use-case



book$(\langle u, m \rangle)$

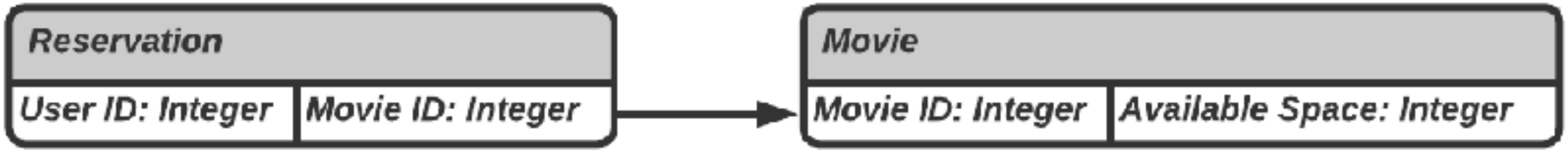
cancelBook$(\langle u, m \rangle)$

offScreen(m)

specialReserve$(\langle m, n \rangle)$

increaseSpace$(\langle m, n \rangle)$

Movie Booking use-case



book$(\langle u, m \rangle)$

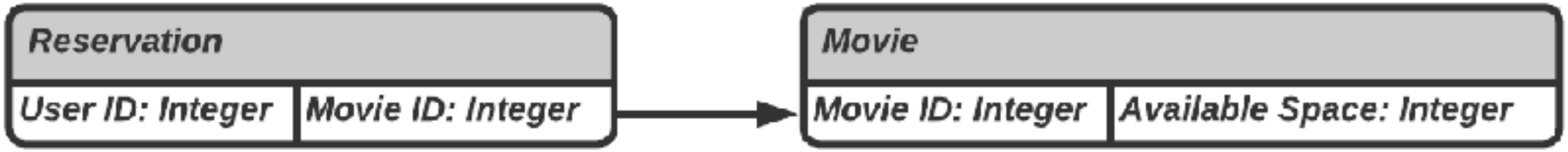
cancelBook$(\langle u, m \rangle)$

offScreen(m)

specialReserve$(\langle m, n \rangle)$

increaseSpace$(\langle m, n \rangle)$

Movie Booking use-case



book($\langle u, m \rangle$)

cancelBook($\langle u, m \rangle$)

offScreen(m)

specialReserve($\langle m, n \rangle$)

increaseSpace($\langle m, n \rangle$)

Staleness bounds specification and inference

$\mathsf{querySpace}(m) := 3\ \lambda\langle rs, ms\rangle.\quad \cdots$

$\mathsf{queryReservations}(u) := 4\ \lambda\langle rs, ms\rangle.\quad \cdots$

$\mathsf{querySpaces}(u) := 6\ \lambda\langle rs, ms\rangle.\quad \cdots$

Staleness bounds specification and inference

$$\text{querySpace}(m) := 3\ \lambda\langle rs, ms\rangle. \quad \cdots$$

$$\text{queryReservations}(u) := 4\ \lambda\langle rs, ms\rangle. \quad \cdots$$

$$\text{querySpaces}(u) := 6\ \lambda\langle rs, ms\rangle. \quad \cdots$$

Staleness bounds specification and inference

$\mathsf{querySpace}(m) := 3\ \lambda\langle rs, ms\rangle. \quad \cdots$

$\boxed{\mathsf{queryReservations}(u)} := \boxed{4}\ \lambda\langle rs, ms\rangle. \quad \cdots$

$\boxed{\mathsf{querySpaces}(u)} := \boxed{6}\ \lambda\langle rs, ms\rangle. \quad \cdots$

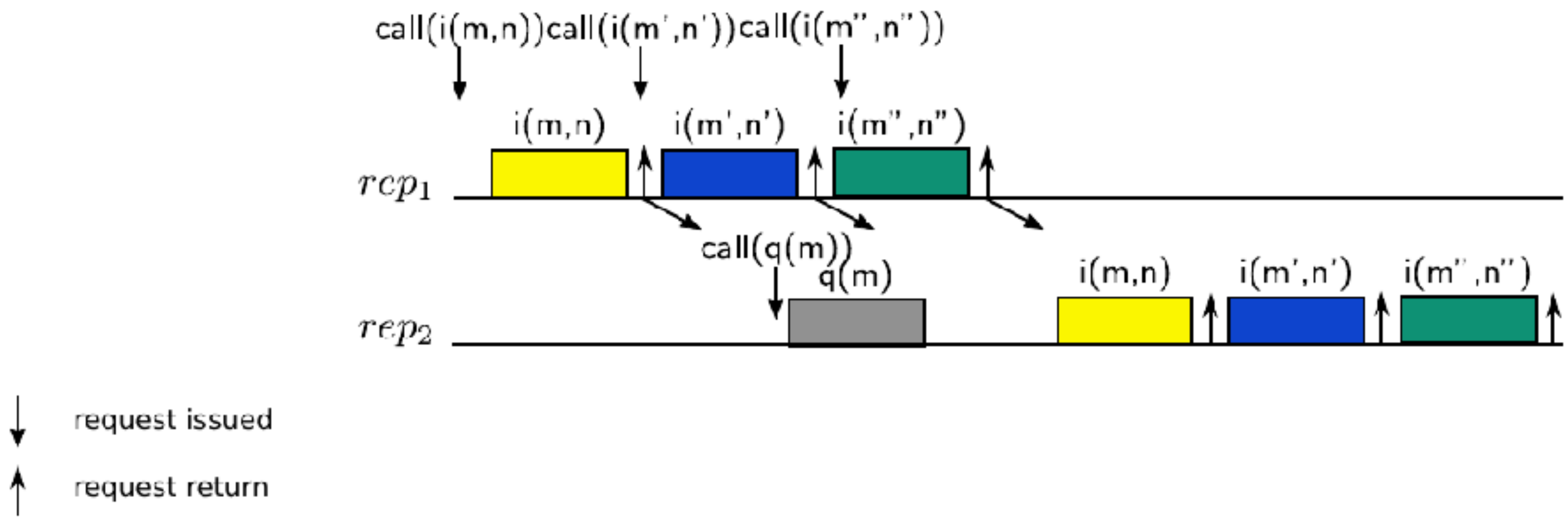
Staleness bounds specification and inference

$$\mathsf{querySpace}(m) := 3\ \lambda\langle rs, ms\rangle. \quad \cdots$$

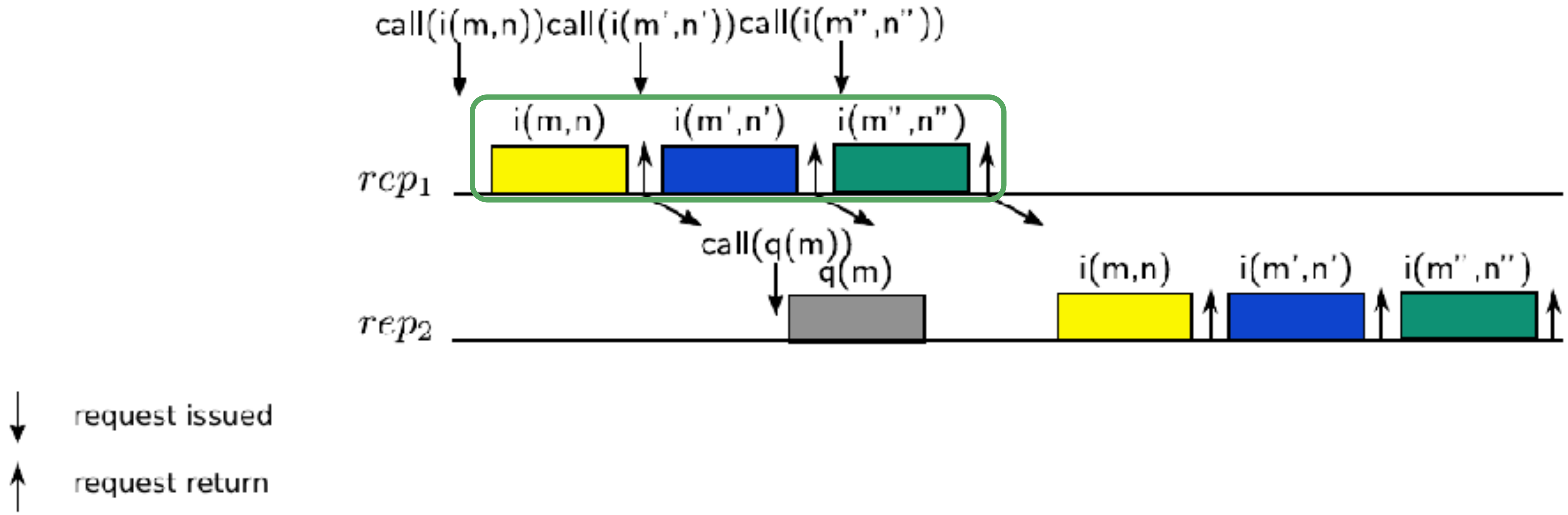
$$\mathsf{queryReservations}(u) := 4\ \lambda\langle rs, ms\rangle. \quad \cdots$$

$$\mathsf{querySpaces}(u) := 6\ \lambda\langle rs, ms\rangle. \quad \cdots$$

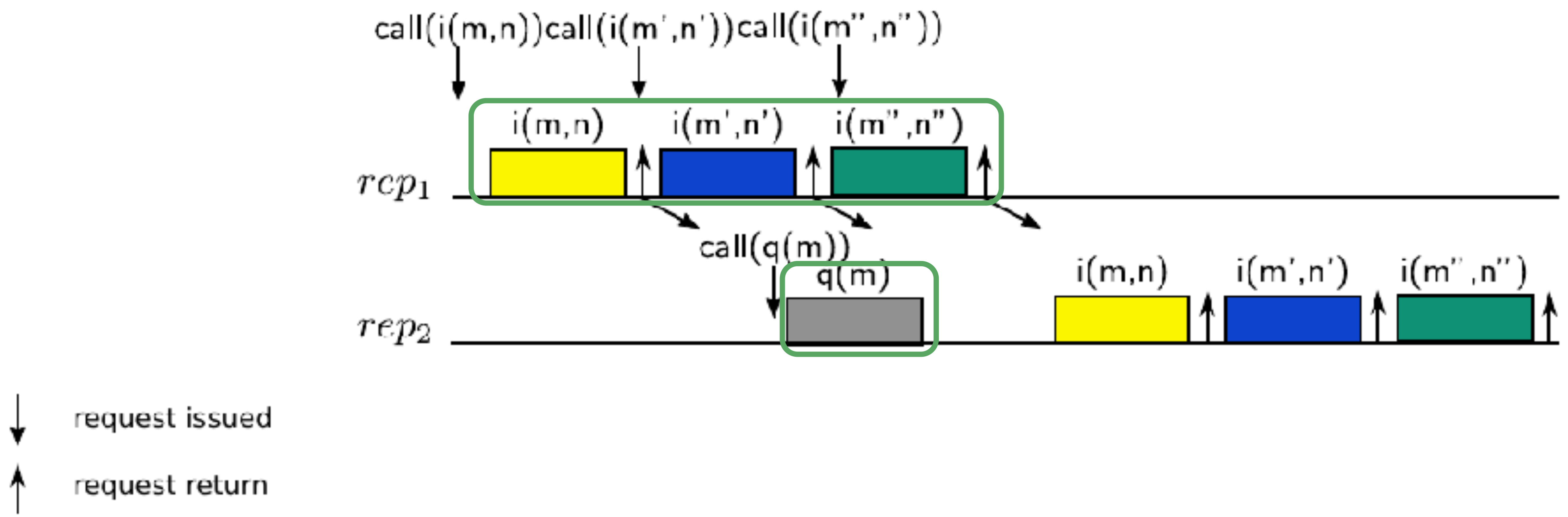
Staleness Bound and Recency



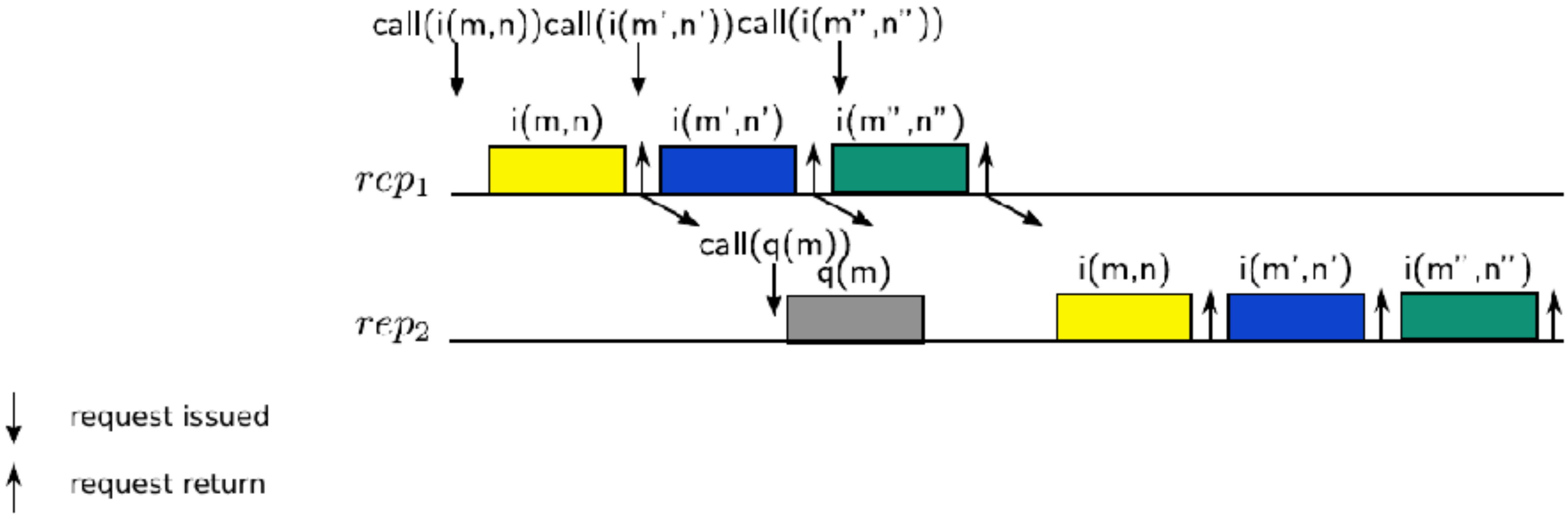
Staleness Bound and Recency



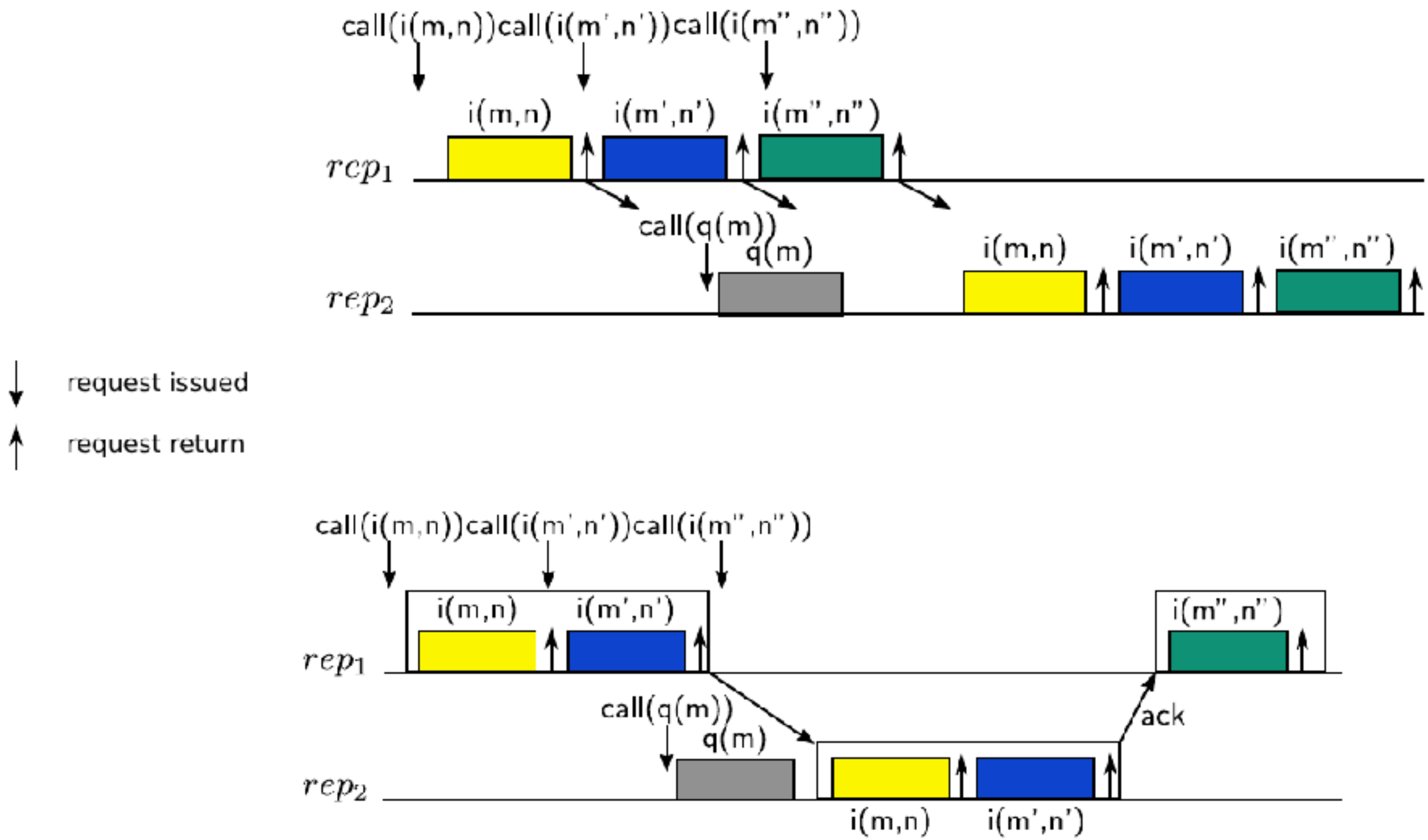
Staleness Bound and Recency



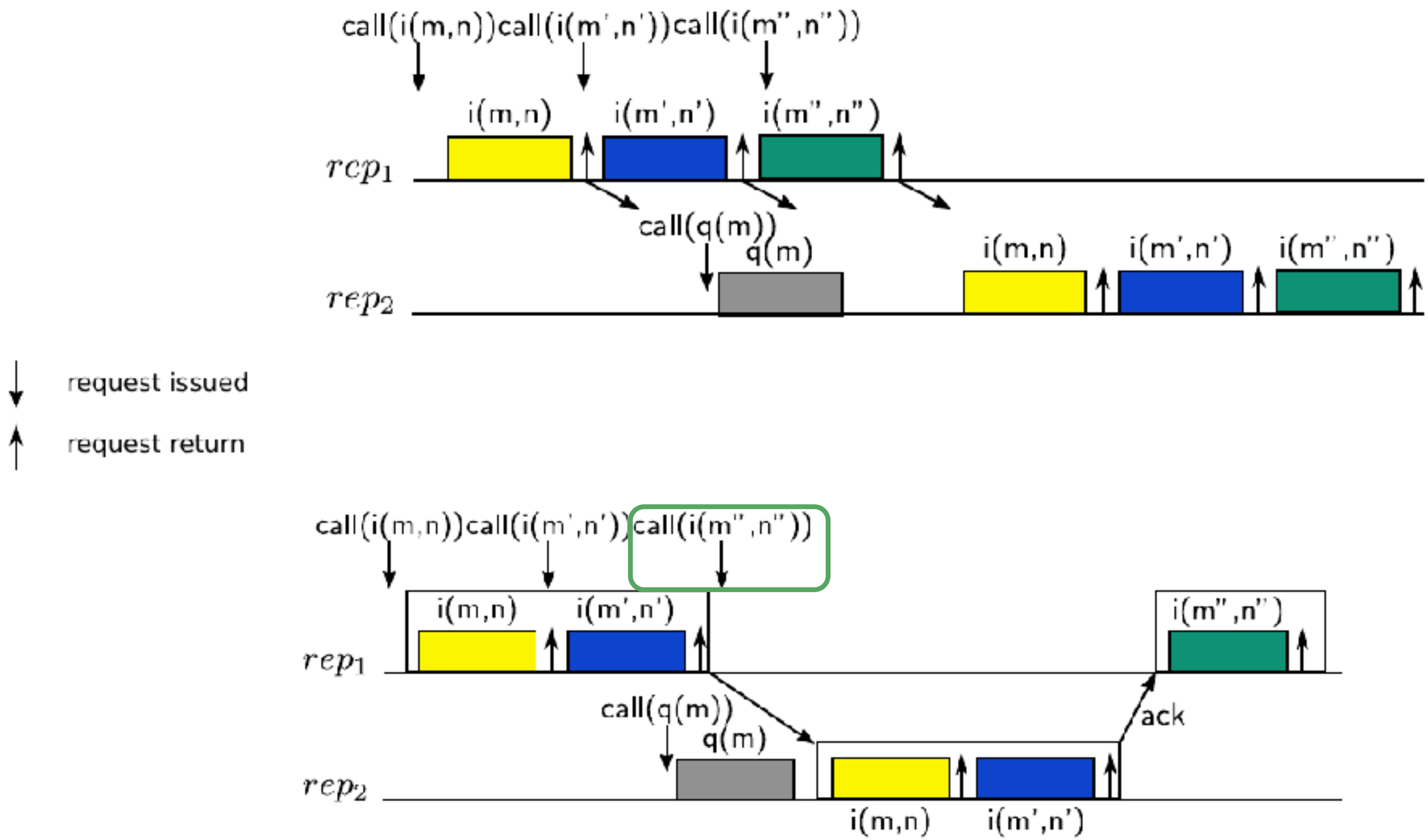
Staleness Bound and Recency



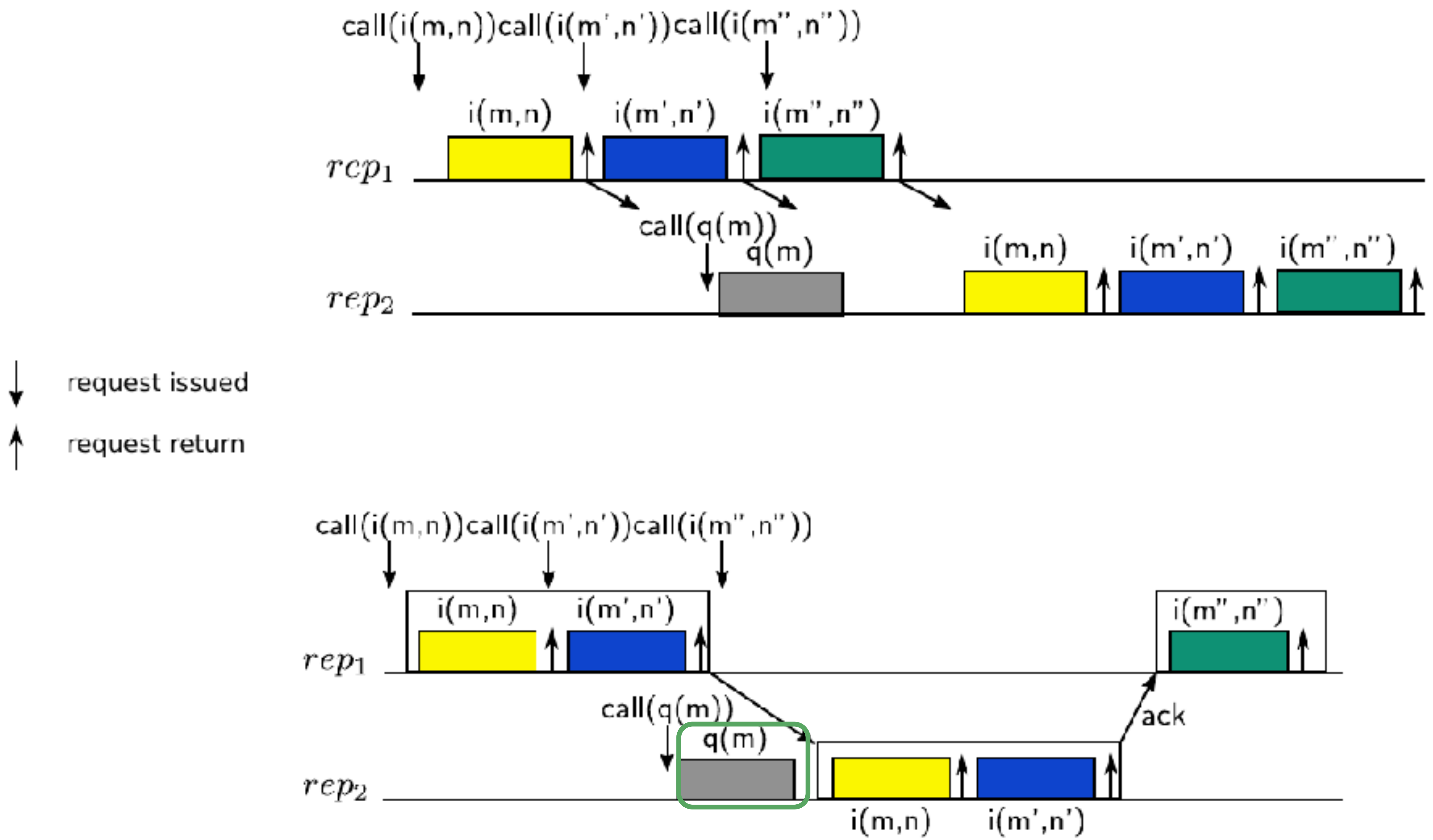
Staleness Bound and Recency



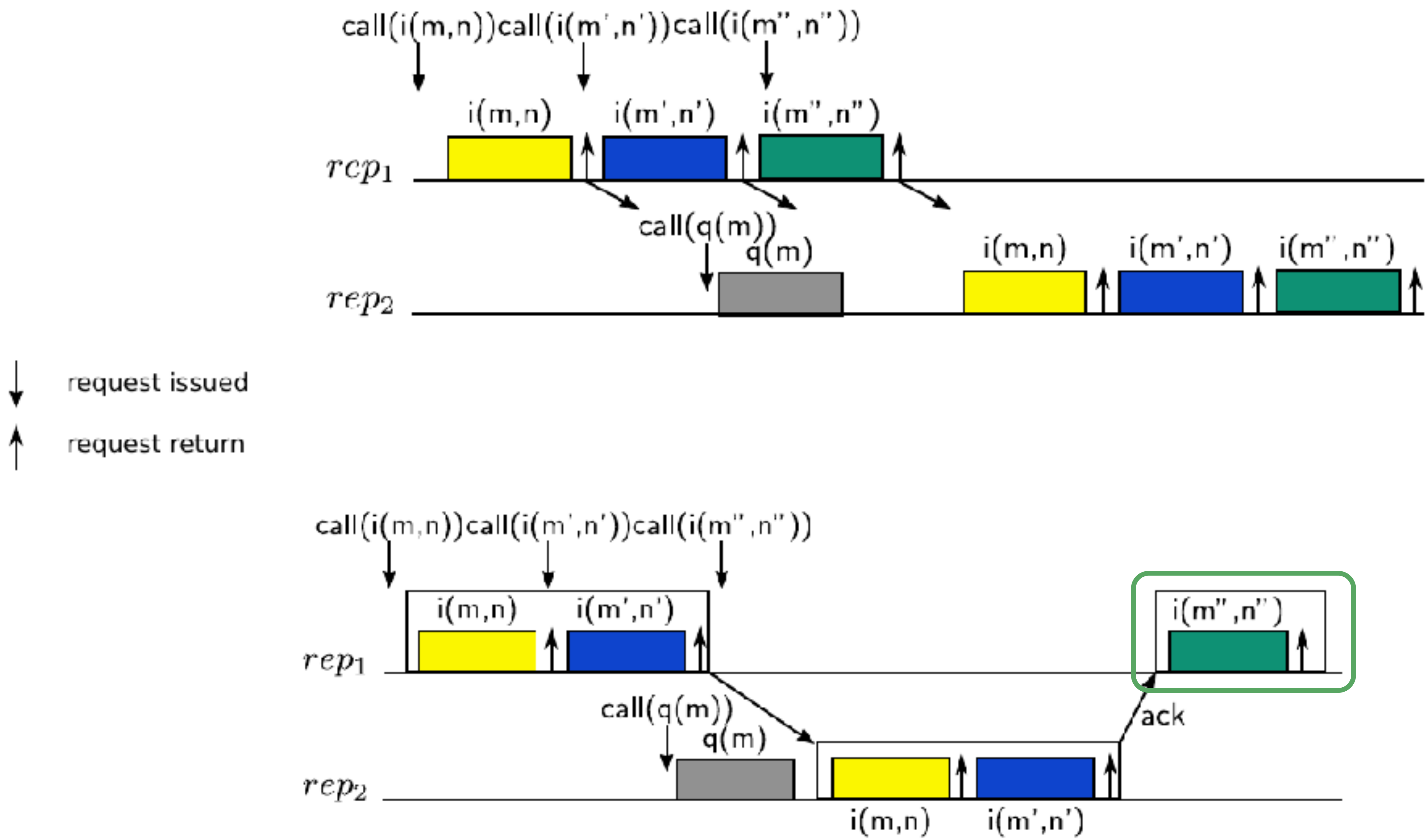
Staleness Bound and Recency



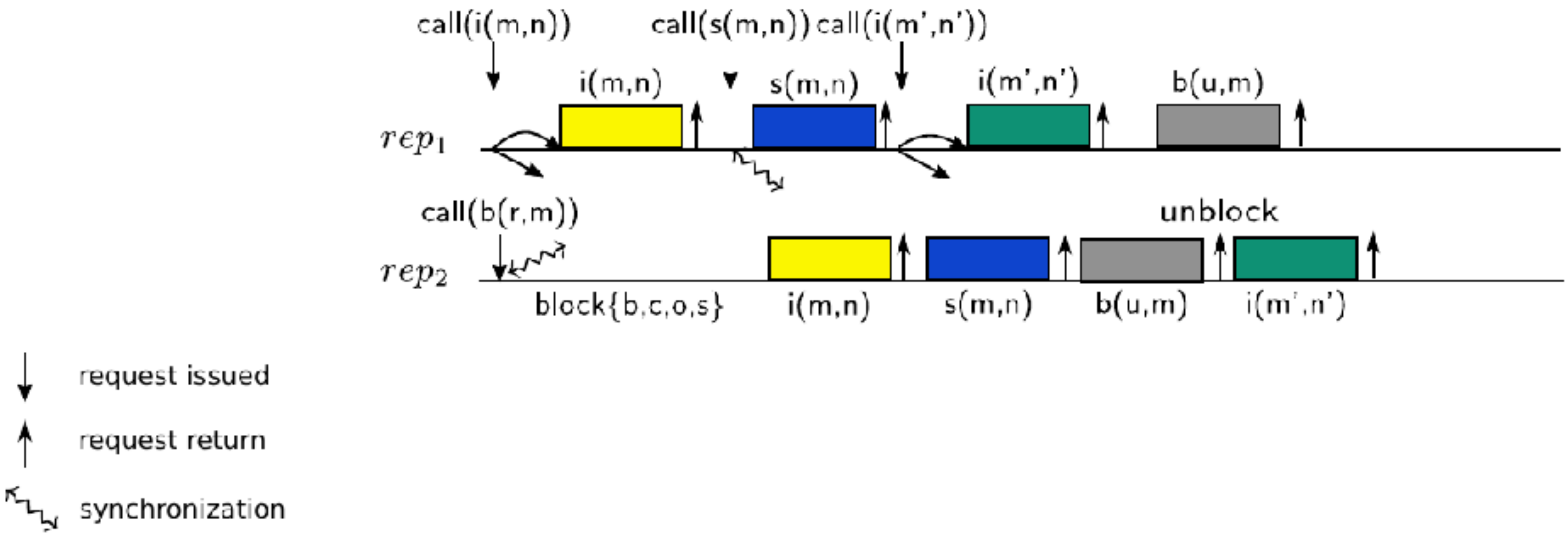
Staleness Bound and Recency



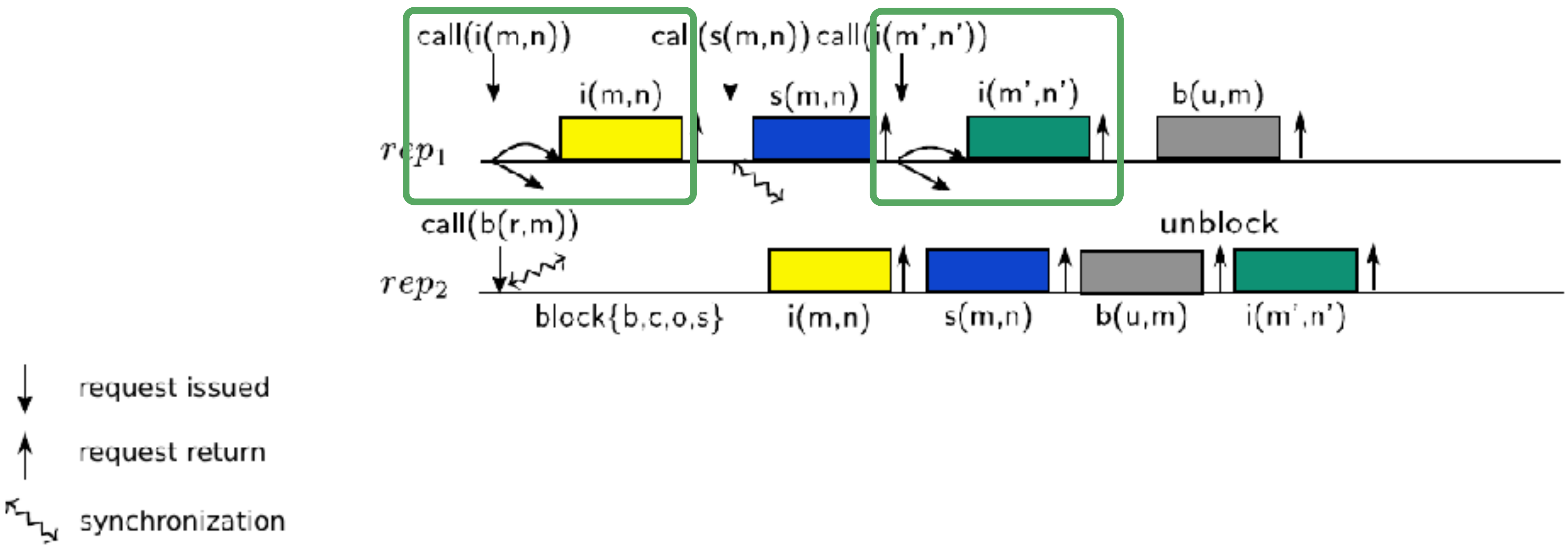
Staleness Bound and Recency



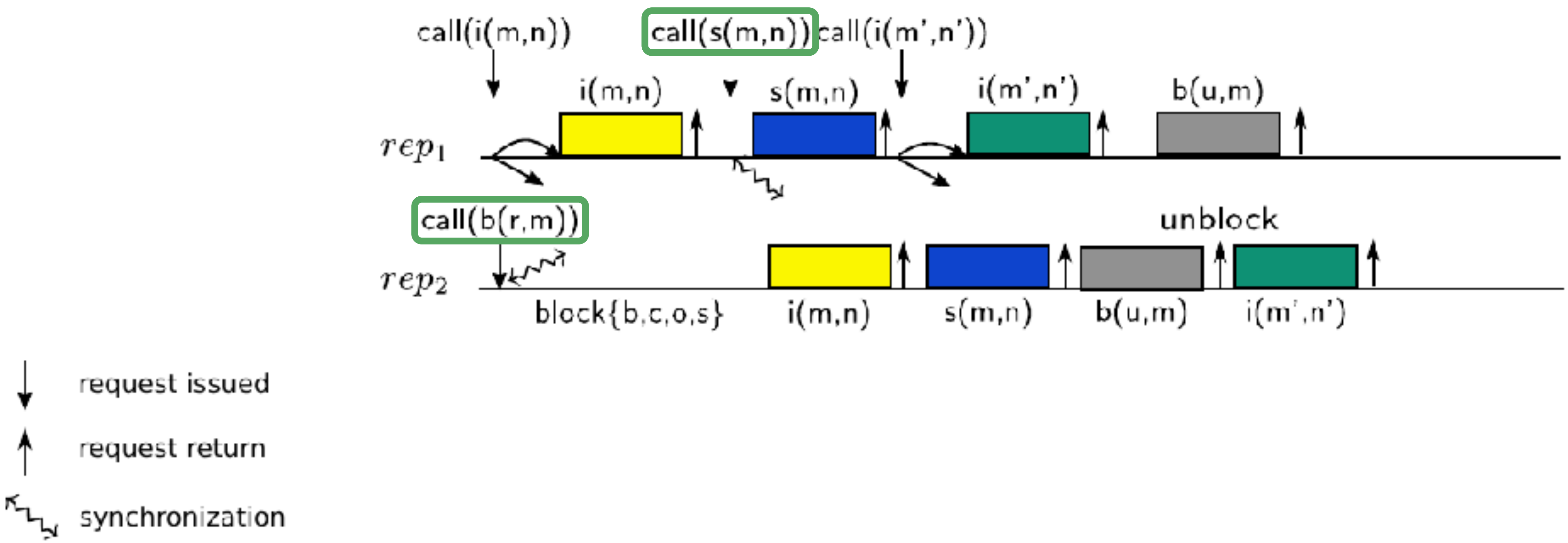
Communication and Synchronization Avoidance



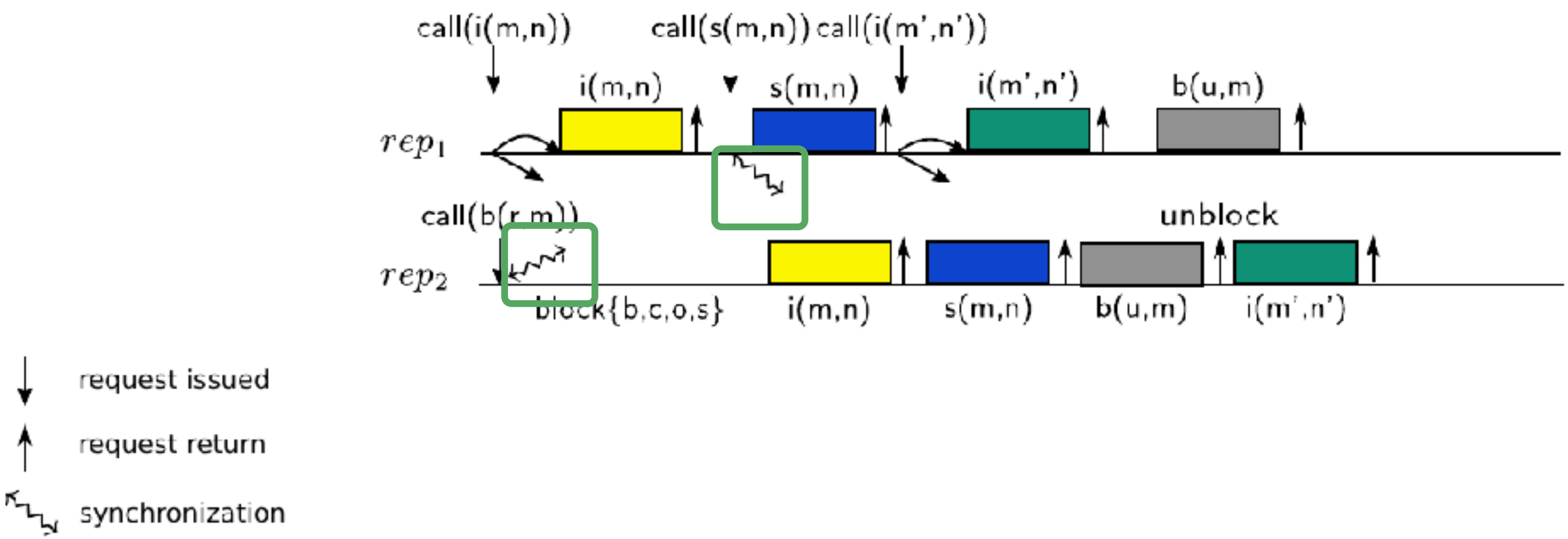
Communication and Synchronization Avoidance

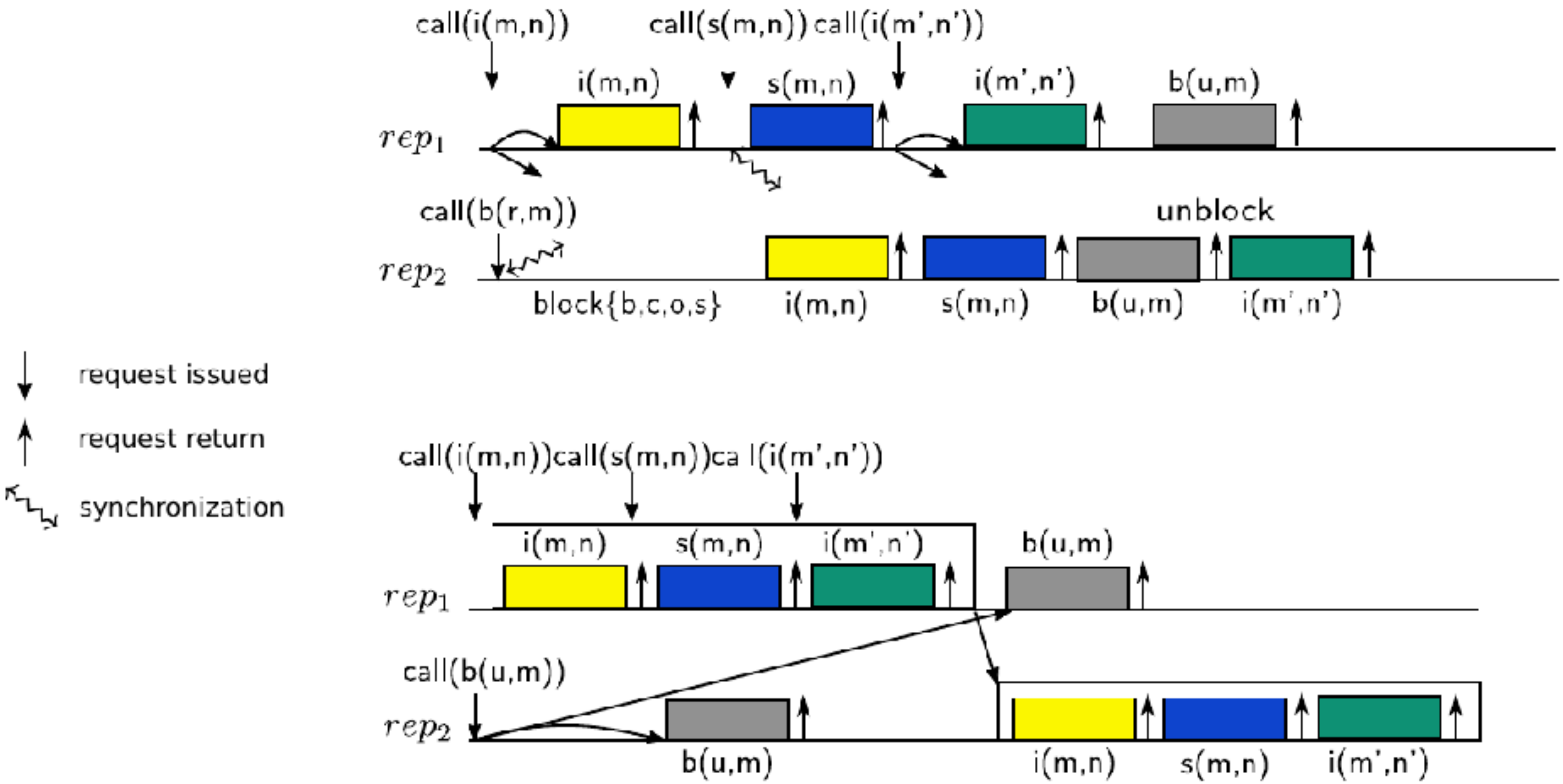


Communication and Synchronization Avoidance

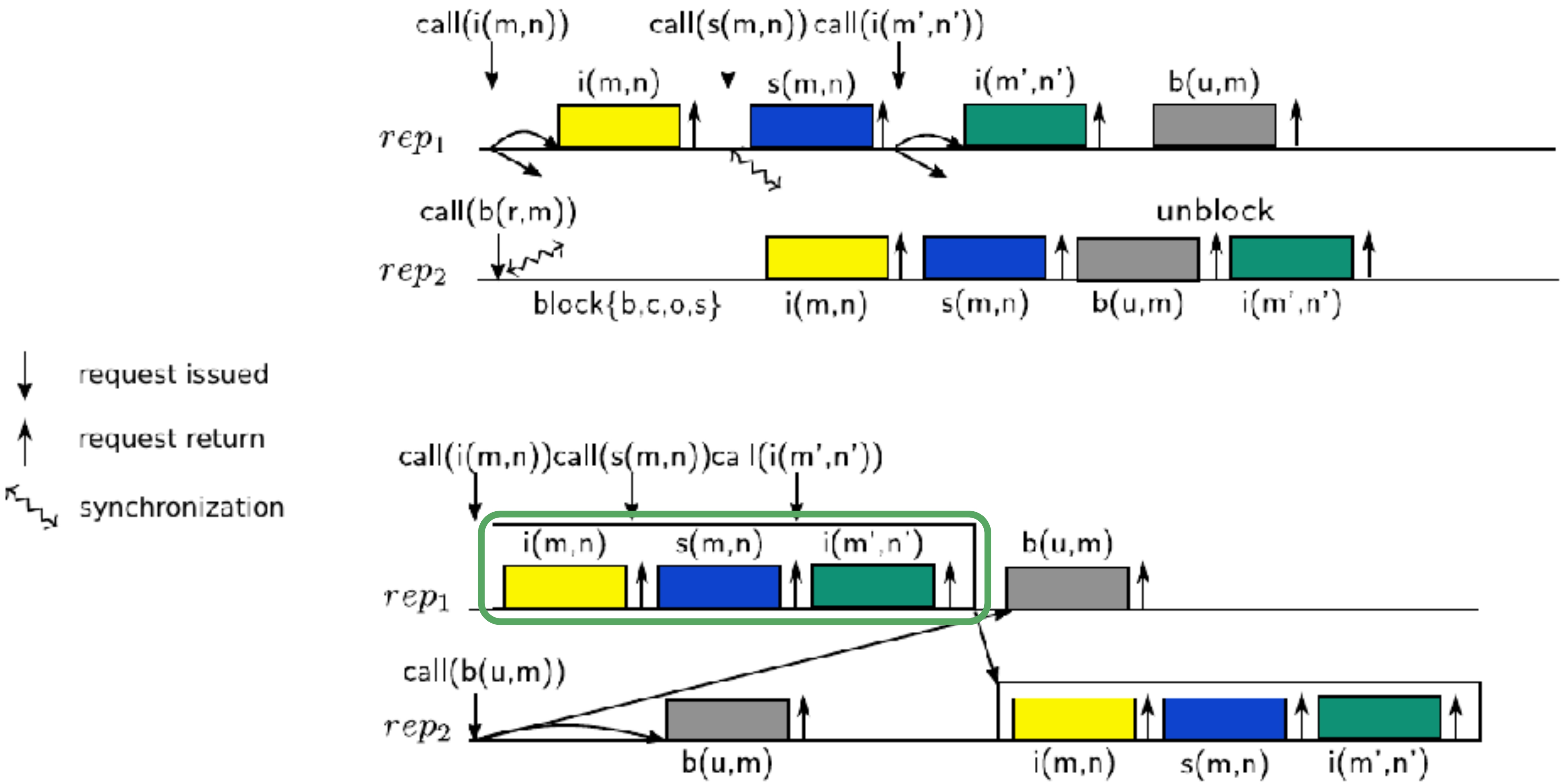


Communication and Synchronization Avoidance

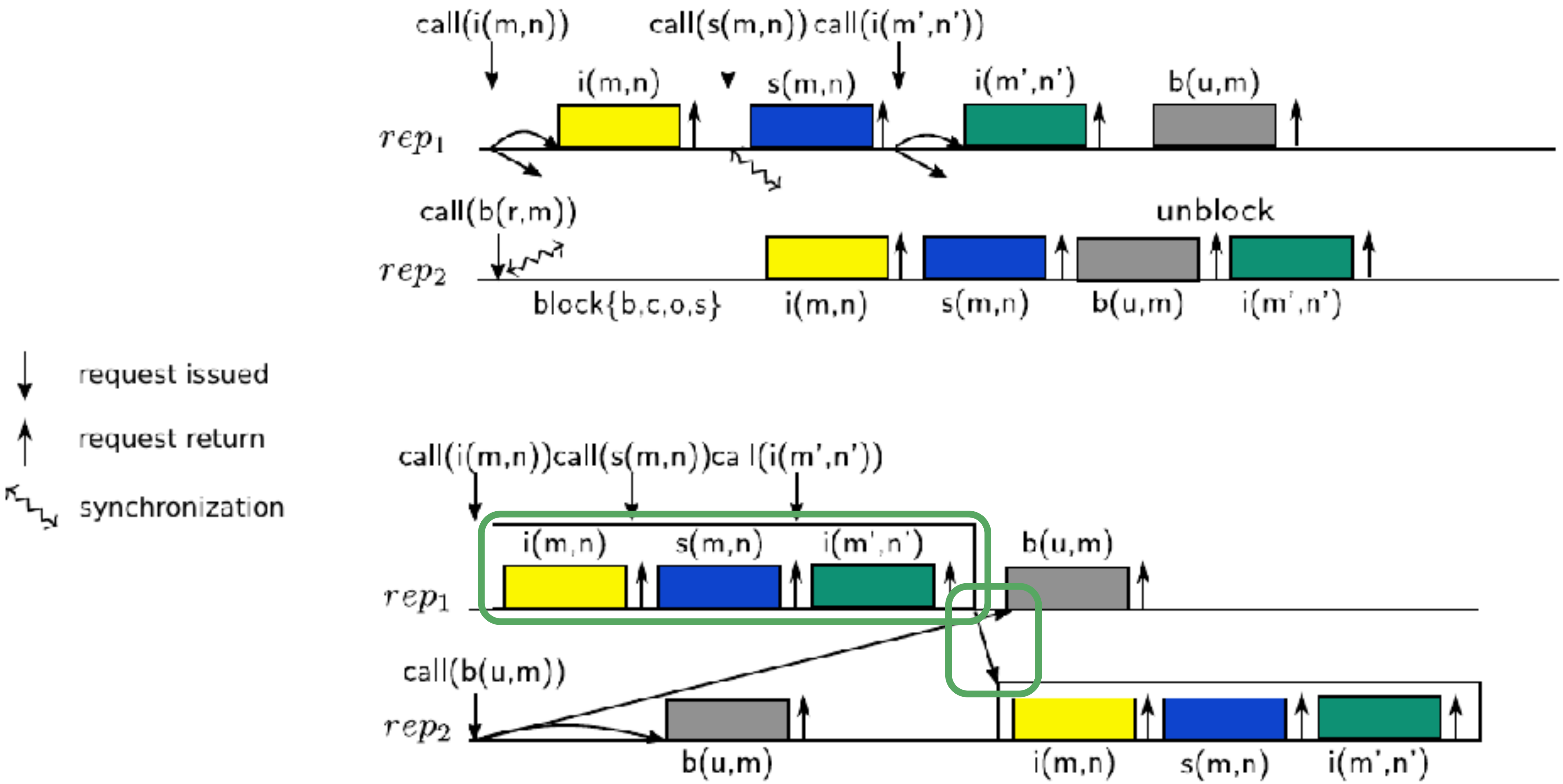



call(i(m,n))
call(s(m,n)) call(i(m',n'))
i(m,n)
s(m,n)
i(m',n')
b(u,m)
rep_1
call(b(r,m))
unblock
rep_2
block{b,c,o,s}
i(m,n)
s(m,n)
b(u,m)
i(m',n')
request issued
request return
synchronization
call(i(m,n))call(s(m,n))ca l(i(m',n'))
i(m,n)
s(m,n)
i(m',n')
b(u,m)
rep_1
call(b(u,m))
rep_2
b(u,m)
i(m,n)
s(m,n)
i(m',n')

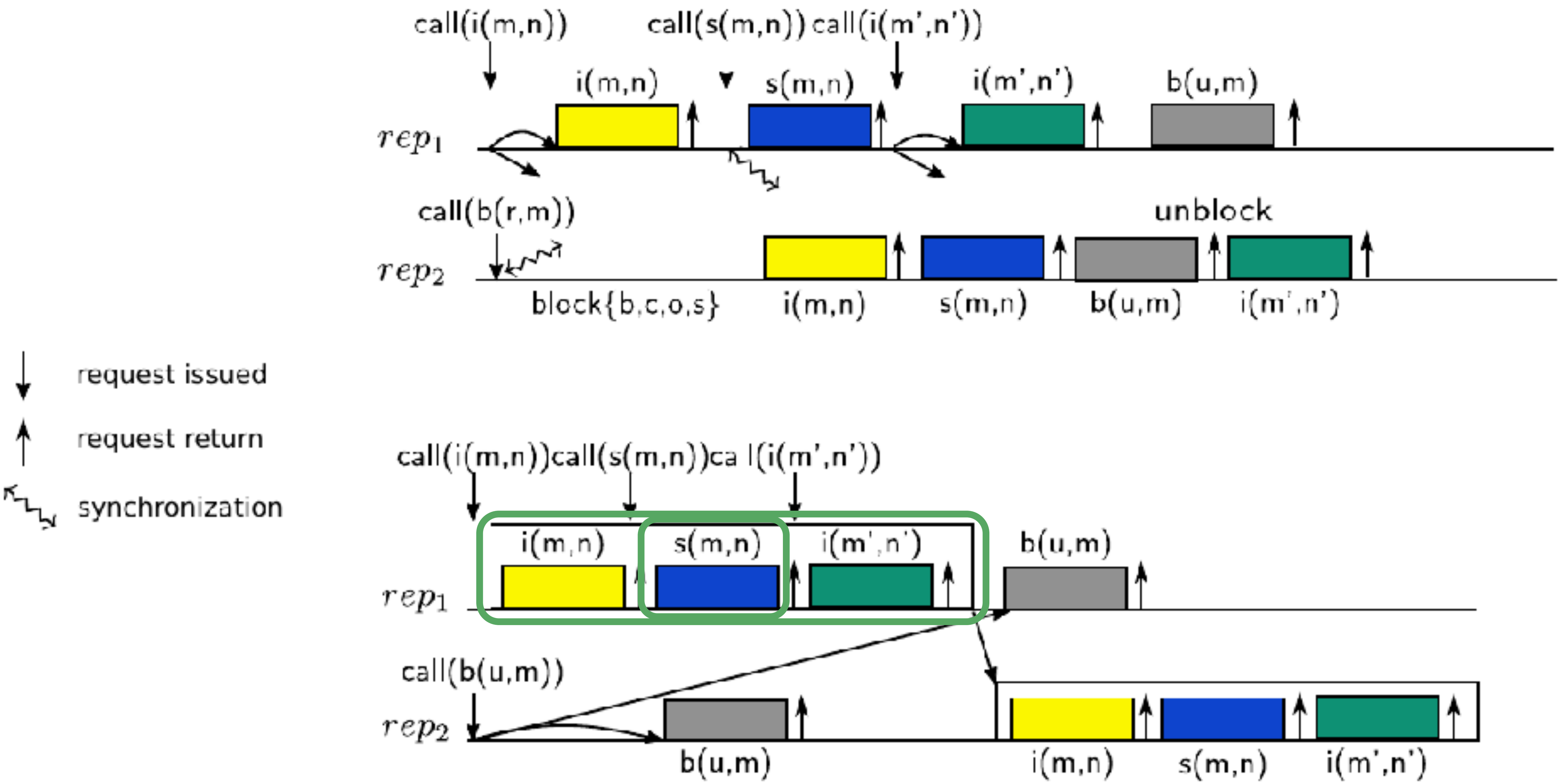
Communication and Synchronization Avoidance



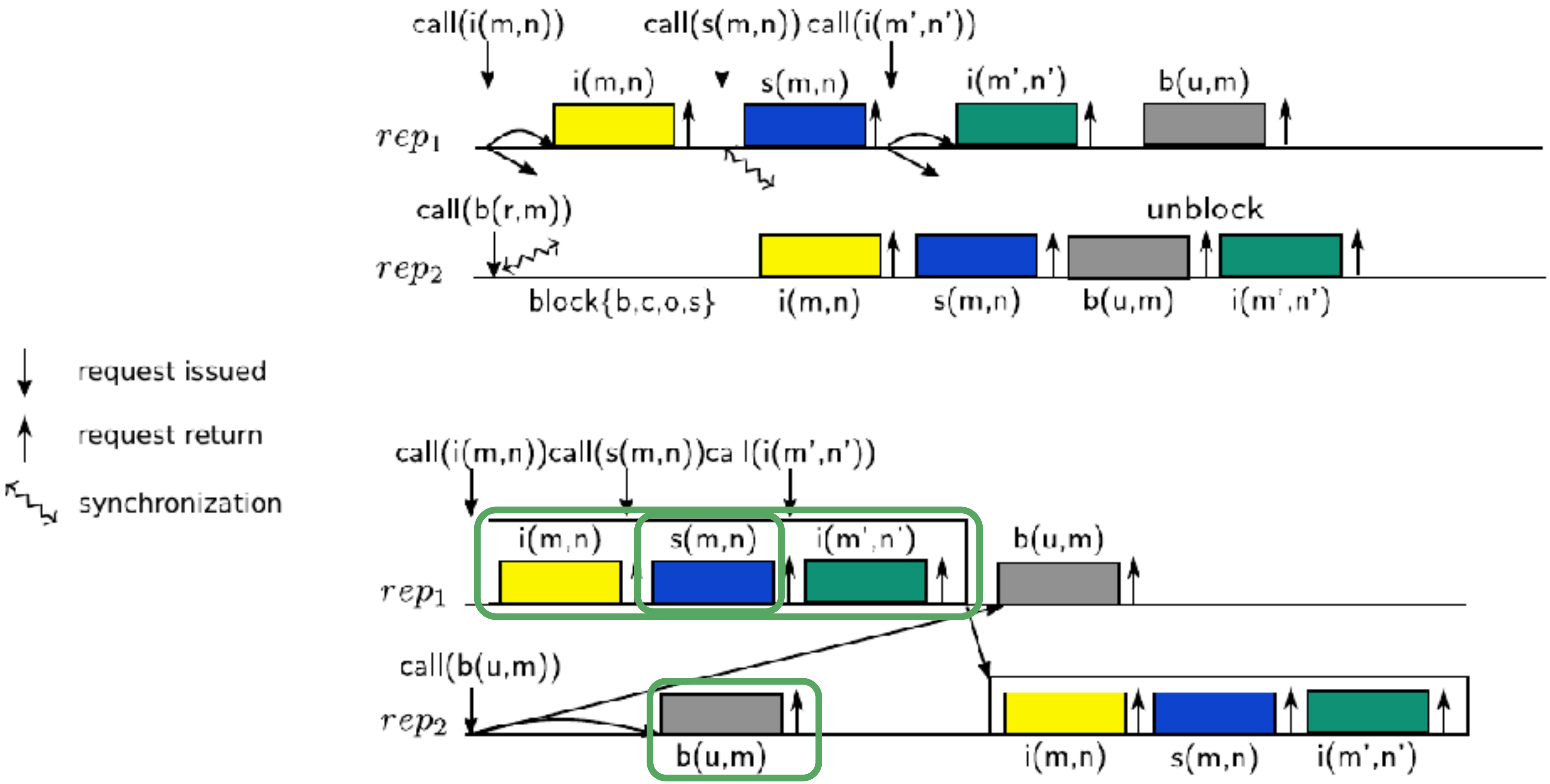
Communication and Synchronization Avoidance



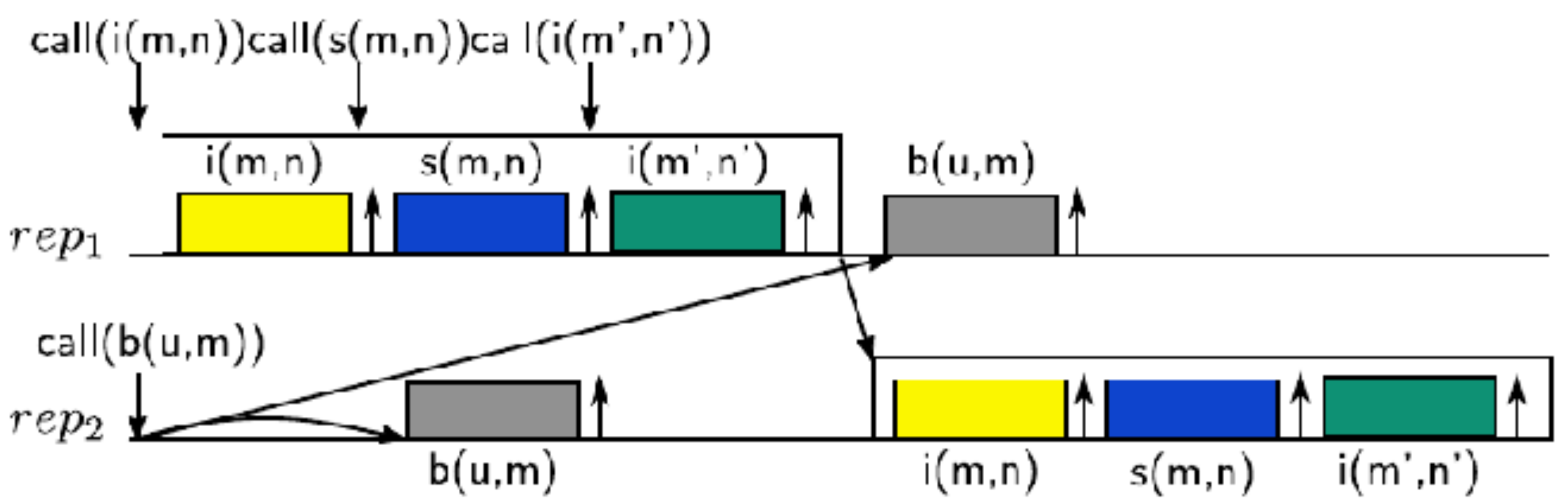
Communication and Synchronization Avoidance



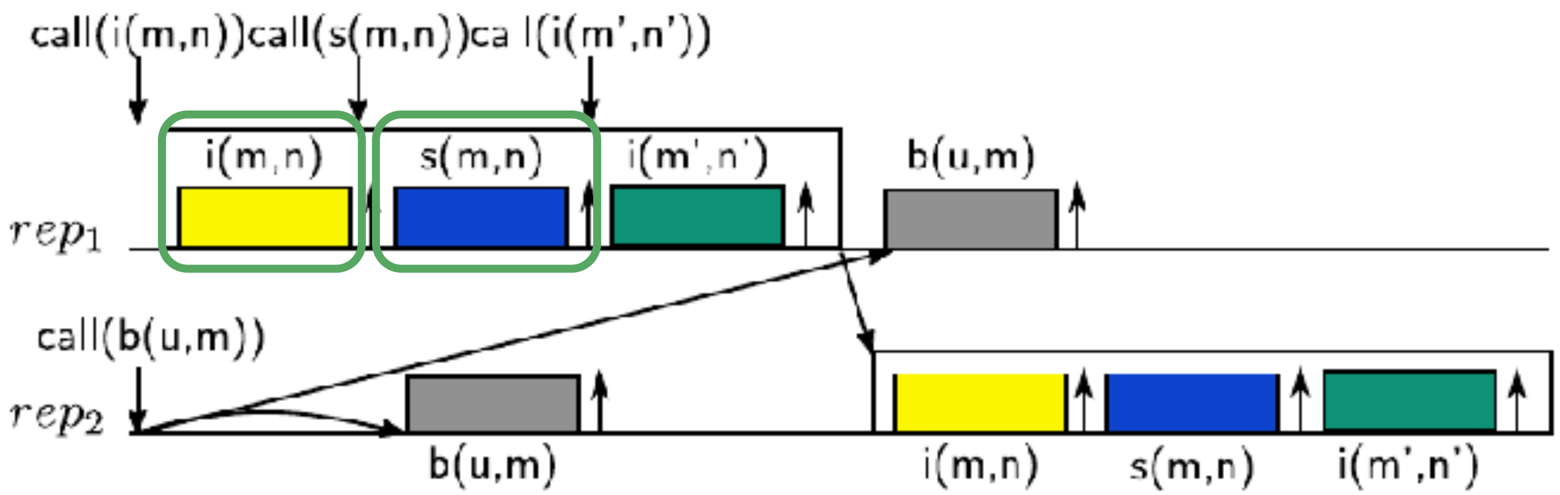
Communication and Synchronization Avoidance



Conditions for Buffering

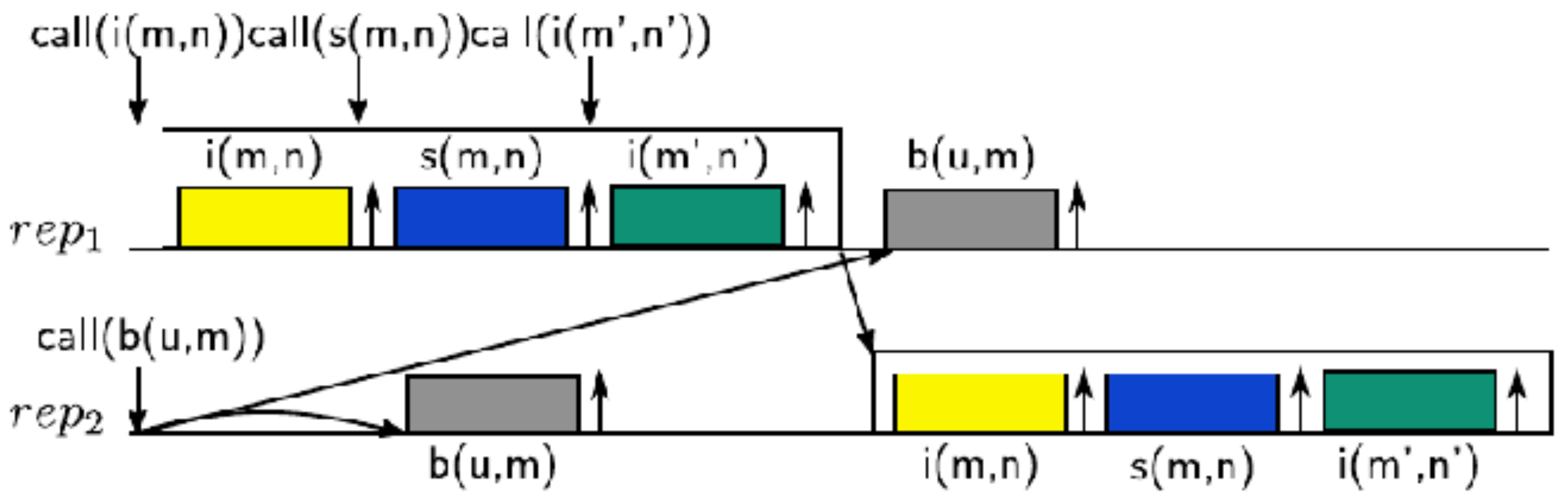


Conditions for Buffering

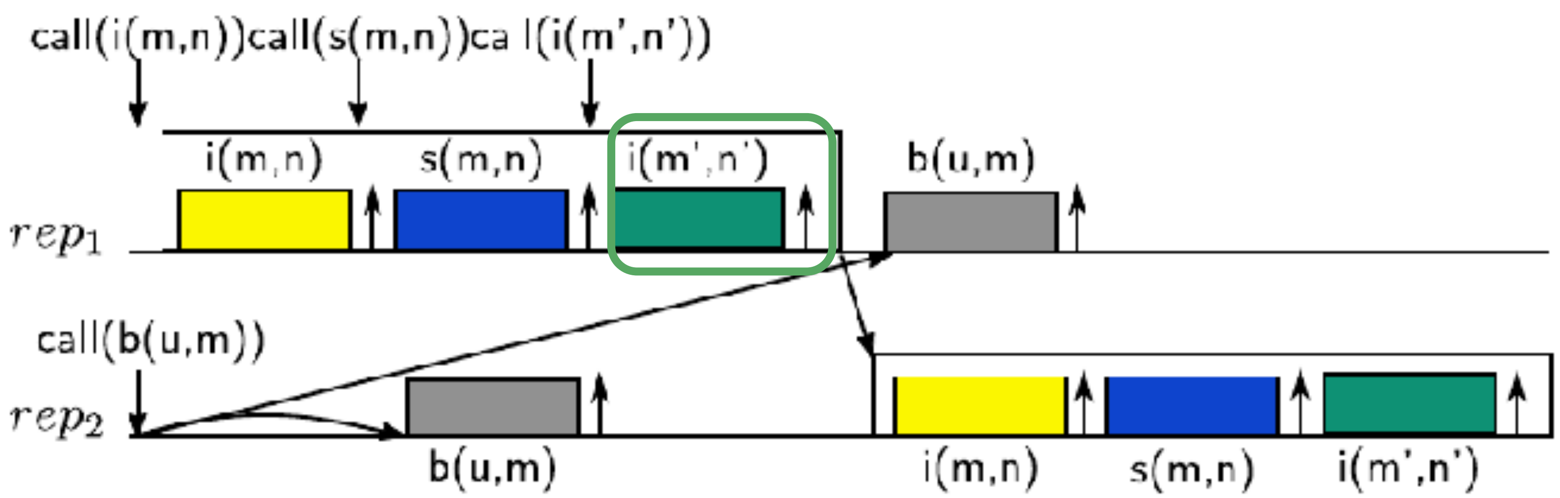


1. All-state-commutativity

Conditions for Buffering

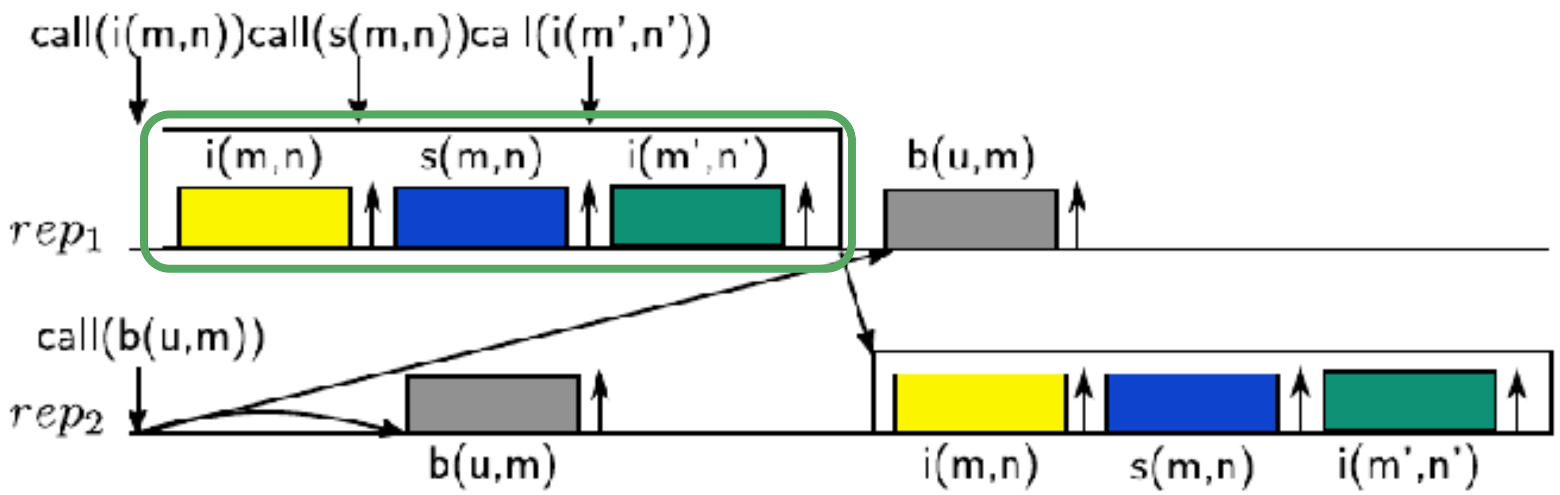


Conditions for Buffering



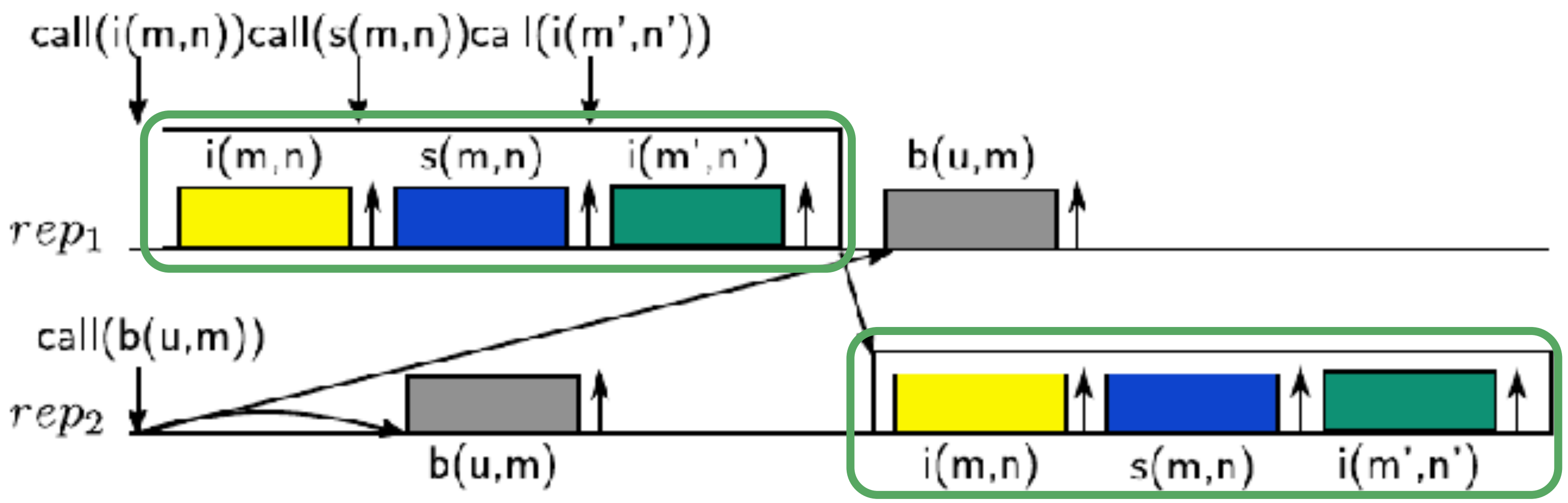
2. In-bound

Conditions for Buffering



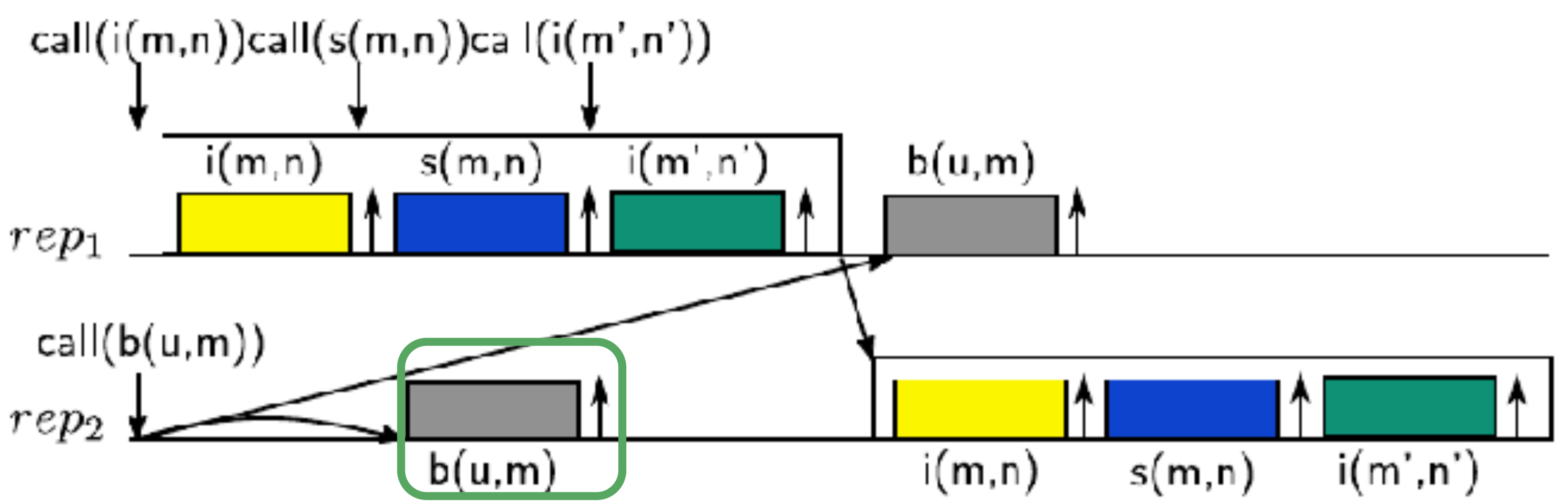
3. Invariant-sufficiency

Conditions for Buffering



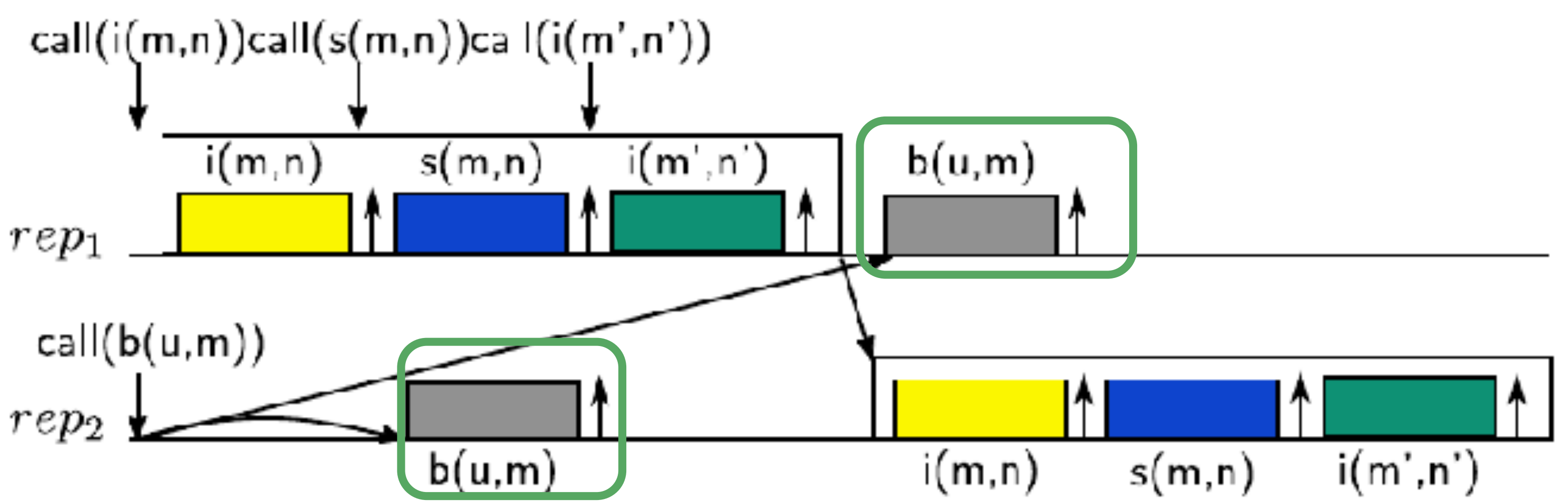
3. Invariant-sufficiency

Conditions for Buffering



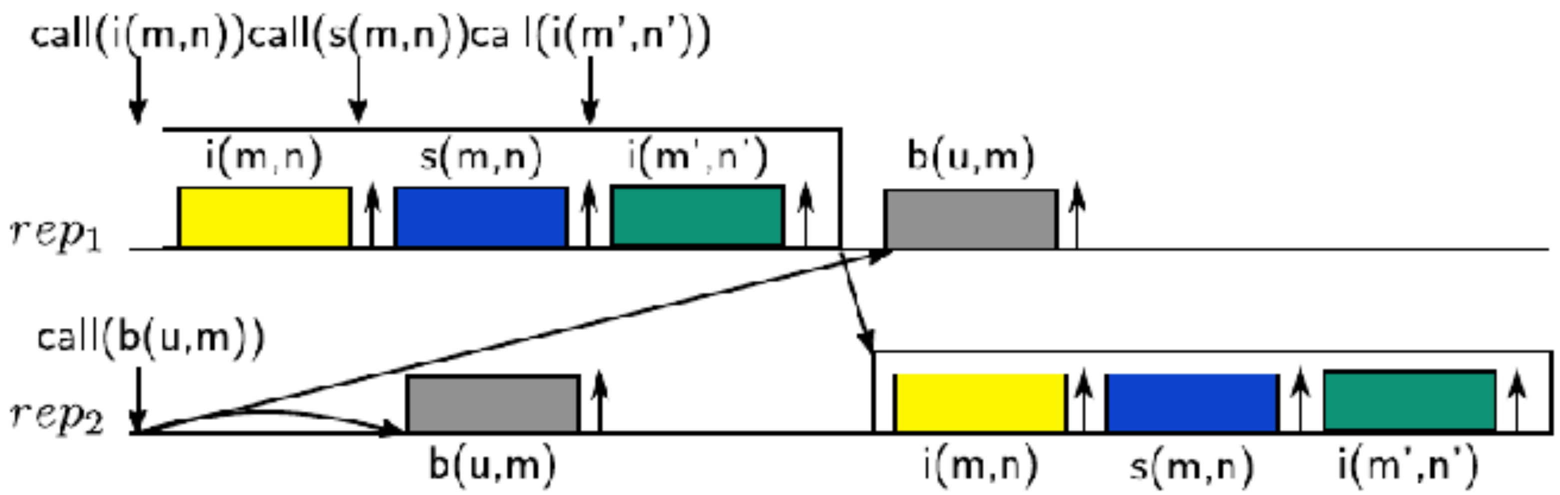
4. Let-P-R-commutativity

Conditions for Buffering



4. Let-P-R-commutativity

Conditions for Buffering

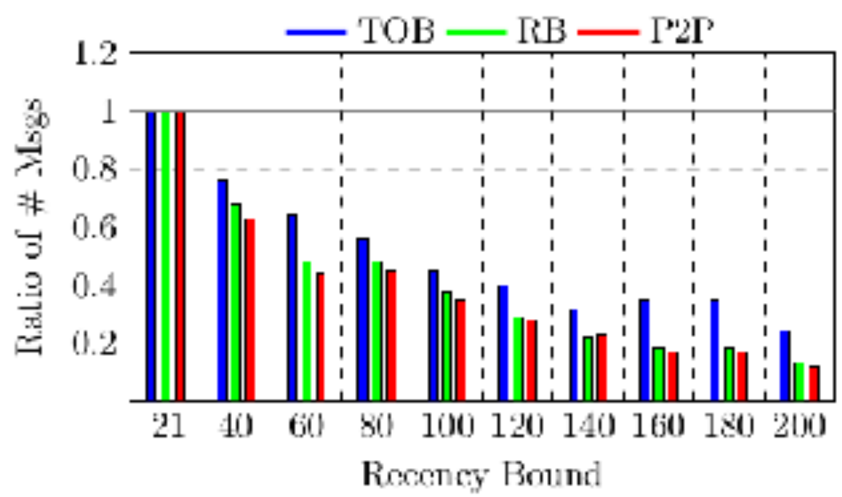


Experimental Results

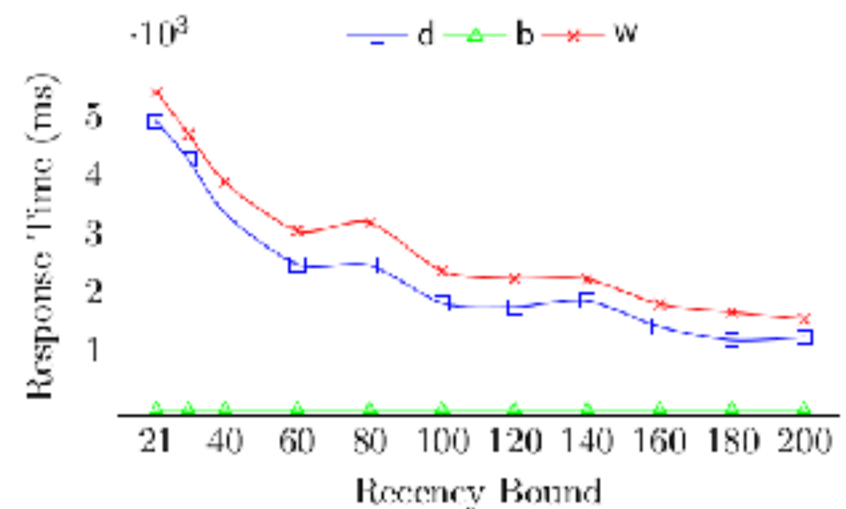
As the recency bound increases,
the coordination overhead and response time decrease.

Bank account

(a)

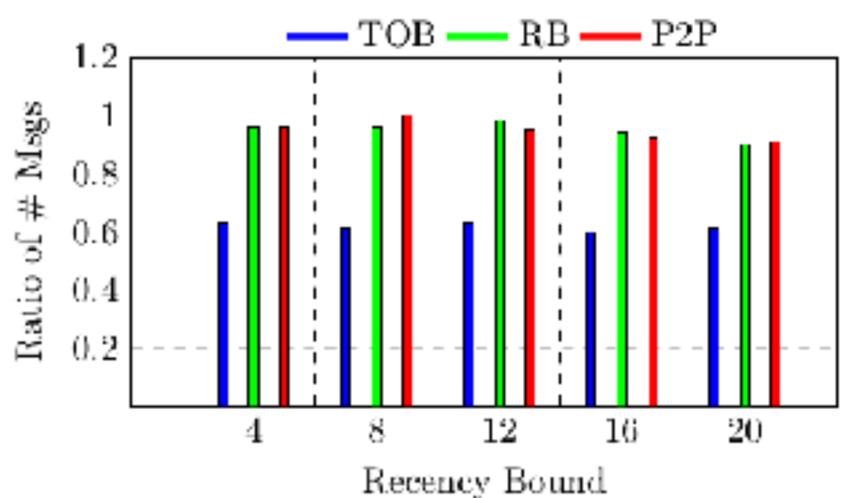


(b)

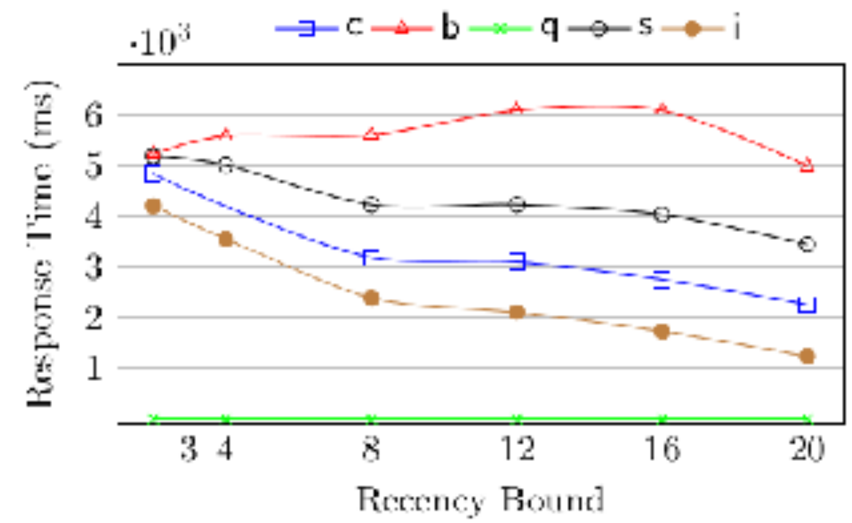


Movie booking

(c)



(d)



Coordination as Commutativity

- Synthesis of replicated objects that
  preserve integrity, convergence and recency, and
  minimize coordination

- Coordination conditions sufficient for these properties
  that are captured as commutativity conditions

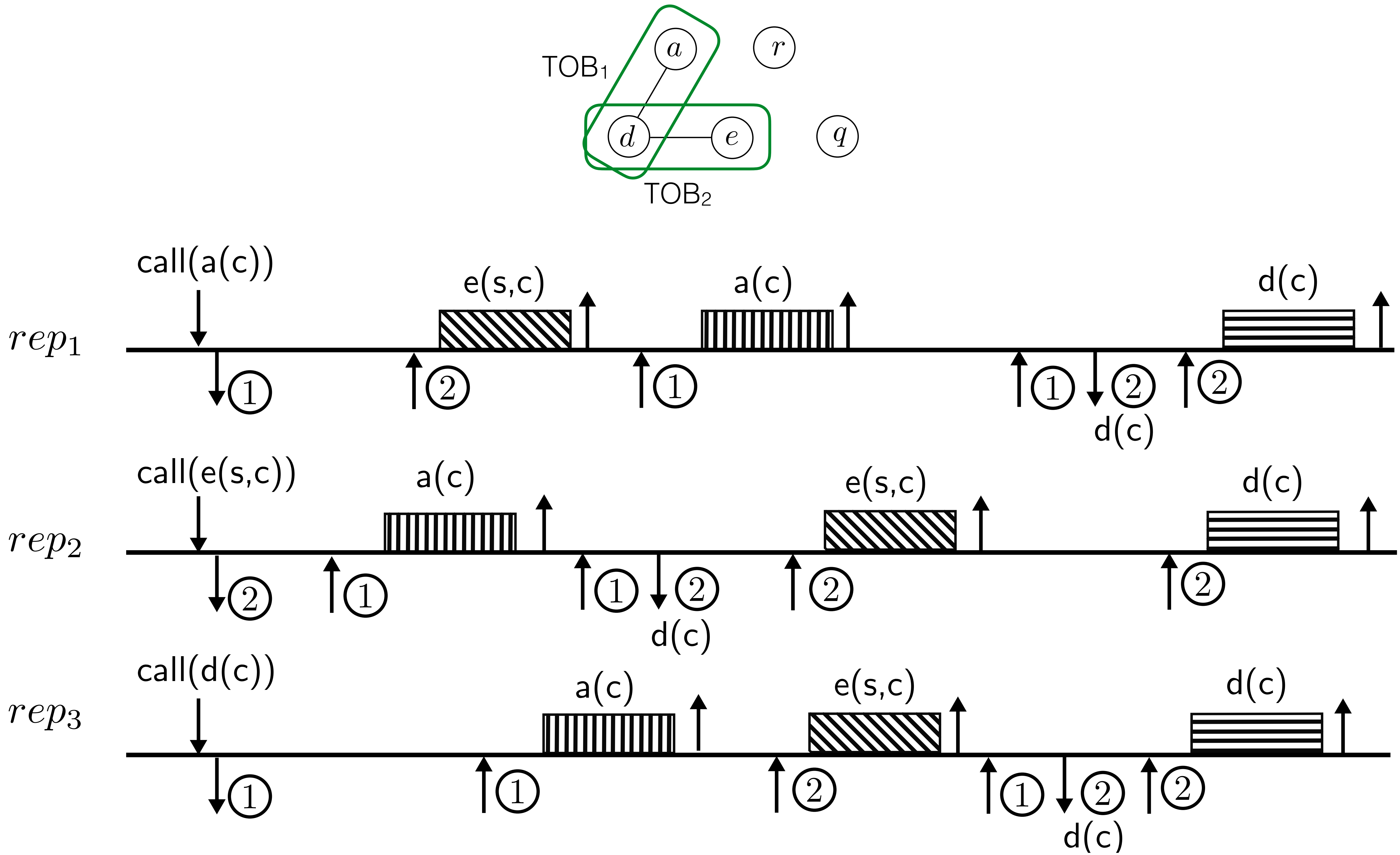
- Reduced coordination minimization to classical graph optimization

- Coordination protocols that preserve these conditions

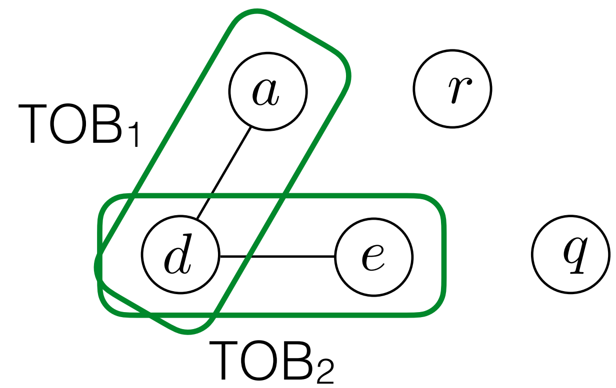
Commutativity Reasoning for Automated Distributed Coordination

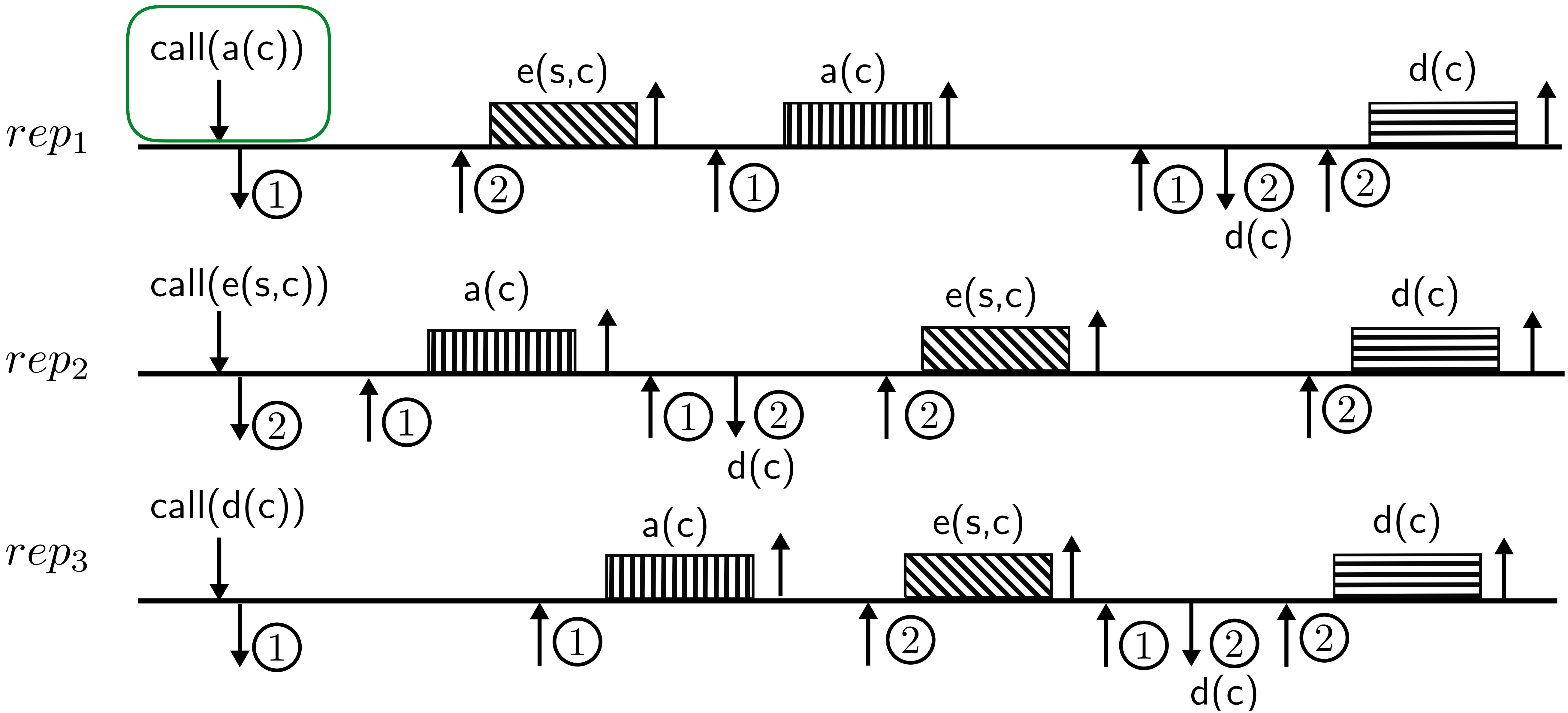
Mohsen Lesani
University of California, Riverside

Non-blocking Protocol

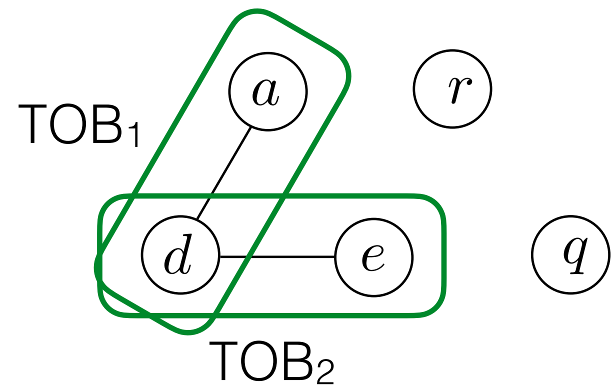


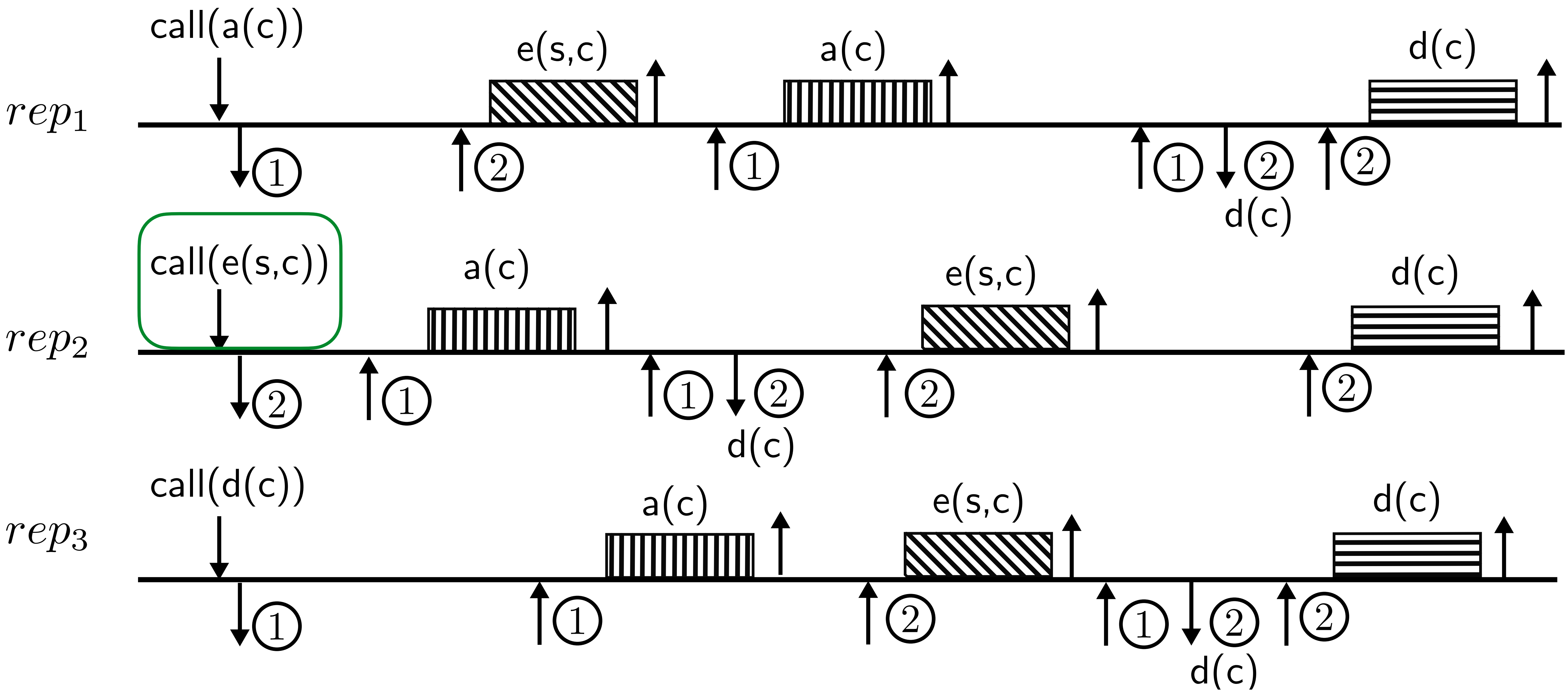
Non-blocking Protocol



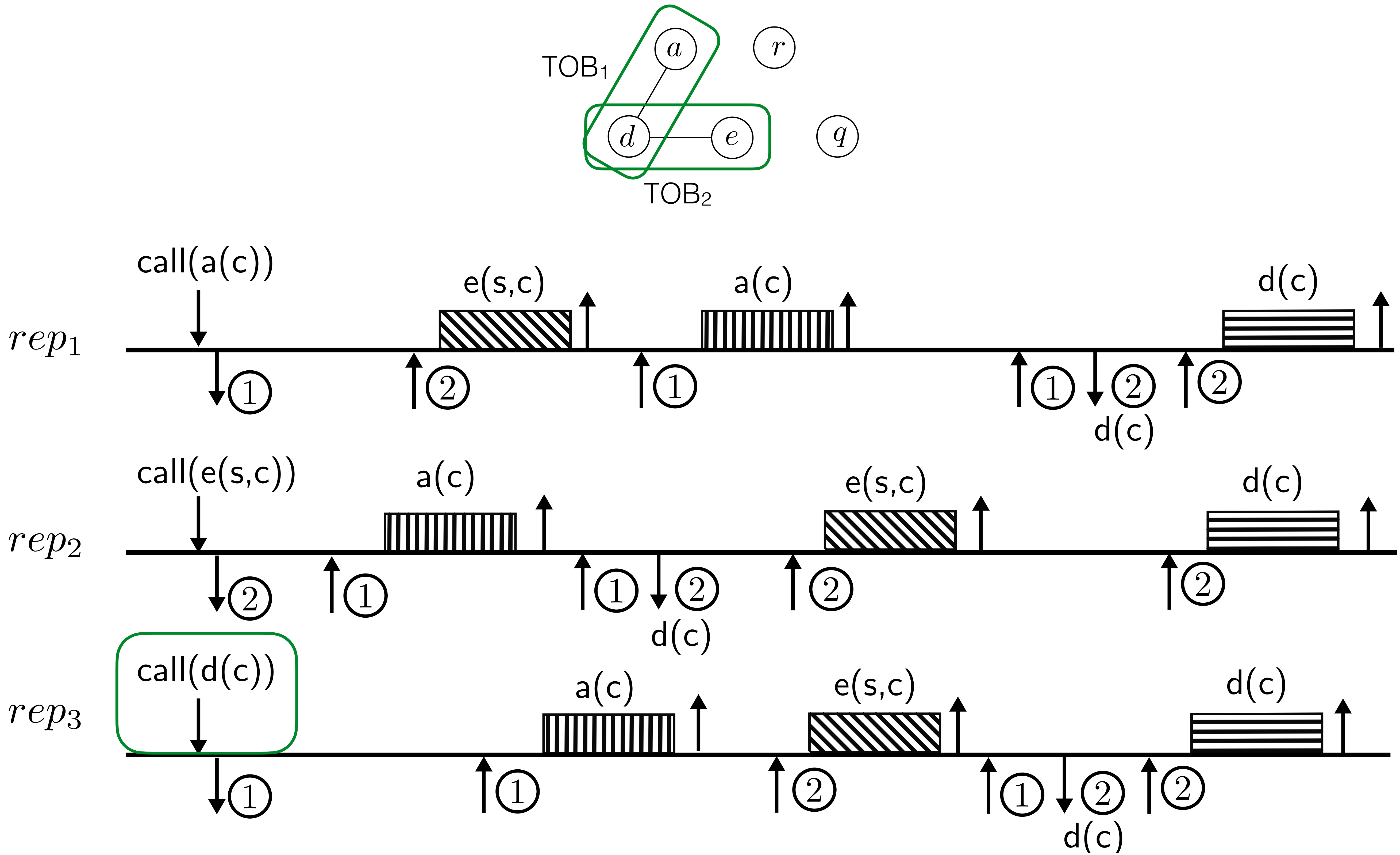


Non-blocking Protocol

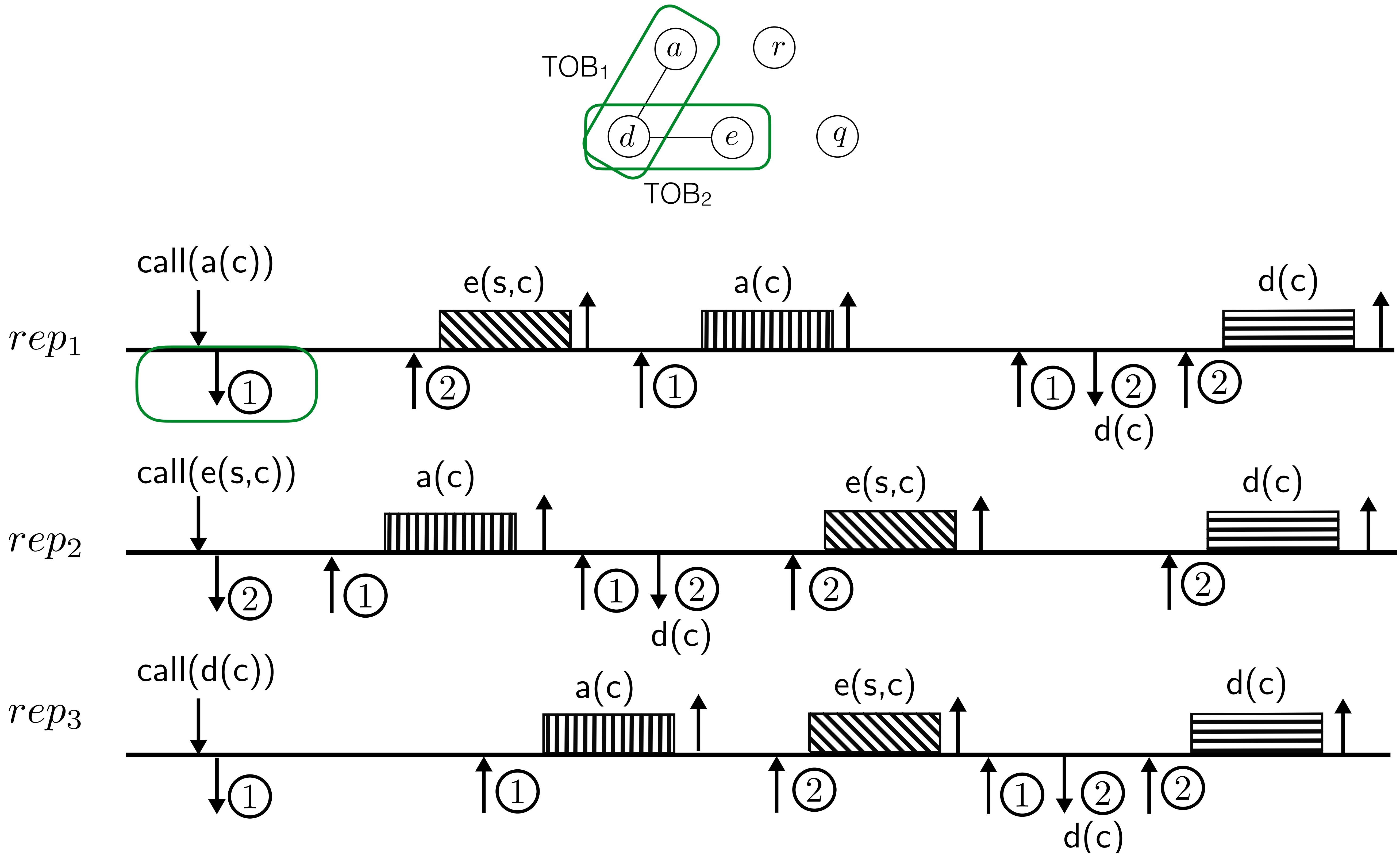




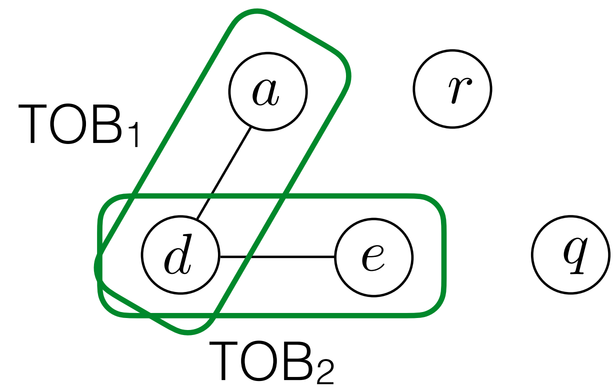
Non-blocking Protocol

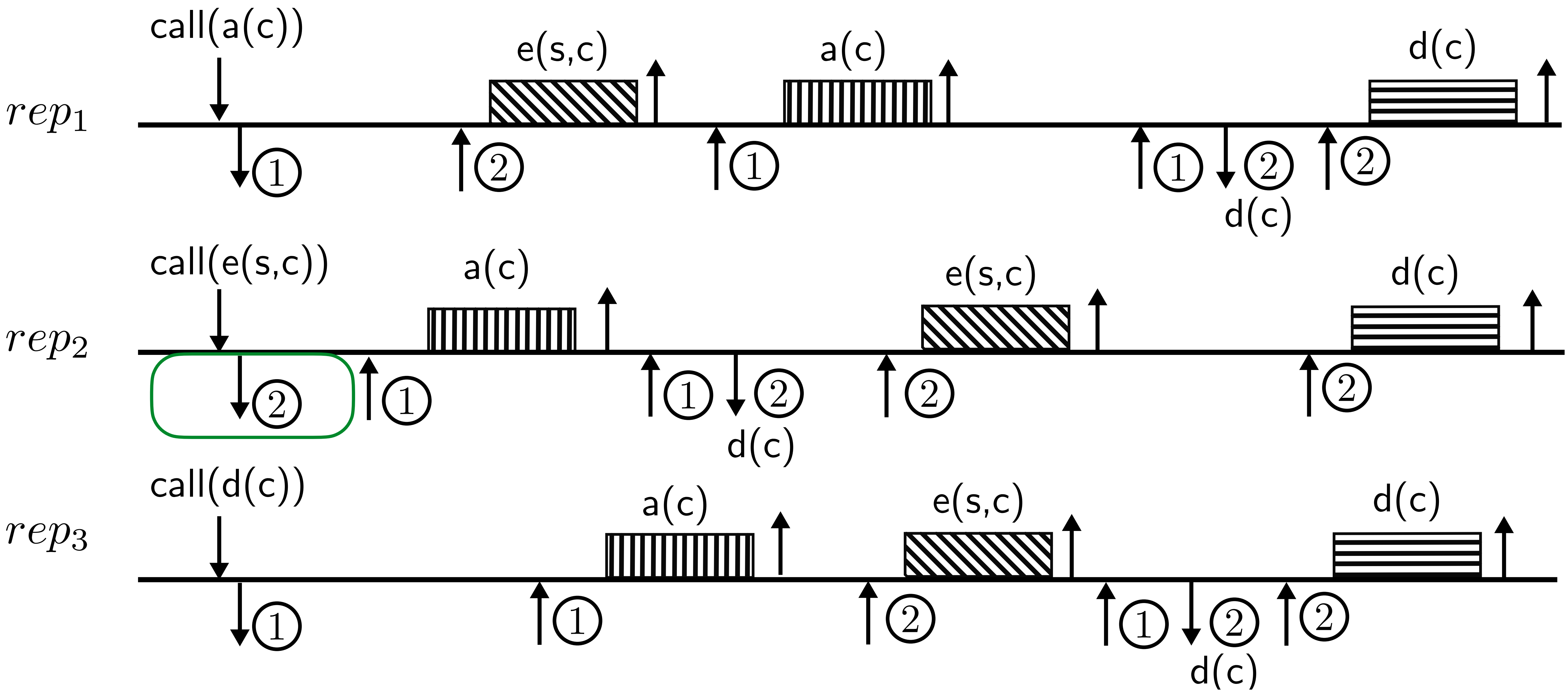


Non-blocking Protocol

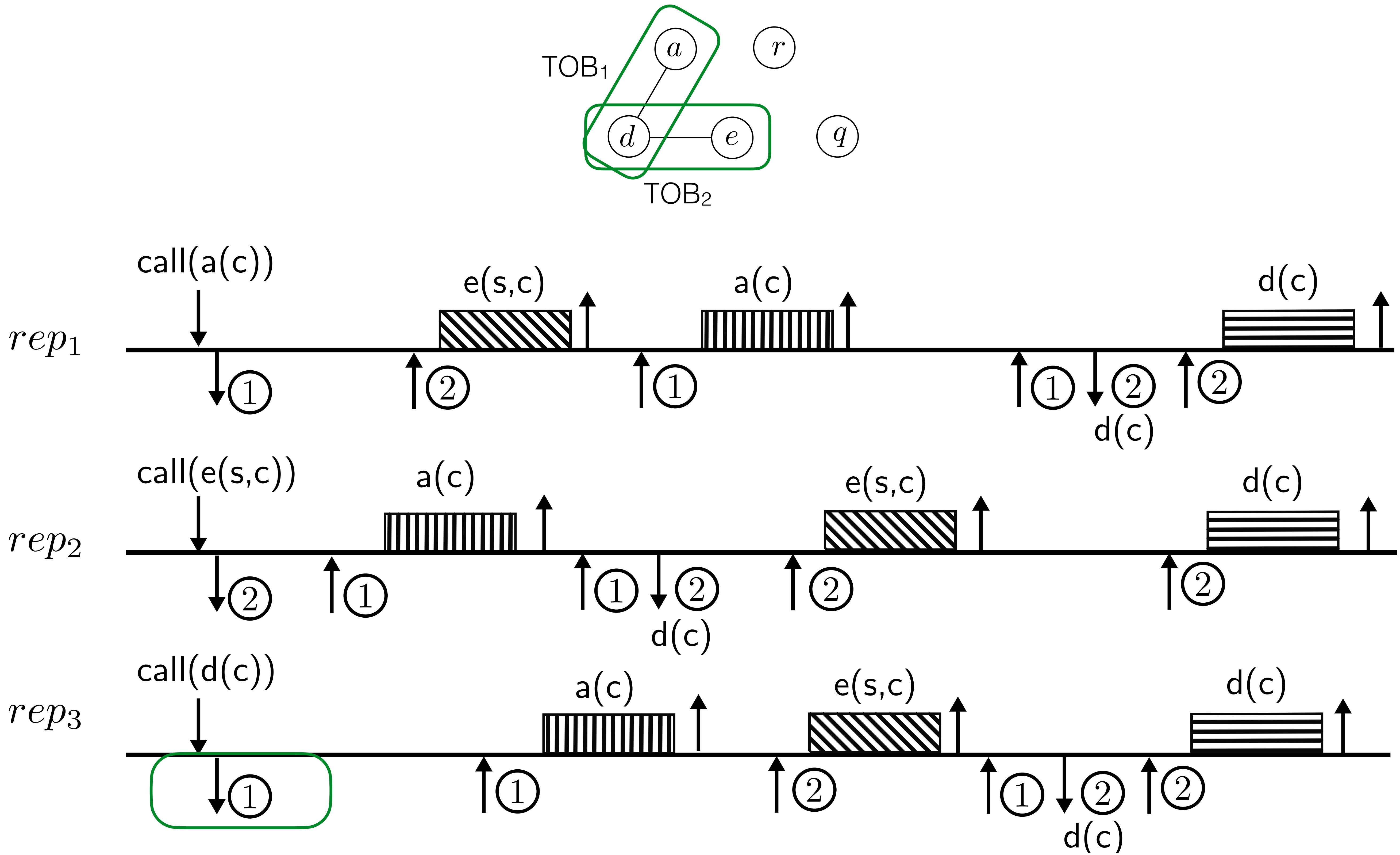


Non-blocking Protocol

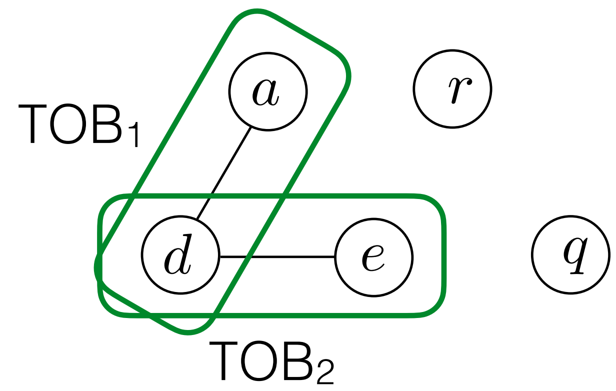


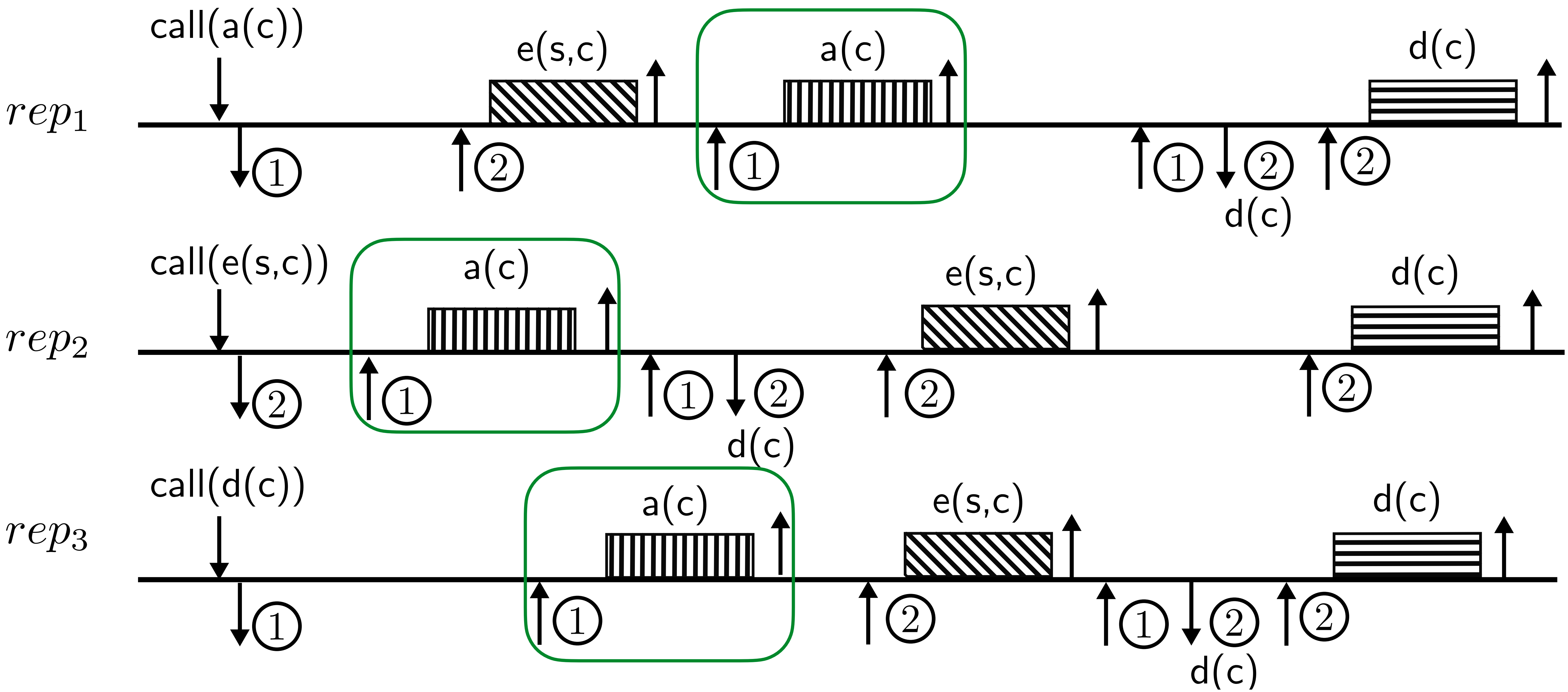


Non-blocking Protocol

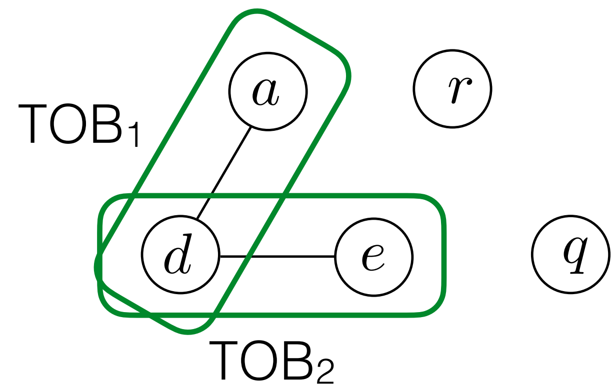


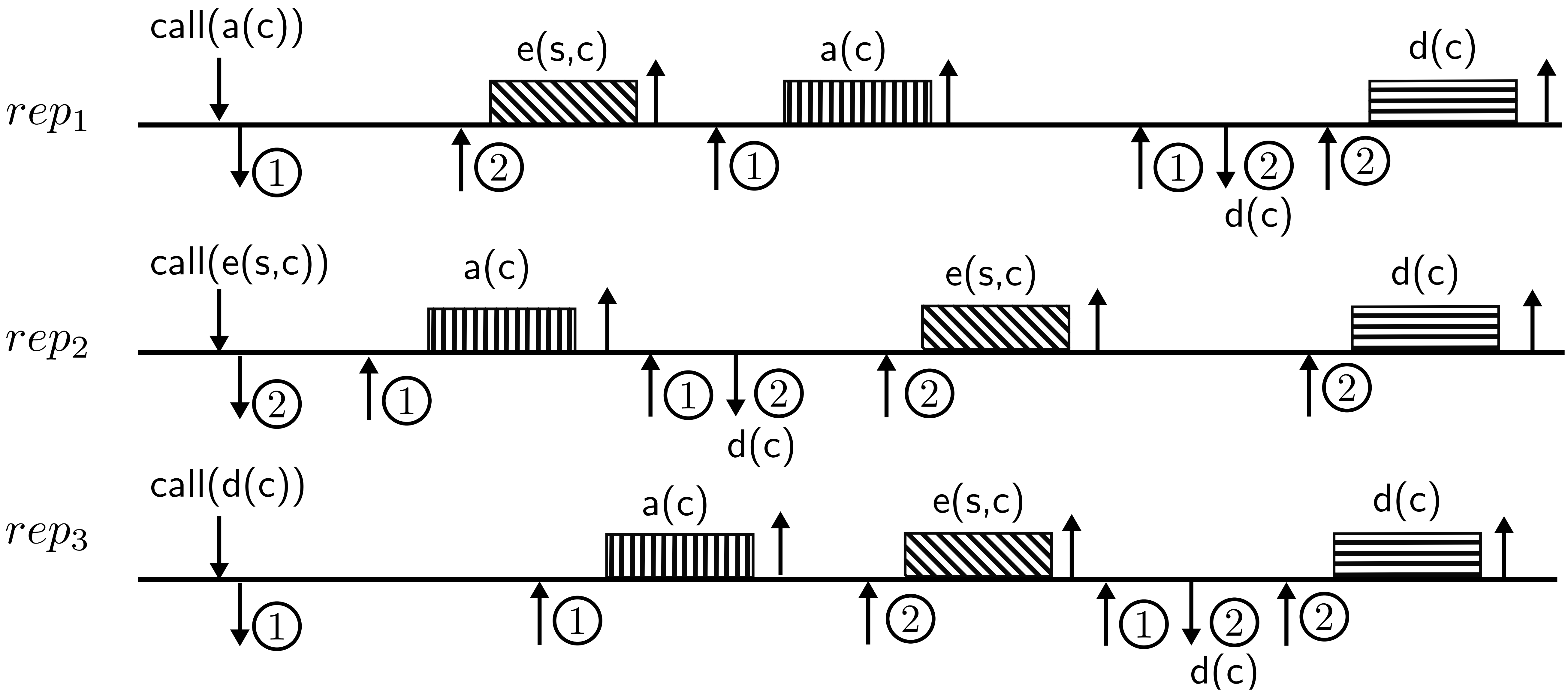
Non-blocking Protocol



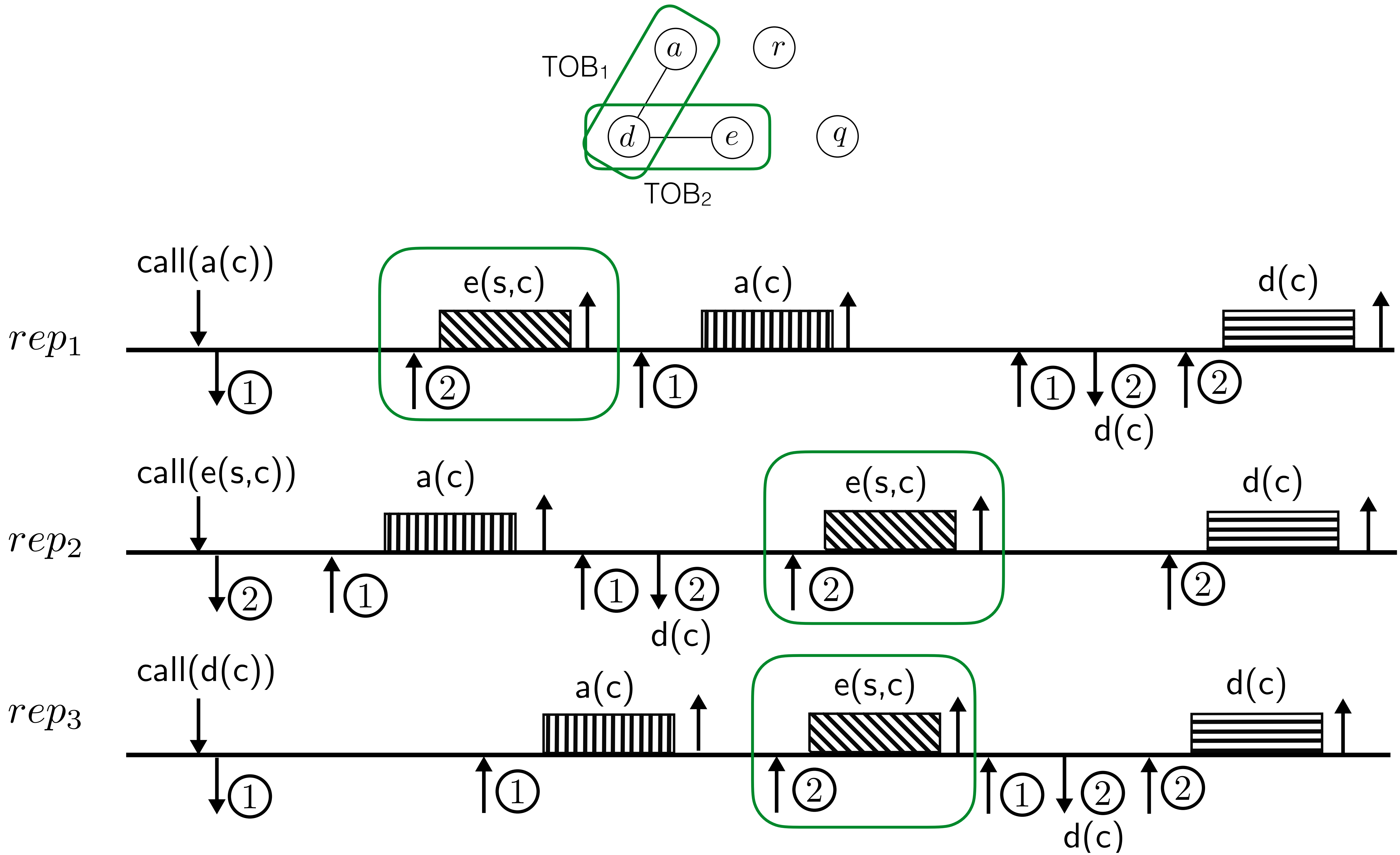


Non-blocking Protocol

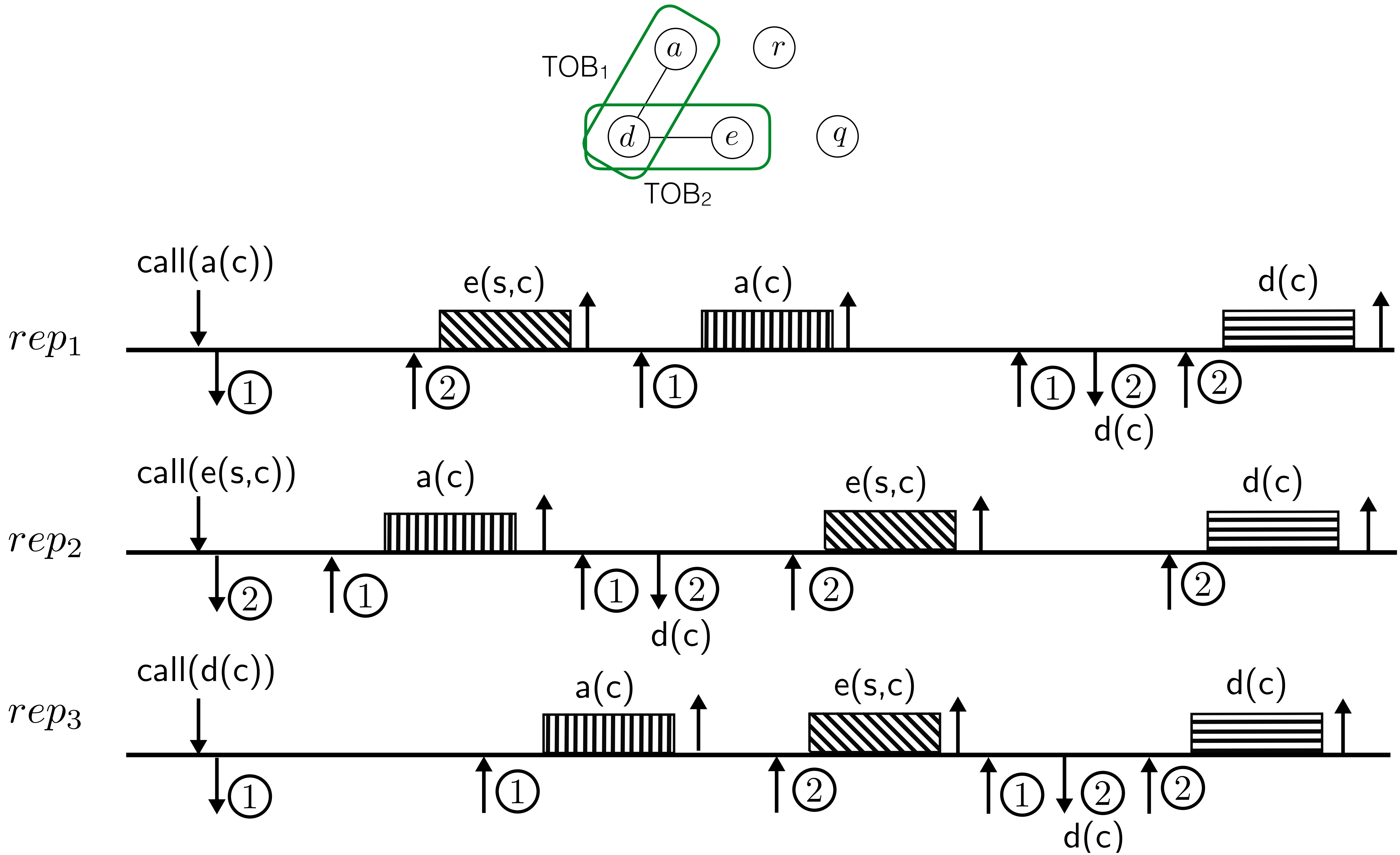




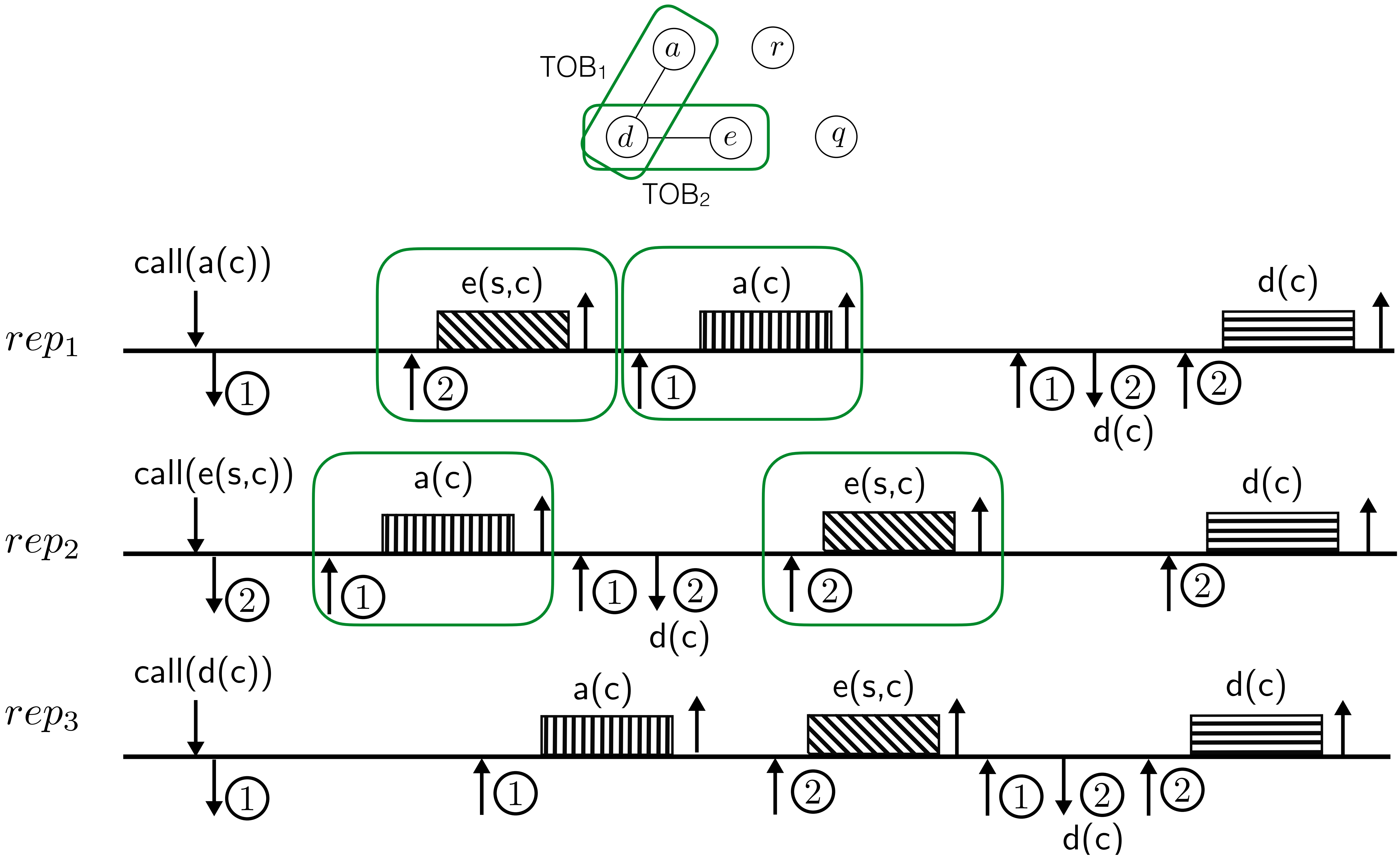
Non-blocking Protocol



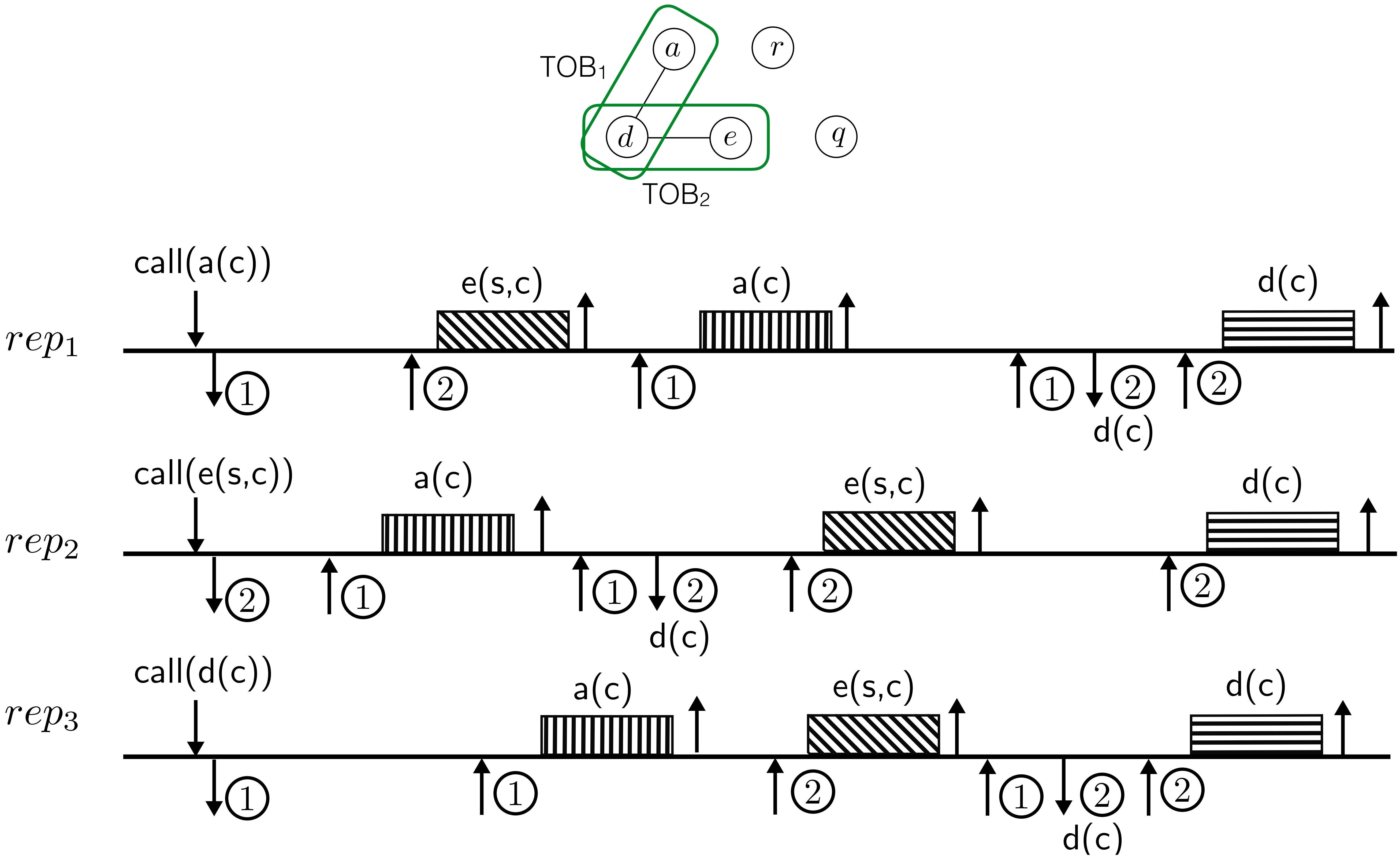
Non-blocking Protocol



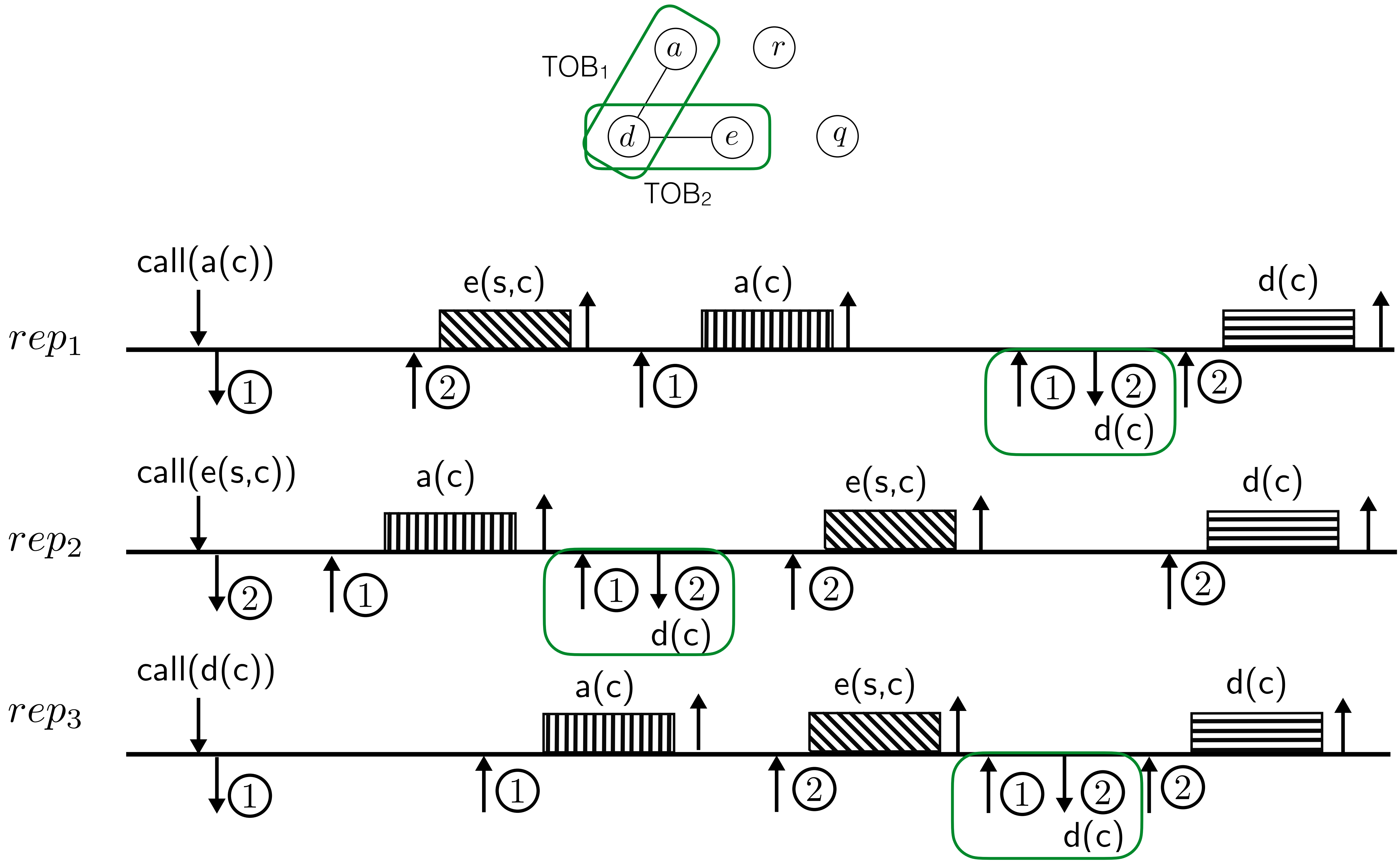
Non-blocking Protocol



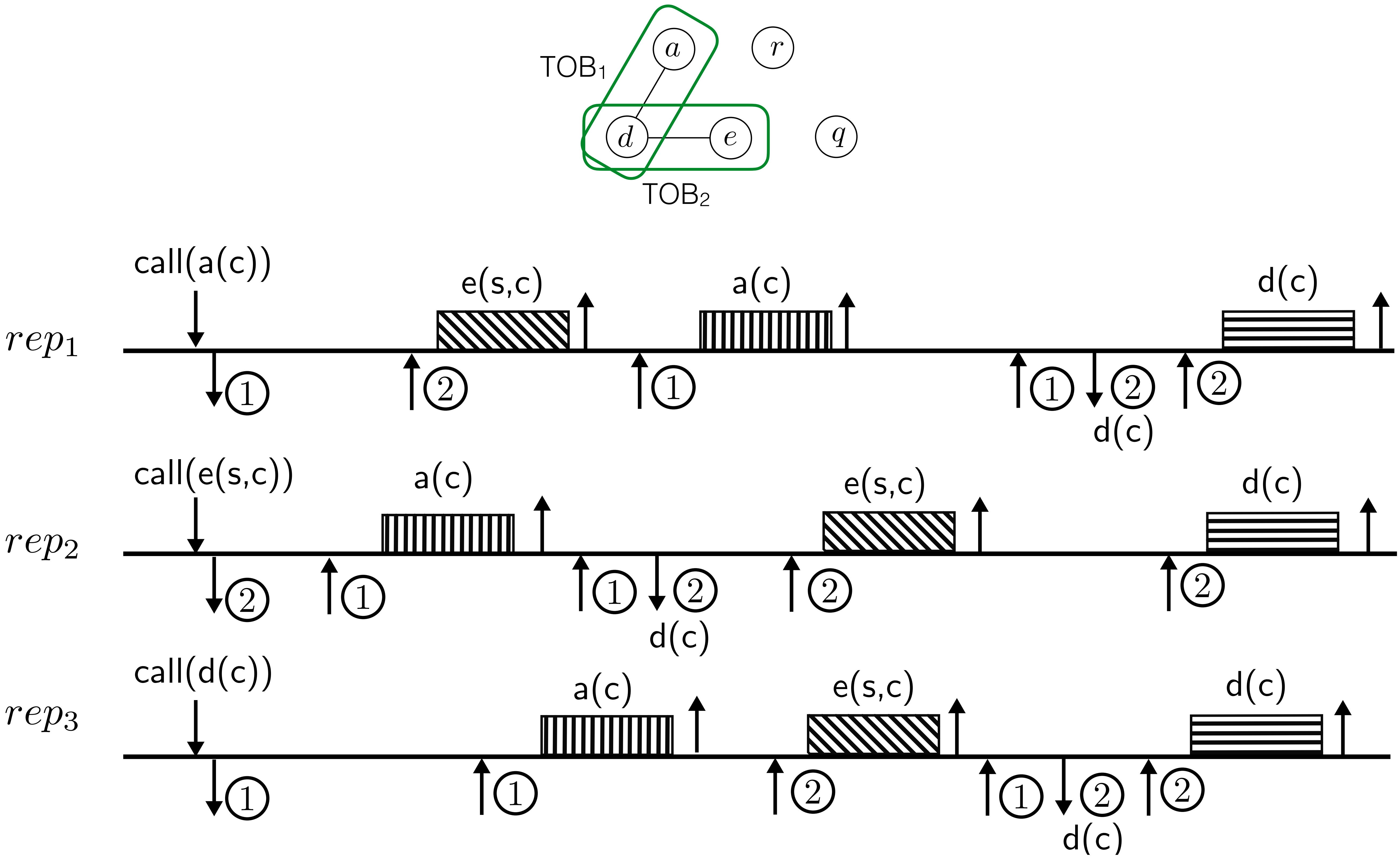
Non-blocking Protocol



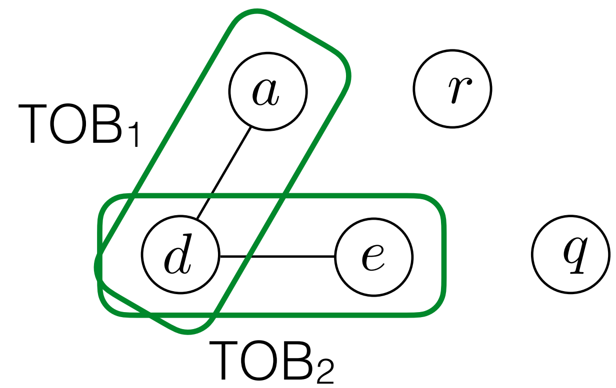
Non-blocking Protocol

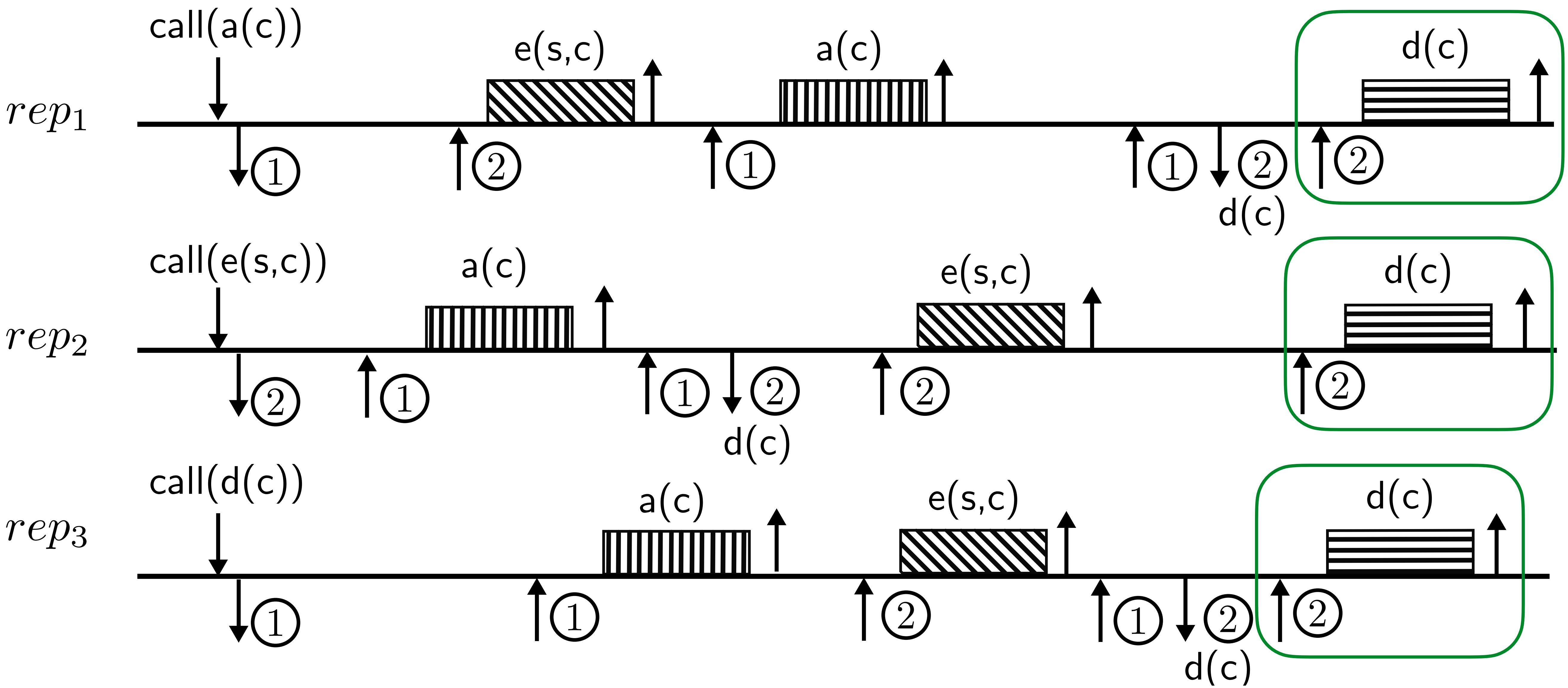


Non-blocking Protocol

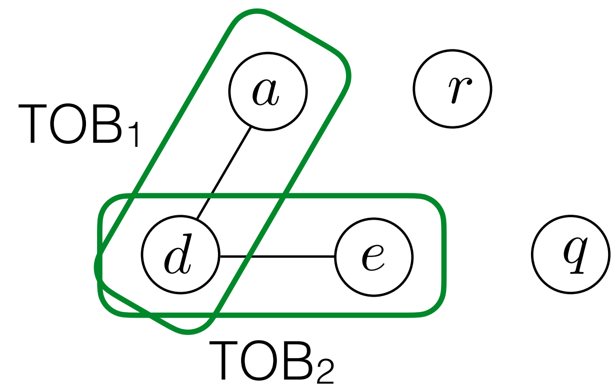


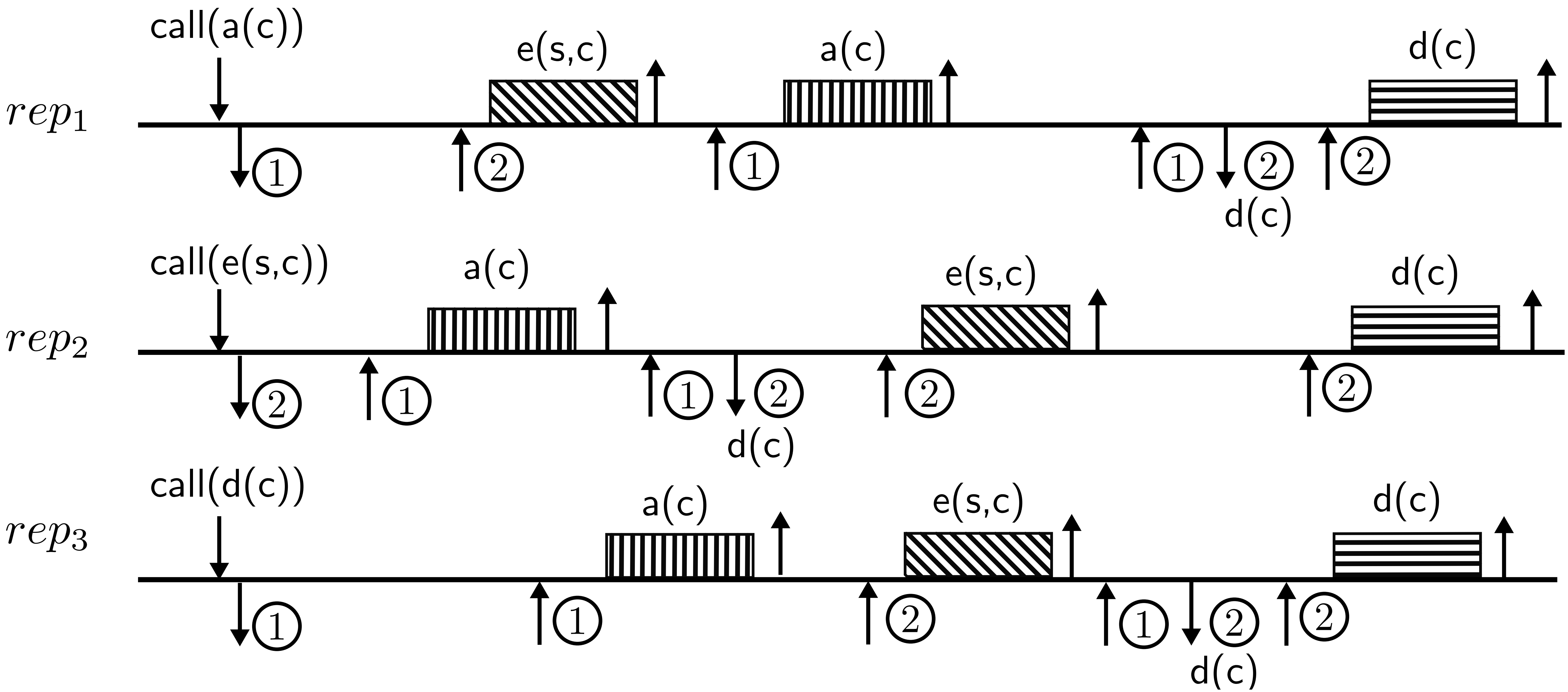
Non-blocking Protocol



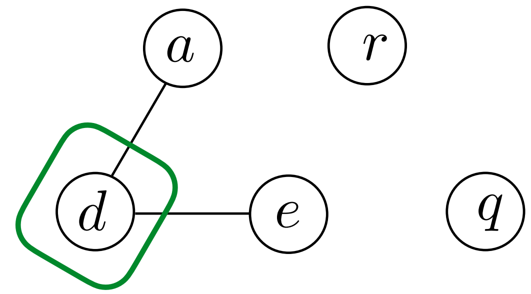


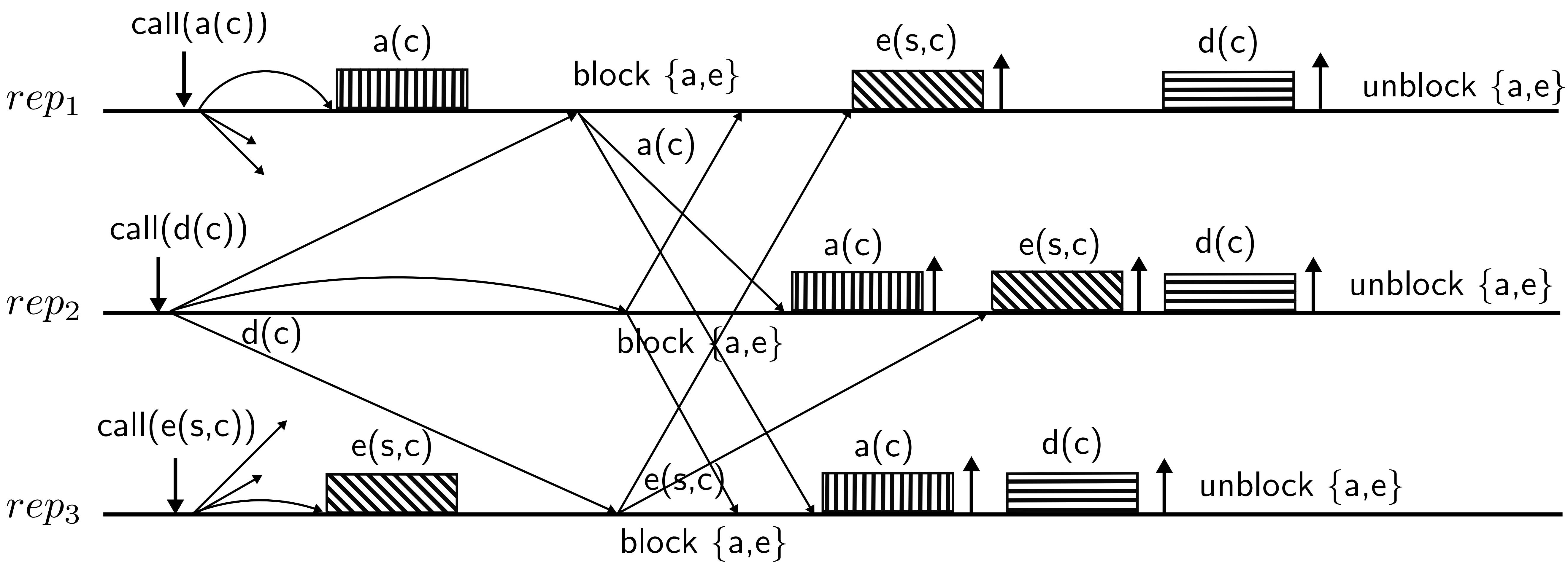
Non-blocking Protocol



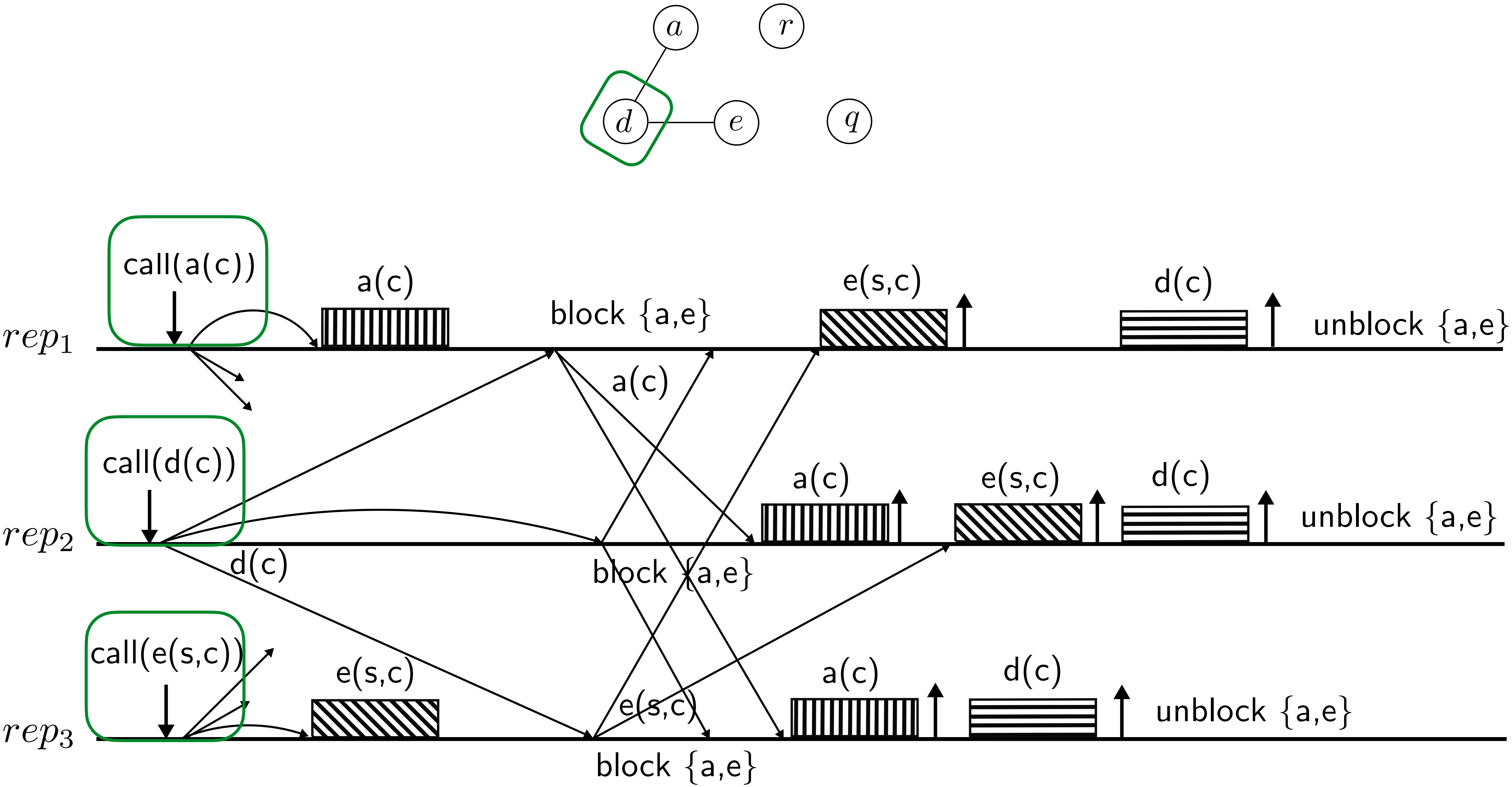


Blocking Protocol

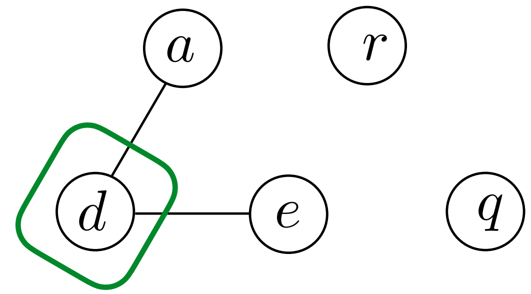


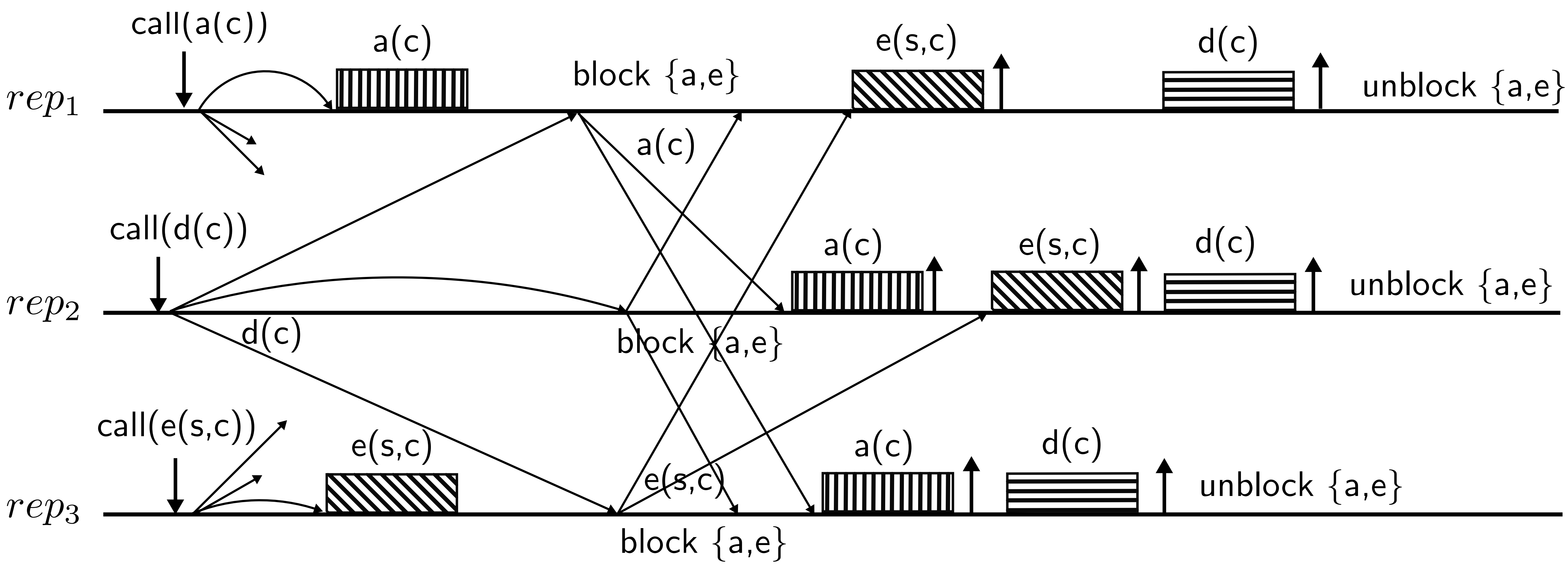


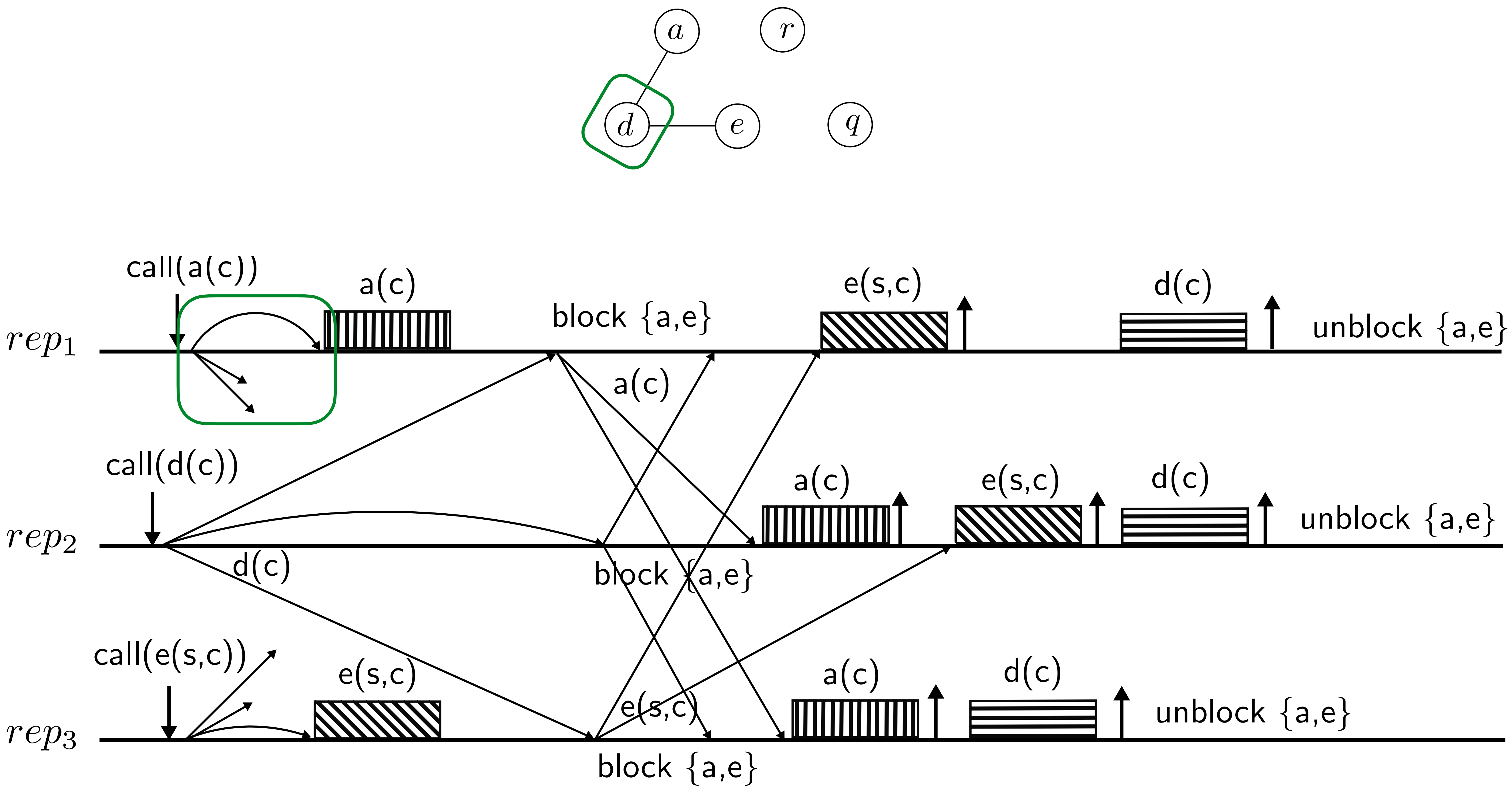
Blocking Protocol



Blocking Protocol

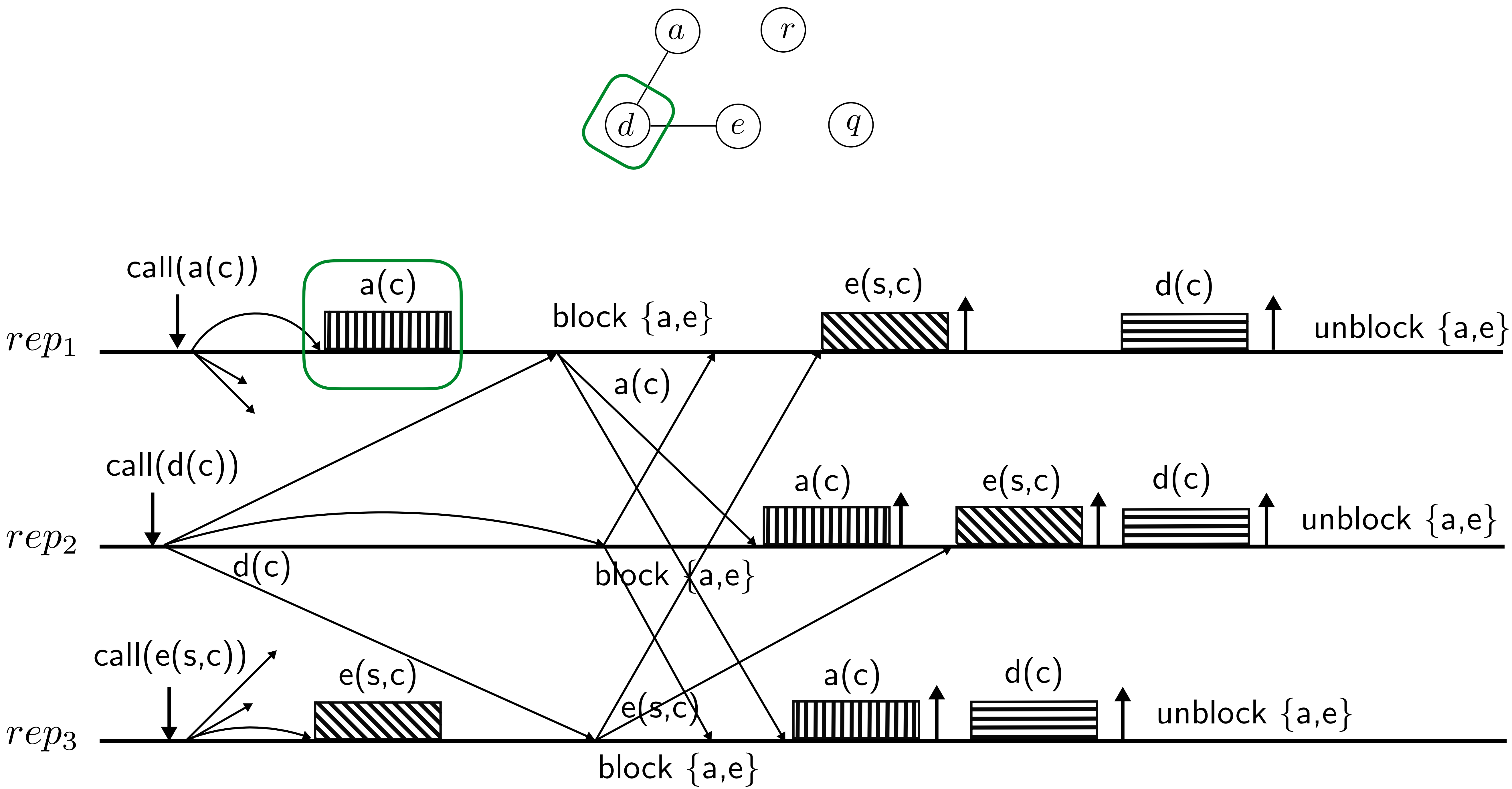




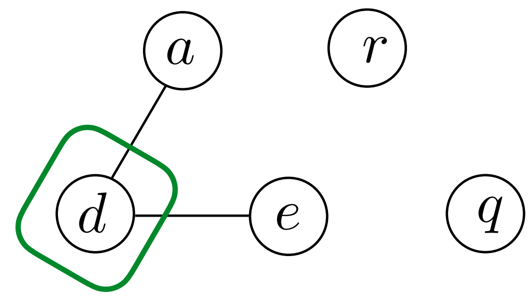


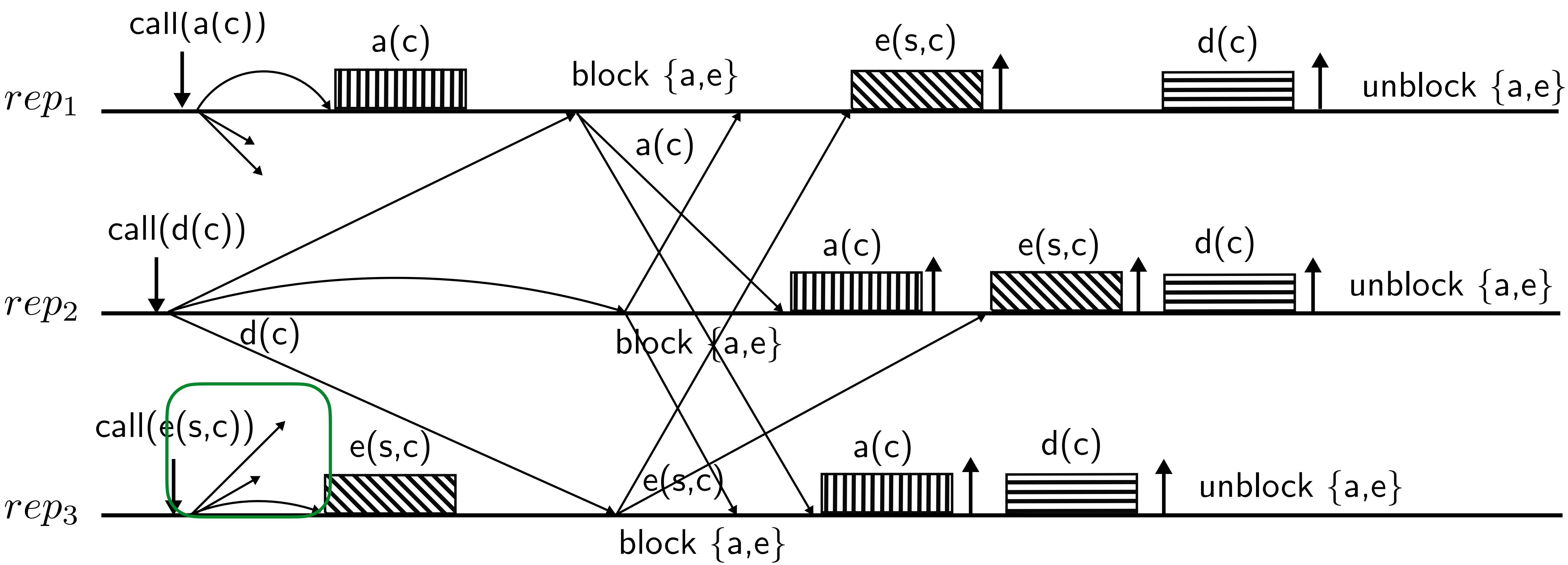
a
r
d
e
q
call(a(c))
a(c)
block {a,e}
e(s,c)
d(c)
unblock {a,e}
rep_1
a(c)
call(d(c))
a(c)
e(s,c)
d(c)
unblock {a,e}
rep_2
d(c)
block {a,e}
call(e(s,c))
e(s,c)
e(s,c)
a(c)
d(c)
unblock {a,e}
rep_3
block {a,e}

Blocking Protocol

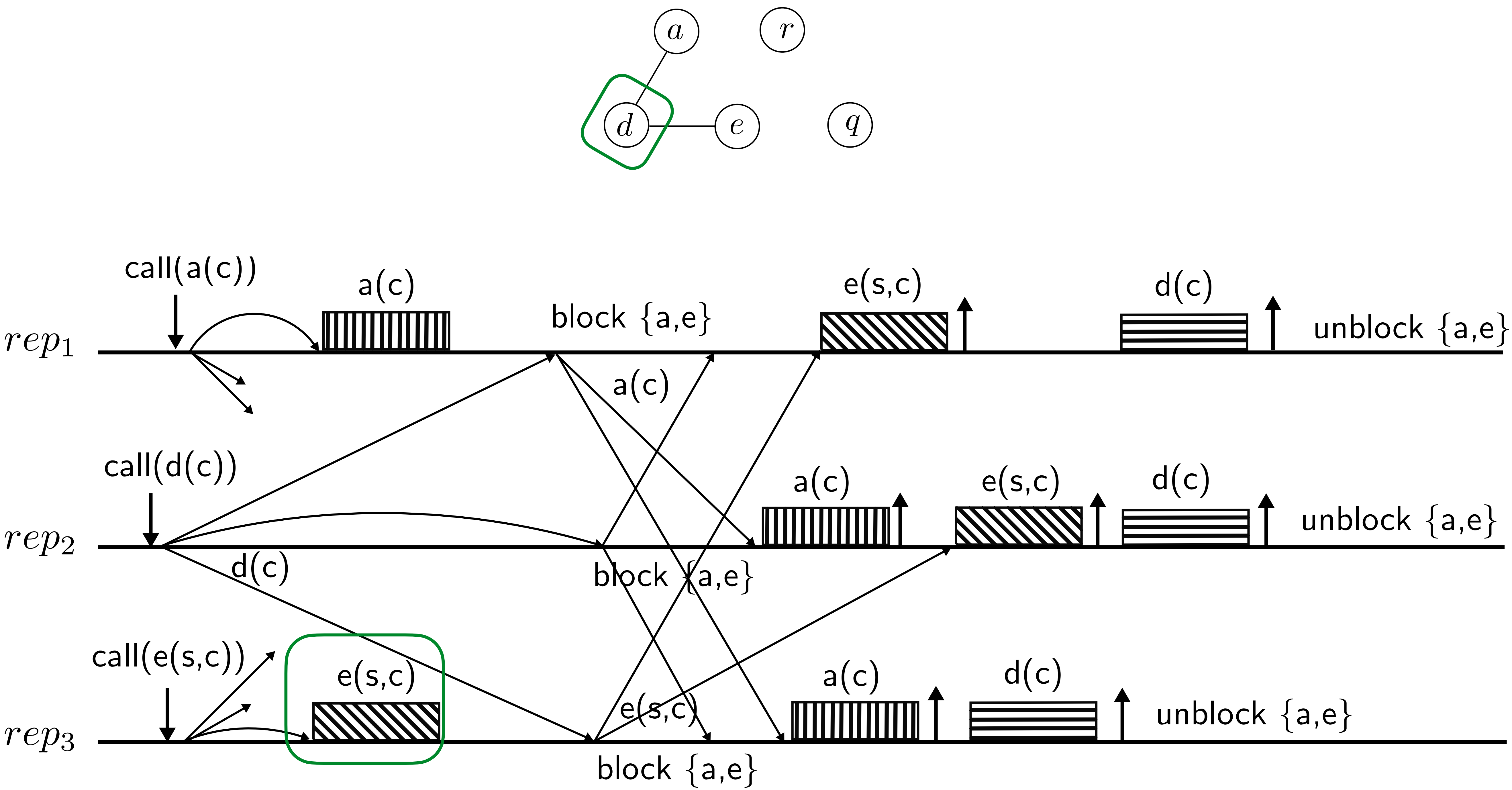


Blocking Protocol

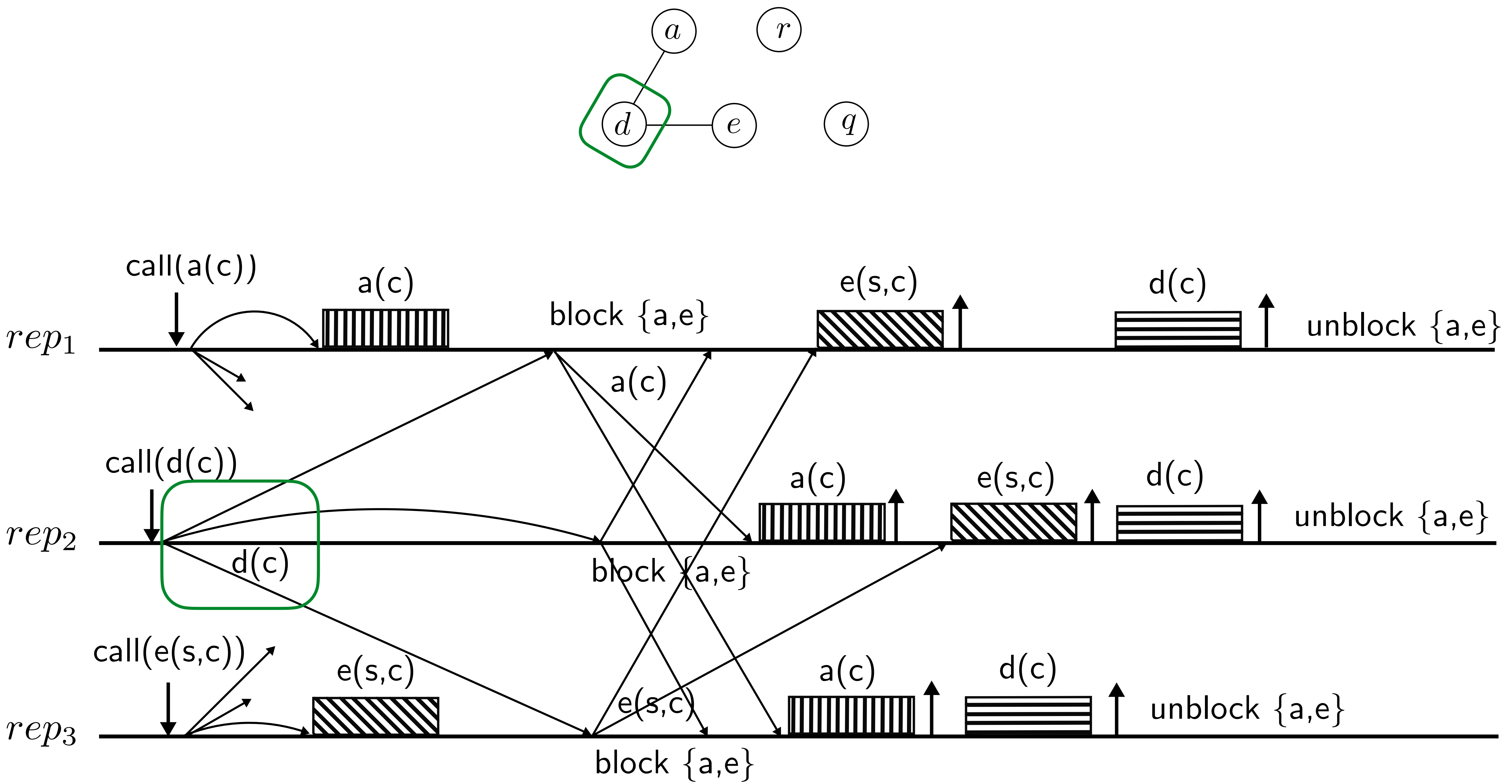




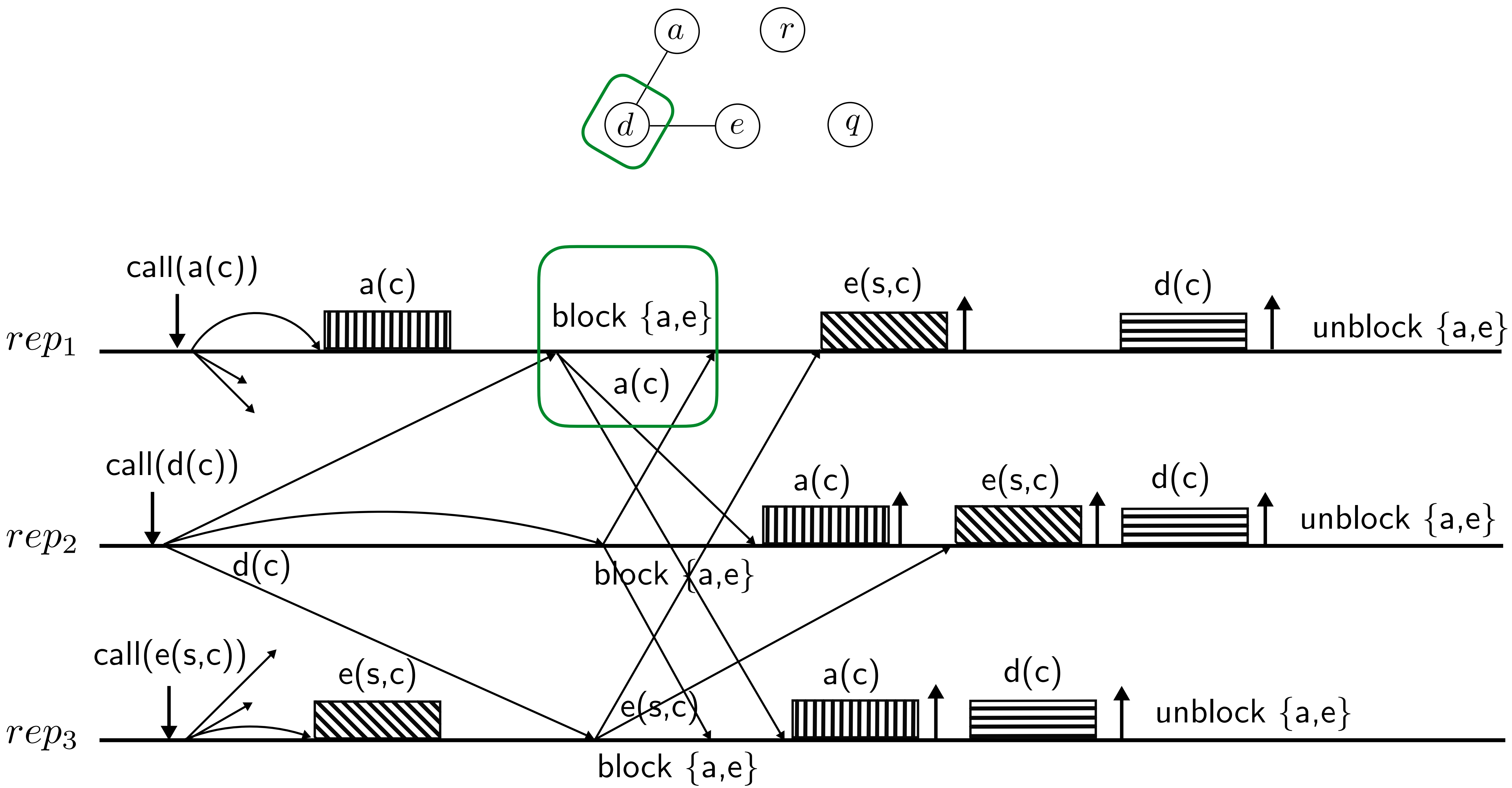
Blocking Protocol



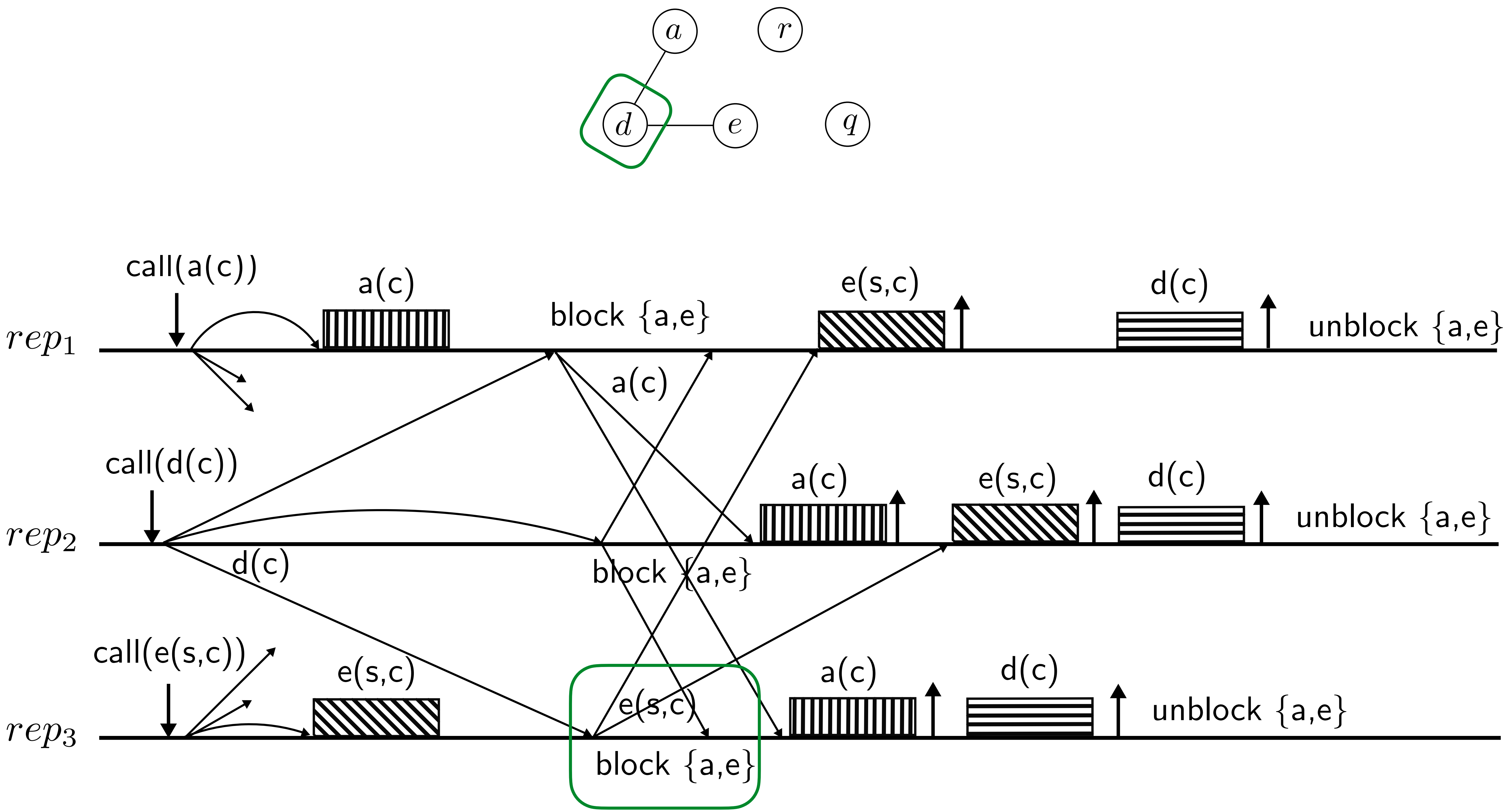
Blocking Protocol



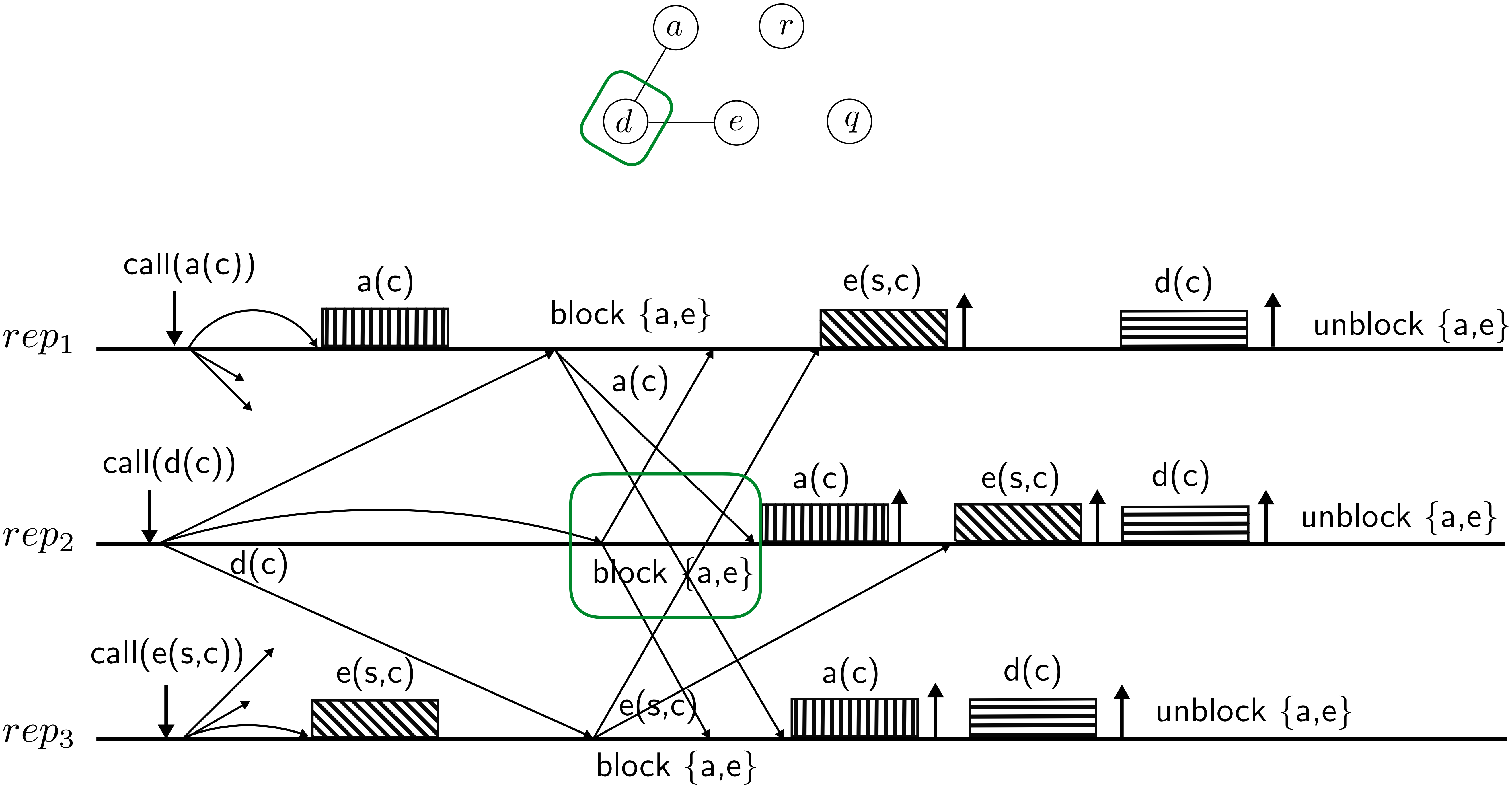
Blocking Protocol



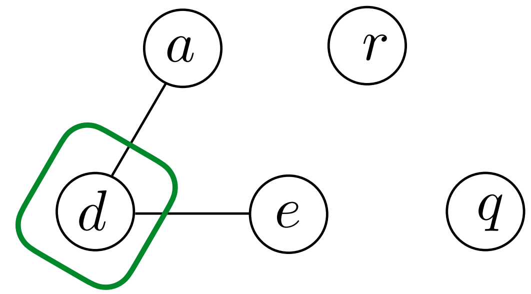
Blocking Protocol

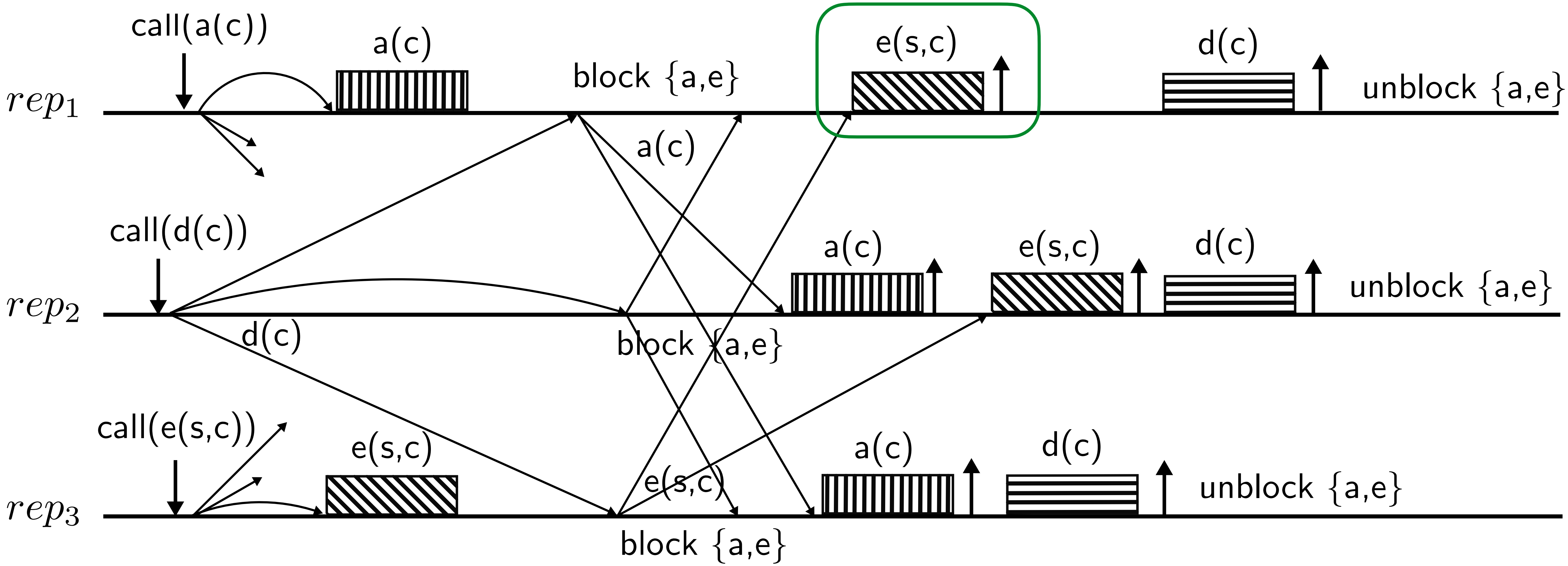


Blocking Protocol

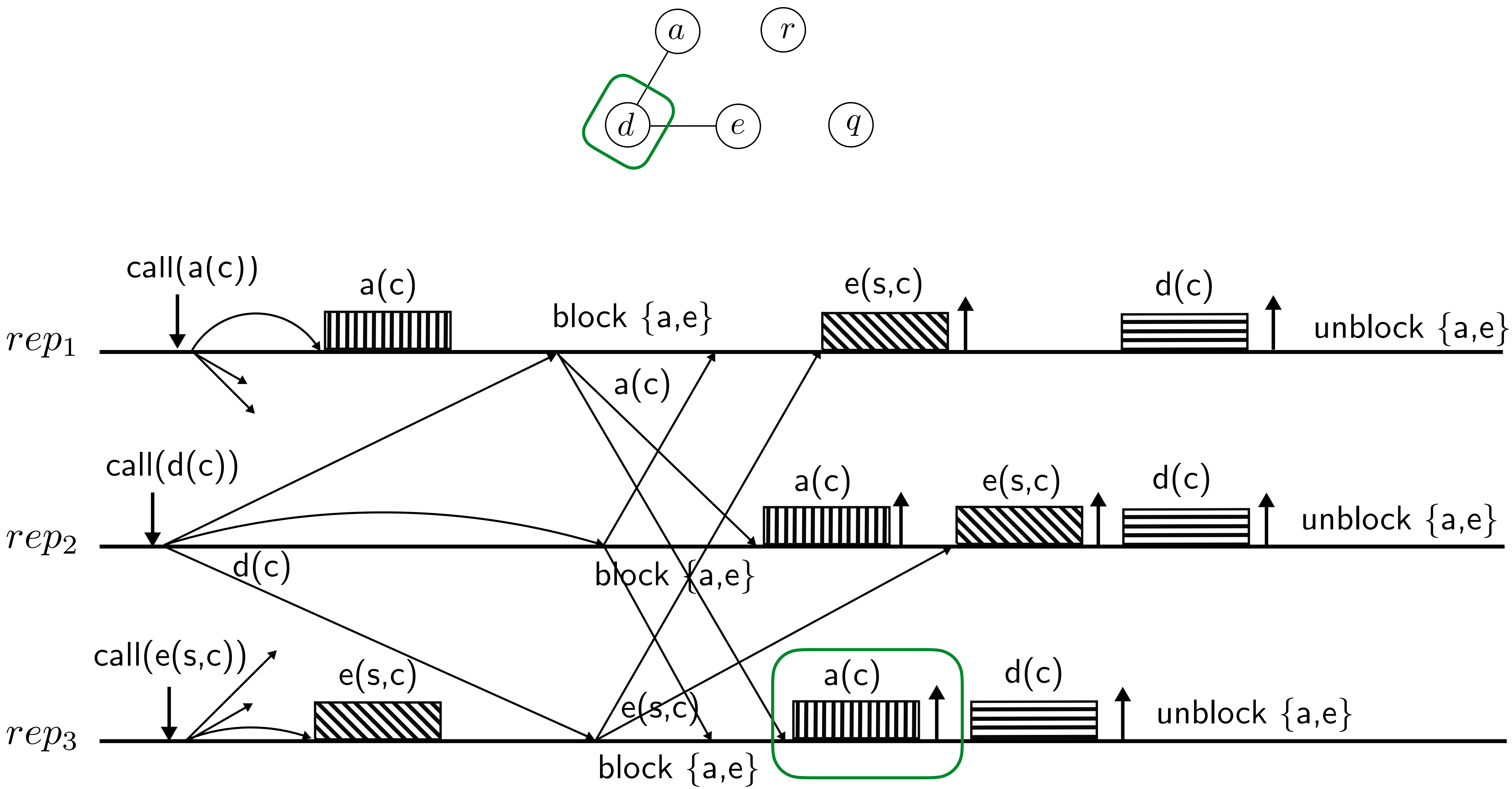


Blocking Protocol

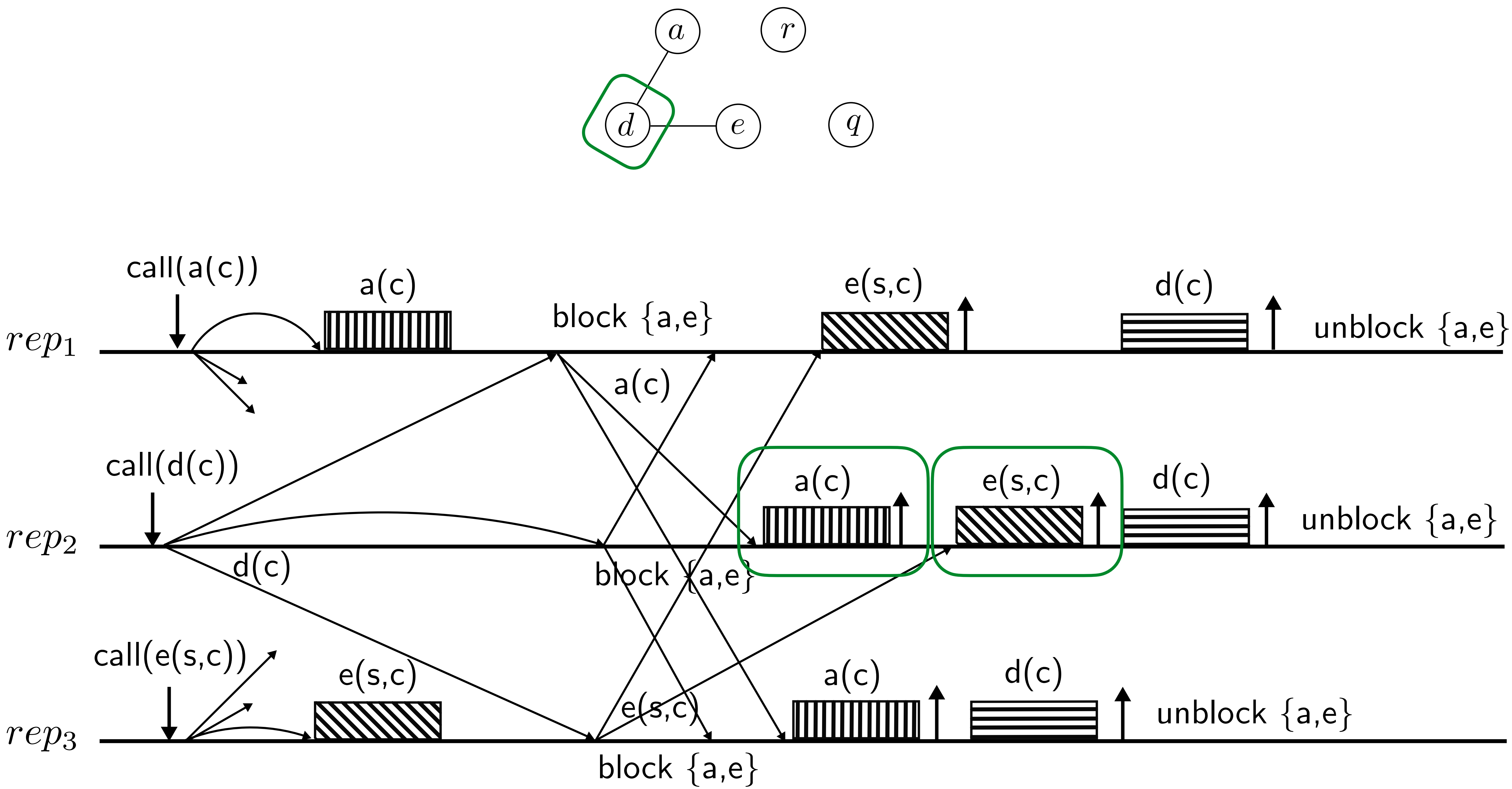




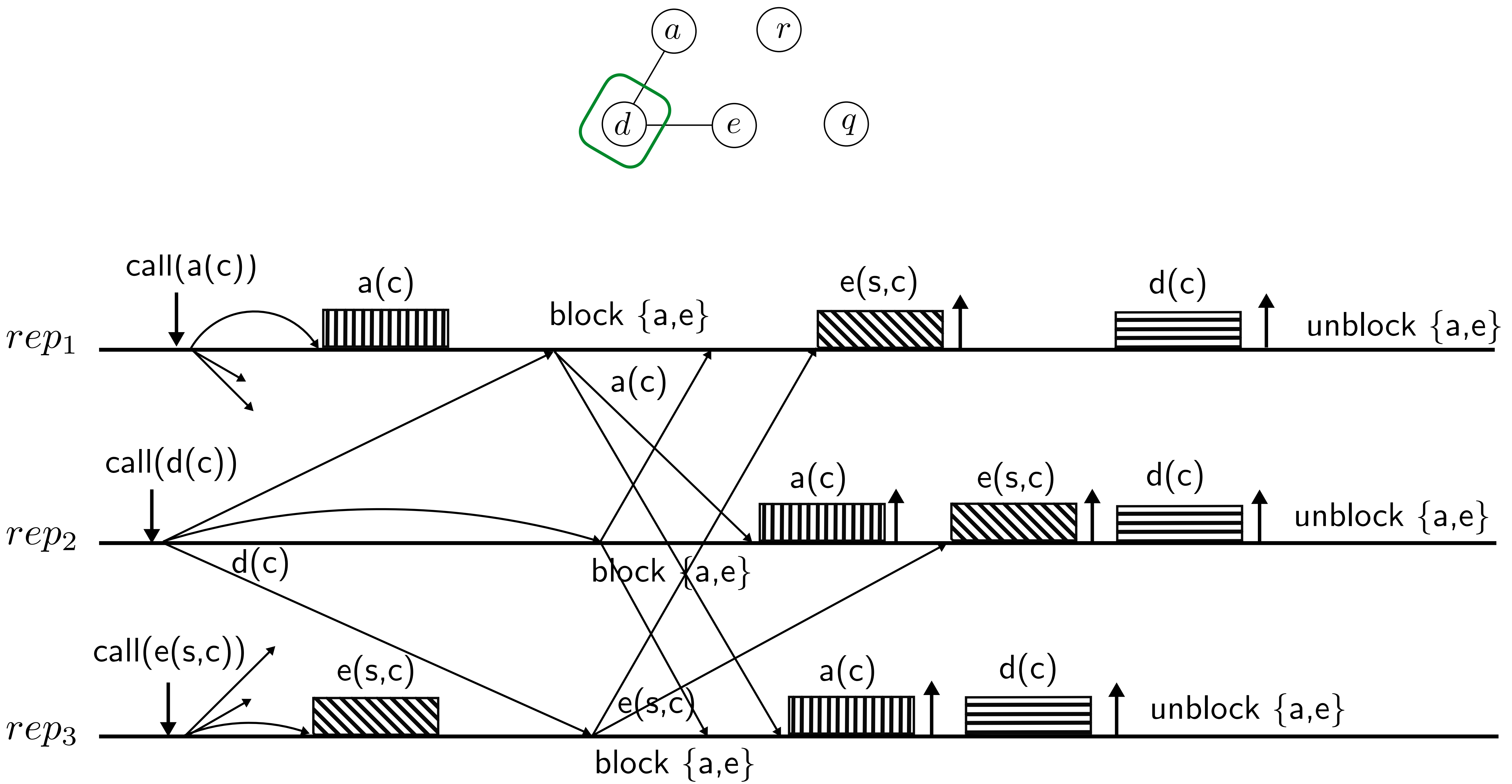
Blocking Protocol



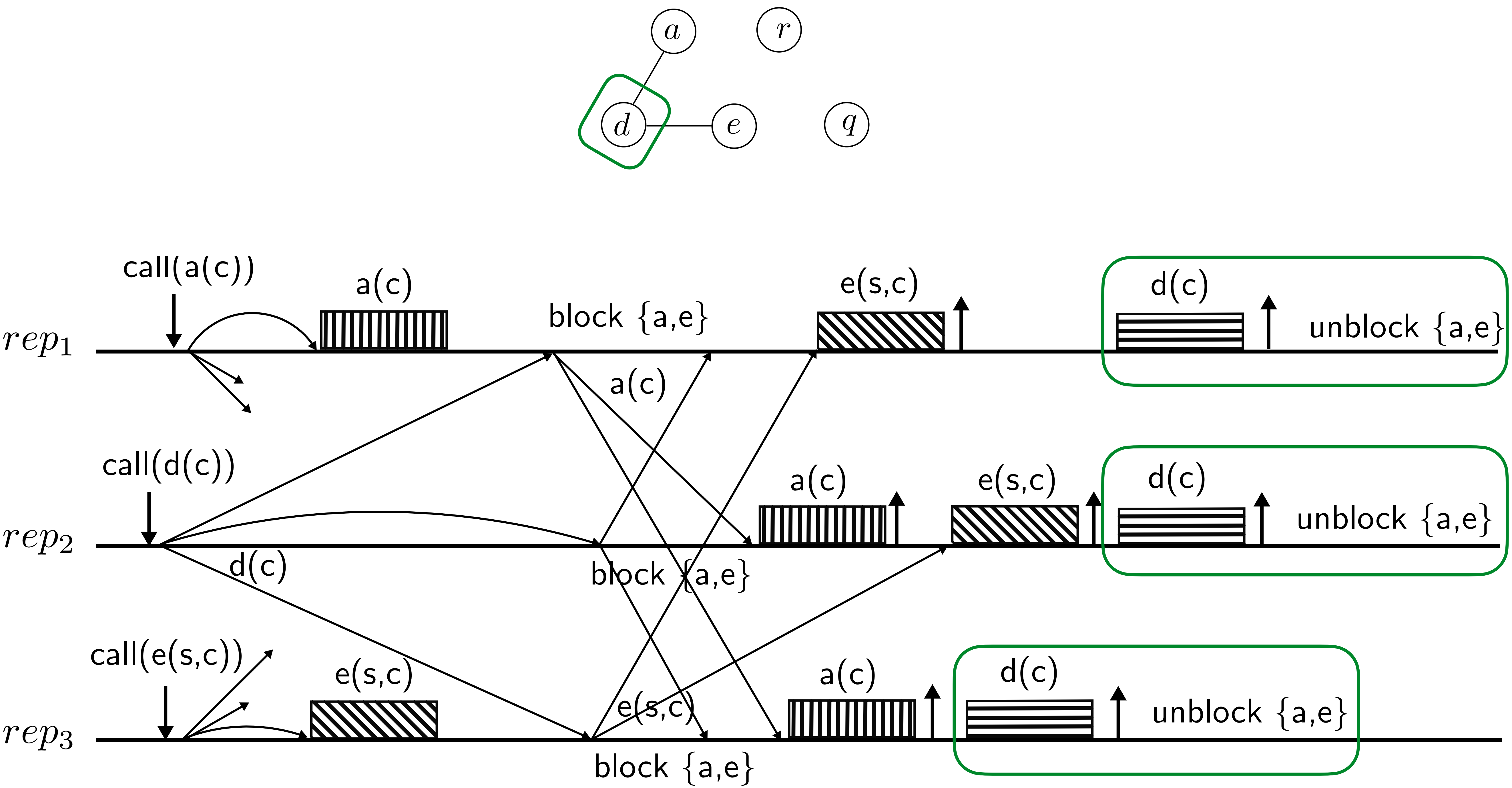
Blocking Protocol



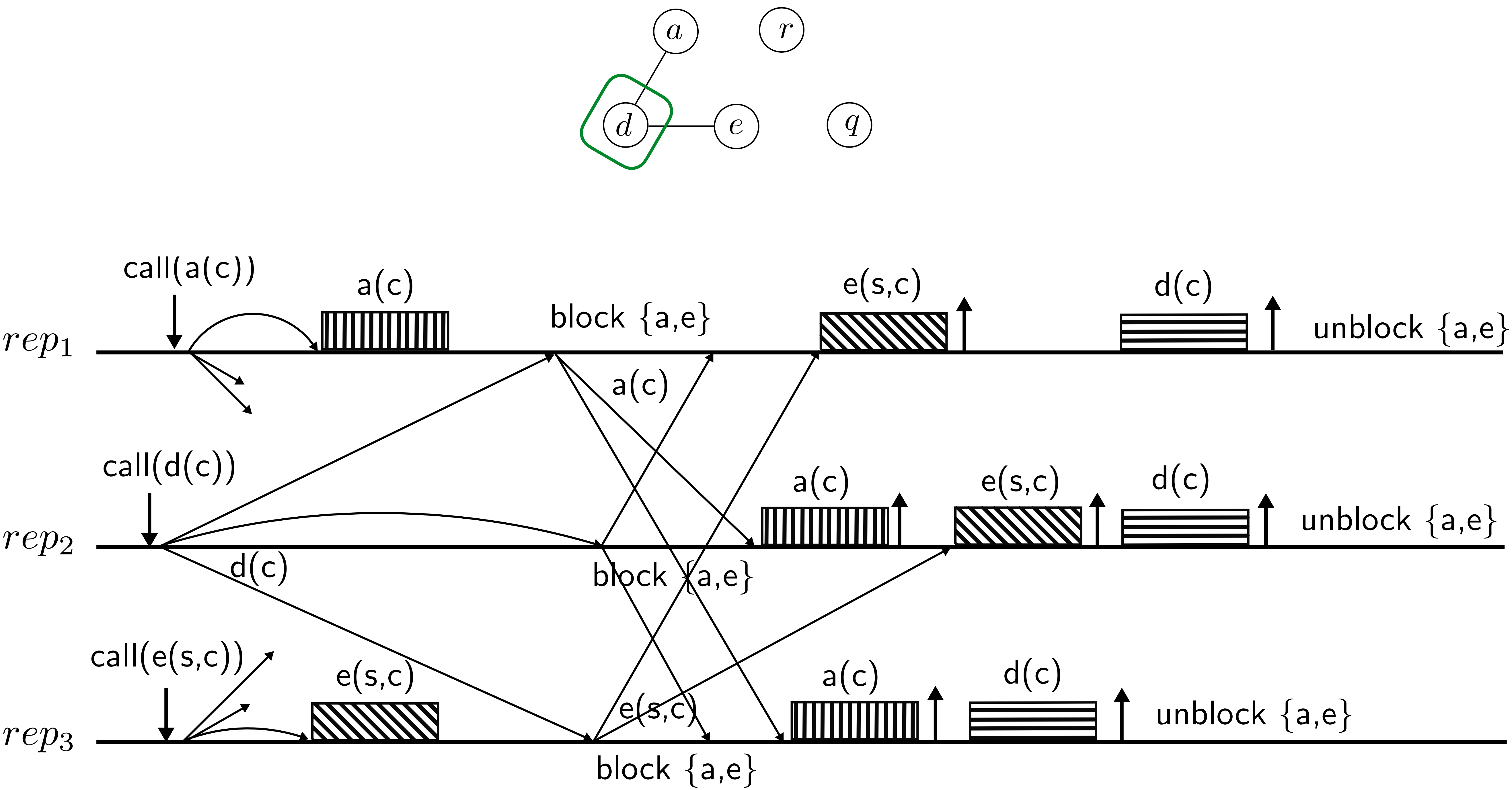
Blocking Protocol



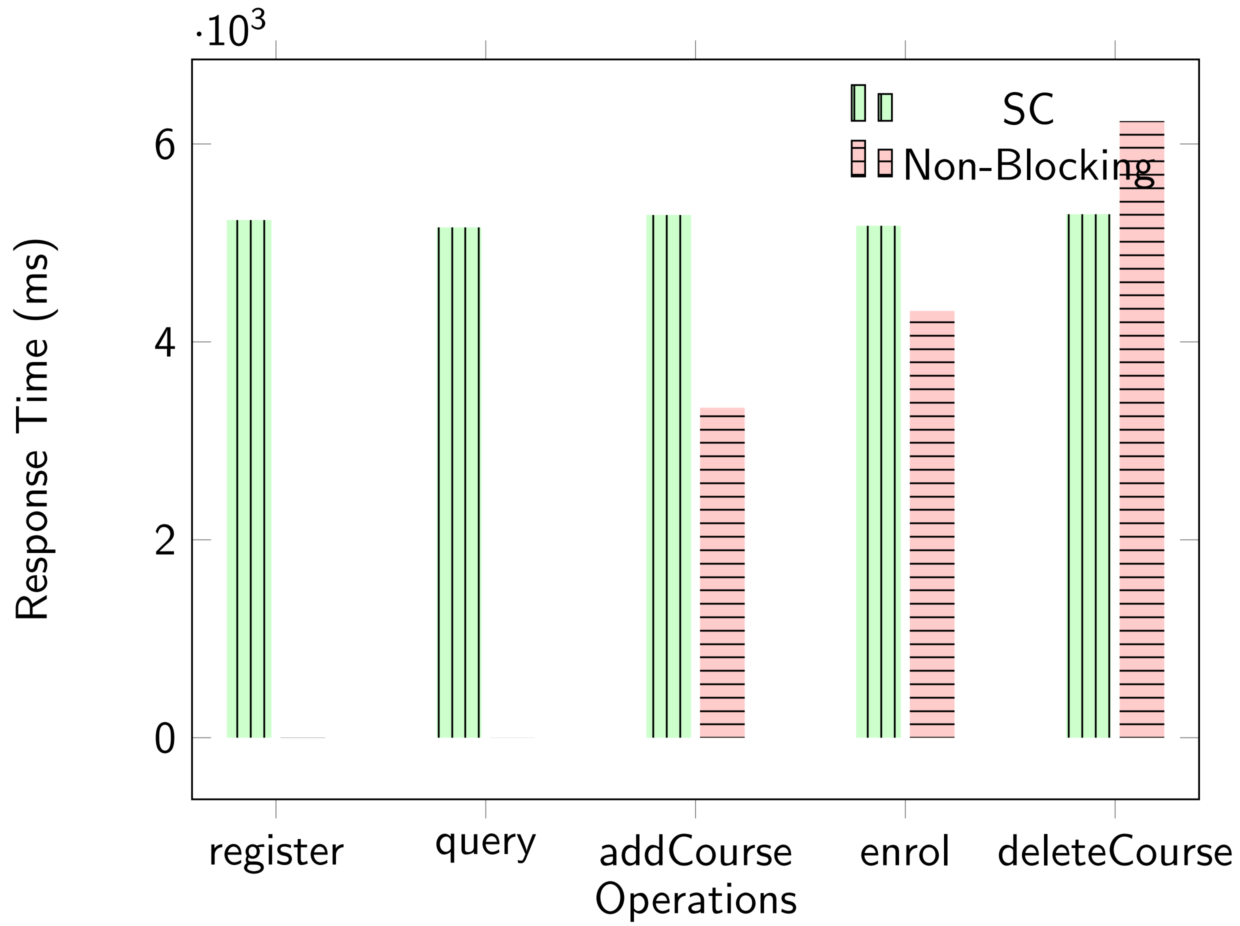
Blocking Protocol

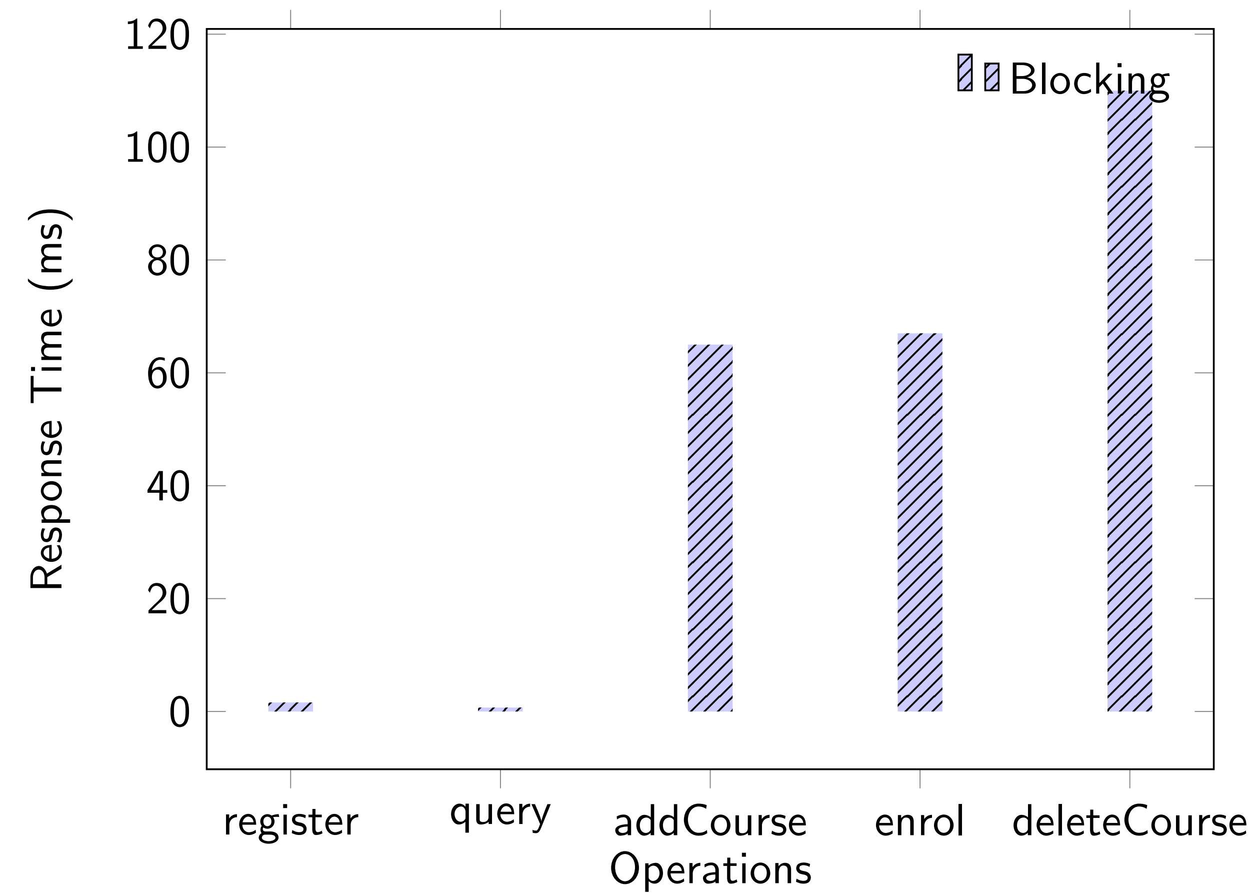


Blocking Protocol



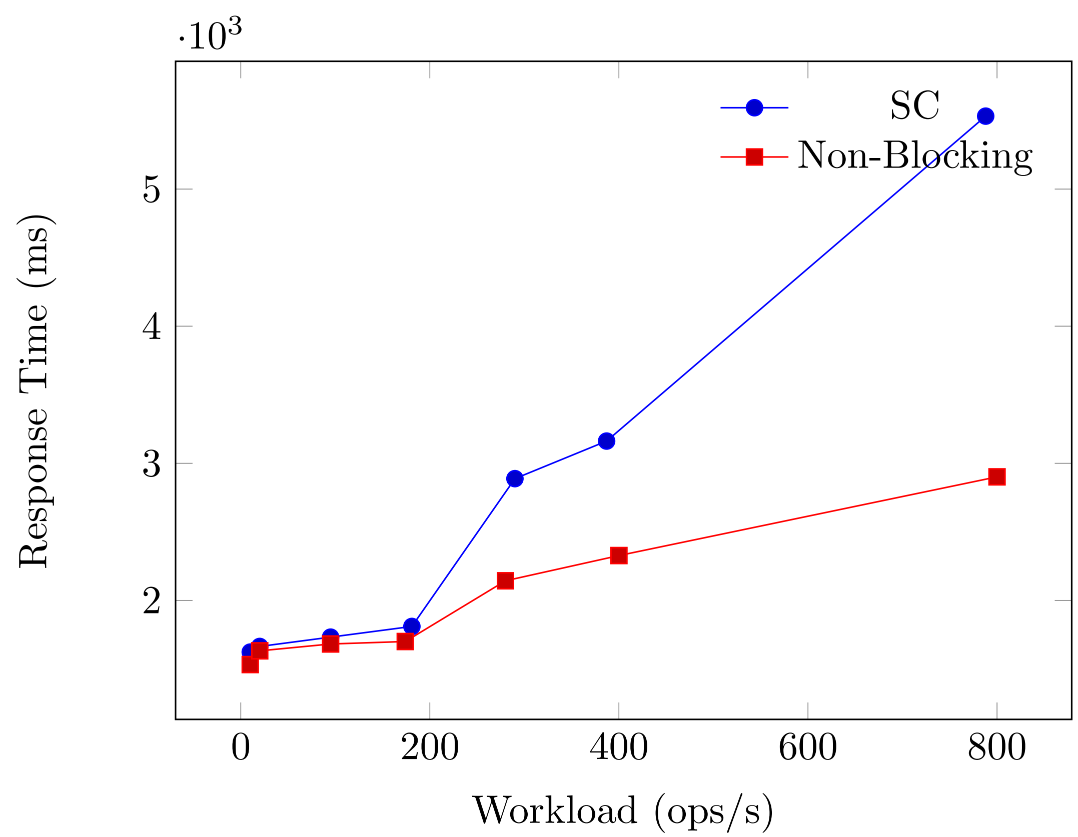
Experiments





We execute 500 calls evenly distributed on the methods.
We issue one call per millisecond and measure the average response time of the calls on each method.

Experiments



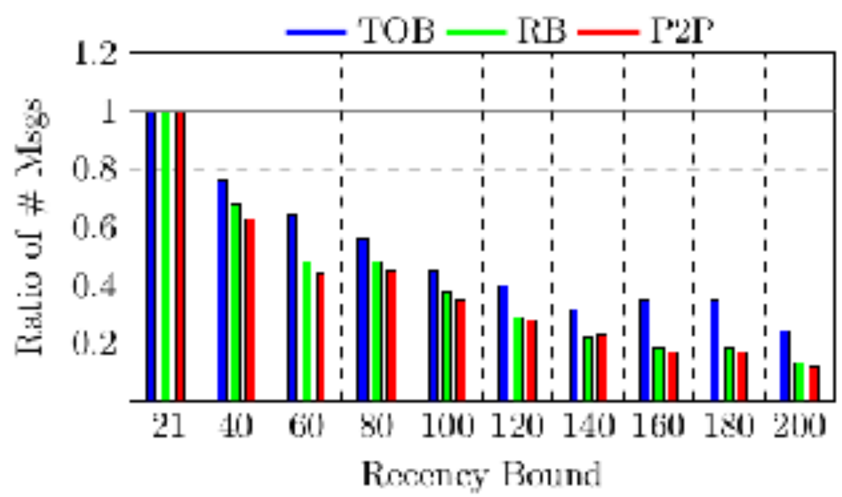
We execute 500 calls evenly distributed on the methods.
We increase the workload from 10 to 800 calls per second and measure the average response time over all the calls.

Experimental Results

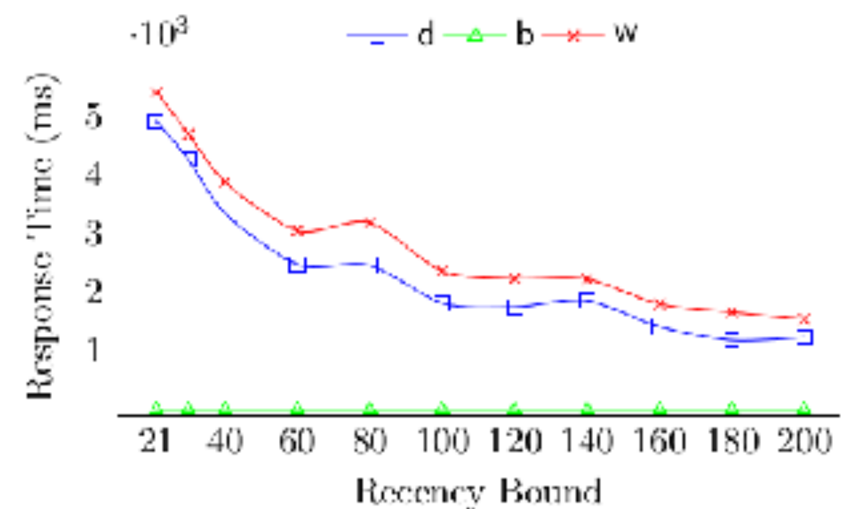
As the recency bound increases,
the coordination overhead and response time decrease.

Bank account

(a)

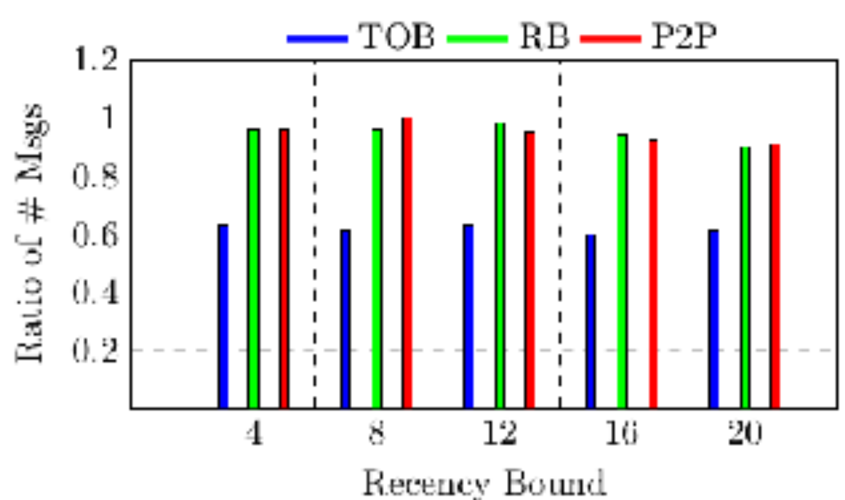


(b)

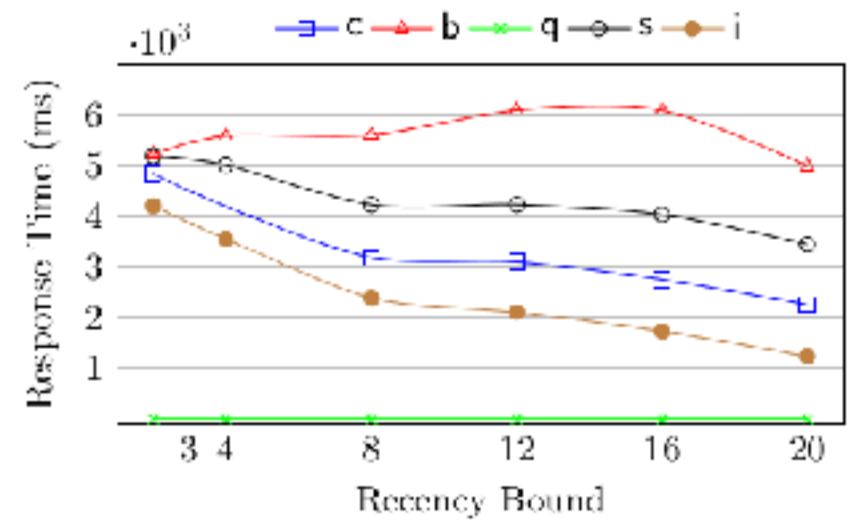


Movie booking

(c)



(d)

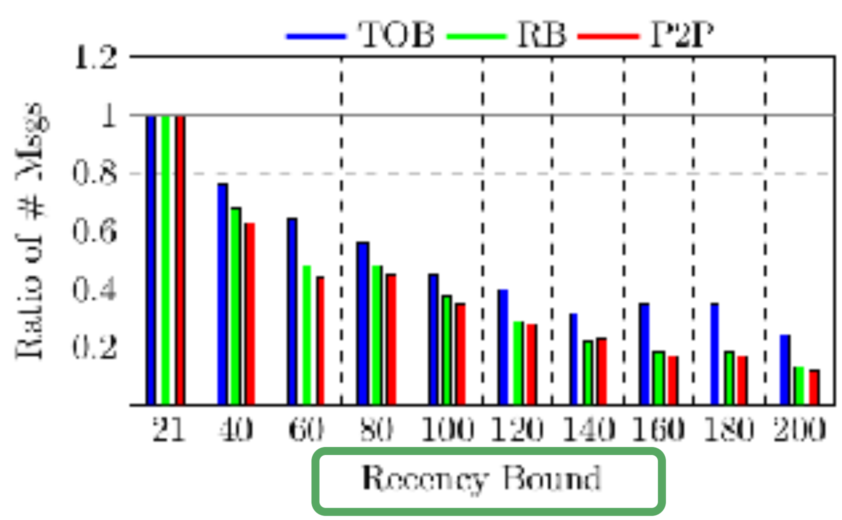


Experimental Results

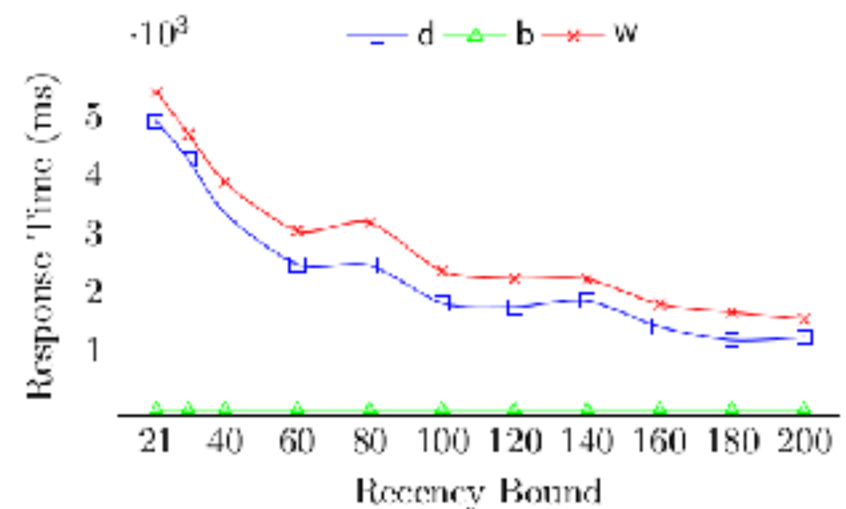
As the recency bound increases,
the coordination overhead and response time decrease.

Bank account

(a)

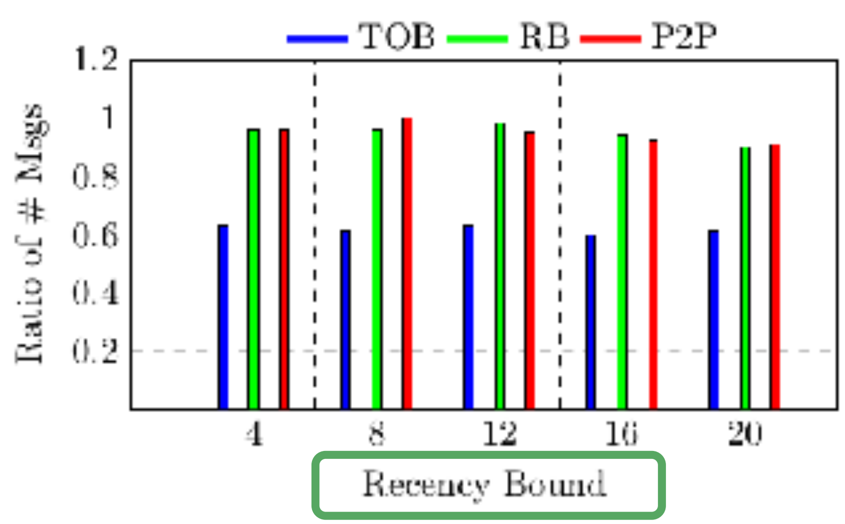


(b)

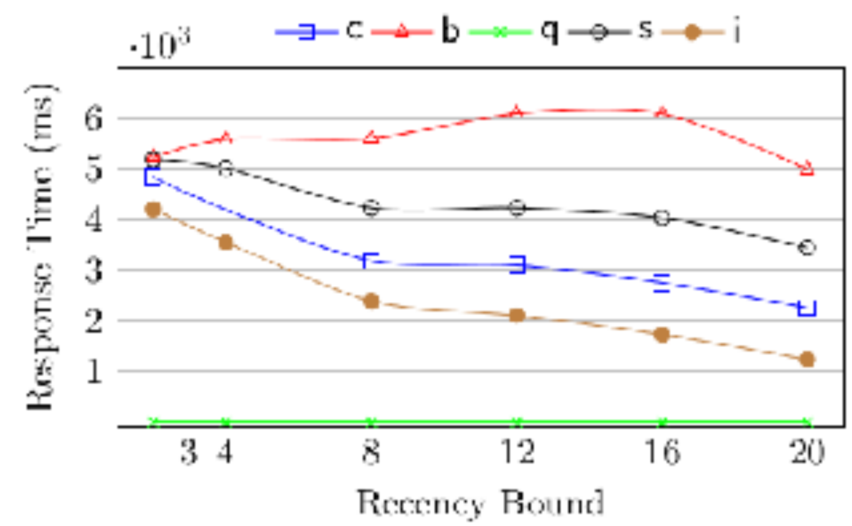


Movie booking

(c)



(d)

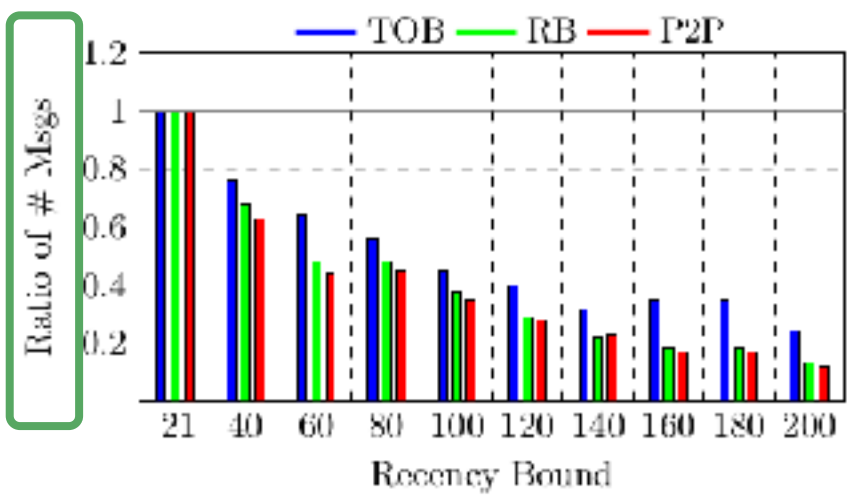


Experimental Results

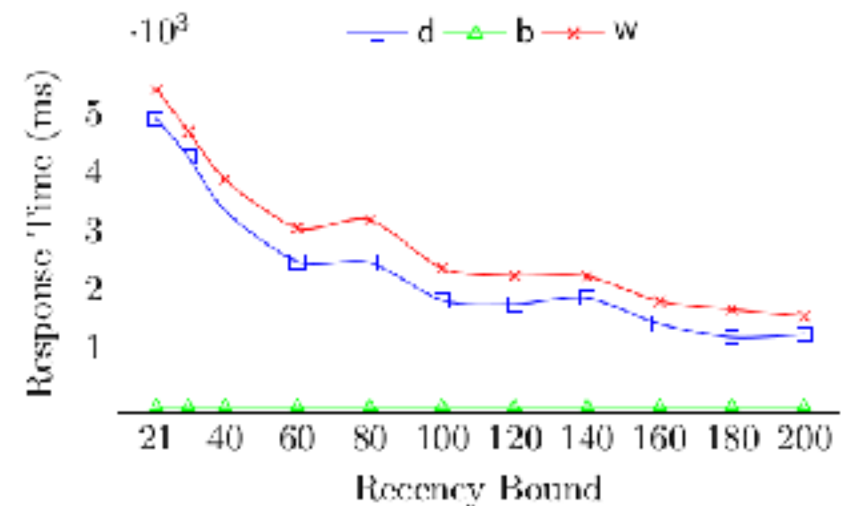
As the recency bound increases,
the coordination overhead and response time decrease.

Bank account

(a)

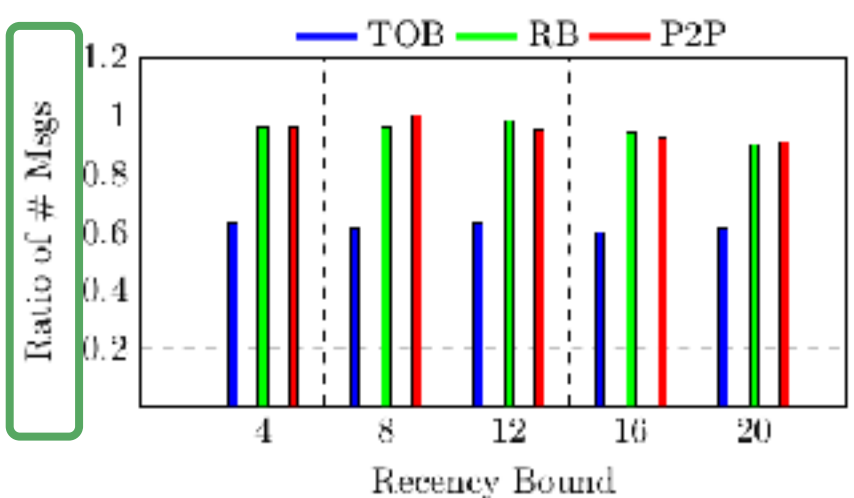


(b)

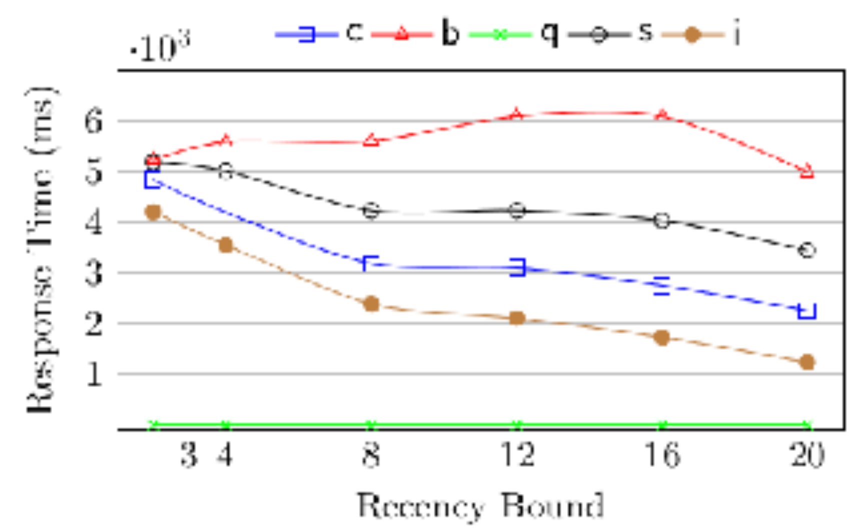


Movie booking

(c)

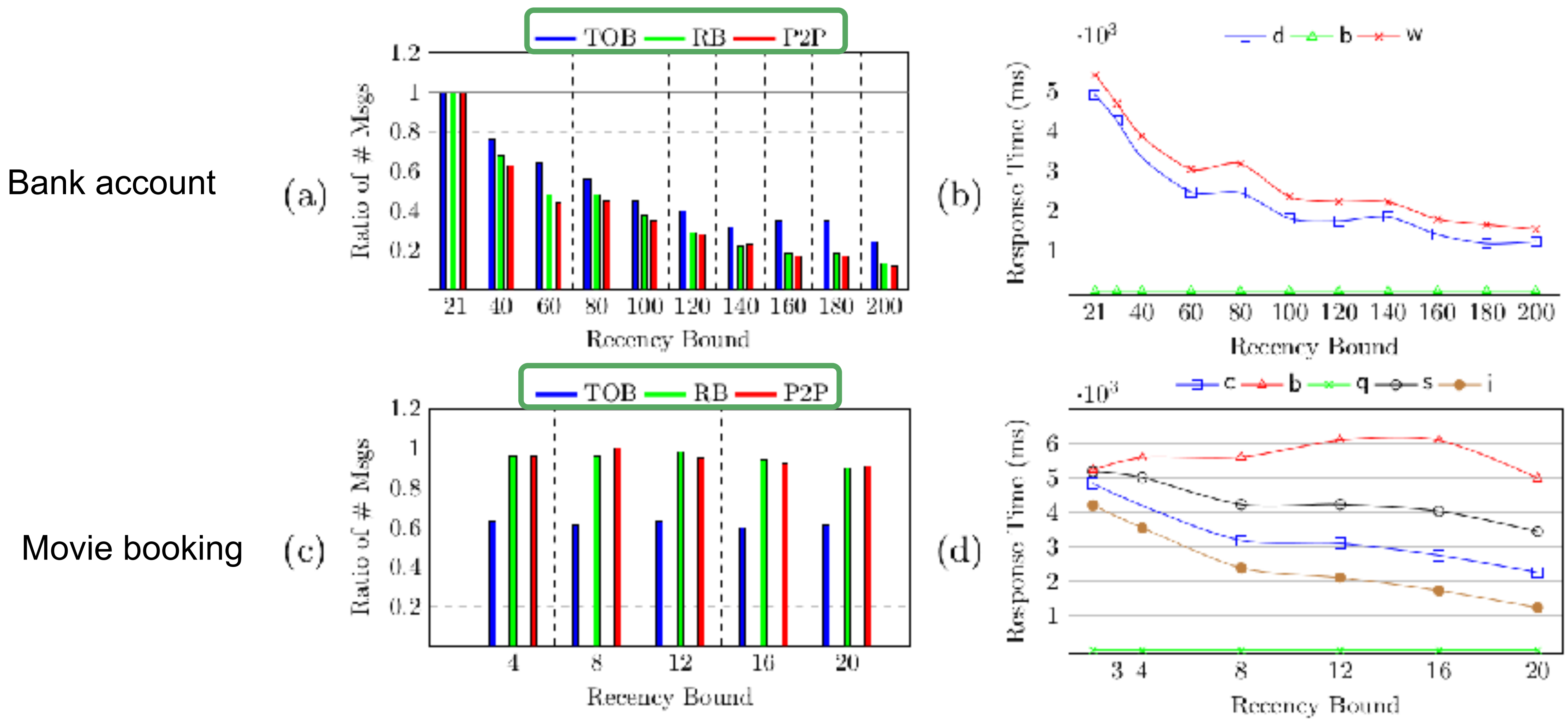


(d)



Experimental Results

As the recency bound increases,
the coordination overhead and response time decrease.

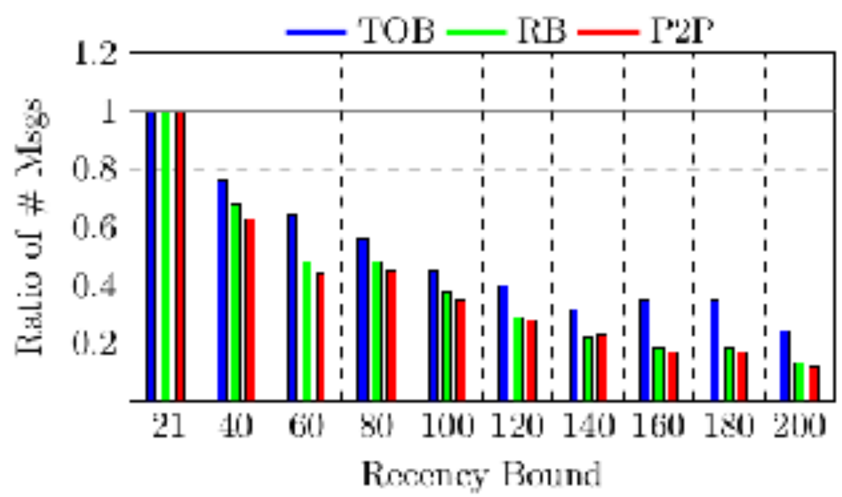


Experimental Results

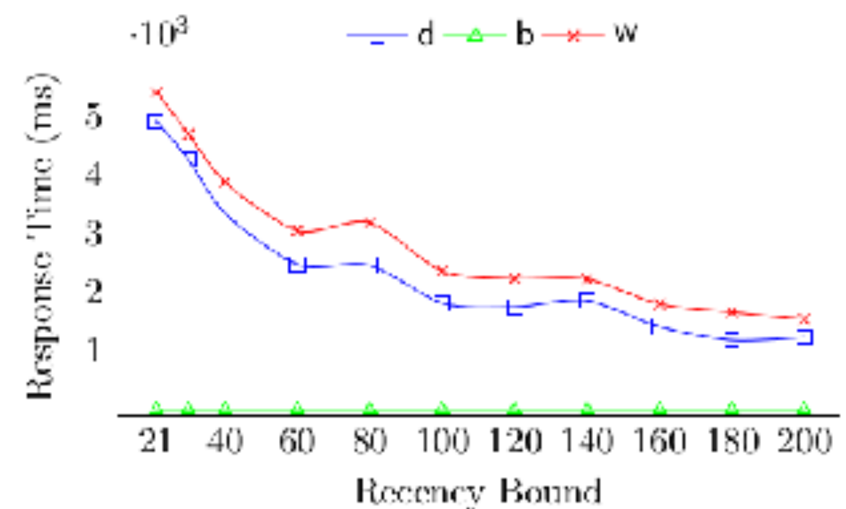
As the recency bound increases,
the coordination overhead and response time decrease.

Bank account

(a)

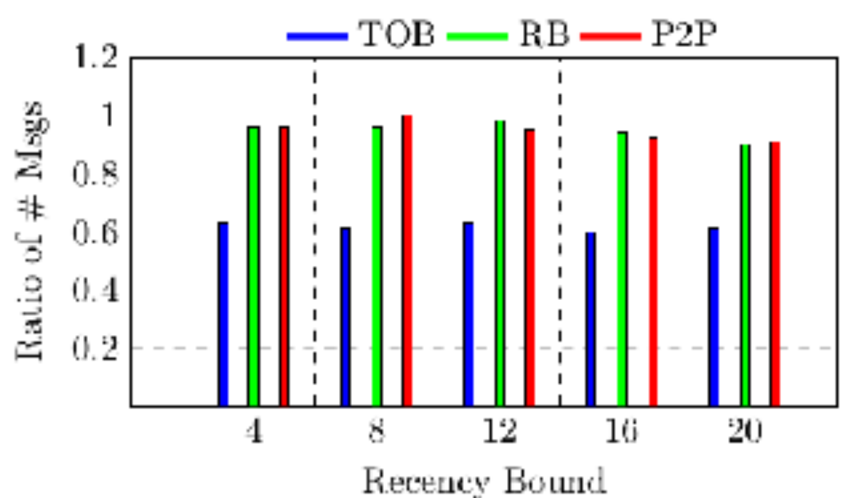


(b)

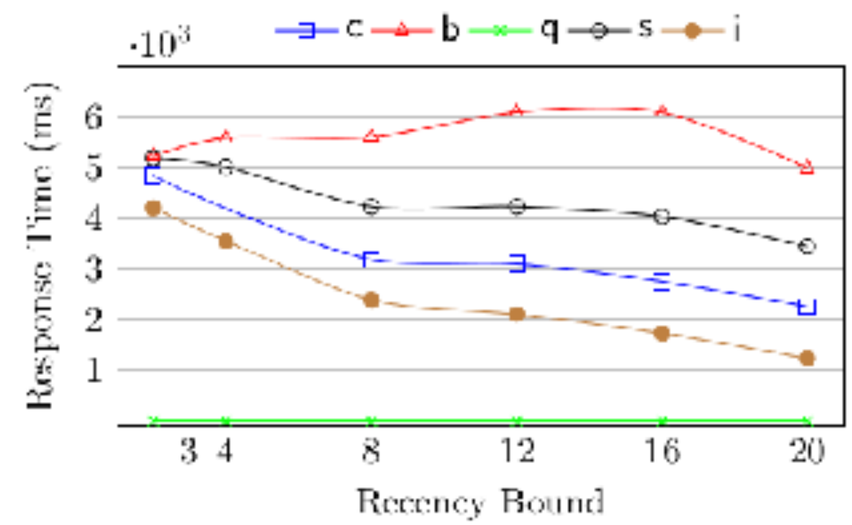


Movie booking

(c)



(d)

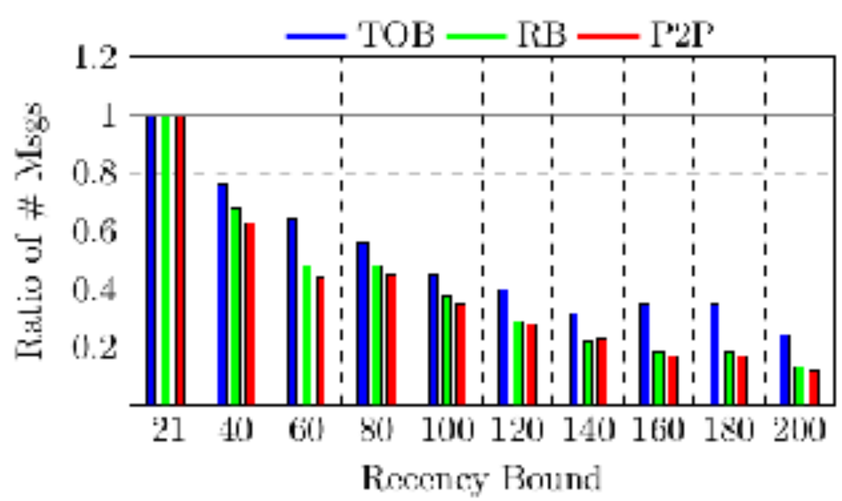


Experimental Results

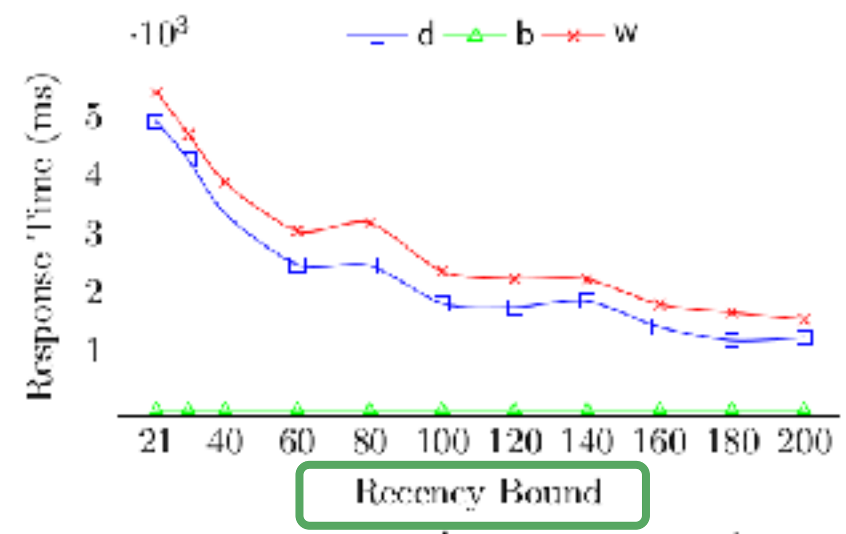
As the recency bound increases,
the coordination overhead and response time decrease.

Bank account

(a)

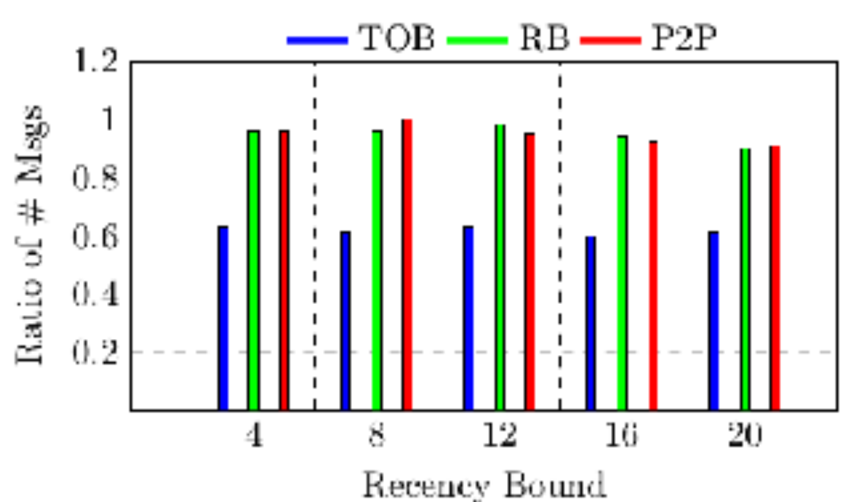


(b)

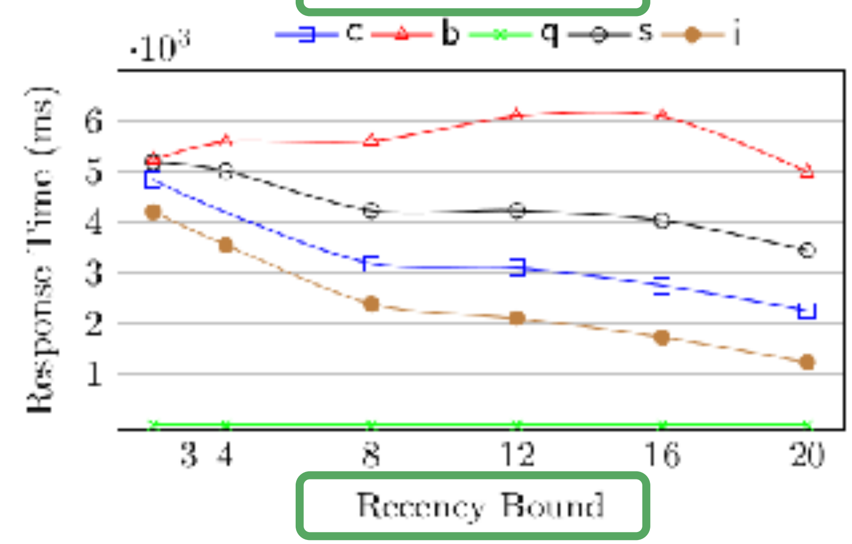


Movie booking

(c)



(d)

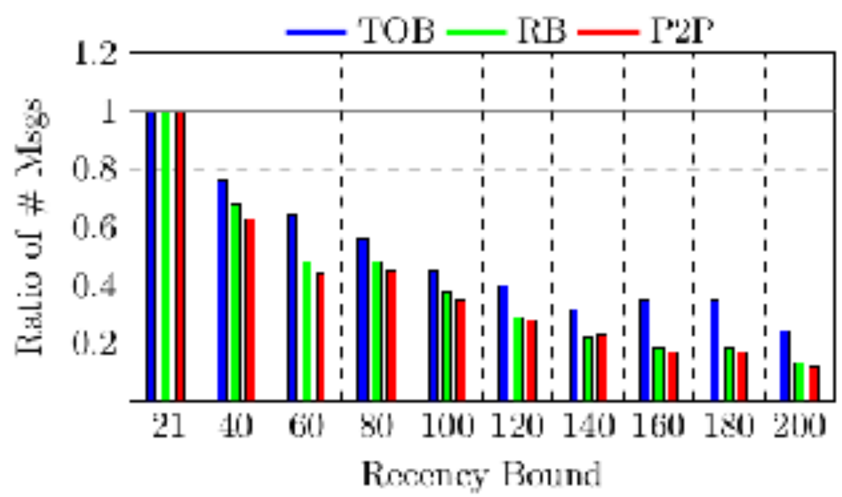


Experimental Results

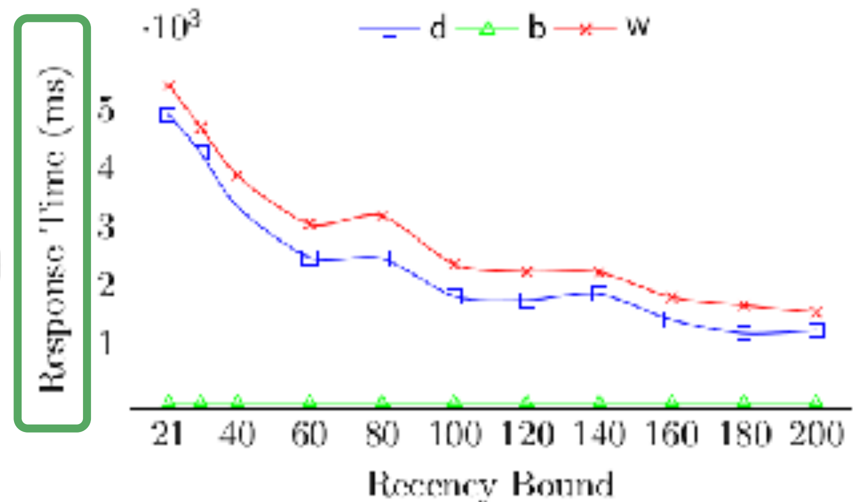
As the recency bound increases,
the coordination overhead and response time decrease.

Bank account

(a)

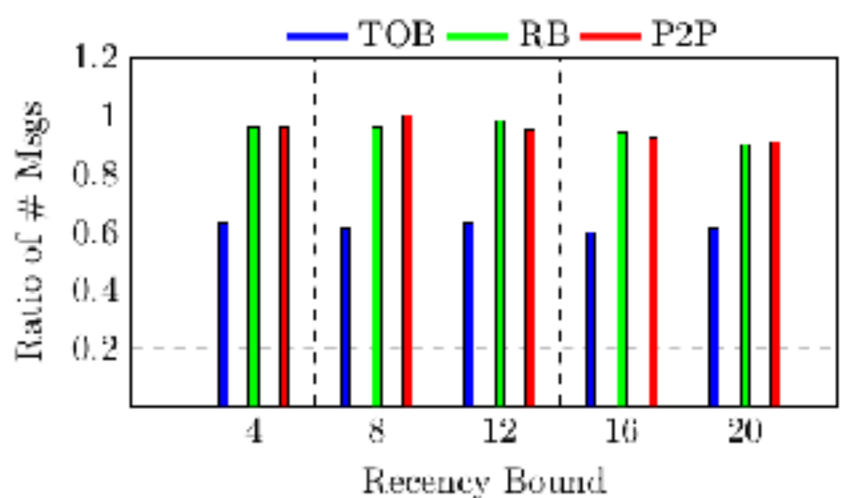


(b)

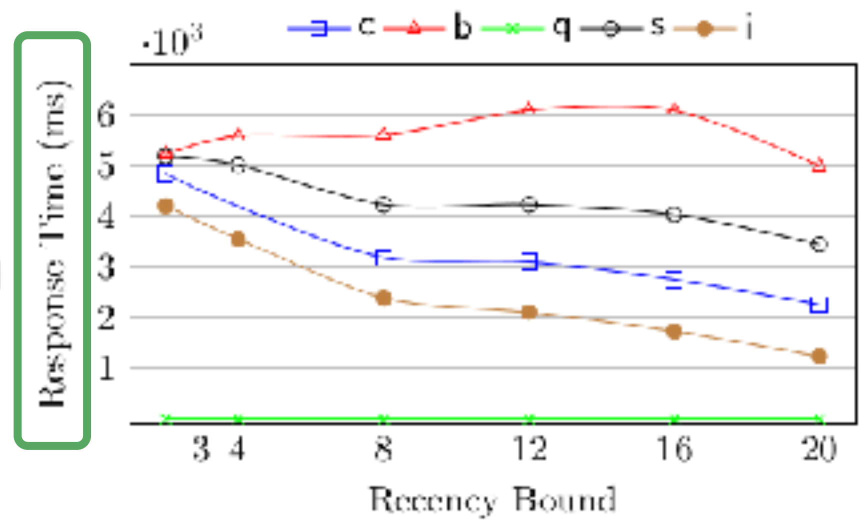


Movie booking

(c)



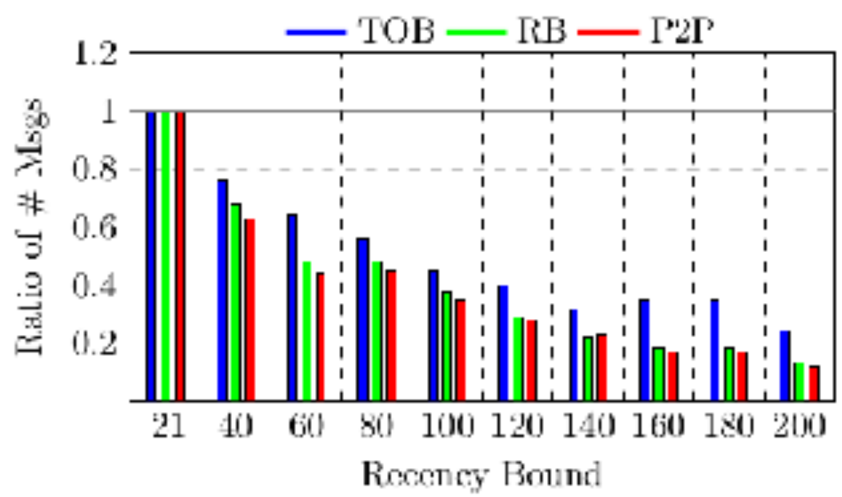
(d)



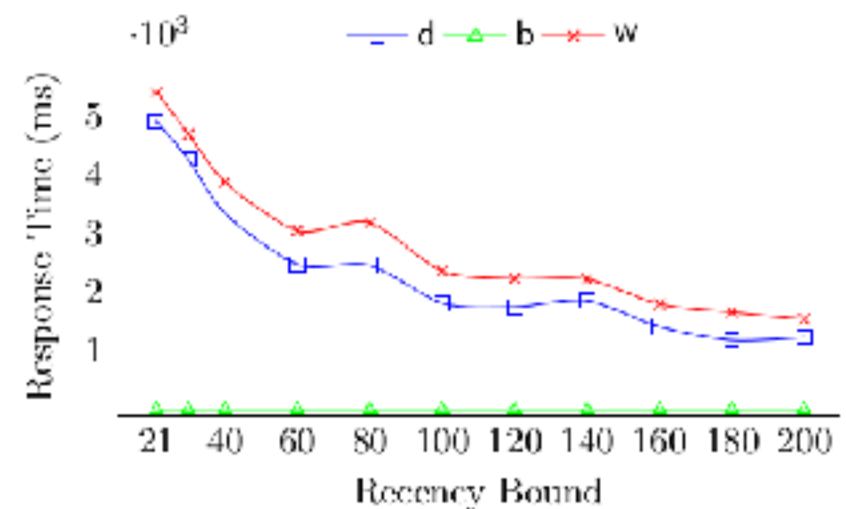
Experimental Results

As the recency bound increases,
the coordination overhead and response time decrease.

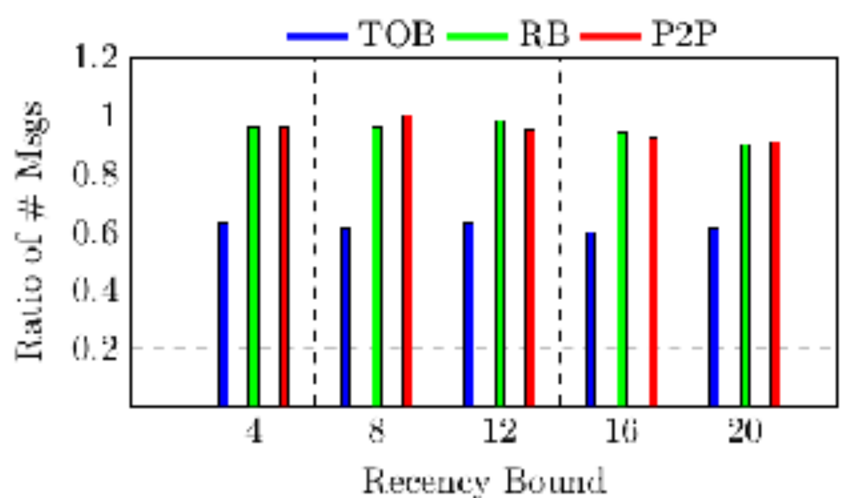
Bank account (a)



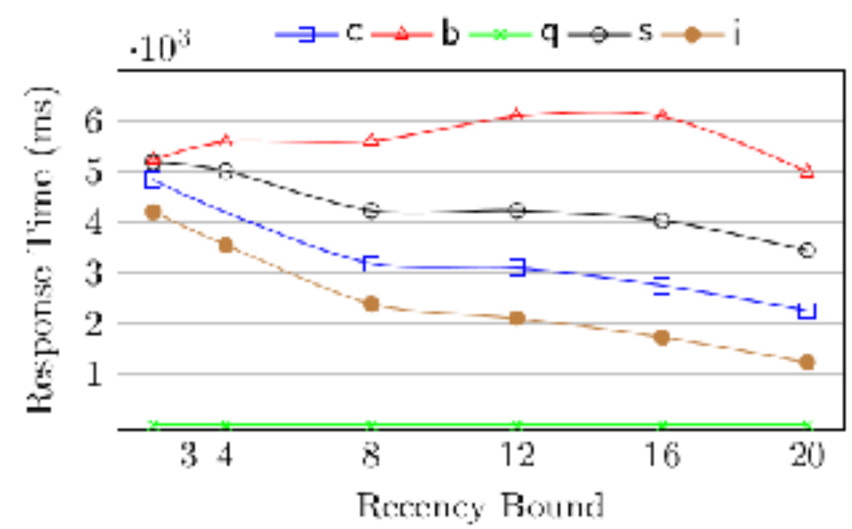
(b)



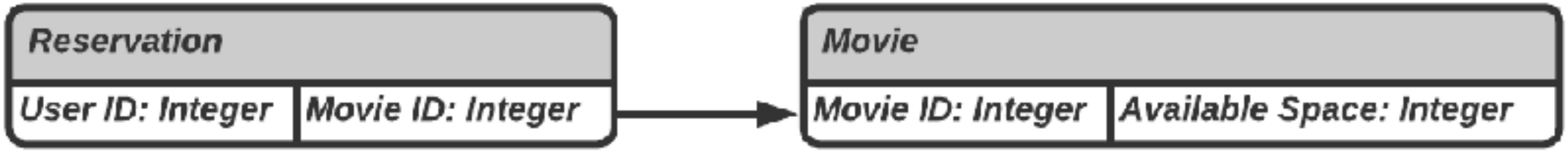
Movie booking (c)



(d)



Movie Booking use-case



Class MovieBooking

$\Sigma := \text{let } rs := \text{Set } \mathbb{N} \times \mathbb{N} \text{ in}$ $\triangleright$ Reservation: user identifier and movie identifier

$\quad \text{let } ms := \text{Set } \mathbb{N} \times \mathbb{N} \text{ in}$ $\triangleright$ Movie: movie identifier and available space

$\quad \langle rs, ms \rangle$

$\mathcal{I} := \lambda \langle rs, ms \rangle.\ \text{unique}\,(ms, \lambda \langle m, a \rangle.\ m) \wedge$

$\quad \text{refIntegrity}\,(rs, \lambda \langle u, m \rangle.\ m, ms, \lambda \langle m, a \rangle.\ m) \wedge$

$\quad \text{rowIntegrity}\,(ms, \lambda \langle m, a \rangle.\ a \geq 0)$

$\text{book}(\langle u, m \rangle) := 0\ \lambda \langle rs, ms \rangle.$

$\quad \langle \langle u, m \rangle \notin rs, \quad \langle rs \cup \langle u, m \rangle,\ \mathcal{U}_{\lambda \langle m', a \rangle.\ \langle m' = m, \langle m, a-1 \rangle \rangle}\ ms \rangle, \quad \bot \rangle$

$\text{cancelBook}(\langle u, m \rangle) := 0\ \lambda \langle rs, ms \rangle.$

$\quad \langle \text{True}, \quad \langle rs \setminus \langle u, m \rangle,\ \mathcal{U}_{\lambda \langle m', a \rangle.\ \langle m' = m, \langle m, a+1 \rangle \rangle}\ ms \rangle, \quad \bot \rangle$

$\text{offScreen}(m) := 0\ \lambda \langle rs, ms \rangle.$

$\quad \langle \text{True}, \quad \langle rs,\ ms \setminus \sigma_{\lambda \langle m', a \rangle.\ m' = m}\ ms \rangle, \quad \bot \rangle$

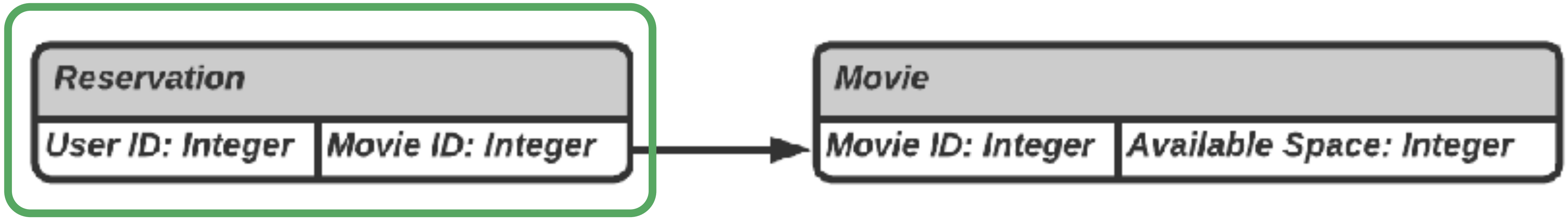
$\text{specialReserve}(\langle m, n \rangle) := 0\ \lambda \langle rs, ms \rangle.$

$\quad \langle n > 0, \quad \langle rs,\ \mathcal{U}_{\lambda \langle m', a \rangle.\ \langle m' = m, \langle m, a-n \rangle \rangle}\ ms \rangle, \quad \bot \rangle$

$\text{increaseSpace}(\langle m, n \rangle) := 0\ \lambda \langle rs, ms \rangle.$

$\quad \langle n > 0, \quad \langle rs,\ \mathcal{U}_{\lambda \langle m', a \rangle.\ \langle m' = m, \langle m, a+n \rangle \rangle}\ ms \rangle, \quad \bot \rangle$

Movie Booking use-case



Class MovieBooking

$$\Sigma := \text{let } rs := \text{Set } \mathbb{N} \times \mathbb{N} \text{ in} \quad \triangleright \text{Reservation: user identifier and movie identifier}$$
$$\text{let } ms := \text{Set } \mathbb{N} \times \mathbb{N} \text{ in} \quad \triangleright \text{Movie: movie identifier and available space}$$
$$\langle rs, ms \rangle$$

$$\mathcal{I} := \lambda \langle rs, ms \rangle .\ \text{unique}\,(ms, \lambda \langle m, a \rangle .\ m) \wedge$$
$$\text{refIntegrity}\,(rs, \lambda \langle u, m \rangle .\ m, ms, \lambda \langle m, a \rangle .\ m) \wedge$$
$$\text{rowIntegrity}\,(ms, \lambda \langle m, a \rangle .\ a \geq 0)$$

$$\text{book}(\langle u, m \rangle) := 0\ \lambda \langle rs, ms \rangle .$$
$$\langle \langle u, m \rangle \notin rs, \quad \langle rs \cup \langle u, m \rangle, \ \mathcal{U}_{\lambda \langle m', a \rangle .\ \langle m' = m, \langle m, a-1 \rangle \rangle}\, ms \rangle, \quad \bot \rangle$$

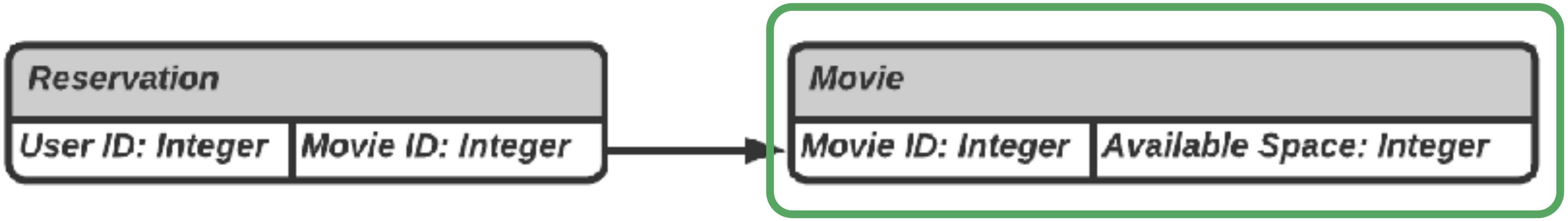
$$\text{cancelBook}(\langle u, m \rangle) := 0\ \lambda \langle rs, ms \rangle .$$
$$\langle \text{True}, \quad \langle rs \setminus \langle u, m \rangle, \ \mathcal{U}_{\lambda \langle m', a \rangle .\ \langle m' = m, \langle m, a+1 \rangle \rangle}\, ms \rangle, \quad \bot \rangle$$

$$\text{offScreen}(m) := 0\ \lambda \langle rs, ms \rangle .$$
$$\langle \text{True}, \quad \langle rs, \ ms \setminus \sigma_{\lambda \langle m', a \rangle .\ m' = m}\, ms \rangle, \quad \bot \rangle$$

$$\text{specialReserve}(\langle m, n \rangle) := 0\ \lambda \langle rs, ms \rangle .$$
$$\langle n > 0, \quad \langle rs, \ \mathcal{U}_{\lambda \langle m', a \rangle .\ \langle m' = m, \langle m, a-n \rangle \rangle}\, ms \rangle, \quad \bot \rangle$$

$$\text{increaseSpace}(\langle m, n \rangle) := 0\ \lambda \langle rs, ms \rangle .$$
$$\langle n > 0, \quad \langle rs, \ \mathcal{U}_{\lambda \langle m', a \rangle .\ \langle m' = m, \langle m, a+n \rangle \rangle}\, ms \rangle, \quad \bot \rangle$$

Movie Booking use-case



Class MovieBooking

$\Sigma := \text{let } rs := \text{Set } \mathbb{N} \times \mathbb{N} \text{ in}$ $\triangleright$ Reservation: user identifier and movie identifier

$\quad \text{let } ms := \text{Set } \mathbb{N} \times \mathbb{N} \text{ in}$ $\triangleright$ Movie: movie identifier and available space

$\quad \langle rs, ms \rangle$

$\mathcal{I} := \lambda\langle rs, ms\rangle.\ \text{unique}\,(ms, \lambda\langle m, a\rangle.\ m) \wedge$

$\qquad \text{refIntegrity}\,(rs, \lambda\langle u, m\rangle.\ m, ms, \lambda\langle m, a\rangle.\ m) \wedge$

$\qquad \text{rowIntegrity}\,(ms, \lambda\langle m, a\rangle.\ a \geq 0)$

$\text{book}(\langle u, m\rangle) := 0\ \lambda\langle rs, ms\rangle.$

$\quad \langle \langle u, m\rangle \notin rs,\quad \langle rs \cup \langle u, m\rangle,\ \mathcal{U}_{\lambda\langle m', a\rangle.\,\langle m'=m, \langle m, a-1\rangle\rangle}\, ms\rangle,\quad \bot\rangle$

$\text{cancelBook}(\langle u, m\rangle) := 0\ \lambda\langle rs, ms\rangle.$

$\quad \langle \text{True},\quad \langle rs \setminus \langle u, m\rangle,\ \mathcal{U}_{\lambda\langle m', a\rangle.\,\langle m'=m, \langle m, a+1\rangle\rangle}\, ms\rangle,\quad \bot\rangle$

$\text{offScreen}(m) := 0\ \lambda\langle rs, ms\rangle.$

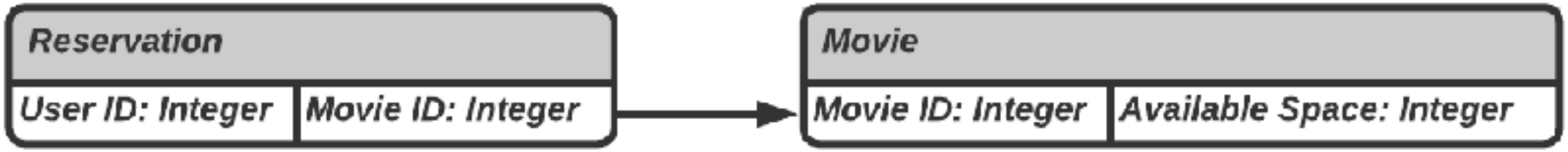
$\quad \langle \text{True},\quad \langle rs,\ ms \setminus \sigma_{\lambda\langle m', a\rangle.\, m'=m}\, ms\rangle,\quad \bot\rangle$

$\text{specialReserve}(\langle m, n\rangle) := 0\ \lambda\langle rs, ms\rangle.$

$\quad \langle n > 0,\quad \langle rs,\ \mathcal{U}_{\lambda\langle m', a\rangle.\,\langle m'=m, \langle m, a-n\rangle\rangle}\, ms\rangle,\quad \bot\rangle$

$\text{increaseSpace}(\langle m, n\rangle) := 0\ \lambda\langle rs, ms\rangle.$

$\quad \langle n > 0,\quad \langle rs,\ \mathcal{U}_{\lambda\langle m', a\rangle.\,\langle m'=m, \langle m, a+n\rangle\rangle}\, ms\rangle,\quad \bot\rangle$



Class MovieBooking

$\Sigma := \mathsf{let}\ rs := \mathsf{Set}\ \mathbb{N} \times \mathbb{N}\ \mathsf{in}$ $\triangleright$ Reservation: user identifier and movie identifier

$\quad \mathsf{let}\ ms := \mathsf{Set}\ \mathbb{N} \times \mathbb{N}\ \mathsf{in}$ $\triangleright$ Movie: movie identifier and available space

$\quad \langle rs, ms \rangle$

$\mathcal{I} := \lambda \langle rs, ms \rangle.\ \mathsf{unique}\,(ms, \lambda \langle m, a \rangle.\ m) \wedge$

$\quad \mathsf{refIntegrity}\,(rs, \lambda \langle u, m \rangle.\ m, ms, \lambda \langle m, a \rangle.\ m) \wedge$

$\quad \mathsf{rowIntegrity}\,(ms, \lambda \langle m, a \rangle.\ a \geq 0)$

$\mathsf{book}(\langle u, m \rangle) := 0\ \lambda \langle rs, ms \rangle.$

$\quad \langle \langle u, m \rangle \notin rs, \quad \langle rs \cup \langle u, m \rangle,\ \mathcal{U}_{\lambda \langle m', a \rangle.\ \langle m' = m, \langle m, a-1 \rangle \rangle}\ ms \rangle, \quad \bot \rangle$

$\mathsf{cancelBook}(\langle u, m \rangle) := 0\ \lambda \langle rs, ms \rangle.$

$\quad \langle \mathsf{True}, \quad \langle rs \setminus \langle u, m \rangle,\ \mathcal{U}_{\lambda \langle m', a \rangle.\ \langle m' = m, \langle m, a+1 \rangle \rangle}\ ms \rangle, \quad \bot \rangle$

$\mathsf{offScreen}(m) := 0\ \lambda \langle rs, ms \rangle.$

$\quad \langle \mathsf{True}, \quad \langle rs,\ ms \setminus \sigma_{\lambda \langle m', a \rangle.\ m' = m}\ ms \rangle, \quad \bot \rangle$

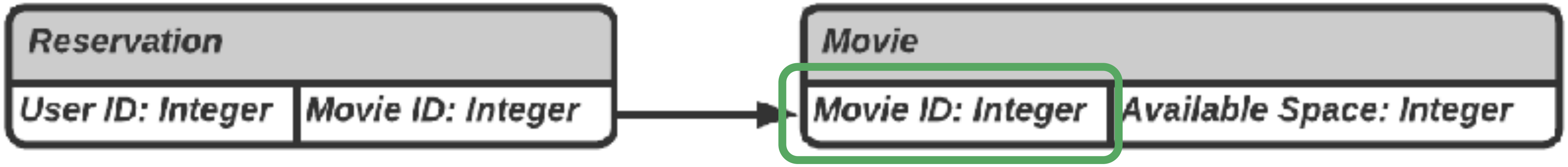
$\mathsf{specialReserve}(\langle m, n \rangle) := 0\ \lambda \langle rs, ms \rangle.$

$\quad \langle n > 0, \quad \langle rs,\ \mathcal{U}_{\lambda \langle m', a \rangle.\ \langle m' = m, \langle m, a-n \rangle \rangle}\ ms \rangle, \quad \bot \rangle$

$\mathsf{increaseSpace}(\langle m, n \rangle) := 0\ \lambda \langle rs, ms \rangle.$

$\quad \langle n > 0, \quad \langle rs,\ \mathcal{U}_{\lambda \langle m', a \rangle.\ \langle m' = m, \langle m, a+n \rangle \rangle}\ ms \rangle, \quad \bot \rangle$

Movie Booking use-case



Class MovieBooking

$\Sigma :=$ let $rs :=$ Set $\mathbb{N} \times \mathbb{N}$ in $\quad \triangleright$ Reservation: user identifier and movie identifier

let $ms :=$ Set $\mathbb{N} \times \mathbb{N}$ in $\quad \triangleright$ Movie: movie identifier and available space

$\langle rs, ms \rangle$

$\mathcal{I} := \lambda \langle rs, ms \rangle.$ unique $(ms, \lambda \langle m, a \rangle.\ m) \wedge$

refIntegrity $(rs, \lambda \langle u, m \rangle.\ m, ms, \lambda \langle m, a \rangle.\ m) \wedge$

rowIntegrity $(ms, \lambda \langle m, a \rangle.\ a \geq 0)$

book$(\langle u, m \rangle) := 0\ \lambda \langle rs, ms \rangle.$

$\langle \langle u, m \rangle \notin rs, \quad \langle rs \cup \langle u, m \rangle, \mathcal{U}_{\lambda \langle m', a \rangle.\ \langle m' = m, \langle m, a-1 \rangle \rangle}\ ms \rangle, \quad \bot \rangle$

cancelBook$(\langle u, m \rangle) := 0\ \lambda \langle rs, ms \rangle.$

$\langle$True, $\quad \langle rs \setminus \langle u, m \rangle, \mathcal{U}_{\lambda \langle m', a \rangle.\ \langle m' = m, \langle m, a+1 \rangle \rangle}\ ms \rangle, \quad \bot \rangle$

offScreen$(m) := 0\ \lambda \langle rs, ms \rangle.$

$\langle$True, $\quad \langle rs,\ ms \setminus \sigma_{\lambda \langle m', a \rangle.\ m' = m}\ ms \rangle, \quad \bot \rangle$

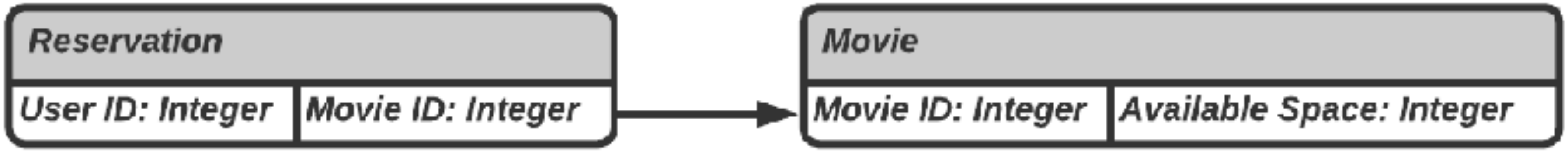
specialReserve$(\langle m, n \rangle) := 0\ \lambda \langle rs, ms \rangle.$

$\langle n > 0, \quad \langle rs,\ \mathcal{U}_{\lambda \langle m', a \rangle.\ \langle m' = m, \langle m, a-n \rangle \rangle}\ ms \rangle, \quad \bot \rangle$

increaseSpace$(\langle m, n \rangle) := 0\ \lambda \langle rs, ms \rangle.$

$\langle n > 0, \quad \langle rs,\ \mathcal{U}_{\lambda \langle m', a \rangle.\ \langle m' = m, \langle m, a+n \rangle \rangle}\ ms \rangle, \quad \bot \rangle$

Movie Booking use-case



Class MovieBooking

$\Sigma :=$ let $rs :=$ Set $\mathbb{N} \times \mathbb{N}$ in ▷ Reservation: user identifier and movie identifier

let $ms :=$ Set $\mathbb{N} \times \mathbb{N}$ in ▷ Movie: movie identifier and available space

$\langle rs, ms \rangle$

$\mathcal{I} := \lambda \langle rs, ms \rangle.$ unique $(ms, \lambda \langle m, a \rangle.\ m) \wedge$

reflntegrity $(rs, \lambda \langle u, m \rangle.\ m, ms, \lambda \langle m, a \rangle.\ m) \wedge$

rowIntegrity $(ms, \lambda \langle m, a \rangle.\ a \geq 0)$

book$(\langle u, m \rangle) := 0\ \lambda \langle rs, ms \rangle.$

$\langle \langle u, m \rangle \notin rs, \quad \langle rs \cup \langle u, m \rangle, \mathcal{U}_{\lambda \langle m', a \rangle.\ \langle m' = m, \langle m, a-1 \rangle \rangle}\ ms \rangle, \quad \bot \rangle$

cancelBook$(\langle u, m \rangle) := 0\ \lambda \langle rs, ms \rangle.$

$\langle$True, $\quad \langle rs \setminus \langle u, m \rangle, \mathcal{U}_{\lambda \langle m', a \rangle.\ \langle m' = m, \langle m, a+1 \rangle \rangle}\ ms \rangle, \quad \bot \rangle$

offScreen$(m) := 0\ \lambda \langle rs, ms \rangle.$

$\langle$True, $\quad \langle rs,\ ms \setminus \sigma_{\lambda \langle m', a \rangle.\ m' = m}\ ms \rangle, \quad \bot \rangle$

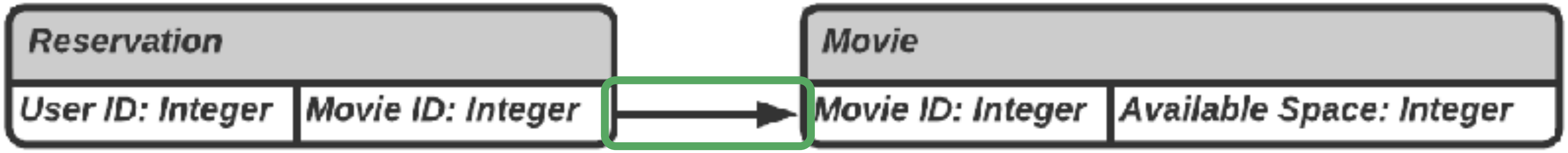
specialReserve$(\langle m, n \rangle) := 0\ \lambda \langle rs, ms \rangle.$

$\langle n > 0, \quad \langle rs,\ \mathcal{U}_{\lambda \langle m', a \rangle.\ \langle m' = m, \langle m, a-n \rangle \rangle}\ ms \rangle, \quad \bot \rangle$

increaseSpace$(\langle m, n \rangle) := 0\ \lambda \langle rs, ms \rangle.$

$\langle n > 0, \quad \langle rs,\ \mathcal{U}_{\lambda \langle m', a \rangle.\ \langle m' = m, \langle m, a+n \rangle \rangle}\ ms \rangle, \quad \bot \rangle$

Movie Booking use-case



Class MovieBooking

$\Sigma := \mathsf{let}\ rs := \mathsf{Set}\ \mathbb{N} \times \mathbb{N}\ \mathsf{in}$ $\triangleright$ Reservation: user identifier and movie identifier

$\quad \mathsf{let}\ ms := \mathsf{Set}\ \mathbb{N} \times \mathbb{N}\ \mathsf{in}$ $\triangleright$ Movie: movie identifier and available space

$\quad \langle rs, ms \rangle$

$\mathcal{I} := \lambda\langle rs, ms\rangle.\ \mathsf{unique}\,(ms, \lambda\langle m, a\rangle.\ m) \wedge$

$\quad \mathsf{refIntegrity}\,(rs, \lambda\langle u, m\rangle.\ m, ms, \lambda\langle m, a\rangle.\ m) \wedge$

$\quad \mathsf{rowIntegrity}\,(ms, \lambda\langle m, a\rangle.\ a \geq 0)$

$\mathsf{book}(\langle u, m\rangle) := 0\ \lambda\langle rs, ms\rangle.$

$\quad \langle \langle u, m\rangle \notin rs, \quad \langle rs \cup \langle u, m\rangle, \mathcal{U}_{\lambda\langle m', a\rangle.\ \langle m' = m, \langle m, a-1\rangle\rangle}\ ms\rangle, \quad \bot\rangle$

$\mathsf{cancelBook}(\langle u, m\rangle) := 0\ \lambda\langle rs, ms\rangle.$

$\quad \langle \mathsf{True}, \quad \langle rs \setminus \langle u, m\rangle, \mathcal{U}_{\lambda\langle m', a\rangle.\ \langle m' = m, \langle m, a+1\rangle\rangle}\ ms\rangle, \quad \bot\rangle$

$\mathsf{offScreen}(m) := 0\ \lambda\langle rs, ms\rangle.$

$\quad \langle \mathsf{True}, \quad \langle rs,\ ms \setminus \sigma_{\lambda\langle m', a\rangle.\ m' = m}\ ms\rangle, \quad \bot\rangle$

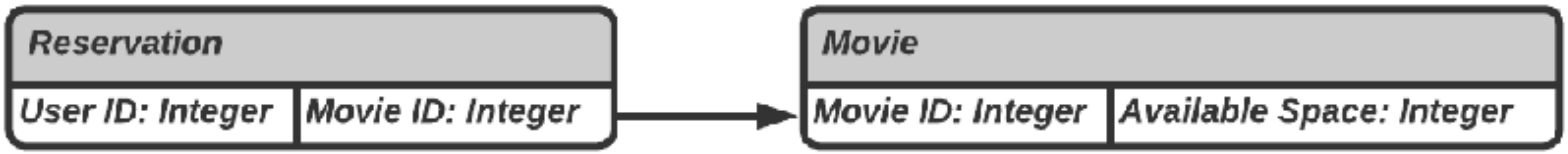
$\mathsf{specialReserve}(\langle m, n\rangle) := 0\ \lambda\langle rs, ms\rangle.$

$\quad \langle n > 0, \quad \langle rs,\ \mathcal{U}_{\lambda\langle m', a\rangle.\ \langle m' = m, \langle m, a-n\rangle\rangle}\ ms\rangle, \quad \bot\rangle$

$\mathsf{increaseSpace}(\langle m, n\rangle) := 0\ \lambda\langle rs, ms\rangle.$

$\quad \langle n > 0, \quad \langle rs,\ \mathcal{U}_{\lambda\langle m', a\rangle.\ \langle m' = m, \langle m, a+n\rangle\rangle}\ ms\rangle, \quad \bot\rangle$

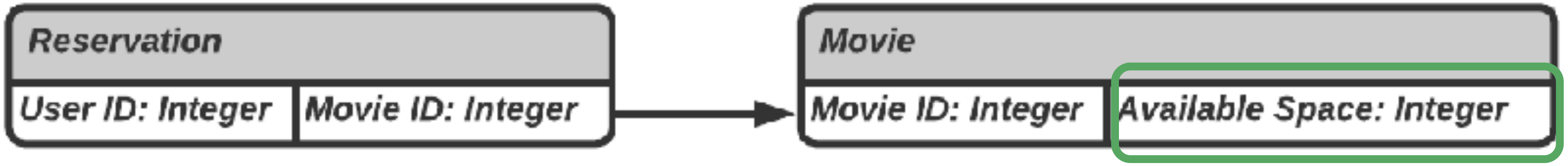
Movie Booking use-case



Class MovieBooking

$$
\begin{aligned}
&\Sigma := \text{let } rs := \text{Set } \mathbb{N} \times \mathbb{N} \text{ in} \quad \triangleright \text{Reservation: user identifier and movie identifier} \\
&\qquad\quad \text{let } ms := \text{Set } \mathbb{N} \times \mathbb{N} \text{ in} \quad \triangleright \text{Movie: movie identifier and available space} \\
&\qquad\quad \langle rs, ms \rangle \\
&\mathcal{I} := \lambda\langle rs, ms\rangle.\ \text{unique}\,(ms, \lambda\langle m, a\rangle.\ m) \wedge \\
&\qquad\qquad\qquad \text{refIntegrity}\,(rs, \lambda\langle u, m\rangle.\ m, ms, \lambda\langle m, a\rangle.\ m) \wedge \\
&\qquad\qquad\qquad \text{rowIntegrity}\,(ms, \lambda\langle m, a\rangle.\ a \geq 0) \\
&\text{book}(\langle u, m\rangle) := 0\ \lambda\langle rs, ms\rangle. \\
&\quad \langle \langle u, m\rangle \notin rs, \quad \langle rs \cup \langle u, m\rangle,\ \mathcal{U}_{\lambda\langle m', a\rangle.\ \langle m'=m, \langle m, a-1\rangle\rangle}\ ms\rangle, \quad \bot\rangle \\
&\text{cancelBook}(\langle u, m\rangle) := 0\ \lambda\langle rs, ms\rangle. \\
&\quad \langle \text{True}, \quad \langle rs \setminus \langle u, m\rangle,\ \mathcal{U}_{\lambda\langle m', a\rangle.\ \langle m'=m, \langle m, a+1\rangle\rangle}\ ms\rangle, \quad \bot\rangle \\
&\text{offScreen}(m) := 0\ \lambda\langle rs, ms\rangle. \\
&\quad \langle \text{True}, \quad \langle rs,\ ms \setminus \sigma_{\lambda\langle m', a\rangle.\ m'=m}\ ms\rangle, \quad \bot\rangle \\
&\text{specialReserve}(\langle m, n\rangle) := 0\ \lambda\langle rs, ms\rangle. \\
&\quad \langle n > 0, \quad \langle rs,\ \mathcal{U}_{\lambda\langle m', a\rangle.\ \langle m'=m, \langle m, a-n\rangle\rangle}\ ms\rangle, \quad \bot\rangle \\
&\text{increaseSpace}(\langle m, n\rangle) := 0\ \lambda\langle rs, ms\rangle. \\
&\quad \langle n > 0, \quad \langle rs,\ \mathcal{U}_{\lambda\langle m', a\rangle.\ \langle m'=m, \langle m, a+n\rangle\rangle}\ ms\rangle, \quad \bot\rangle
\end{aligned}
$$

Movie Booking use-case



Class MovieBooking

$\Sigma :=$ let $rs :=$ Set $\mathbb{N} \times \mathbb{N}$ in ▷ Reservation: user identifier and movie identifier

let $ms :=$ Set $\mathbb{N} \times \mathbb{N}$ in ▷ Movie: movie identifier and available space

$\langle rs, ms \rangle$

$\mathcal{I} := \lambda\langle rs, ms\rangle.$ unique $(ms, \lambda\langle m, a\rangle.\ m) \wedge$

refIntegrity $(rs, \lambda\langle u, m\rangle.\ m, ms, \lambda\langle m, a\rangle.\ m) \wedge$

rowIntegrity $(ms, \lambda\langle m, a\rangle.\ a \geq 0)$

book$(\langle u, m\rangle) := 0\ \lambda\langle rs, ms\rangle.$

$\langle \langle u, m\rangle \notin rs, \quad \langle rs \cup \langle u, m\rangle, \mathcal{U}_{\lambda\langle m', a\rangle.\ \langle m'=m, \langle m, a-1\rangle\rangle}\ ms\rangle, \quad \bot\rangle$

cancelBook$(\langle u, m\rangle) := 0\ \lambda\langle rs, ms\rangle.$

$\langle \mathsf{True}, \quad \langle rs \setminus \langle u, m\rangle, \mathcal{U}_{\lambda\langle m', a\rangle.\ \langle m'=m, \langle m, a+1\rangle\rangle}\ ms\rangle, \quad \bot\rangle$

offScreen$(m) := 0\ \lambda\langle rs, ms\rangle.$

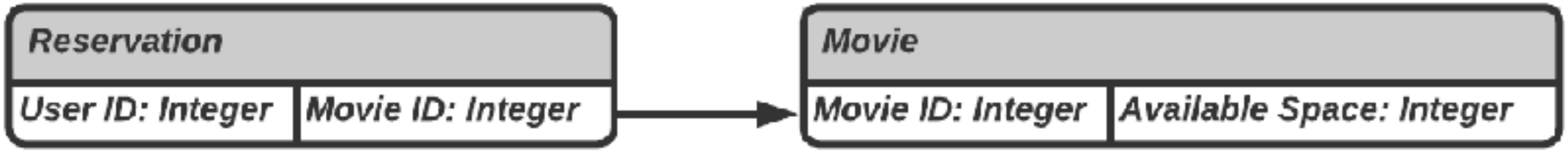
$\langle \mathsf{True}, \quad \langle rs,\ ms \setminus \sigma_{\lambda\langle m', a\rangle.\ m'=m}\ ms\rangle, \quad \bot\rangle$

specialReserve$(\langle m, n\rangle) := 0\ \lambda\langle rs, ms\rangle.$

$\langle n > 0, \quad \langle rs,\ \mathcal{U}_{\lambda\langle m', a\rangle.\ \langle m'=m, \langle m, a-n\rangle\rangle}\ ms\rangle, \quad \bot\rangle$

increaseSpace$(\langle m, n\rangle) := 0\ \lambda\langle rs, ms\rangle.$

$\langle n > 0, \quad \langle rs,\ \mathcal{U}_{\lambda\langle m', a\rangle.\ \langle m'=m, \langle m, a+n\rangle\rangle}\ ms\rangle, \quad \bot\rangle$



Class MovieBooking

$\Sigma := \text{let } rs := \text{Set } \mathbb{N} \times \mathbb{N} \text{ in}$ $\triangleright$ Reservation: user identifier and movie identifier

$\quad \text{let } ms := \text{Set } \mathbb{N} \times \mathbb{N} \text{ in}$ $\triangleright$ Movie: movie identifier and available space

$\quad \langle rs, ms \rangle$

$\mathcal{I} := \lambda\langle rs, ms\rangle.\ \text{unique}\,(ms, \lambda\langle m, a\rangle.\ m) \wedge$

$\quad \text{refIntegrity}\,(rs, \lambda\langle u, m\rangle.\ m, ms, \lambda\langle m, a\rangle.\ m) \wedge$

$\quad \text{rowIntegrity}\,(ms, \lambda\langle m, a\rangle.\ a \geq 0)$

$\text{book}(\langle u, m\rangle) := 0\ \lambda\langle rs, ms\rangle.$

$\quad \langle \langle u, m\rangle \notin rs, \quad \langle rs \cup \langle u, m\rangle, \mathcal{U}_{\lambda\langle m', a\rangle.\,\langle m'=m, \langle m, a-1\rangle\rangle}\, ms\rangle, \quad \bot\rangle$

$\text{cancelBook}(\langle u, m\rangle) := 0\ \lambda\langle rs, ms\rangle.$

$\quad \langle \text{True}, \quad \langle rs \setminus \langle u, m\rangle, \mathcal{U}_{\lambda\langle m', a\rangle.\,\langle m'=m, \langle m, a+1\rangle\rangle}\, ms\rangle, \quad \bot\rangle$

$\text{offScreen}(m) := 0\ \lambda\langle rs, ms\rangle.$

$\quad \langle \text{True}, \quad \langle rs,\ ms \setminus \sigma_{\lambda\langle m', a\rangle.\, m'=m}\, ms\rangle, \quad \bot\rangle$

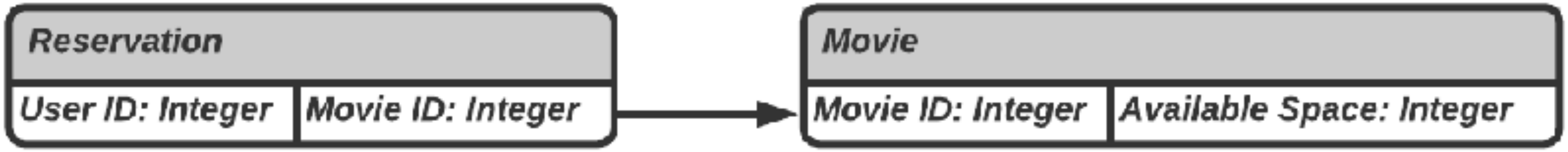
$\text{specialReserve}(\langle m, n\rangle) := 0\ \lambda\langle rs, ms\rangle.$

$\quad \langle n > 0, \quad \langle rs,\ \mathcal{U}_{\lambda\langle m', a\rangle.\,\langle m'=m, \langle m, a-n\rangle\rangle}\, ms\rangle, \quad \bot\rangle$

$\text{increaseSpace}(\langle m, n\rangle) := 0\ \lambda\langle rs, ms\rangle.$

$\quad \langle n > 0, \quad \langle rs,\ \mathcal{U}_{\lambda\langle m', a\rangle.\,\langle m'=m, \langle m, a+n\rangle\rangle}\, ms\rangle, \quad \bot\rangle$

Movie Booking use-case



Class MovieBooking

$\Sigma := \text{let } rs := \text{Set } \mathbb{N} \times \mathbb{N} \text{ in}$ ▷ Reservation: user identifier and movie identifier

$\quad \text{let } ms := \text{Set } \mathbb{N} \times \mathbb{N} \text{ in}$ ▷ Movie: movie identifier and available space

$\quad \langle rs, ms \rangle$

$\mathcal{I} := \lambda\langle rs, ms\rangle.\ \text{unique}\,(ms, \lambda\langle m, a\rangle.\ m) \wedge$

$\quad \text{refIntegrity}\,(rs, \lambda\langle u, m\rangle.\ m, ms, \lambda\langle m, a\rangle.\ m) \wedge$

$\quad \text{rowIntegrity}\,(ms, \lambda\langle m, a\rangle.\ a \geq 0)$

$\text{book}(\langle u, m\rangle) := 0\ \lambda\langle rs, ms\rangle.$

$\quad \langle \langle u, m\rangle \notin rs, \quad \langle rs \cup \langle u, m\rangle,\ \mathcal{U}_{\lambda\langle m', a\rangle.\ \langle m'=m, \langle m, a-1\rangle\rangle}\ ms\rangle, \quad \bot\rangle$

$\text{cancelBook}(\langle u, m\rangle) := 0\ \lambda\langle rs, ms\rangle.$

$\quad \langle \text{True}, \quad \langle rs \setminus \langle u, m\rangle,\ \mathcal{U}_{\lambda\langle m', a\rangle.\ \langle m'=m, \langle m, a+1\rangle\rangle}\ ms\rangle, \quad \bot\rangle$

$\text{offScreen}(m) := 0\ \lambda\langle rs, ms\rangle.$

$\quad \langle \text{True}, \quad \langle rs,\ ms \setminus \sigma_{\lambda\langle m', a\rangle.\ m'=m}\ ms\rangle, \quad \bot\rangle$

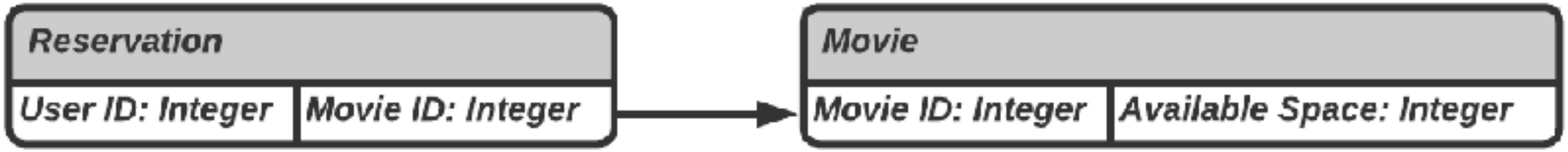
$\text{specialReserve}(\langle m, n\rangle) := 0\ \lambda\langle rs, ms\rangle.$

$\quad \langle n > 0, \quad \langle rs,\ \mathcal{U}_{\lambda\langle m', a\rangle.\ \langle m'=m, \langle m, a-n\rangle\rangle}\ ms\rangle, \quad \bot\rangle$

$\text{increaseSpace}(\langle m, n\rangle) := 0\ \lambda\langle rs, ms\rangle.$

$\quad \langle n > 0, \quad \langle rs,\ \mathcal{U}_{\lambda\langle m', a\rangle.\ \langle m'=m, \langle m, a+n\rangle\rangle}\ ms\rangle, \quad \bot\rangle$

Movie Booking use-case



Class MovieBooking

$\Sigma :=$ let $rs :=$ Set $\mathbb{N} \times \mathbb{N}$ in $\quad \triangleright$ Reservation: user identifier and movie identifier

let $ms :=$ Set $\mathbb{N} \times \mathbb{N}$ in $\quad \triangleright$ Movie: movie identifier and available space

$\langle rs, ms \rangle$

$\mathcal{I} := \lambda \langle rs, ms \rangle.$ unique $(ms, \lambda \langle m, a \rangle.\ m) \wedge$

refIntegrity $(rs, \lambda \langle u, m \rangle.\ m, ms, \lambda \langle m, a \rangle.\ m) \wedge$

rowIntegrity $(ms, \lambda \langle m, a \rangle.\ a \geq 0)$

book$(\langle u, m \rangle) := 0\ \lambda \langle rs, ms \rangle.$

$\langle \langle u, m \rangle \notin rs, \quad \langle rs \cup \langle u, m \rangle, \mathcal{U}_{\lambda \langle m', a \rangle.\ \langle m' = m, \langle m, a-1 \rangle \rangle}\ ms \rangle, \quad \bot \rangle$

cancelBook$(\langle u, m \rangle) := 0\ \lambda \langle rs, ms \rangle.$

$\langle$True, $\quad \langle rs \setminus \langle u, m \rangle, \mathcal{U}_{\lambda \langle m', a \rangle.\ \langle m' = m, \langle m, a+1 \rangle \rangle}\ ms \rangle, \quad \bot \rangle$

offScreen$(m) := 0\ \lambda \langle rs, ms \rangle.$

$\langle$True, $\quad \langle rs,\ ms \setminus \sigma_{\lambda \langle m', a \rangle.\ m' = m}\ ms \rangle, \quad \bot \rangle$

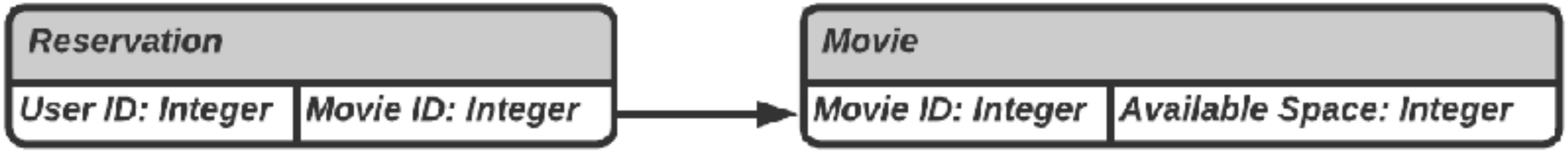
specialReserve$(\langle m, n \rangle) := 0\ \lambda \langle rs, ms \rangle.$

$\langle n > 0, \quad \langle rs, \mathcal{U}_{\lambda \langle m', a \rangle.\ \langle m' = m, \langle m, a-n \rangle \rangle}\ ms \rangle, \quad \bot \rangle$

increaseSpace$(\langle m, n \rangle) := 0\ \lambda \langle rs, ms \rangle.$

$\langle n > 0, \quad \langle rs, \mathcal{U}_{\lambda \langle m', a \rangle.\ \langle m' = m, \langle m, a+n \rangle \rangle}\ ms \rangle, \quad \bot \rangle$

Movie Booking use-case



Class MovieBooking

$\Sigma :=$ let $rs :=$ Set $\mathbb{N} \times \mathbb{N}$ in $\quad \triangleright$ Reservation: user identifier and movie identifier

$\quad$ let $ms :=$ Set $\mathbb{N} \times \mathbb{N}$ in $\quad \triangleright$ Movie: movie identifier and available space

$\quad \langle rs, ms \rangle$

$\mathcal{I} := \lambda \langle rs, ms \rangle.$ unique $(ms, \lambda \langle m, a \rangle.\ m) \wedge$

$\quad$ refIntegrity $(rs, \lambda \langle u, m \rangle.\ m, ms, \lambda \langle m, a \rangle.\ m) \wedge$

$\quad$ rowIntegrity $(ms, \lambda \langle m, a \rangle.\ a \geq 0)$

book$(\langle u, m \rangle) := 0\ \lambda \langle rs, ms \rangle.$

$\quad \langle \langle u, m \rangle \notin rs, \quad \langle rs \cup \langle u, m \rangle, \mathcal{U}_{\lambda \langle m', a \rangle.\ \langle m'=m, \langle m, a-1 \rangle \rangle}\ ms \rangle, \quad \bot \rangle$

cancelBook$(\langle u, m \rangle) := 0\ \lambda \langle rs, ms \rangle.$

$\quad \langle$True, $\quad \langle rs \setminus \langle u, m \rangle, \mathcal{U}_{\lambda \langle m', a \rangle.\ \langle m'=m, \langle m, a+1 \rangle \rangle}\ ms \rangle, \quad \bot \rangle$

offScreen$(m) := 0\ \lambda \langle rs, ms \rangle.$

$\quad \langle$True, $\quad \langle rs,\ ms \setminus \sigma_{\lambda \langle m', a \rangle.\ m'=m}\ ms \rangle, \quad \bot \rangle$

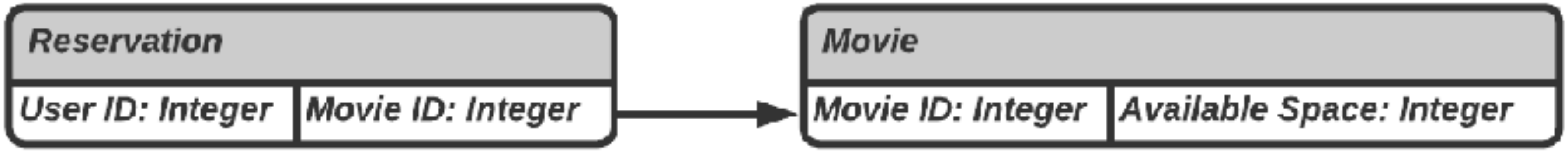
specialReserve$(\langle m, n \rangle) := 0\ \lambda \langle rs, ms \rangle.$

$\quad \langle n > 0, \quad \langle rs,\ \mathcal{U}_{\lambda \langle m', a \rangle.\ \langle m'=m, \langle m, a-n \rangle \rangle}\ ms \rangle, \quad \bot \rangle$

increaseSpace$(\langle m, n \rangle) := 0\ \lambda \langle rs, ms \rangle.$

$\quad \langle n > 0, \quad \langle rs,\ \mathcal{U}_{\lambda \langle m', a \rangle.\ \langle m'=m, \langle m, a+n \rangle \rangle}\ ms \rangle, \quad \bot \rangle$

Movie Booking use-case



Class MovieBooking

$\Sigma :=$ let $rs :=$ Set $\mathbb{N} \times \mathbb{N}$ in $\quad \triangleright$ Reservation: user identifier and movie identifier

let $ms :=$ Set $\mathbb{N} \times \mathbb{N}$ in $\quad \triangleright$ Movie: movie identifier and available space

$\langle rs, ms \rangle$

$\mathcal{I} := \lambda \langle rs, ms \rangle.$ unique $(ms, \lambda \langle m, a \rangle.\ m) \wedge$

reflntegrity $(rs, \lambda \langle u, m \rangle.\ m, ms, \lambda \langle m, a \rangle.\ m) \wedge$

rowIntegrity $(ms, \lambda \langle m, a \rangle.\ a \geq 0)$

book$(\langle u, m \rangle) := 0\ \lambda \langle rs, ms \rangle.$

$\langle \langle u, m \rangle \notin rs, \quad \langle rs \cup \langle u, m \rangle,\ \mathcal{U}_{\lambda \langle m', a \rangle.\ \langle m' = m, \langle m, a-1 \rangle \rangle}\ ms \rangle, \quad \bot \rangle$

cancelBook$(\langle u, m \rangle) := 0\ \lambda \langle rs, ms \rangle.$

$\langle$True, $\quad \langle rs \setminus \langle u, m \rangle,\ \mathcal{U}_{\lambda \langle m', a \rangle.\ \langle m' = m, \langle m, a+1 \rangle \rangle}\ ms \rangle, \quad \bot \rangle$

offScreen$(m) := 0\ \lambda \langle rs, ms \rangle.$

$\langle$True, $\quad \langle rs,\ ms \setminus \sigma_{\lambda \langle m', a \rangle.\ m' = m}\ ms \rangle, \quad \bot \rangle$

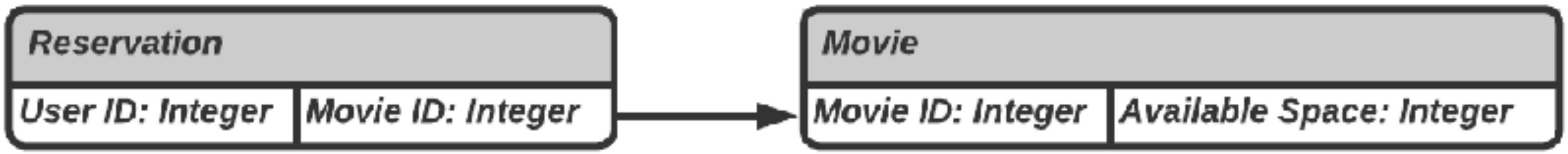
specialReserve$(\langle m, n \rangle) := 0\ \lambda \langle rs, ms \rangle.$

$\langle n > 0, \quad \langle rs,\ \mathcal{U}_{\lambda \langle m', a \rangle.\ \langle m' = m, \langle m, a-n \rangle \rangle}\ ms \rangle, \quad \bot \rangle$

increaseSpace$(\langle m, n \rangle) := 0\ \lambda \langle rs, ms \rangle.$

$\langle n > 0, \quad \langle rs,\ \mathcal{U}_{\lambda \langle m', a \rangle.\ \langle m' = m, \langle m, a+n \rangle \rangle}\ ms \rangle, \quad \bot \rangle$

Movie Booking use-case



Class MovieBooking

$\Sigma := \text{let } rs := \text{Set } \mathbb{N} \times \mathbb{N} \text{ in}$ ▷ Reservation: user identifier and movie identifier

$\quad \text{let } ms := \text{Set } \mathbb{N} \times \mathbb{N} \text{ in}$ ▷ Movie: movie identifier and available space

$\quad \langle rs, ms \rangle$

$\mathcal{I} := \lambda\langle rs, ms\rangle.\ \text{unique}\,(ms, \lambda\langle m, a\rangle.\ m) \wedge$

$\quad \text{refIntegrity}\,(rs, \lambda\langle u, m\rangle.\ m, ms, \lambda\langle m, a\rangle.\ m) \wedge$

$\quad \text{rowIntegrity}\,(ms, \lambda\langle m, a\rangle.\ a \geq 0)$

$\text{book}(\langle u, m\rangle) := 0\ \lambda\langle rs, ms\rangle.$

$\quad \langle \langle u, m\rangle \notin rs, \quad \langle rs \cup \langle u, m\rangle, \ \mathcal{U}_{\lambda\langle m', a\rangle.\ \langle m'=m, \langle m, a-1\rangle\rangle}\ ms\rangle, \quad \bot\rangle$

$\text{cancelBook}(\langle u, m\rangle) := 0\ \lambda\langle rs, ms\rangle.$

$\quad \langle \text{True}, \quad \langle rs \setminus \langle u, m\rangle, \ \mathcal{U}_{\lambda\langle m', a\rangle.\ \langle m'=m, \langle m, a+1\rangle\rangle}\ ms\rangle, \quad \bot\rangle$

$\text{offScreen}(m) := 0\ \lambda\langle rs, ms\rangle.$

$\quad \langle \text{True}, \quad \langle rs, \ ms \setminus \sigma_{\lambda\langle m', a\rangle.\ m'=m}\ ms\rangle, \quad \bot\rangle$

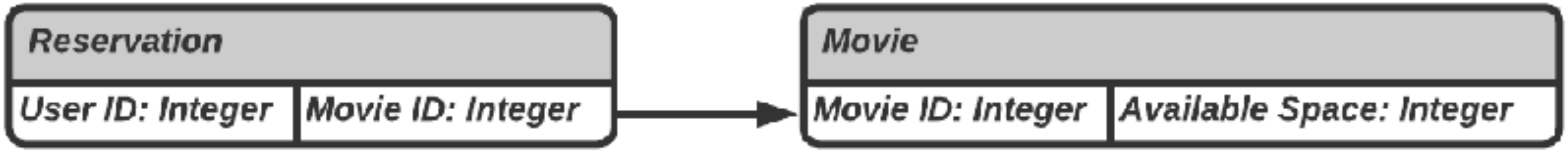
$\text{specialReserve}(\langle m, n\rangle) := 0\ \lambda\langle rs, ms\rangle.$

$\quad \langle n > 0, \quad \langle rs, \ \mathcal{U}_{\lambda\langle m', a\rangle.\ \langle m'=m, \langle m, a-n\rangle\rangle}\ ms\rangle, \quad \bot\rangle$

$\text{increaseSpace}(\langle m, n\rangle) := 0\ \lambda\langle rs, ms\rangle.$

$\quad \langle n > 0, \quad \langle rs, \ \mathcal{U}_{\lambda\langle m', a\rangle.\ \langle m'=m, \langle m, a+n\rangle\rangle}\ ms\rangle, \quad \bot\rangle$

Movie Booking use-case



Class MovieBooking

$\Sigma :=$ let $rs :=$ Set $\mathbb{N} \times \mathbb{N}$ in $\quad \triangleright$ Reservation: user identifier and movie identifier

let $ms :=$ Set $\mathbb{N} \times \mathbb{N}$ in $\quad \triangleright$ Movie: movie identifier and available space

$\langle rs, ms \rangle$

$\mathcal{I} := \lambda \langle rs, ms \rangle.$ unique $(ms, \lambda \langle m, a \rangle.\ m) \wedge$

refIntegrity $(rs, \lambda \langle u, m \rangle.\ m, ms, \lambda \langle m, a \rangle.\ m) \wedge$

rowIntegrity $(ms, \lambda \langle m, a \rangle.\ a \geq 0)$

book$(\langle u, m \rangle) := 0\ \lambda \langle rs, ms \rangle.$

$\langle \langle u, m \rangle \notin rs, \quad \langle rs \cup \langle u, m \rangle, \mathcal{U}_{\lambda \langle m', a \rangle.\ \langle m' = m, \langle m, a-1 \rangle \rangle}\ ms \rangle, \quad \bot \rangle$

cancelBook$(\langle u, m \rangle) := 0\ \lambda \langle rs, ms \rangle.$

$\langle$True, $\quad \langle rs \setminus \langle u, m \rangle, \mathcal{U}_{\lambda \langle m', a \rangle.\ \langle m' = m, \langle m, a+1 \rangle \rangle}\ ms \rangle, \quad \bot \rangle$

offScreen$(m) := 0\ \lambda \langle rs, ms \rangle.$

$\langle$True, $\quad \langle rs,\ ms \setminus \sigma_{\lambda \langle m', a \rangle.\ m' = m}\ ms \rangle, \quad \bot \rangle$

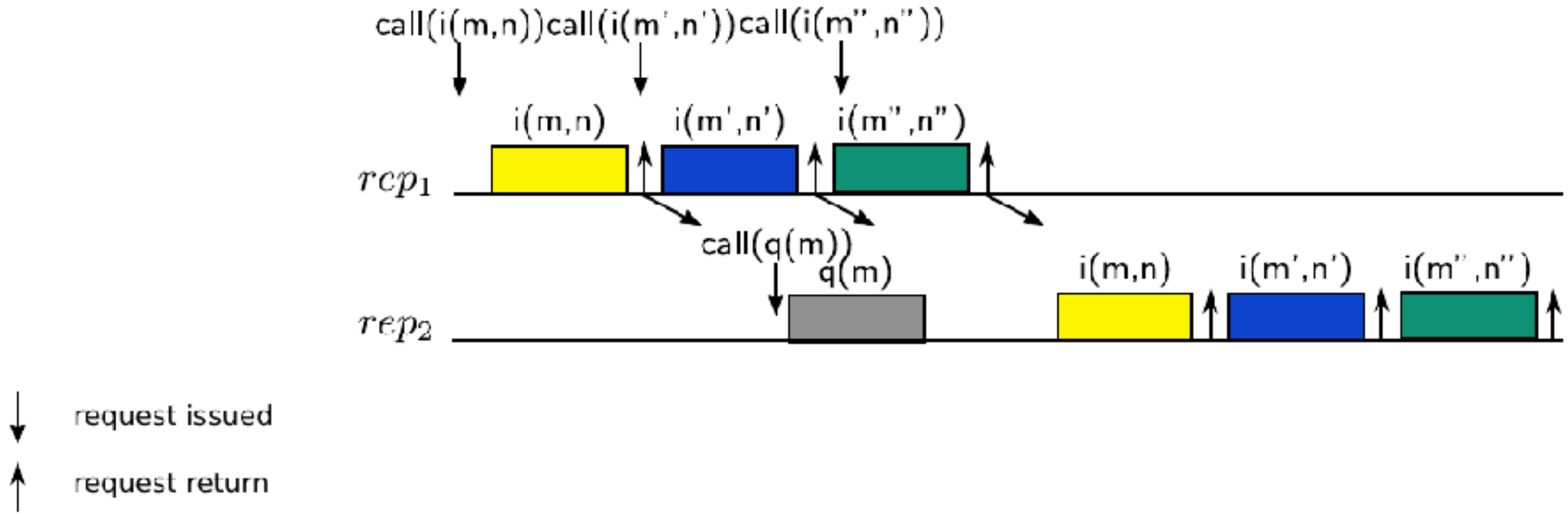
specialReserve$(\langle m, n \rangle) := 0\ \lambda \langle rs, ms \rangle.$

$\langle n > 0, \quad \langle rs,\ \mathcal{U}_{\lambda \langle m', a \rangle.\ \langle m' = m, \langle m, a-n \rangle \rangle}\ ms \rangle, \quad \bot \rangle$

increaseSpace$(\langle m, n \rangle) := 0\ \lambda \langle rs, ms \rangle.$

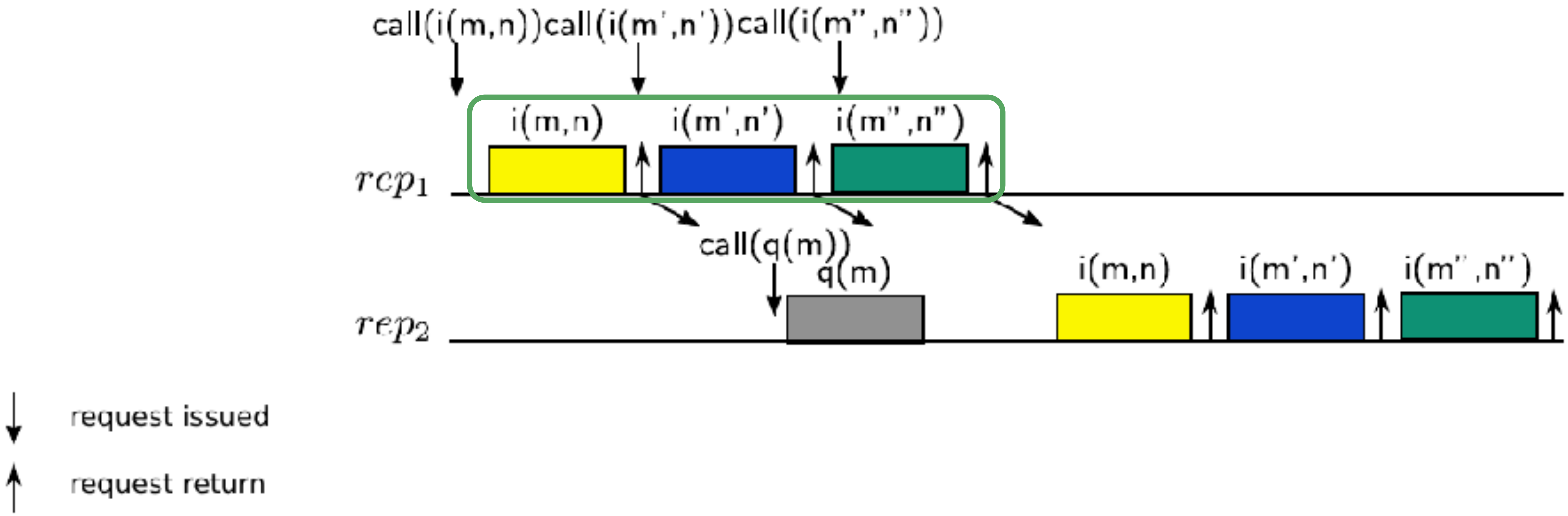
$\langle n > 0, \quad \langle rs,\ \mathcal{U}_{\lambda \langle m', a \rangle.\ \langle m' = m, \langle m, a+n \rangle \rangle}\ ms \rangle, \quad \bot \rangle$

Recency



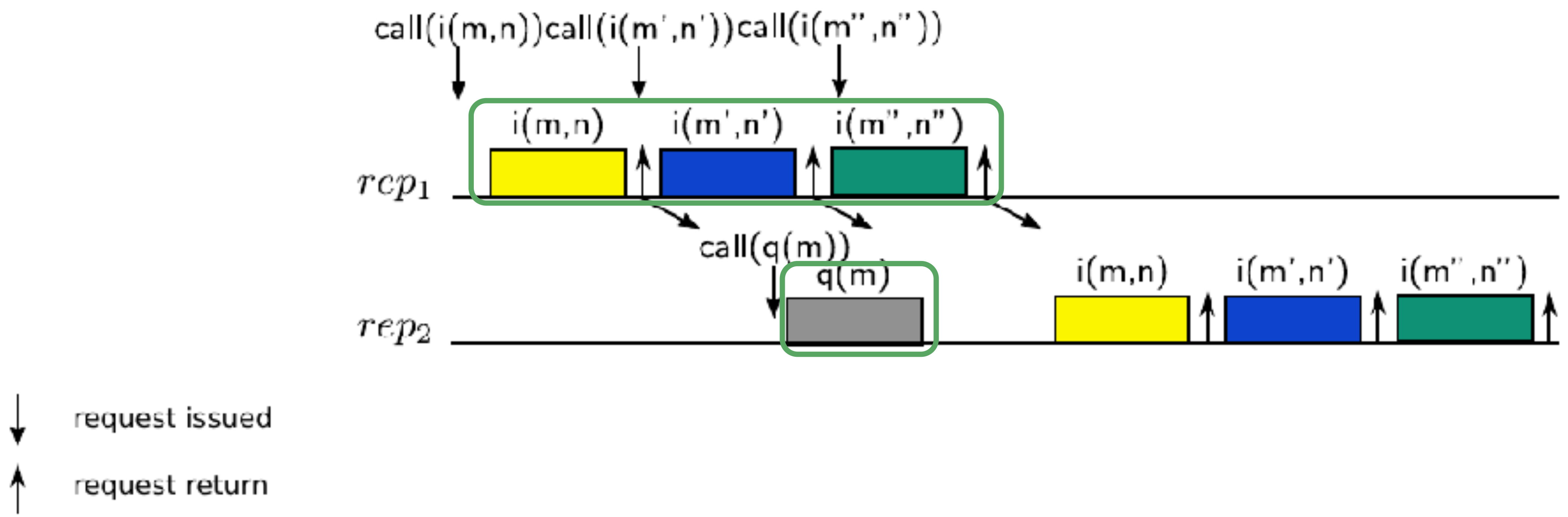
call(i(m,n))call(i(m',n'))call(i(m'',n''))
i(m,n)
i(m',n')
i(m'',n'')
rcp_1
call(q(m))
q(m)
i(m,n)
i(m',n')
i(m'',n'')
rep_2
request issued
request return

Recency



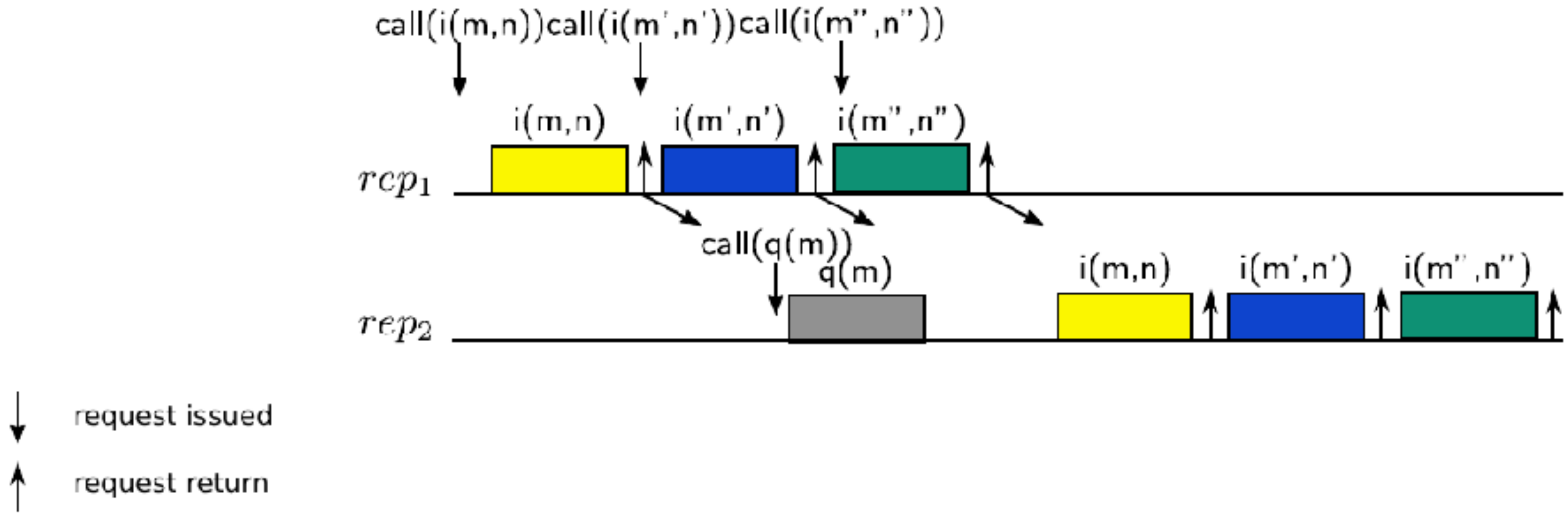
call(i(m,n))call(i(m',n'))call(i(m'',n''))
i(m,n)
i(m',n')
i(m'',n'')
rcp1
call(q(m))
q(m)
i(m,n)
i(m',n')
i(m'',n'')
rep2
request issued
request return

Recency



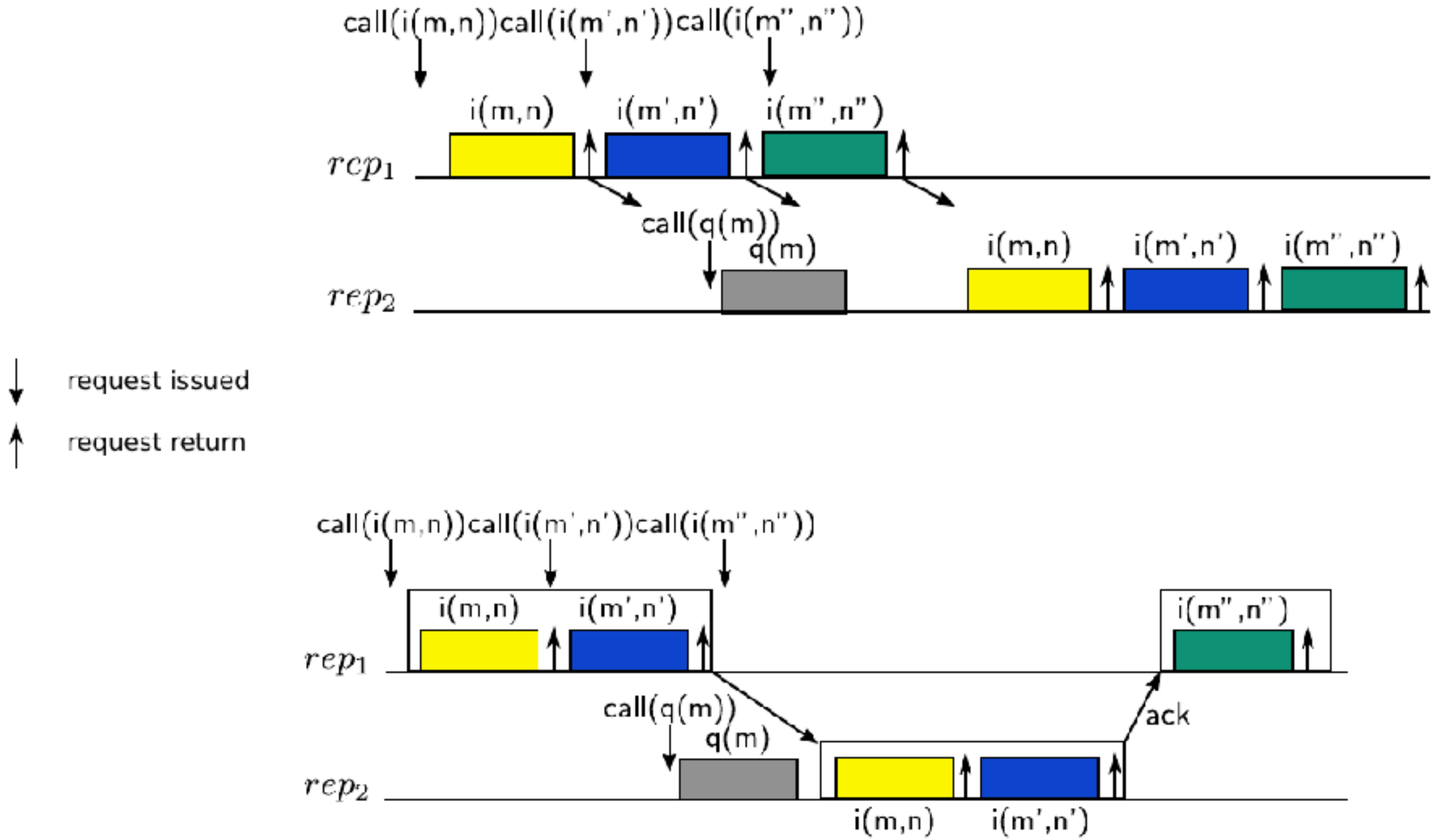
call(i(m,n))call(i(m',n'))call(i(m'',n''))
i(m,n)
i(m',n')
i(m'',n'')
rcp1
call(q(m))
q(m)
i(m,n)
i(m',n')
i(m'',n'')
rep2
request issued
request return

Recency



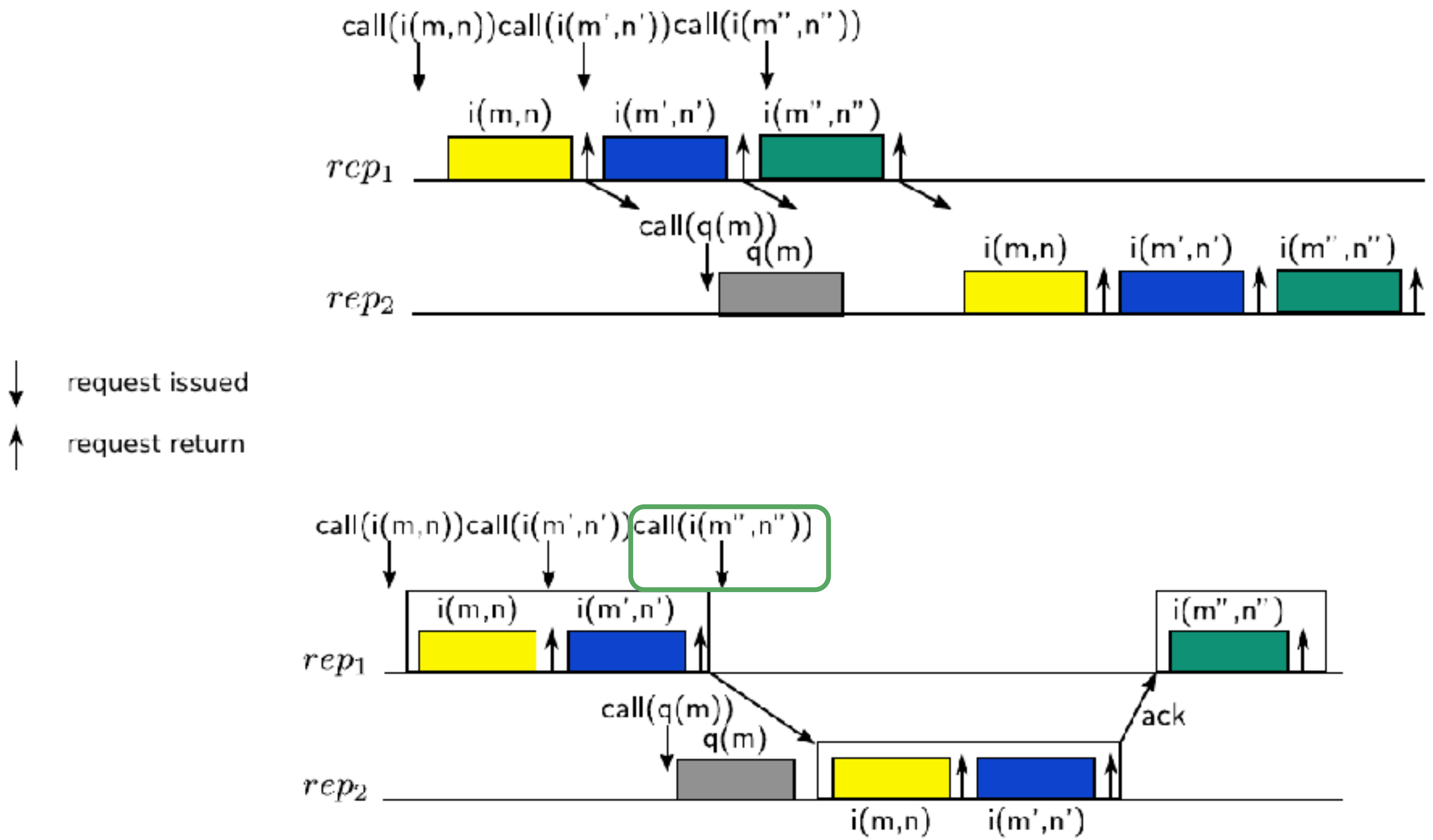
call(i(m,n))call(i(m',n'))call(i(m'',n''))
i(m,n)
i(m',n')
i(m'',n'')
rcp_1
call(q(m))
q(m)
i(m,n)
i(m',n')
i(m'',n'')
rep_2
request issued
request return

Recency



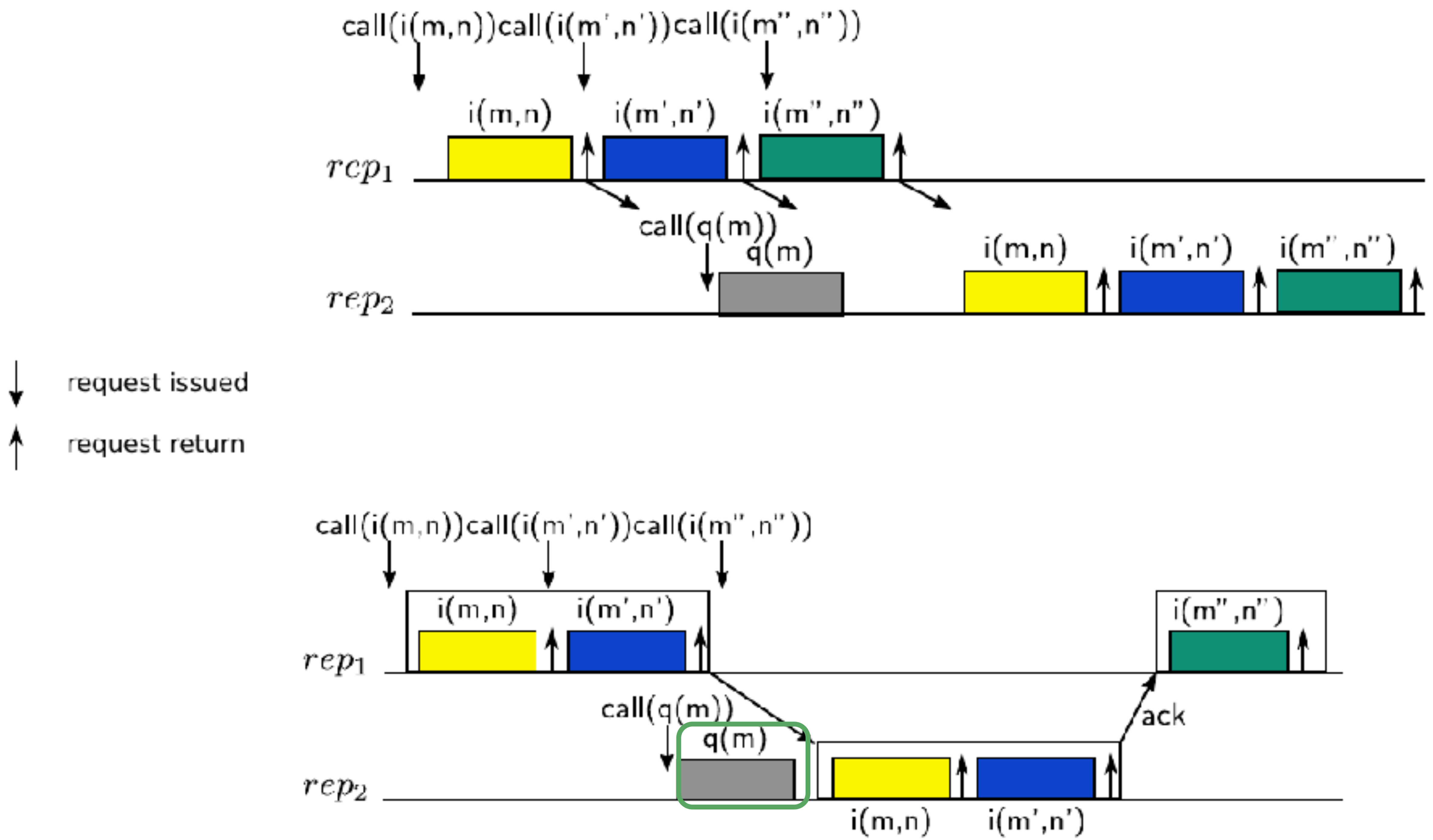
call(i(m,n))call(i(m',n'))call(i(m'',n''))
i(m,n)
i(m',n')
i(m'',n'')
rep_1
call(q(m))
q(m)
i(m,n)
i(m',n')
i(m'',n'')
rep_2
request issued
request return
call(i(m,n))call(i(m',n'))call(i(m'',n''))
i(m,n)
i(m',n')
i(m'',n'')
rep_1
call(q(m))
q(m)
ack
rep_2
i(m,n)
i(m',n')

Recency



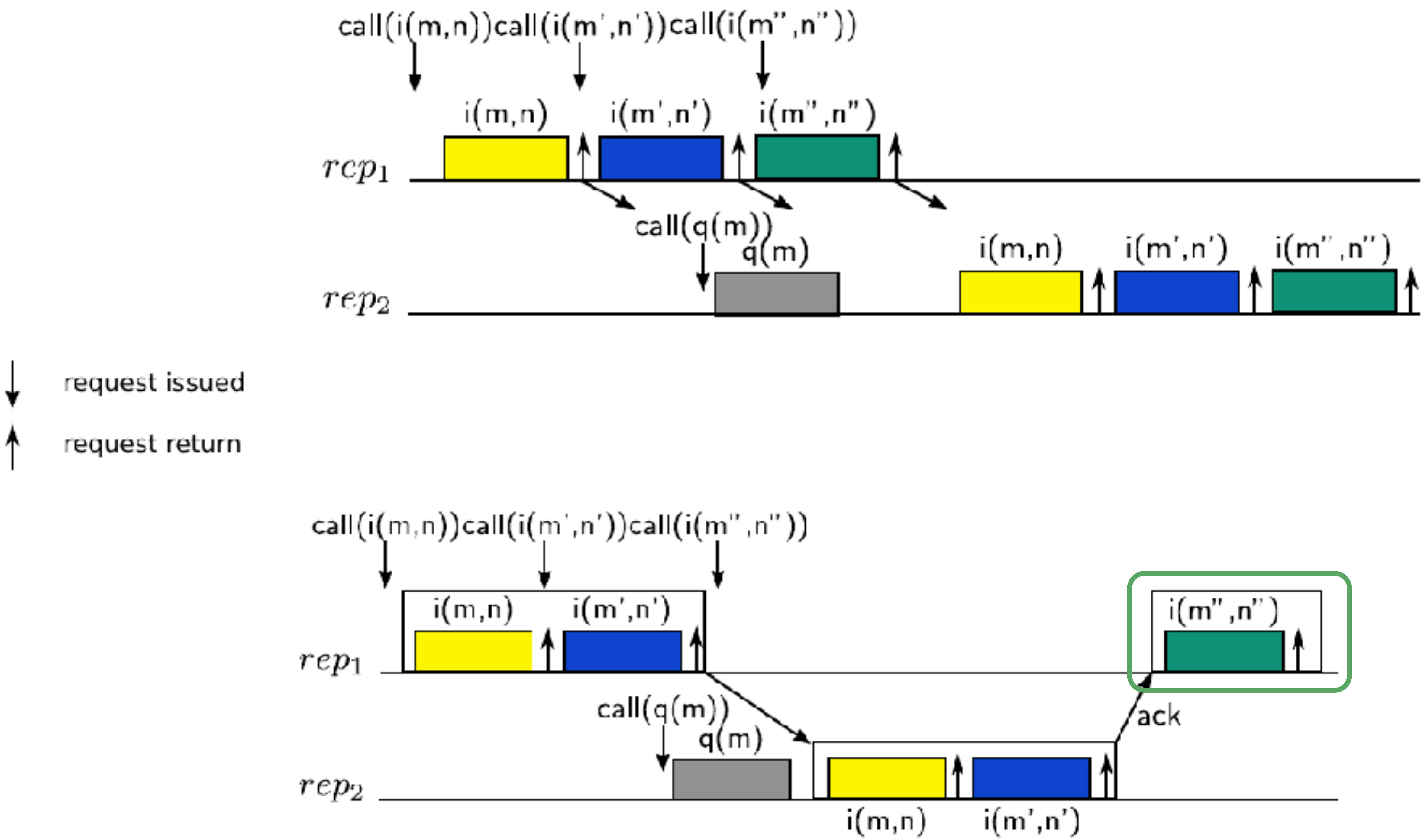
call(i(m,n))call(i(m',n'))call(i(m'',n''))
i(m,n)
i(m',n')
i(m'',n'')
rep1
call(q(m))
q(m)
rep2
request issued
request return
ack

Recency

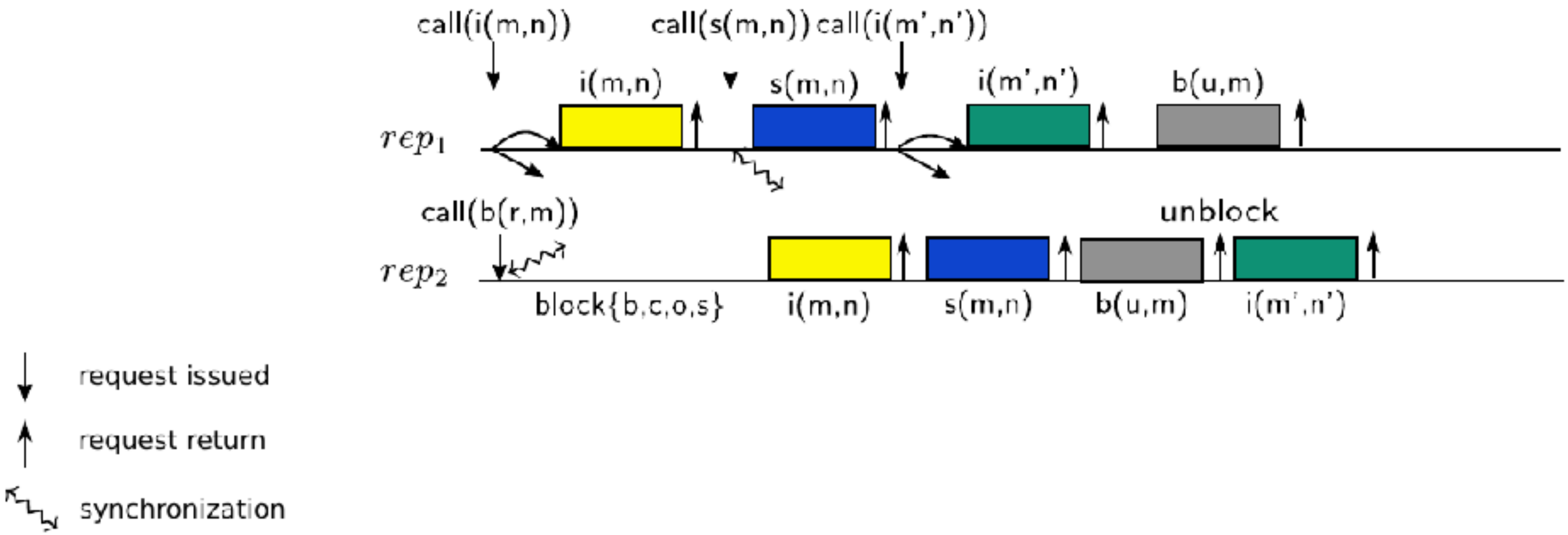


call(i(m,n))call(i(m',n'))call(i(m'',n''))
i(m,n)
i(m',n')
i(m'',n'')
rep1
call(q(m))
q(m)
rep2
request issued
request return
ack

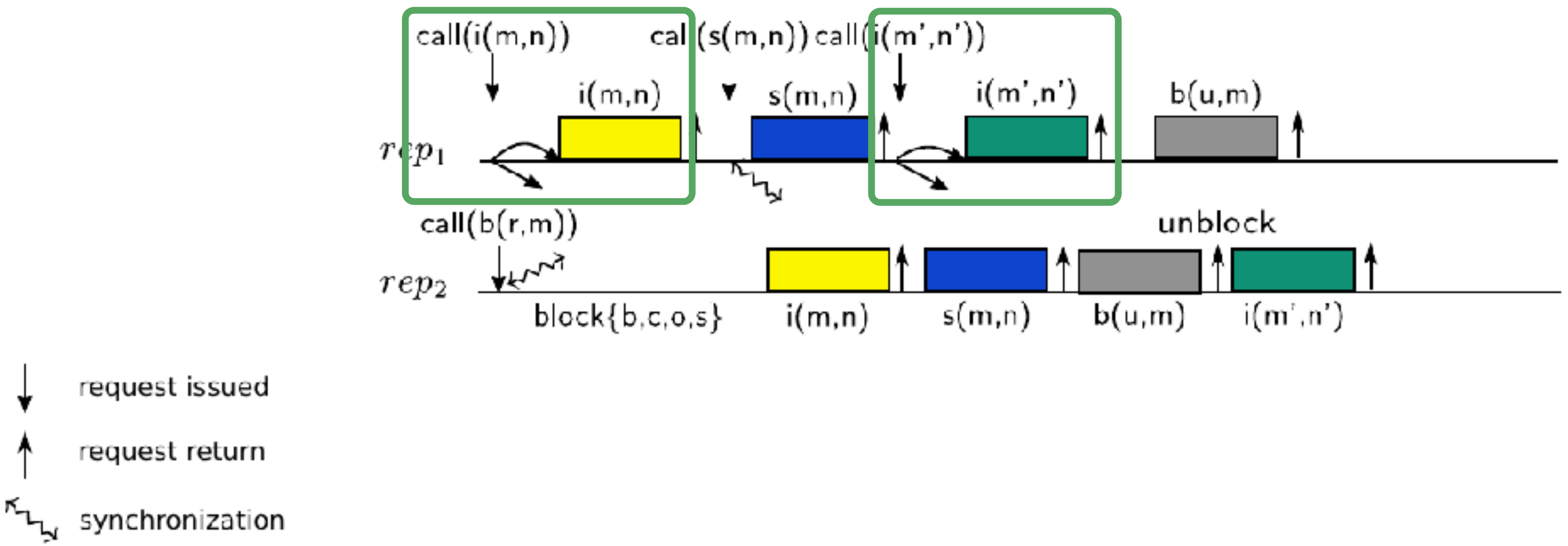
Recency


call(i(m,n))call(i(m',n'))call(i(m",n"))
i(m,n)
i(m',n')
i(m",n")
rep_1
call(q(m))
q(m)
rep_2
request issued
request return
ack

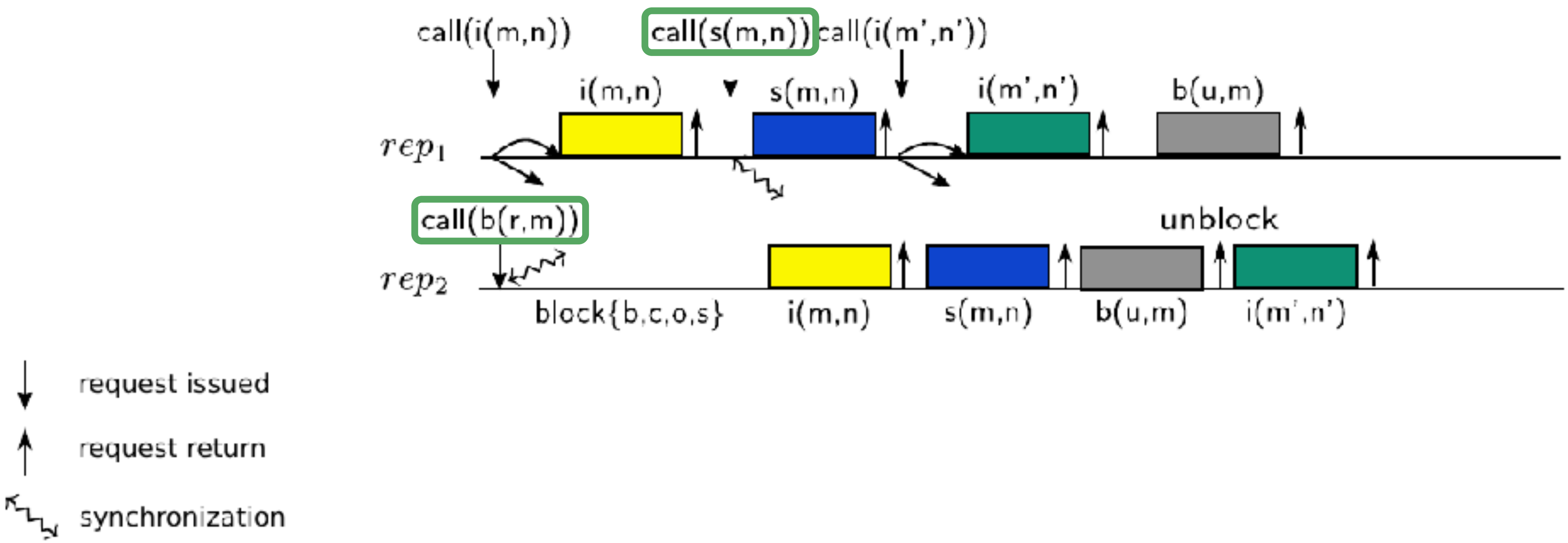
Communication and Synchronization Avoidance



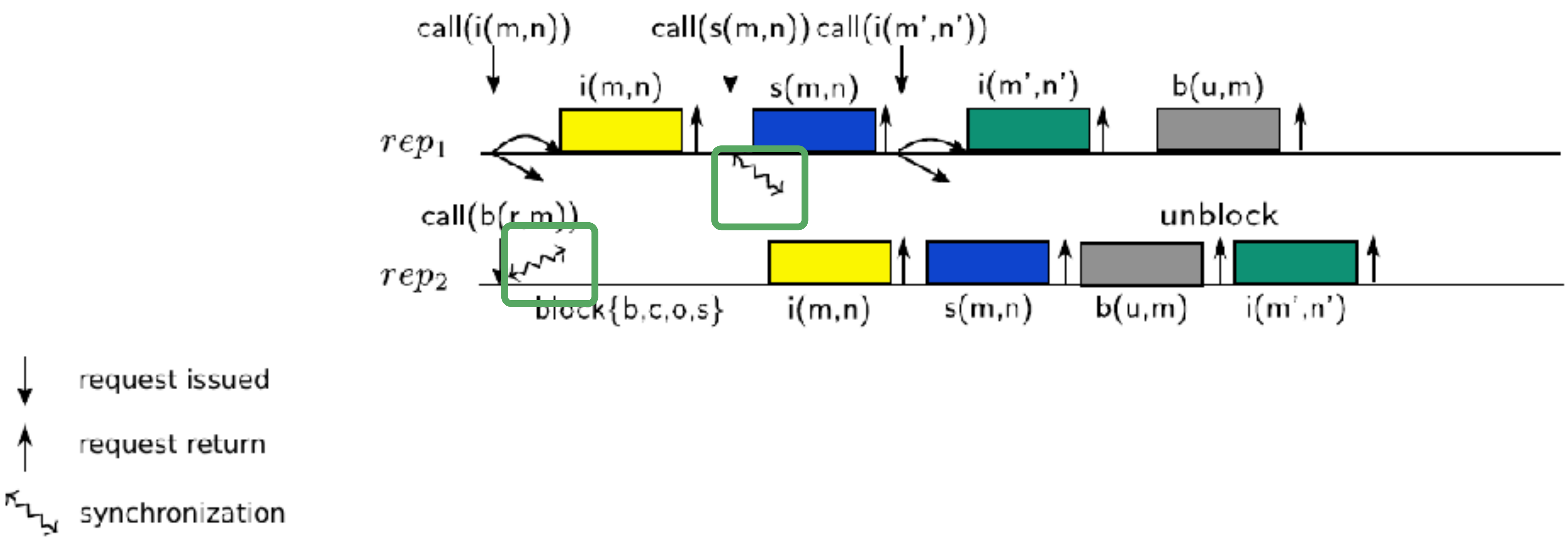
Communication and Synchronization Avoidance



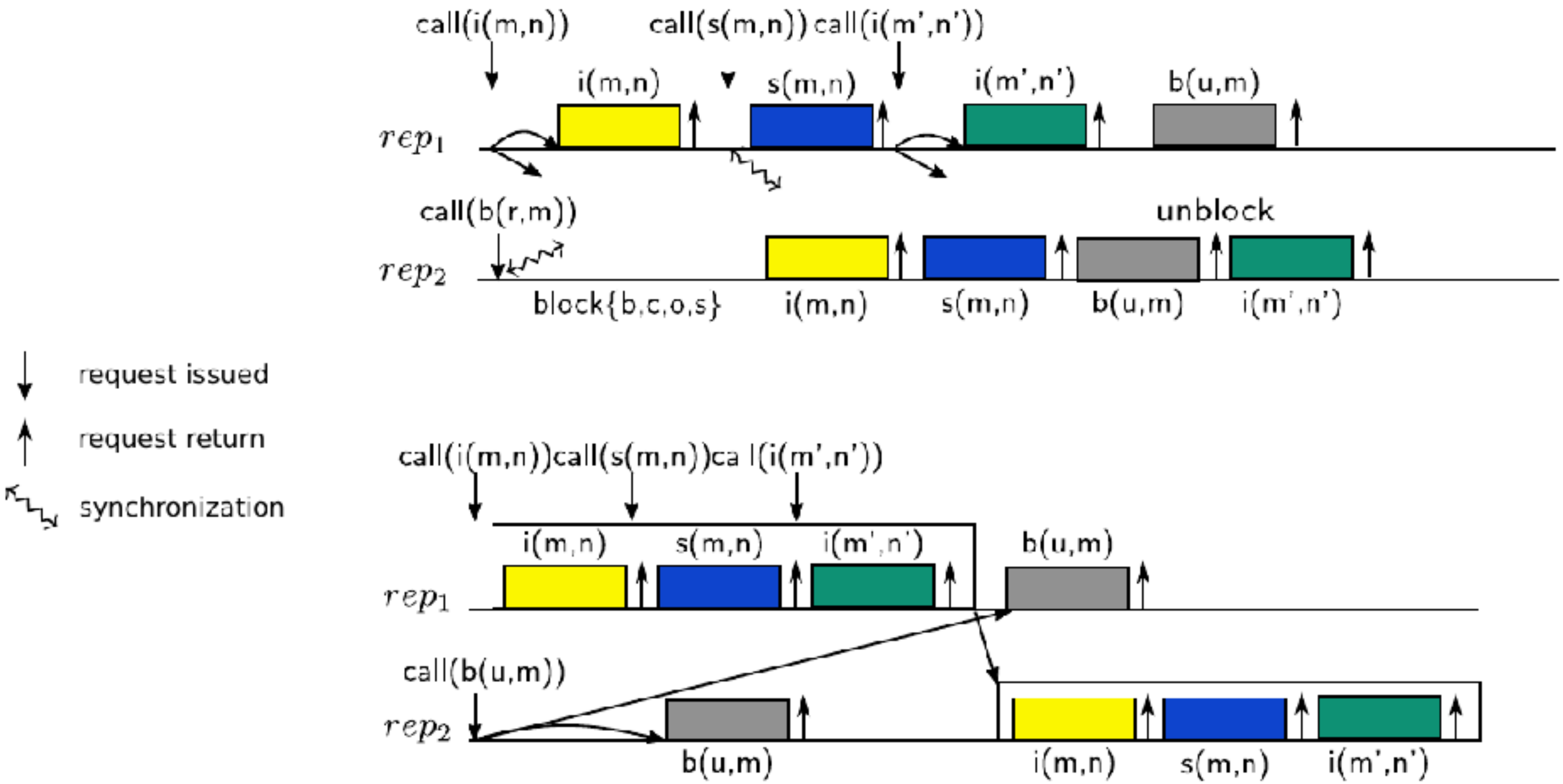
Communication and Synchronization Avoidance



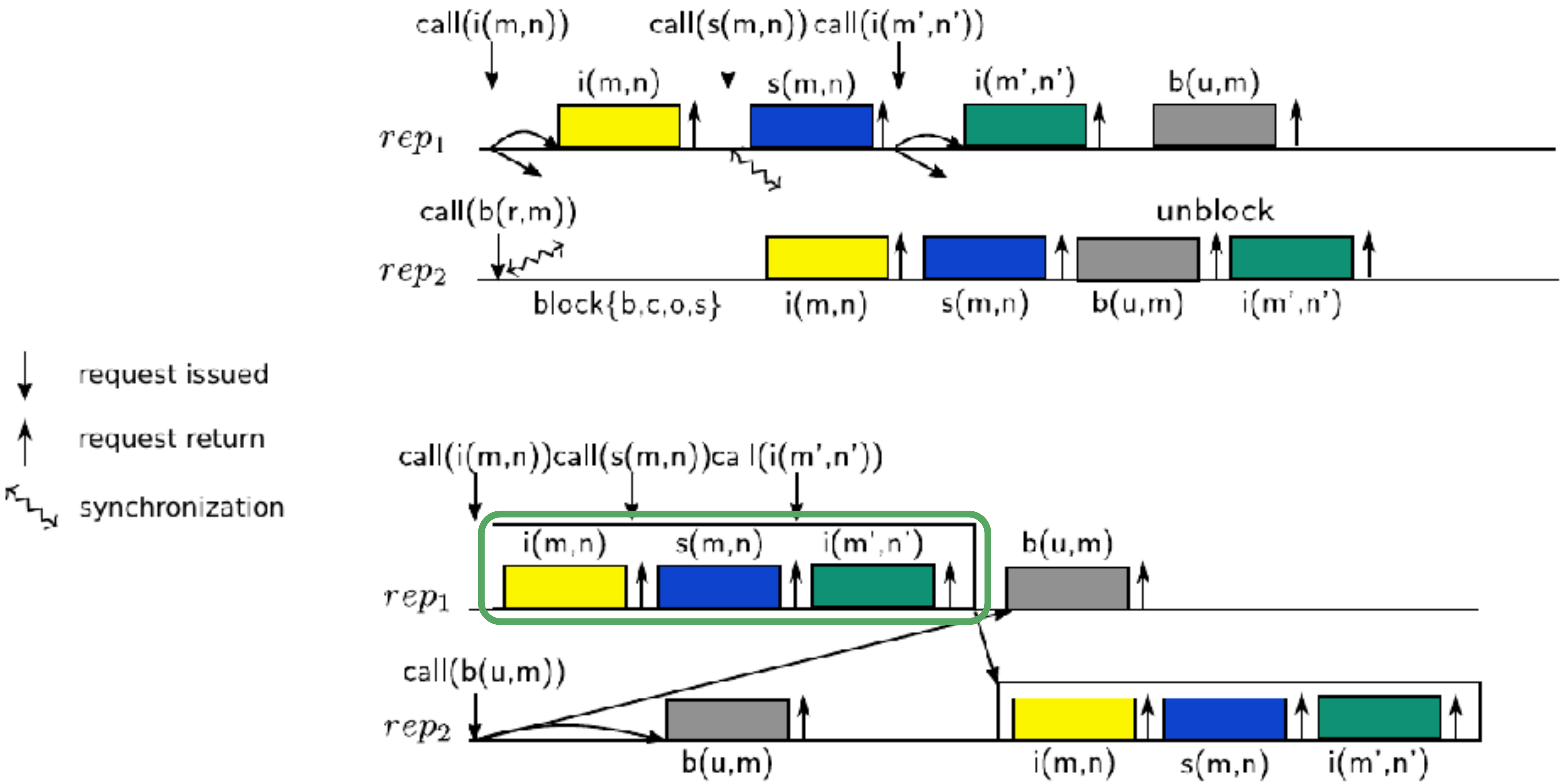
Communication and Synchronization Avoidance



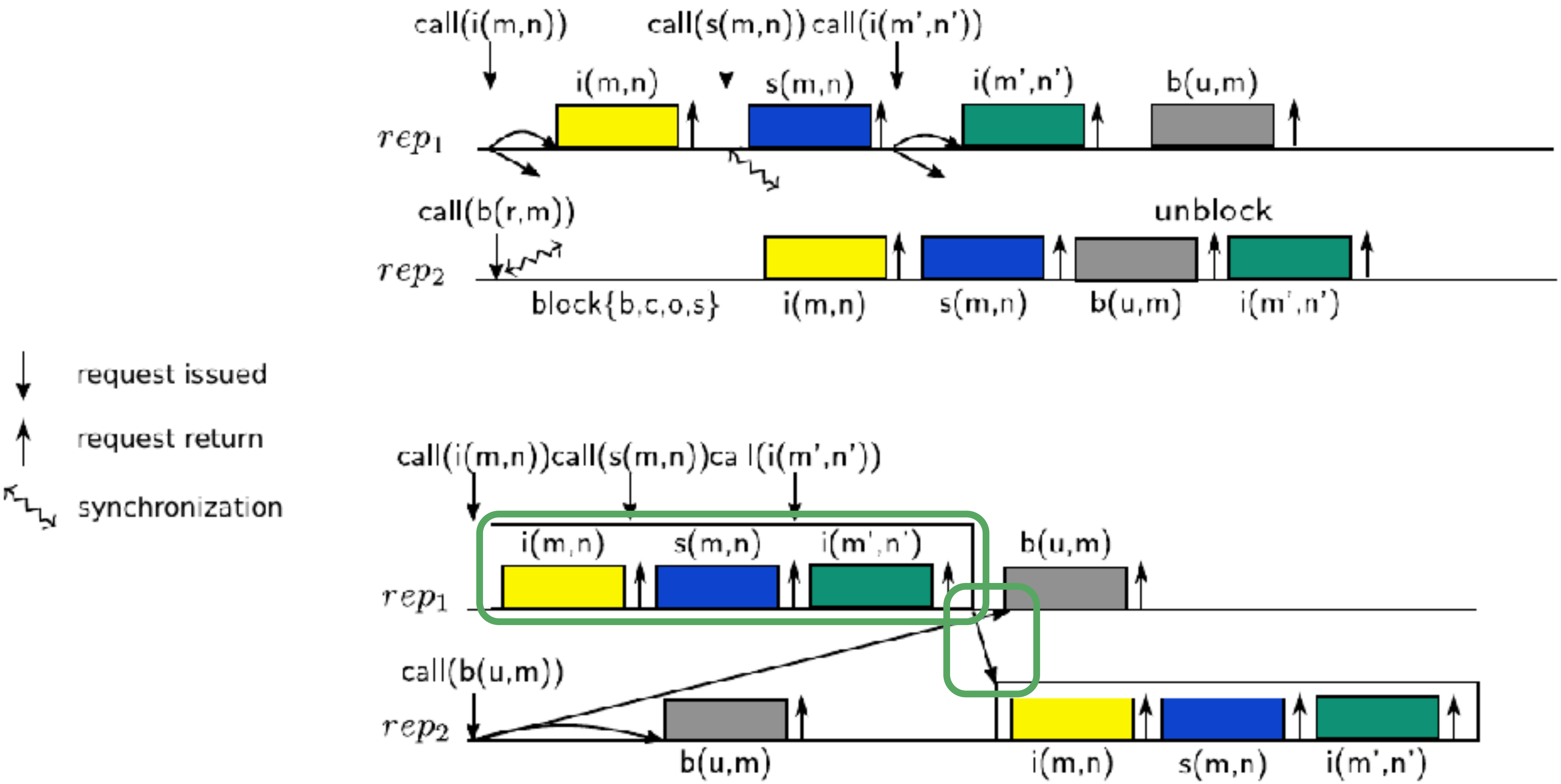
Communication and Synchronization Avoidance



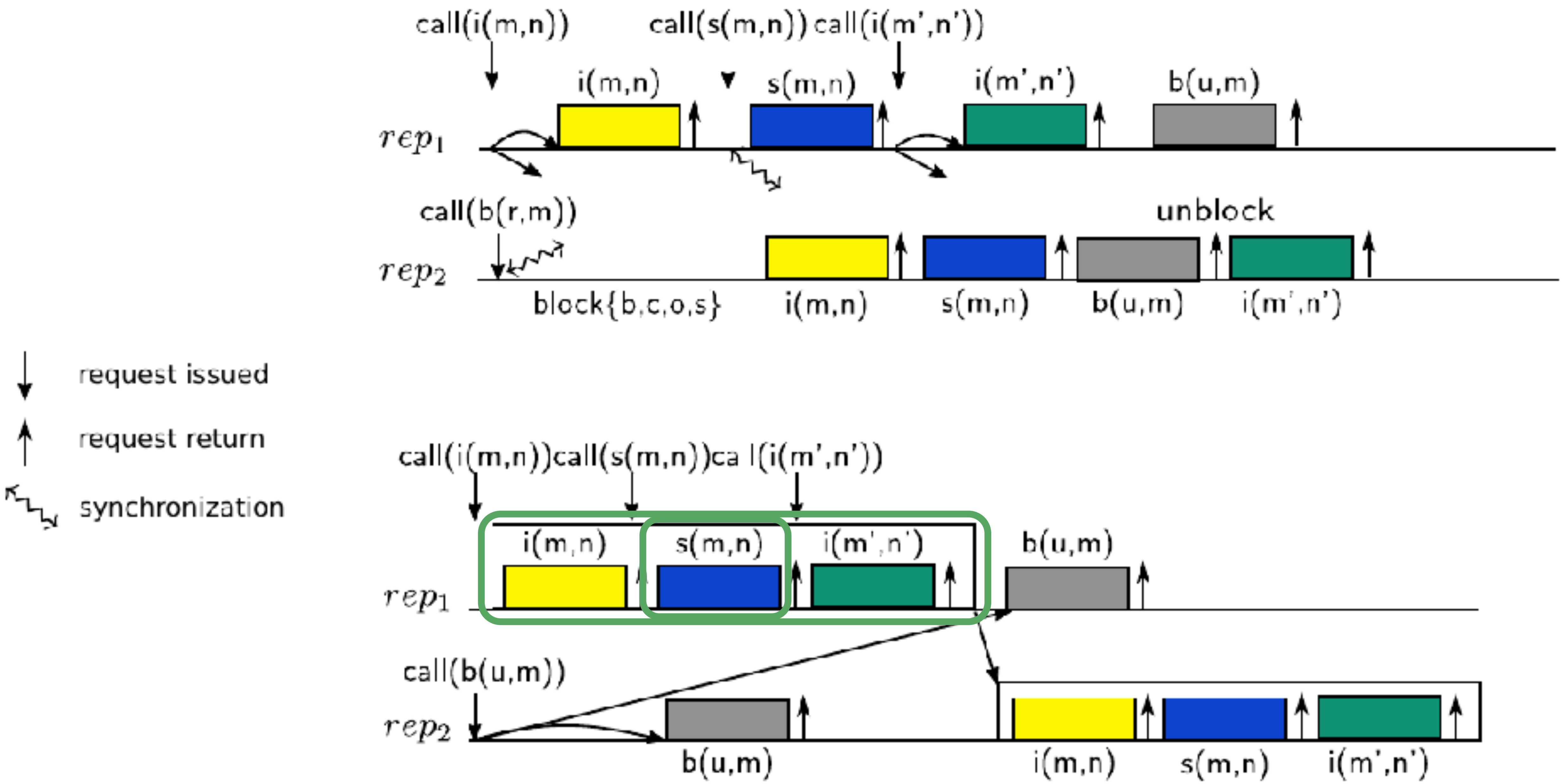
Communication and Synchronization Avoidance



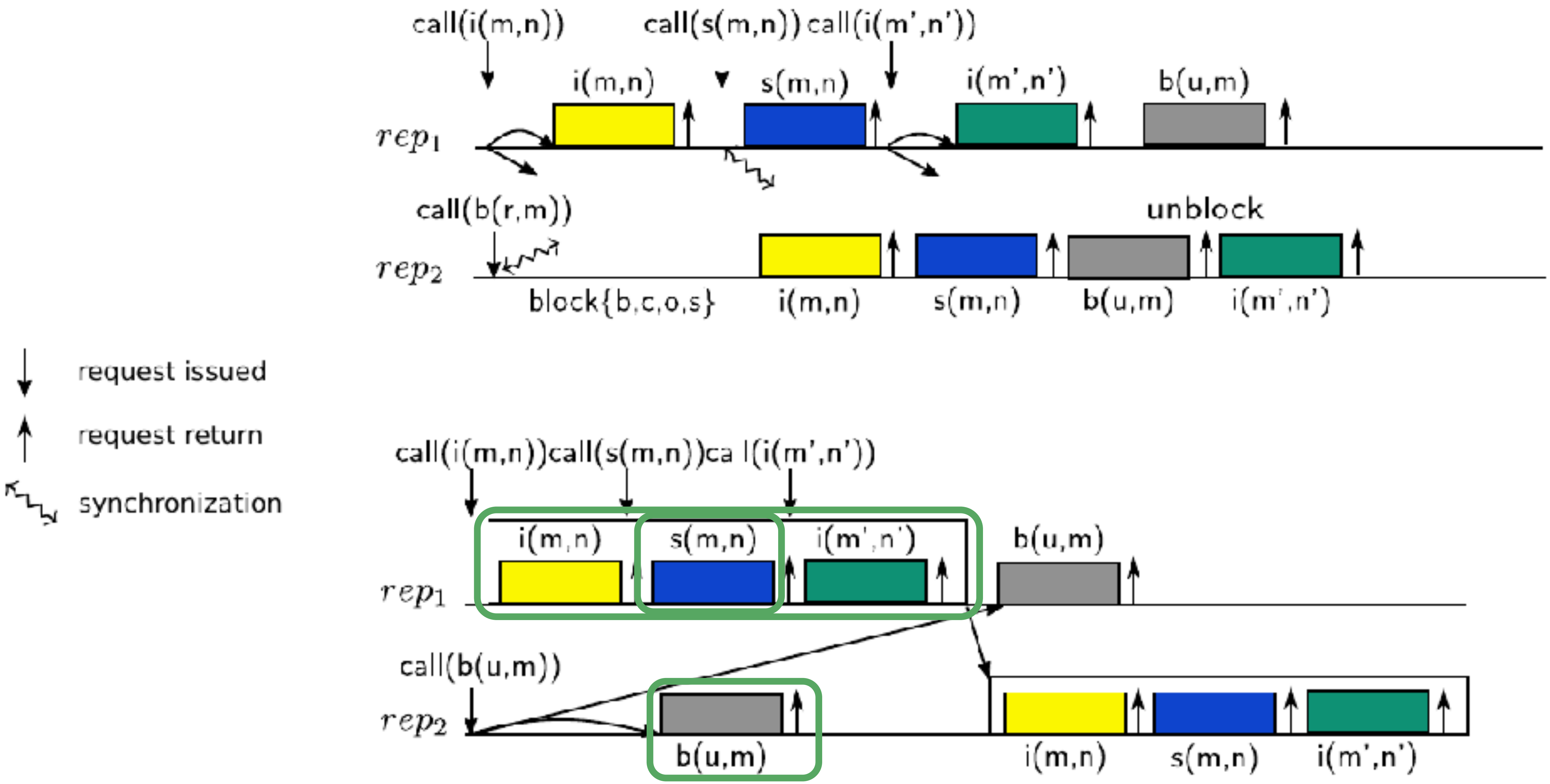
Communication and Synchronization Avoidance



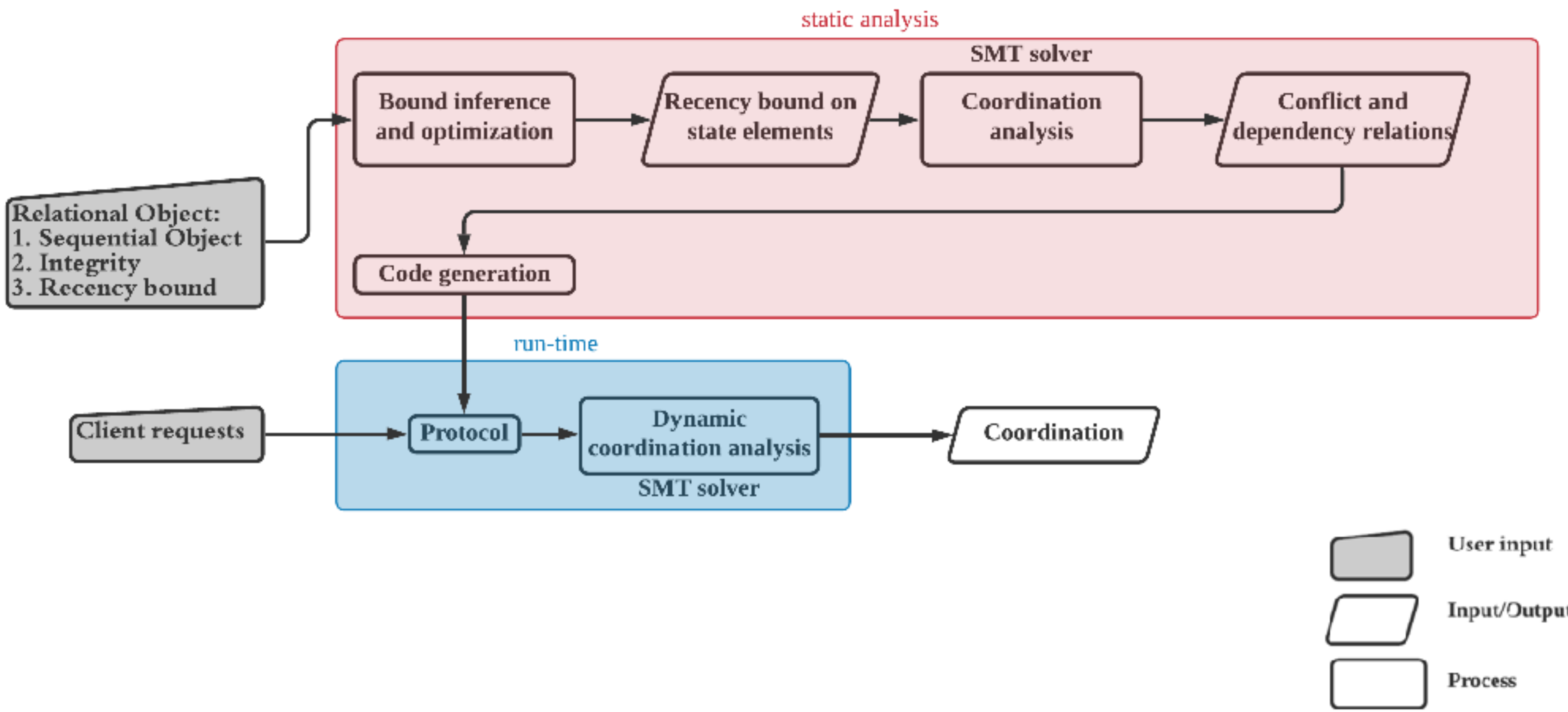
Communication and Synchronization Avoidance



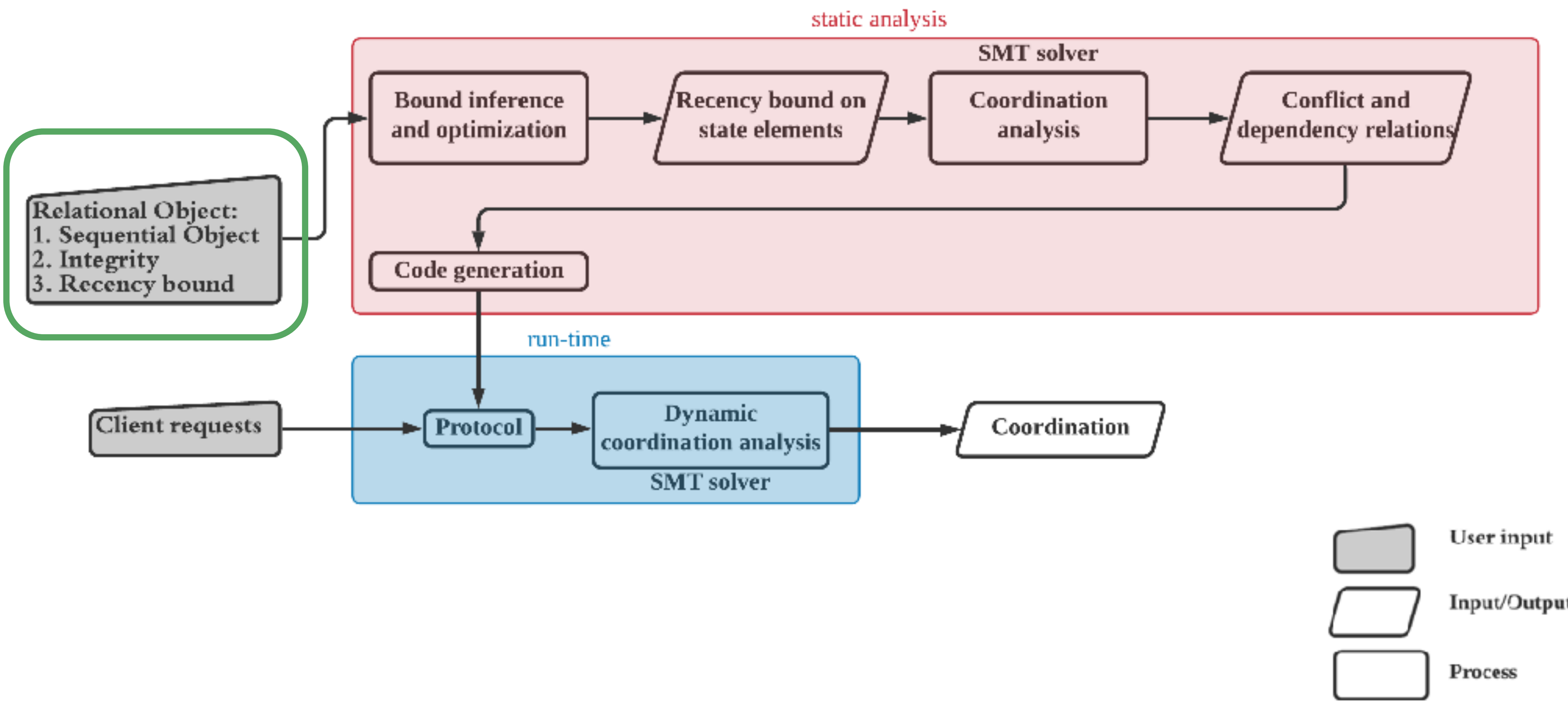
Communication and Synchronization Avoidance



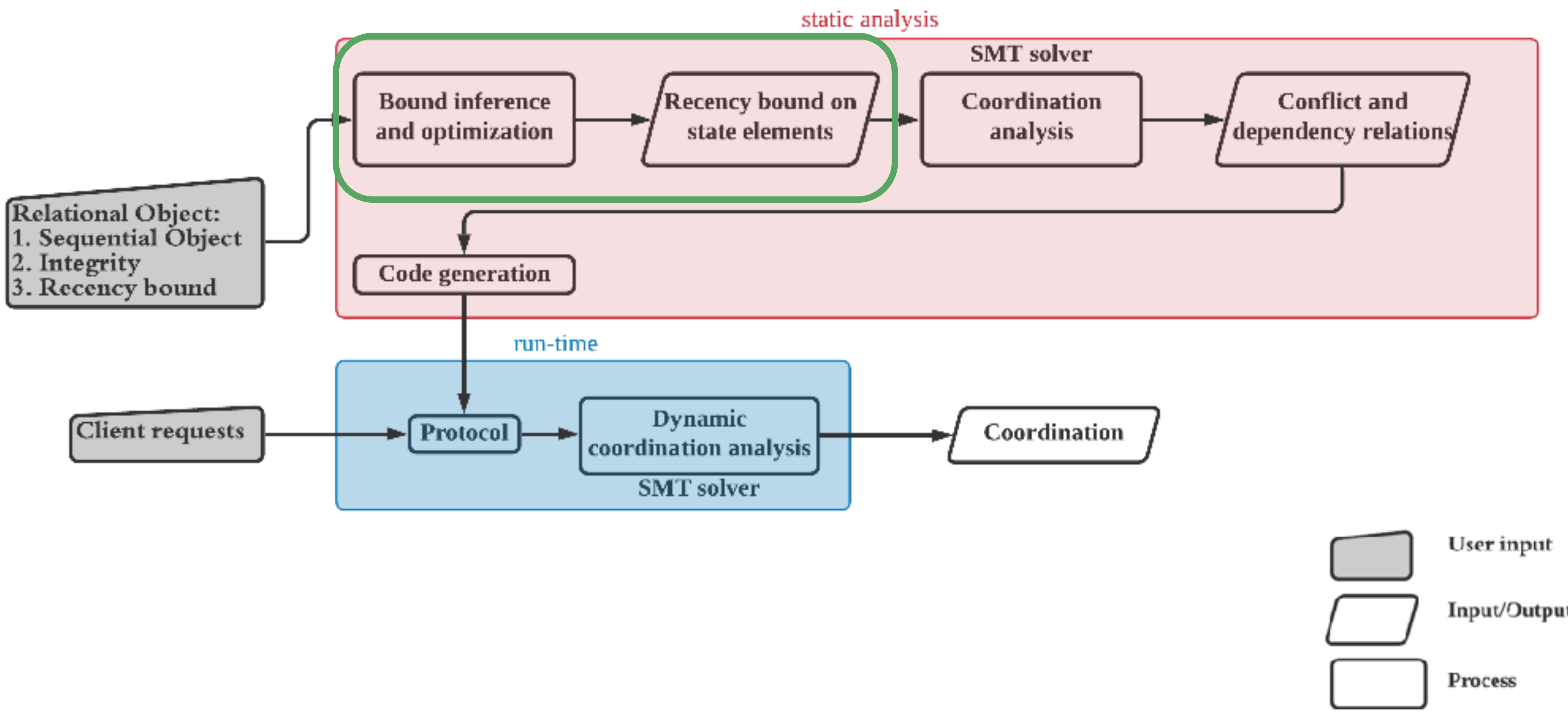
The Anatomy of Hampa



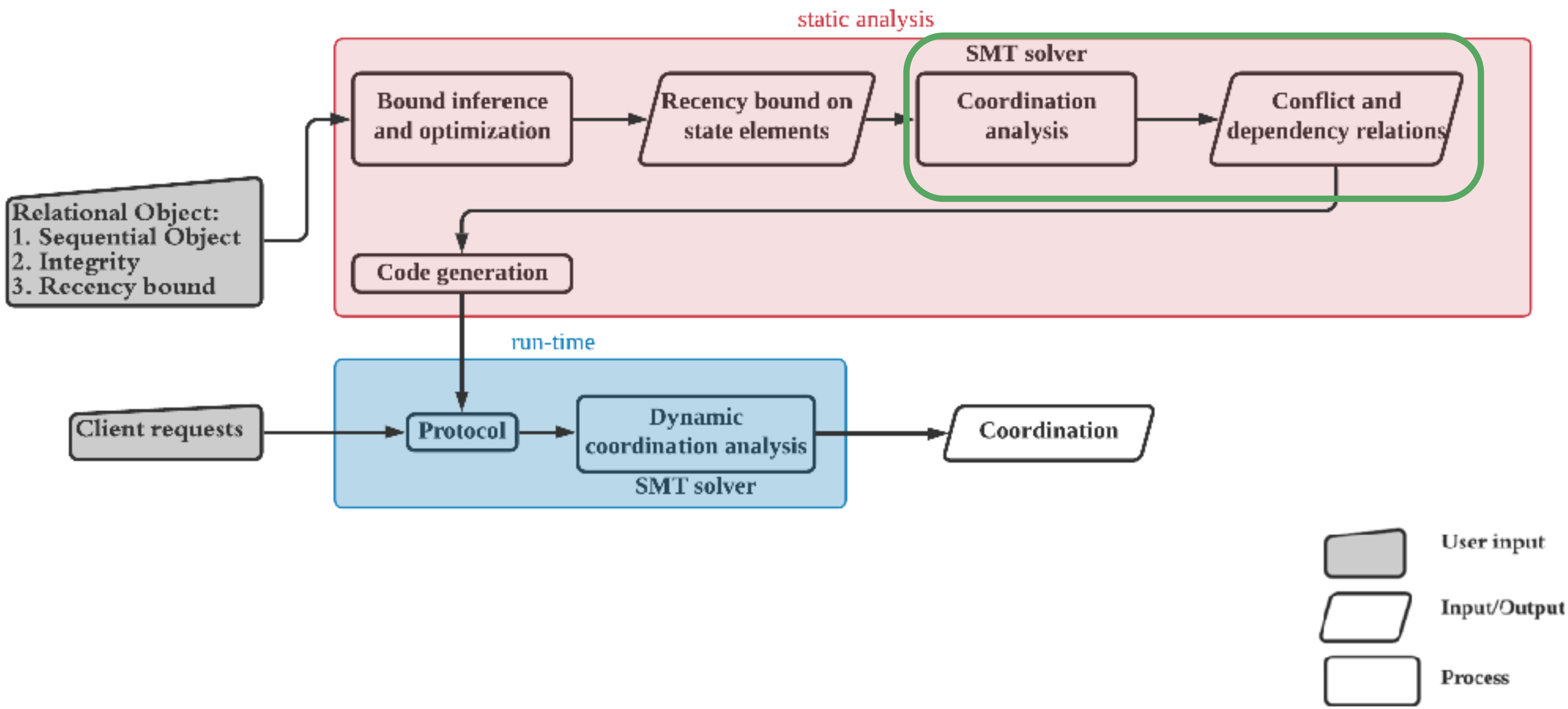
The Anatomy of Hampa



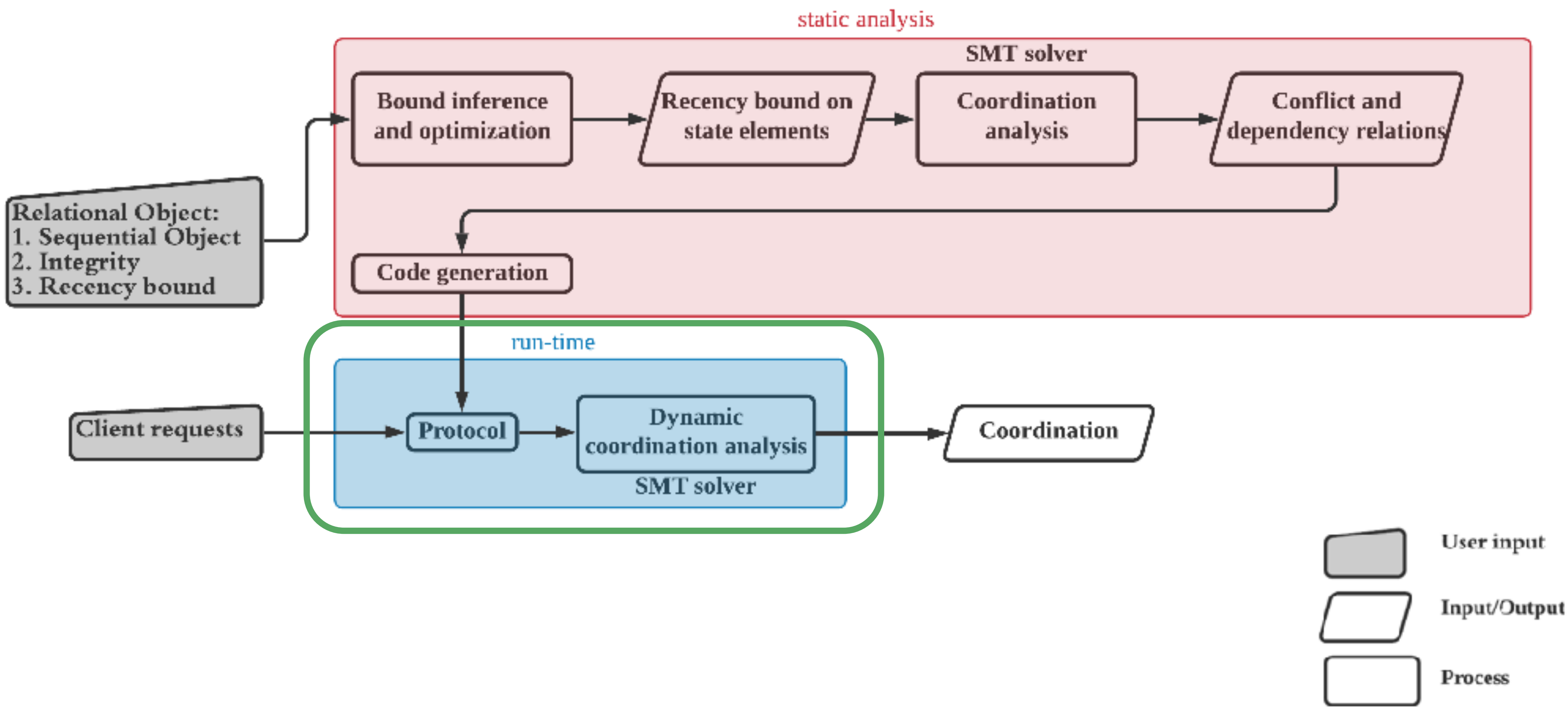
The Anatomy of Hampa



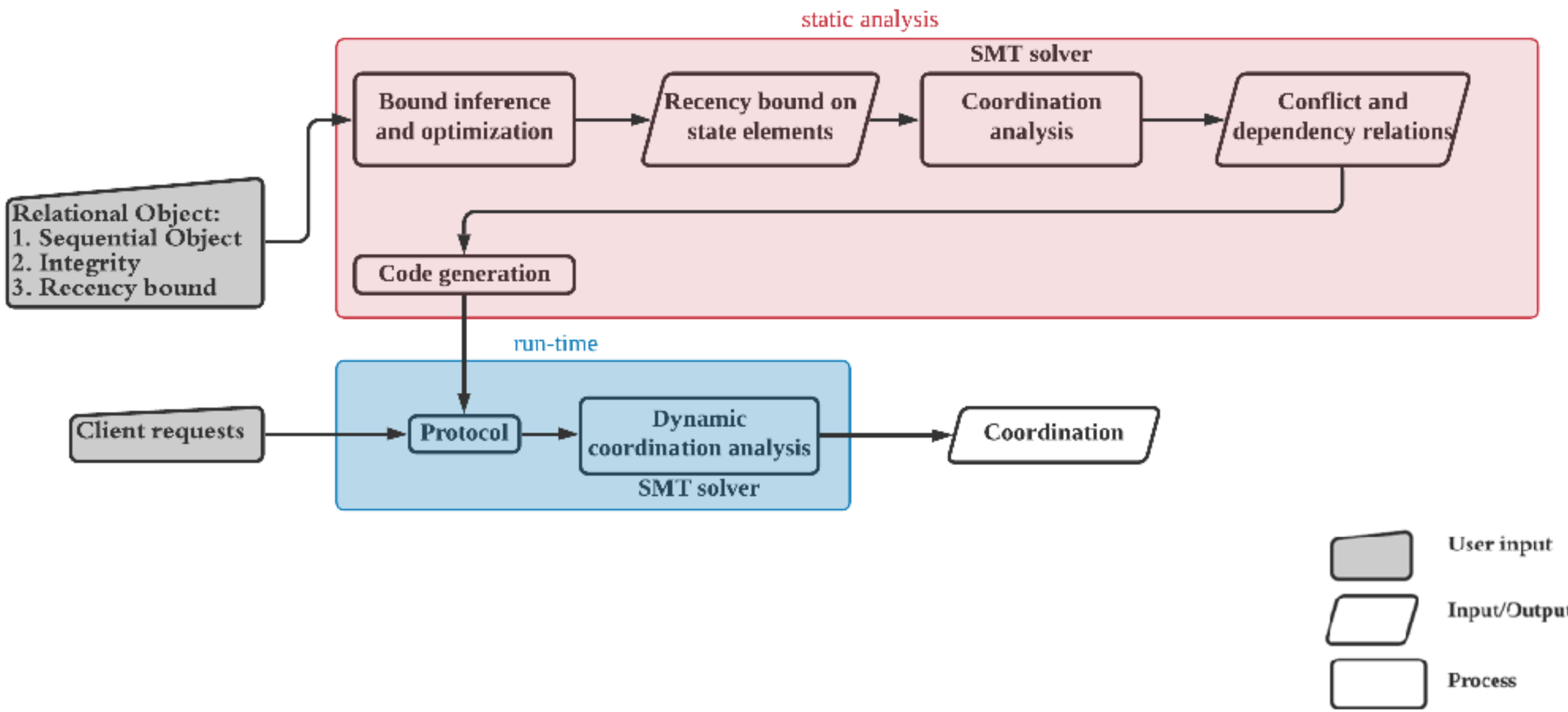
The Anatomy of Hampa



The Anatomy of Hampa



The Anatomy of Hampa



Bound Inference

$\text{querySpace}(m) := 3\ \lambda\langle rs, ms\rangle.$
$\quad \langle \ldots, \quad \ldots, \quad \Pi_{\lambda\langle m', a\rangle.\ \langle a\rangle}\ (\sigma_{\lambda\langle m', a\rangle.\ m'=m}\ ms)\rangle$
$\text{queryReservations}(u) := 4\ \lambda\langle rs, ms\rangle.$
$\quad \langle \ldots, \quad \ldots, \quad \Pi_{\lambda\langle u', m\rangle.\ \langle m\rangle}\ (\sigma_{\lambda\langle u', m\rangle.\ u'=u}\ rs)\rangle$
$\text{querySpaces}(u) := 6\ \lambda\langle rs, ms\rangle.$
$\quad \langle \ldots, \quad \ldots, \quad \Pi_{\lambda\langle u, m, m', a\rangle\ \langle m, a\rangle}\ (rs\ \bowtie_{\lambda\langle u, m\rangle, \langle m', a\rangle.\ m=m'}\ ms)\rangle$

Bound Inference

$\text{querySpace}(m) := 3\ \lambda\langle rs, ms\rangle.$
$\quad\langle \ldots,\quad \ldots,\quad \Pi_{\lambda\langle m',a\rangle.\,\langle a\rangle}\,(\sigma_{\lambda\langle m',a\rangle.\,m'=m}\ ms)\rangle$
$\text{queryReservations}(u) := 4\ \lambda\langle rs, ms\rangle.$
$\quad\langle \ldots,\quad \ldots,\quad \Pi_{\lambda\langle u',m\rangle.\,\langle m\rangle}\,(\sigma_{\lambda\langle u',m\rangle.\,u'=u}\ rs)\rangle$
$\text{querySpaces}(u) := 6\ \lambda\langle rs, ms\rangle.$
$\quad\langle \ldots,\quad \ldots,\quad \Pi_{\lambda\langle u,m,m',a\rangle\,\langle m,a\rangle}\,(rs\ \bowtie_{\lambda\langle u,m\rangle,\langle m',a\rangle.\,m=m'}\ ms)\rangle$

Bound Inference

$\mathsf{querySpace}(m) := 3\ \lambda\langle rs, ms\rangle.$

$\langle \ldots,\quad \ldots,\quad \Pi_{\lambda\langle m',a\rangle.\,\langle a\rangle}\,(\sigma_{\lambda\langle m',a\rangle.\,m'=m}\ ms)\rangle$

$\mathsf{queryReservations}(u) := 4\ \lambda\langle rs, ms\rangle.$

$\langle \ldots,\quad \ldots,\quad \Pi_{\lambda\langle u',m\rangle.\,\langle m\rangle}\,(\sigma_{\lambda\langle u',m\rangle.\,u'=u}\ rs)\rangle$

$\mathsf{querySpaces}(u) := 6\ \lambda\,\langle rs, ms\rangle.$

$\langle \ldots,\quad \ldots,\quad \Pi_{\lambda\langle u,m,m',a\rangle\,\langle m,a\rangle}\,(rs\ \bowtie_{\lambda\langle u,m\rangle,\langle m',a\rangle.\,m=m'}\ ms)\rangle$

Bound Inference

$\text{querySpace}(m) := 3\ \lambda\langle rs, ms\rangle.$

$\quad\langle \ldots,\quad \ldots,\quad \Pi_{\lambda\langle m',a\rangle.\,\langle a\rangle}\,(\sigma_{\lambda\langle m',a\rangle.\,m'=m}\, ms)\rangle$

$\text{queryReservations}(u) := 4\ \lambda\langle rs, ms\rangle.$

$\quad\langle \ldots,\quad \ldots,\quad \Pi_{\lambda\langle u',m\rangle.\,\langle m\rangle}\,(\sigma_{\lambda\langle u',m\rangle.\,u'=u}\, rs)\rangle$

$\text{querySpaces}(u) := 6\ \lambda\langle rs, ms\rangle.$

$\quad\langle \ldots,\quad \ldots,\quad \Pi_{\lambda\langle u,m,m',a\rangle\,\langle m,a\rangle}\,(rs \bowtie_{\lambda\langle u,m\rangle,\langle m',a\rangle.\,m=m'}\ ms)\rangle$

Bound Inference

$\mathsf{querySpace}(m) := 3\ \lambda\langle rs, ms\rangle.$
$\langle \ldots,\quad \ldots,\quad \Pi_{\lambda\langle m',a\rangle.\ \langle a\rangle}\ (\sigma_{\lambda\langle m',a\rangle.\ m'=m}\ ms)\rangle$
$\mathsf{queryReservations}(u) := 4\ \lambda\langle rs, ms\rangle.$
$\langle \ldots,\quad \ldots,\quad \Pi_{\lambda\langle u',m\rangle.\ \langle m\rangle}\ (\sigma_{\lambda\langle u',m\rangle.\ u'=u}\ rs)\rangle$
$\mathsf{querySpaces}(u) := 6\ \lambda\langle rs, ms\rangle.$
$\langle \ldots,\quad \ldots,\quad \Pi_{\lambda\langle u,m,m',a\rangle\ \langle m,a\rangle}\ (rs\ \bowtie_{\lambda\langle u,m\rangle,\langle m',a\rangle.\ m=m'}\ ms)\rangle$

Solution:
Δms = 3
Δrs = 2

Bound Inference

$\mathsf{querySpace}(m) := 3\ \lambda\langle rs, ms\rangle.$
$\quad \langle \ldots, \quad \ldots, \quad \Pi_{\lambda\langle m',a\rangle.\,\langle a\rangle}\,(\sigma_{\lambda\langle m',a\rangle.\,m'=m}\ ms)\rangle$
$\mathsf{queryReservations}(u) := 4\ \lambda\langle rs, ms\rangle.$
$\quad \langle \ldots, \quad \ldots, \quad \Pi_{\lambda\langle u',m\rangle.\,\langle m\rangle}\,(\sigma_{\lambda\langle u',m\rangle.\,u'=u}\ rs)\rangle$
$\mathsf{querySpaces}(u) := 6\ \lambda\langle rs, ms\rangle.$
$\quad \langle \ldots, \quad \ldots, \quad \Pi_{\lambda\langle u,m,m',a\rangle\,\langle m,a\rangle}\,(rs \bowtie_{\lambda\langle u,m\rangle,\langle m',a\rangle.\,m=m'}\ ms)\rangle$

Solution:
Δms = 3
Δrs = 2

Bound Inference

$\text{querySpace}(m) := 3\ \lambda\langle rs, ms\rangle.$
$\quad \langle \dots, \quad \dots, \quad \Pi_{\lambda\langle m', a\rangle.\,\langle a\rangle}\,(\sigma_{\lambda\langle m', a\rangle.\, m'=m}\ ms)\rangle$
$\text{queryReservations}(u) := 4\ \lambda\langle rs, ms\rangle.$
$\quad \langle \dots, \quad \dots, \quad \Pi_{\lambda\langle u', m\rangle.\,\langle m\rangle}\,(\sigma_{\lambda\langle u', m\rangle.\, u'=u}\ rs)\rangle$
$\text{querySpaces}(u) := 6\ \lambda\langle rs, ms\rangle.$
$\quad \langle \dots, \quad \dots, \quad \Pi_{\lambda\langle u, m, m', a\rangle\,\langle m, a\rangle}\,(rs\ \bowtie_{\lambda\langle u, m\rangle, \langle m', a\rangle.\, m=m'}\ ms)\rangle$

Solution:
Δms = 3
Δrs = 2

CProd

$$\frac{\Gamma \vdash e \rhd \delta, C \qquad \Gamma \vdash e' \rhd \delta', C'}{\Gamma \vdash e \times e' \rhd \delta \times \delta', C \wedge C'}$$

Bound Inference

$$\mathsf{querySpace}(m) := 3\ \lambda\langle rs, ms\rangle.$$
$$\langle \ldots,\quad \ldots,\quad \Pi_{\lambda\langle m',a\rangle.\,\langle a\rangle}\,(\sigma_{\lambda\langle m',a\rangle.\,m'=m}\ ms)\rangle$$
$$\mathsf{queryReservations}(u) := 4\ \lambda\langle rs, ms\rangle.$$
$$\langle \ldots,\quad \ldots,\quad \Pi_{\lambda\langle u',m\rangle.\,\langle m\rangle}\,(\sigma_{\lambda\langle u',m\rangle.\,u'=u}\ rs)\rangle$$
$$\mathsf{querySpaces}(u) := 6\ \lambda\langle rs, ms\rangle.$$
$$\langle \ldots,\quad \ldots,\quad \Pi_{\lambda\langle u,m,m',a\rangle\,\langle m,a\rangle}\,(rs\ \bowtie_{\lambda\langle u,m\rangle,\langle m',a\rangle.\,m=m'}\ ms)\rangle$$

Solution:

Δms = 3

Δrs = 2

$$\textsc{CProd}\quad \frac{\boxed{\Gamma \vdash e \rhd \delta, C} \qquad \Gamma \vdash e' \rhd \delta', C'}{\Gamma \vdash e \times e' \rhd \delta \times \delta', C \wedge C'}$$

Bound Inference

$\mathsf{querySpace}(m) := 3\ \lambda\langle rs, ms\rangle.$
$\langle \ldots,\quad \ldots,\quad \Pi_{\lambda\langle m', a\rangle.\,\langle a\rangle}\,(\sigma_{\lambda\langle m', a\rangle.\ m'=m}\ ms)\rangle$
$\mathsf{queryReservations}(u) := 4\ \lambda\langle rs, ms\rangle.$
$\langle \ldots,\quad \ldots,\quad \Pi_{\lambda\langle u', m\rangle.\,\langle m\rangle}\,(\sigma_{\lambda\langle u', m\rangle.\ u'=u}\ rs)\rangle$
$\mathsf{querySpaces}(u) := 6\ \lambda\langle rs, ms\rangle.$
$\langle \ldots,\quad \ldots,\quad \Pi_{\lambda\langle u, m, m', a\rangle\,\langle m, a\rangle}\,(rs \bowtie_{\lambda\langle u, m\rangle, \langle m', a\rangle.\ m=m'}\ ms)\rangle$

Solution:
Δms = 3
Δrs = 2

CProd

$$\frac{\Gamma \vdash e \rhd \delta, C \qquad \Gamma \vdash e' \rhd \delta', C'}{\Gamma \vdash e \times e' \rhd \delta \times \delta', C \wedge C'}$$

Bound Inference

$\mathsf{querySpace}(m) := 3\ \lambda\langle rs, ms\rangle.$
$\quad \langle ..., \quad ..., \quad \Pi_{\lambda\langle m', a\rangle.\,\langle a\rangle}\,(\sigma_{\lambda\langle m', a\rangle.\,m'=m}\ ms)\rangle$
$\mathsf{queryReservations}(u) := 4\ \lambda\langle rs, ms\rangle.$
$\quad \langle ..., \quad ..., \quad \Pi_{\lambda\langle u', m\rangle.\,\langle m\rangle}\,(\sigma_{\lambda\langle u', m\rangle.\,u'=u}\ rs)\rangle$
$\mathsf{querySpaces}(u) := 6\ \lambda\langle rs, ms\rangle.$
$\quad \langle ..., \quad ..., \quad \Pi_{\lambda\langle u, m, m', a\rangle\,\langle m, a\rangle}\,(rs\ \bowtie_{\lambda\langle u, m\rangle, \langle m', a\rangle.\,m=m'}\ ms)\rangle$

Solution:
Δms = 3
Δrs = 2

CProd

$$\frac{\Gamma \vdash e \rhd \delta, C \qquad \Gamma \vdash e' \rhd \delta', C'}{\Gamma \vdash e \times e' \rhd \delta \times \delta', C \wedge C'}$$

Constrains:
Δms × Δrs ≤ 6
Δms ≤ 3
Δrs ≤ 4

Bound Inference

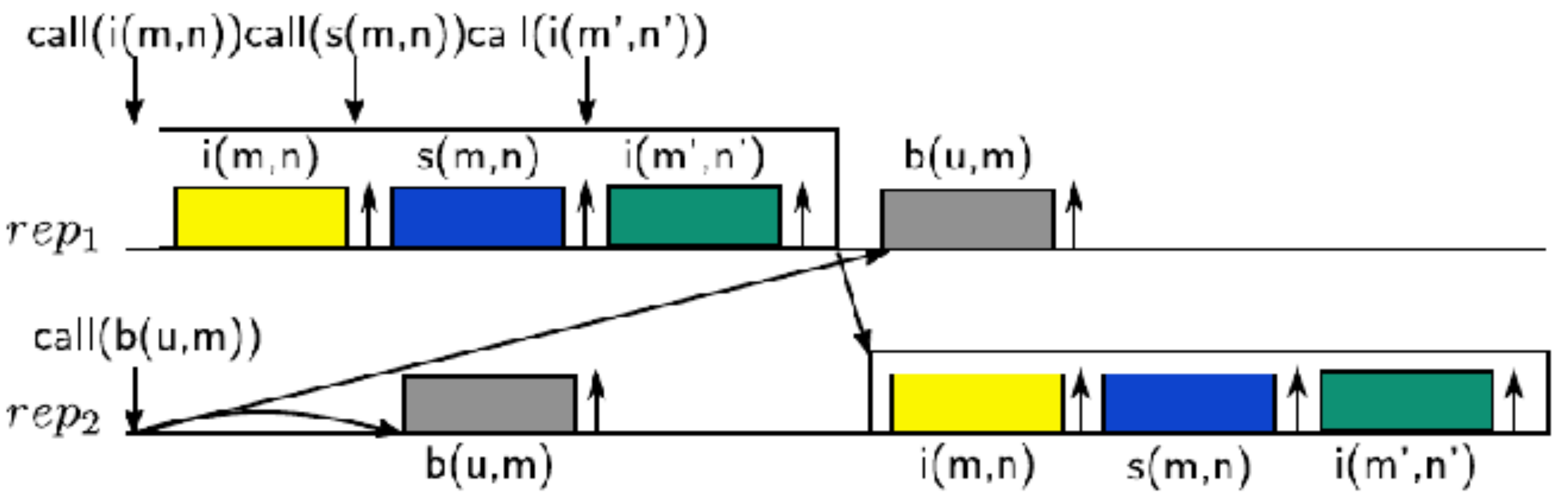
$$\mathsf{querySpace}(m) := 3\ \lambda\langle rs, ms\rangle.$$
$$\langle \ldots,\quad \ldots,\quad \Pi_{\lambda\langle m',a\rangle.\,\langle a\rangle}\,(\sigma_{\lambda\langle m',a\rangle.\, m'=m}\ ms)\rangle$$
$$\mathsf{queryReservations}(u) := 4\ \lambda\langle rs, ms\rangle.$$
$$\langle \ldots,\quad \ldots,\quad \Pi_{\lambda\langle u',m\rangle.\,\langle m\rangle}\,(\sigma_{\lambda\langle u',m\rangle.\, u'=u}\ rs)\rangle$$
$$\mathsf{querySpaces}(u) := 6\ \lambda\langle rs, ms\rangle.$$
$$\langle \ldots,\quad \ldots,\quad \Pi_{\lambda\langle u,m,m',a\rangle\,\langle m,a\rangle}\,(rs\ \bowtie_{\lambda\langle u,m\rangle,\langle m',a\rangle.\, m=m'}\ ms)\rangle$$

Solution:
Δms = 3
Δrs = 2

CProd
$$\frac{\Gamma \vdash e \triangleright \delta, C \qquad \Gamma \vdash e' \triangleright \delta', C'}{\Gamma \vdash e \times e' \triangleright \delta \times \delta', C \wedge C'}$$

Constrains:
Δms $\times$ Δrs ≤ 6
Δms ≤ 3
Δrs ≤ 4

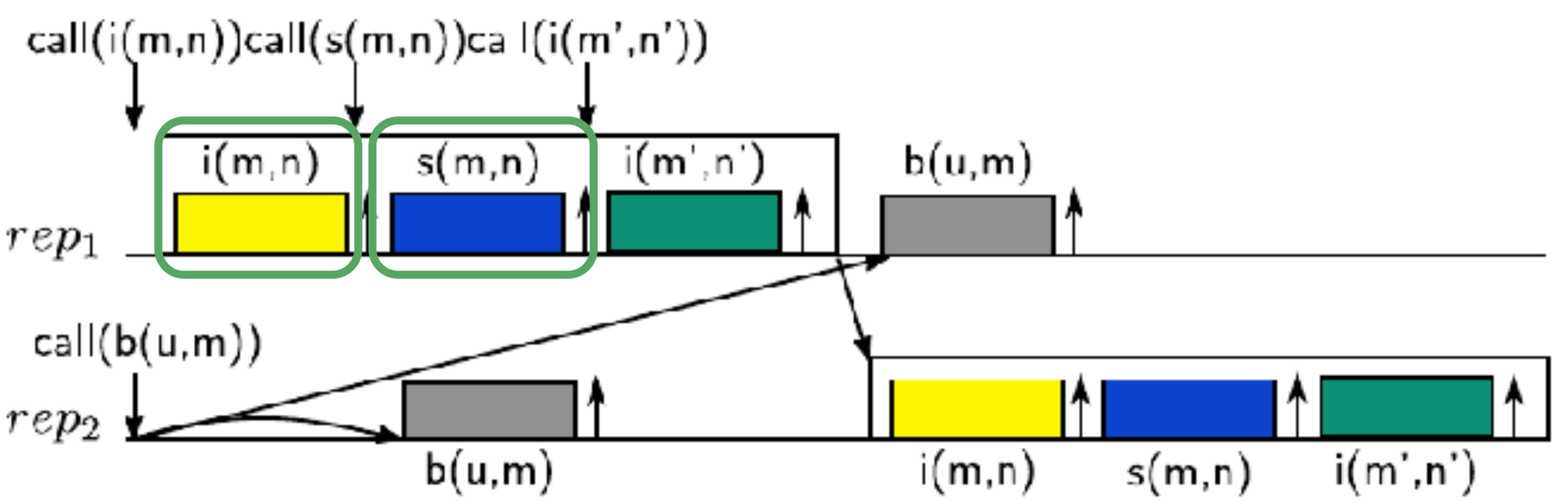
Coordination Conditions and Operational Semantics



CallLocal

$$\frac{\begin{array}{c} \mathcal{P}(\sigma, c) \qquad c(\sigma) = \langle \_, \sigma', v\rangle \\ c' = c \cdot \mathsf{call}(r) \\ \mathsf{AllSComm}(c) \\ \mathsf{InvSuff}(c') \qquad \mathsf{LetPRComm}(c') \\ \mathsf{call}' = \mathsf{call}[r \mapsto c'] \\ \mathsf{xs}' = \begin{cases} \mathsf{xs}[n \mapsto (\mathsf{xs}(n) ::: r)] & \text{if } \mathsf{call}(r) = \mathsf{id} \\ \mathsf{xs} & \text{else} \end{cases} \\ \mathsf{InBound}_{\langle \mathsf{orig}, \mathsf{call}' \rangle}(\mathsf{xs}', n) \end{array}}{\begin{array}{c} (h[n \mapsto (x \leftarrow c; s,\ \sigma,\ r)], t, \mathsf{xs}, \mathsf{orig}, \mathsf{call}) \\ \xrightarrow{n, r, c} \\ (h[n \mapsto (s[x \mapsto v],\ \sigma',\ r)], t, \mathsf{xs}', \mathsf{orig}, \mathsf{call}') \end{array}}$$

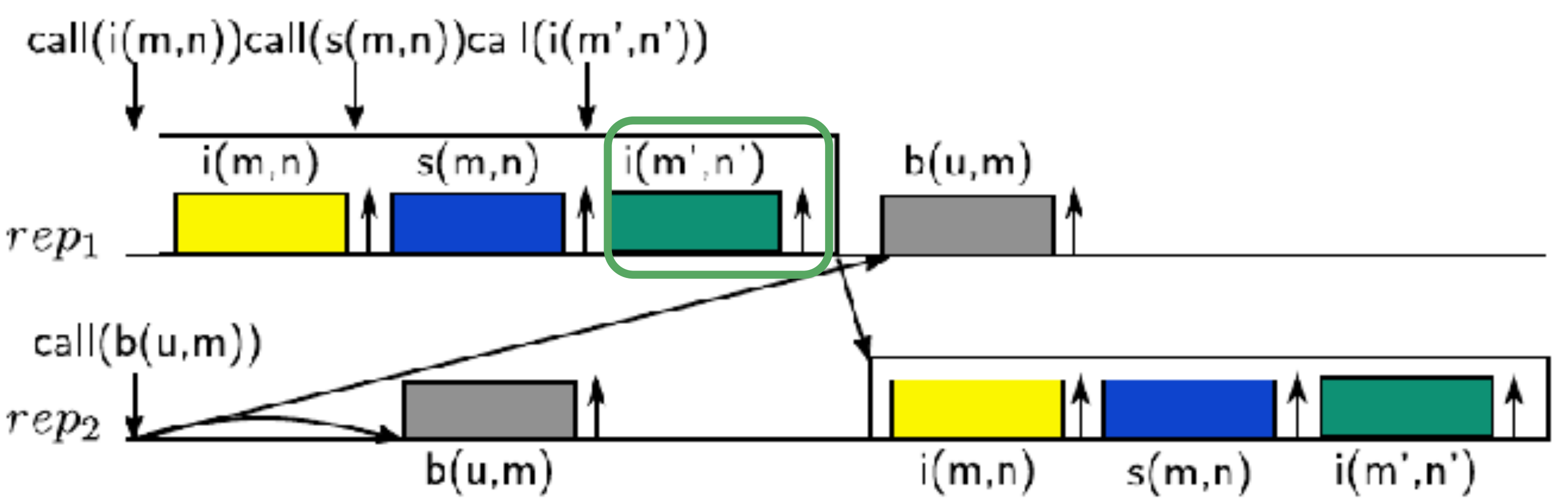
Coordination Conditions and Operational Semantics



CallLocal

$$\frac{\begin{array}{c} \mathcal{P}(\sigma, c) \qquad c(\sigma) = \langle \_, \sigma', v\rangle \\ c' = c \cdot \mathsf{call}(r) \\ \boxed{\mathsf{AllSComm}(c)} \\ \mathsf{InvSuff}(c') \qquad \mathsf{LetPRComm}(c') \\ \mathsf{call}' = \mathsf{call}[r \mapsto c'] \\ \mathsf{xs}' = \begin{cases} \mathsf{xs}[n \mapsto (\mathsf{xs}(n) ::: r)] & \text{if } \mathsf{call}(r) = \mathsf{id} \\ \mathsf{xs} & \text{else} \end{cases} \\ \mathsf{InBound}_{\langle \mathsf{orig}, \mathsf{call}' \rangle}(\mathsf{xs}', n) \end{array}}{\begin{array}{c} (h[n \mapsto (x \leftarrow c; s,\ \sigma,\ r)], t, \mathsf{xs}, \mathsf{orig}, \mathsf{call}) \\ \xrightarrow{n, r, c} \\ (h[n \mapsto (s[x \mapsto v],\ \sigma',\ r)], t, \mathsf{xs}', \mathsf{orig}, \mathsf{call}') \end{array}}$$

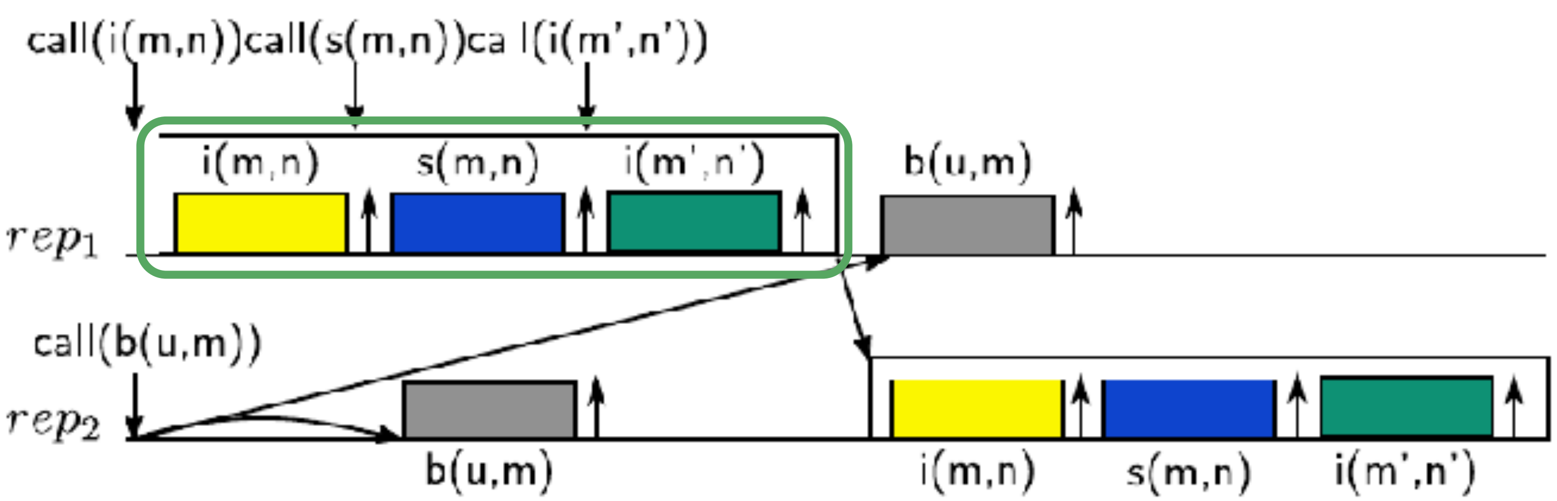
Coordination Conditions and Operational Semantics



CallLocal

$$\frac{\begin{array}{c} \mathcal{P}(\sigma, c) \qquad c(\sigma) = \langle \_, \sigma', v \rangle \\ c' = c \cdot \mathsf{call}(r) \\ \mathsf{AllSComm}(c) \\ \mathsf{InvSuff}(c') \qquad \mathsf{LetPRComm}(c') \\ \mathsf{call}' = \mathsf{call}[r \mapsto c'] \\ \mathsf{xs}' = \begin{cases} \mathsf{xs}[n \mapsto (\mathsf{xs}(n) ::: r)] & \text{if } \mathsf{call}(r) = \mathsf{id} \\ \mathsf{xs} & \text{else} \end{cases} \\ \boxed{\mathsf{InBound}_{\langle \mathsf{orig}, \mathsf{call}' \rangle}(\mathsf{xs}', n)} \end{array}}{\begin{array}{c} (h[n \mapsto (x \leftarrow c; s,\ \sigma,\ r)], t, \mathsf{xs}, \mathsf{orig}, \mathsf{call}) \\ \xrightarrow{n, r, c} \\ (h[n \mapsto (s[x \mapsto v],\ \sigma',\ r)], t, \mathsf{xs}', \mathsf{orig}, \mathsf{call}') \end{array}}$$

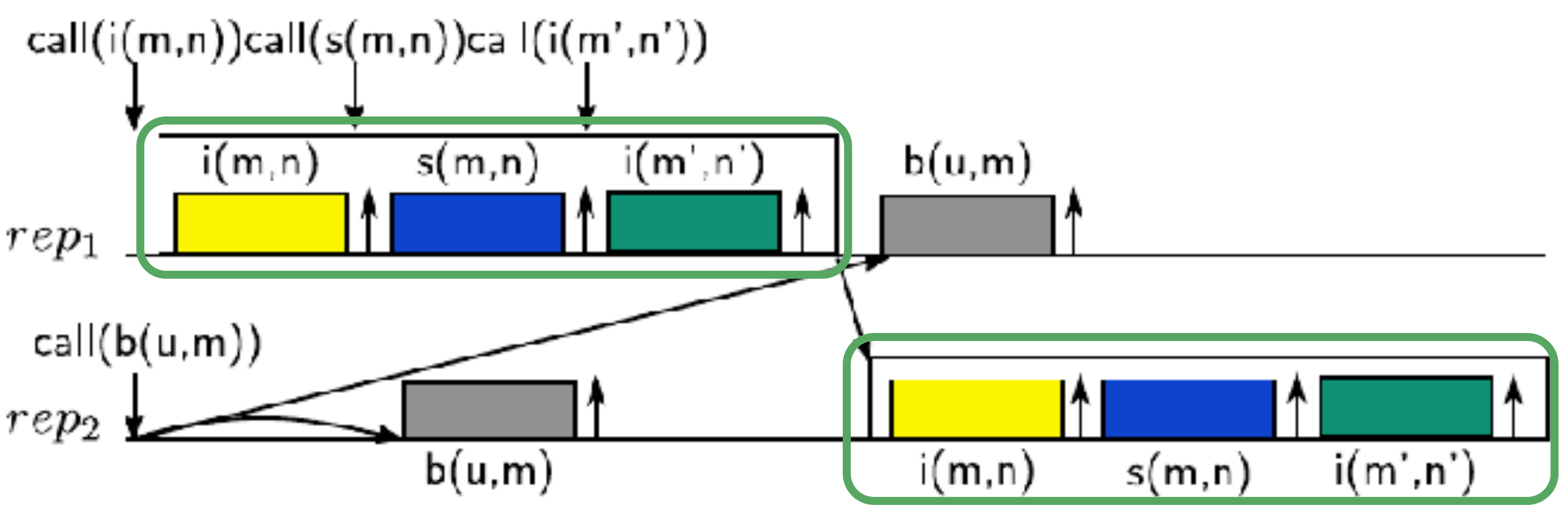
Coordination Conditions and Operational Semantics



CALLLOCAL

$$\frac{\begin{array}{c} \mathcal{P}(\sigma, c) \qquad c(\sigma) = \langle \_, \sigma', v \rangle \\ c' = c \cdot \mathsf{call}(r) \\ \mathsf{AllSComm}(c) \\ \boxed{\mathsf{InvSuff}(c')} \qquad \mathsf{LetPRComm}(c') \\ \mathsf{call}' = \mathsf{call}[r \mapsto c'] \\ \mathsf{xs}' = \begin{cases} \mathsf{xs}[n \mapsto (\mathsf{xs}(n) ::: r)] & \text{if } \mathsf{call}(r) = \mathsf{id} \\ \mathsf{xs} & \text{else} \end{cases} \\ \mathsf{InBound}_{\langle \mathsf{orig}, \mathsf{call}' \rangle}(\mathsf{xs}', n) \end{array}}{\begin{array}{c} (h[n \mapsto (x \leftarrow c; s,\ \sigma,\ r)], t, \mathsf{xs}, \mathsf{orig}, \mathsf{call}) \\ \xrightarrow{n, r, c} \\ (h[n \mapsto (s[x \mapsto v],\ \sigma',\ r)], t, \mathsf{xs}', \mathsf{orig}, \mathsf{call}') \end{array}}$$

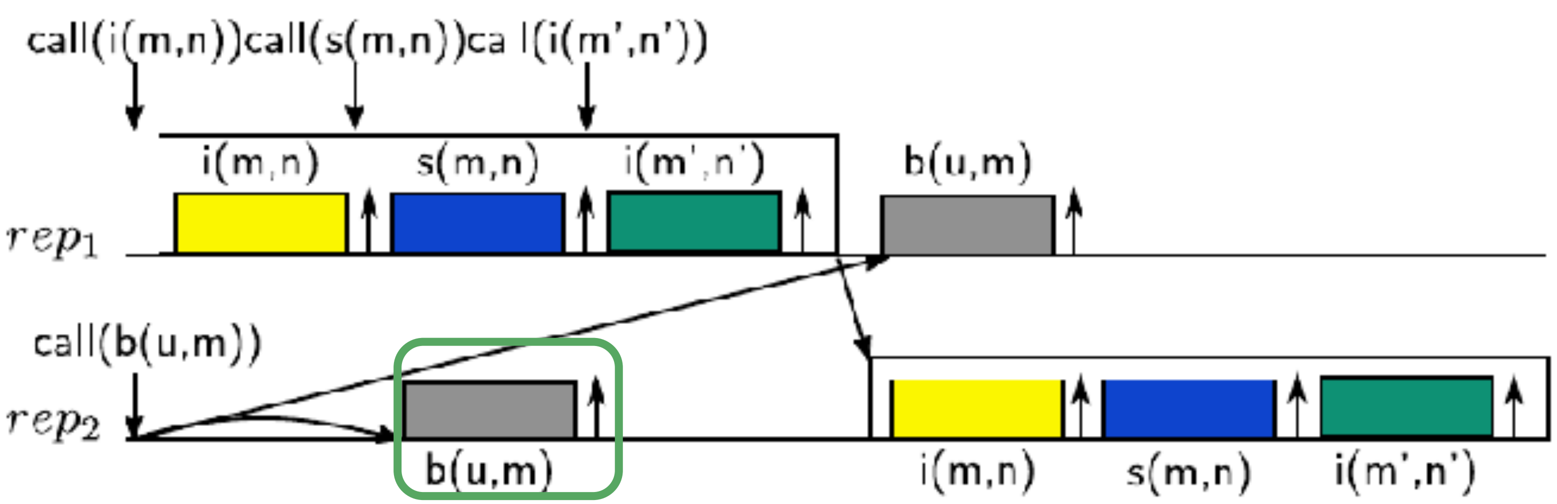
Coordination Conditions and Operational Semantics



CallLocal

$$\frac{\begin{array}{c} \mathcal{P}(\sigma, c) \qquad c(\sigma) = \langle \_, \sigma', v \rangle \\ c' = c \cdot \mathsf{call}(r) \\ \mathsf{AllSComm}(c) \\ \mathsf{InvSuff}(c') \qquad \mathsf{LetPRComm}(c') \\ \mathsf{call}' = \mathsf{call}[r \mapsto c'] \\ \mathsf{xs}' = \begin{cases} \mathsf{xs}[n \mapsto (\mathsf{xs}(n) ::: r)] & \text{if } \mathsf{call}(r) = \mathsf{id} \\ \mathsf{xs} & \text{else} \end{cases} \\ \mathsf{InBound}_{\langle \mathsf{orig}, \mathsf{call}' \rangle}(\mathsf{xs}', n) \end{array}}{\begin{array}{c} (h[n \mapsto (x \leftarrow c; s,\ \sigma,\ r)], t, \mathsf{xs}, \mathsf{orig}, \mathsf{call}) \\ \xrightarrow{n,r,c} \\ (h[n \mapsto (s[x \mapsto v],\ \sigma',\ r)], t, \mathsf{xs}', \mathsf{orig}, \mathsf{call}') \end{array}}$$

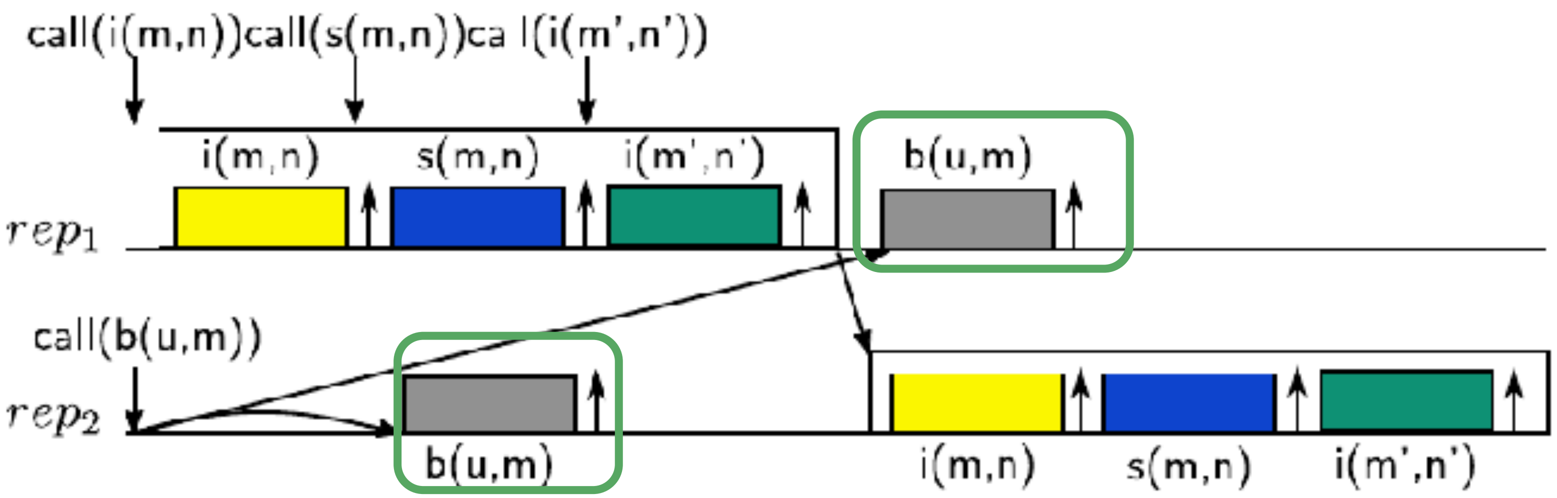
Coordination Conditions and Operational Semantics



CallLocal

$$\frac{\begin{array}{c}\mathcal{P}(\sigma, c) \qquad c(\sigma) = \langle \_, \sigma', v\rangle \\ c' = c \cdot \mathsf{call}(r) \\ \mathsf{AllSComm}(c) \\ \mathsf{InvSuff}(c') \qquad \boxed{\mathsf{LetPRComm}(c')} \\ \mathsf{call}' = \mathsf{call}[r \mapsto c'] \\ \mathsf{xs}' = \begin{cases} \mathsf{xs}[n \mapsto (\mathsf{xs}(n) ::: r)] & \text{if } \mathsf{call}(r) = \mathsf{id} \\ \mathsf{xs} & \text{else} \end{cases} \\ \mathsf{InBound}_{\langle \mathsf{orig}, \mathsf{call}' \rangle}(\mathsf{xs}', n)\end{array}}{\begin{array}{c}(h[n \mapsto (x \leftarrow c; s,\ \sigma,\ r)], t, \mathsf{xs}, \mathsf{orig}, \mathsf{call}) \\ \xrightarrow{n,r,c} \\ (h[n \mapsto (s[x \mapsto v],\ \sigma',\ r)], t, \mathsf{xs}', \mathsf{orig}, \mathsf{call}')\end{array}}$$

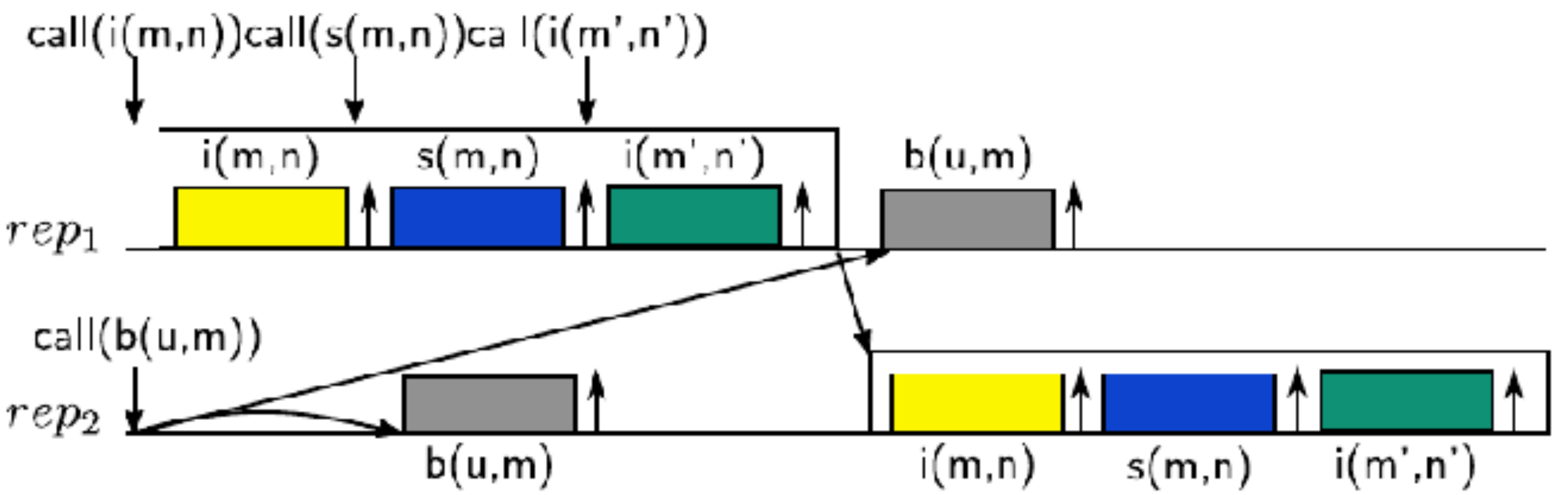
Coordination Conditions and Operational Semantics



CallLocal

$$\frac{\begin{array}{c} \mathcal{P}(\sigma, c) \qquad c(\sigma) = \langle \_, \sigma', v\rangle \\ c' = c \cdot \mathsf{call}(r) \\ \mathsf{AllSComm}(c) \\ \mathsf{InvSuff}(c') \qquad \boxed{\mathsf{LetPRComm}(c')} \\ \mathsf{call}' = \mathsf{call}[r \mapsto c'] \\ \mathsf{xs}' = \begin{cases} \mathsf{xs}[n \mapsto (\mathsf{xs}(n) ::: r)] & \text{if } \mathsf{call}(r) = \mathsf{id} \\ \mathsf{xs} & \text{else} \end{cases} \\ \mathsf{InBound}_{\langle \mathsf{orig}, \mathsf{call}' \rangle}(\mathsf{xs}', n) \end{array}}{\begin{array}{c} (h[n \mapsto (x \leftarrow c; s,\ \sigma,\ r)], t, \mathsf{xs}, \mathsf{orig}, \mathsf{call}) \\ \xrightarrow{n, r, c} \\ (h[n \mapsto (s[x \mapsto v],\ \sigma',\ r)], t, \mathsf{xs}', \mathsf{orig}, \mathsf{call}') \end{array}}$$

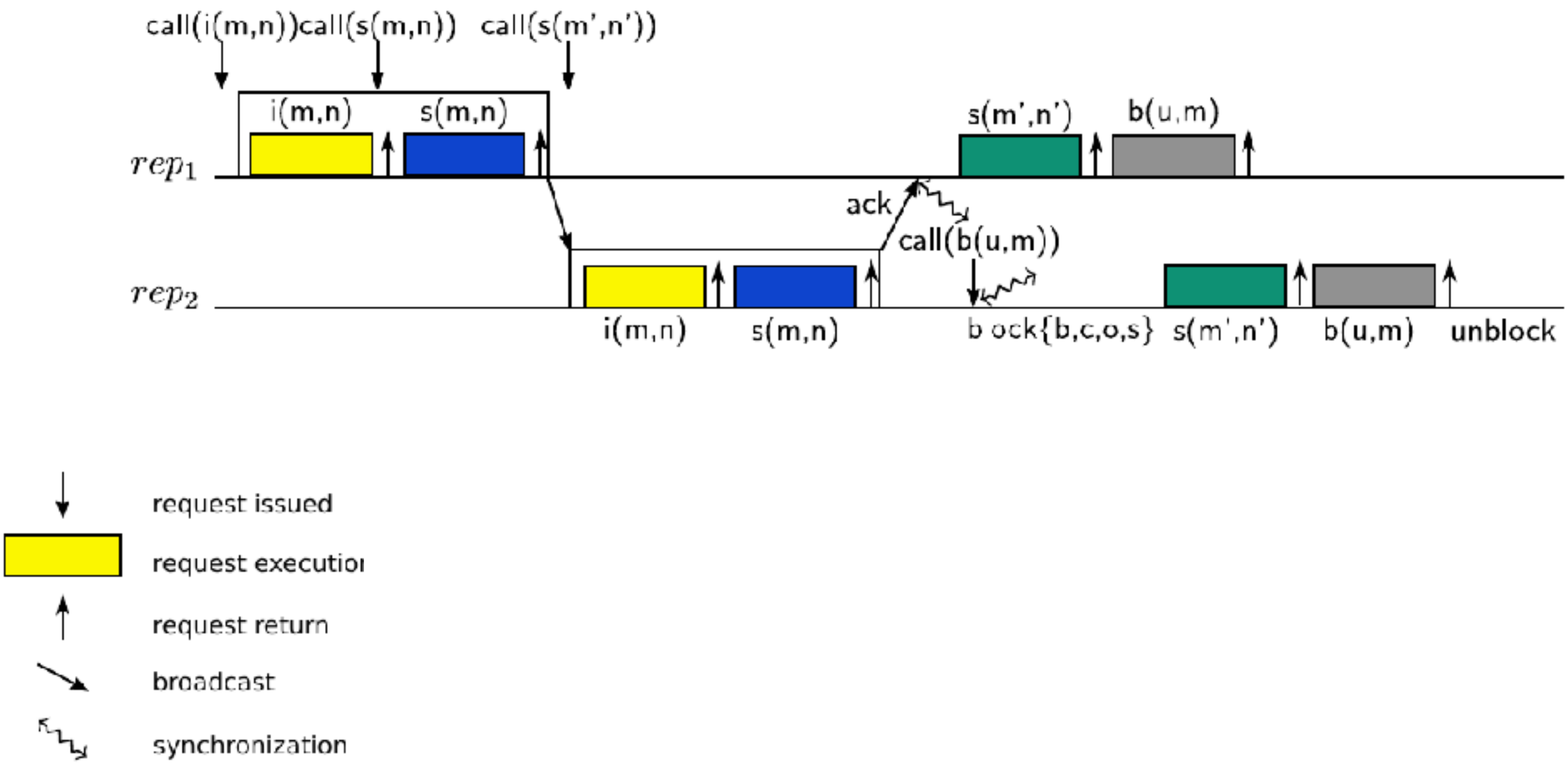
Coordination Conditions and Operational Semantics



CallLocal

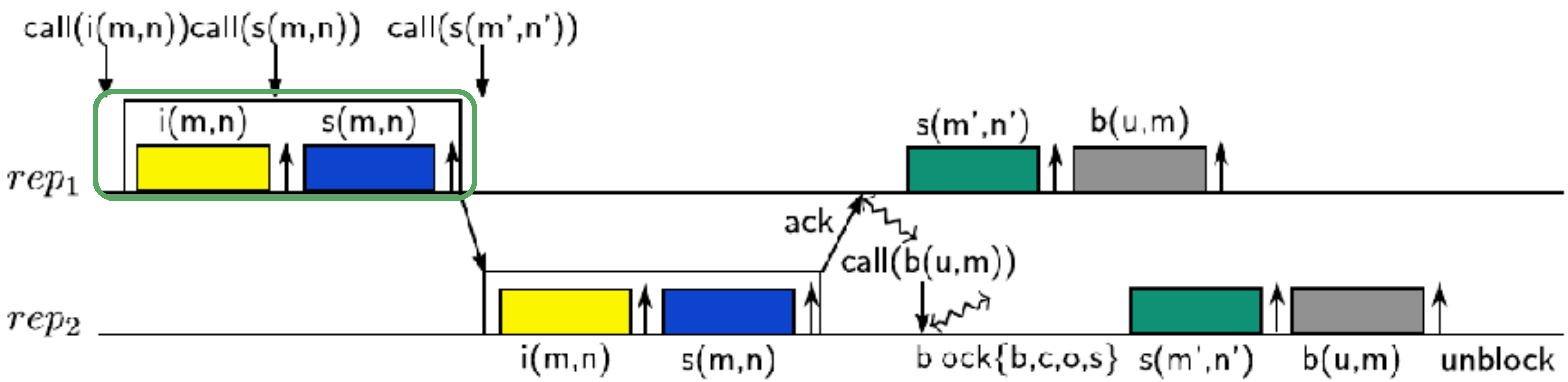
$$\frac{\begin{array}{c} \mathcal{P}(\sigma, c) \qquad c(\sigma) = \langle \_, \sigma', v\rangle \\ c' = c \cdot \mathsf{call}(r) \\ \mathsf{AllSComm}(c) \\ \mathsf{InvSuff}(c') \qquad \mathsf{LetPRComm}(c') \\ \mathsf{call}' = \mathsf{call}[r \mapsto c'] \\ \mathsf{xs}' = \begin{cases} \mathsf{xs}[n \mapsto (\mathsf{xs}(n) ::: r)] & \text{if } \mathsf{call}(r) = \mathsf{id} \\ \mathsf{xs} & \text{else} \end{cases} \\ \mathsf{InBound}_{\langle \mathsf{orig}, \mathsf{call}'\rangle}(\mathsf{xs}', n) \end{array}}{\begin{array}{c} (h[n \mapsto (x \leftarrow c; s,\ \sigma,\ r)], t, \mathsf{xs}, \mathsf{orig}, \mathsf{call}) \\ \xrightarrow{n, r, c} \\ (h[n \mapsto (s[x \mapsto v],\ \sigma',\ r)], t, \mathsf{xs}', \mathsf{orig}, \mathsf{call}') \end{array}}$$

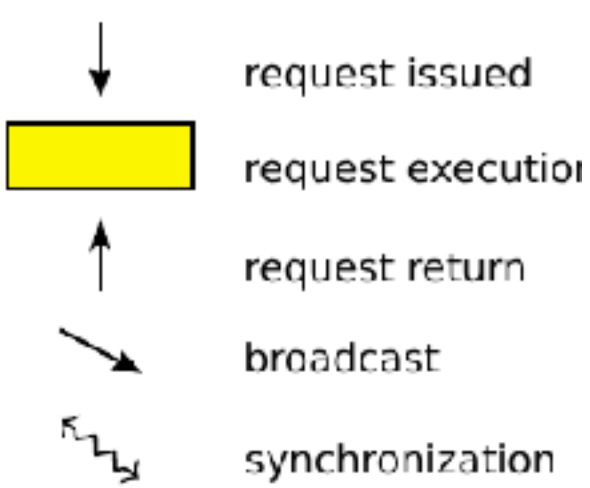
Protocol



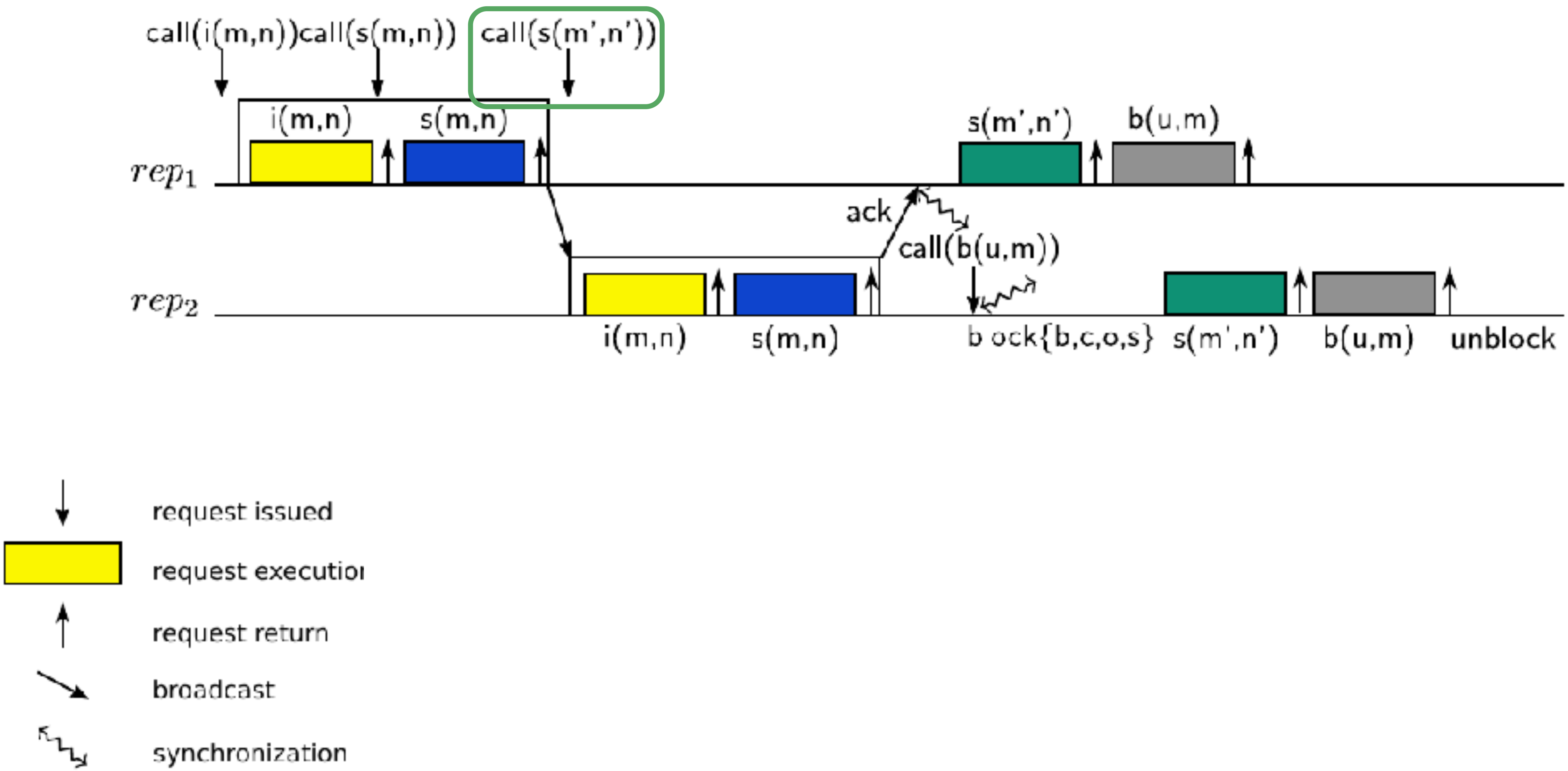
call(i(m,n))
call(s(m,n))
call(s(m',n'))
i(m,n)
s(m,n)
s(m',n')
b(u,m)
rep_1
rep_2
ack
call(b(u,m))
i(m,n)
s(m,n)
b ock{b,c,o,s}
s(m',n')
b(u,m)
unblock
request issued
request executio
request return
broadcast
synchronization

Protocol


call(i(m,n))call(s(m,n)) call(s(m',n'))
i(m,n)
s(m,n)
s(m',n')
b(u,m)
rep1
ack
call(b(u,m))
rep2
i(m,n)
s(m,n)
b ock{b,c,o,s}
s(m',n')
b(u,m)
unblock

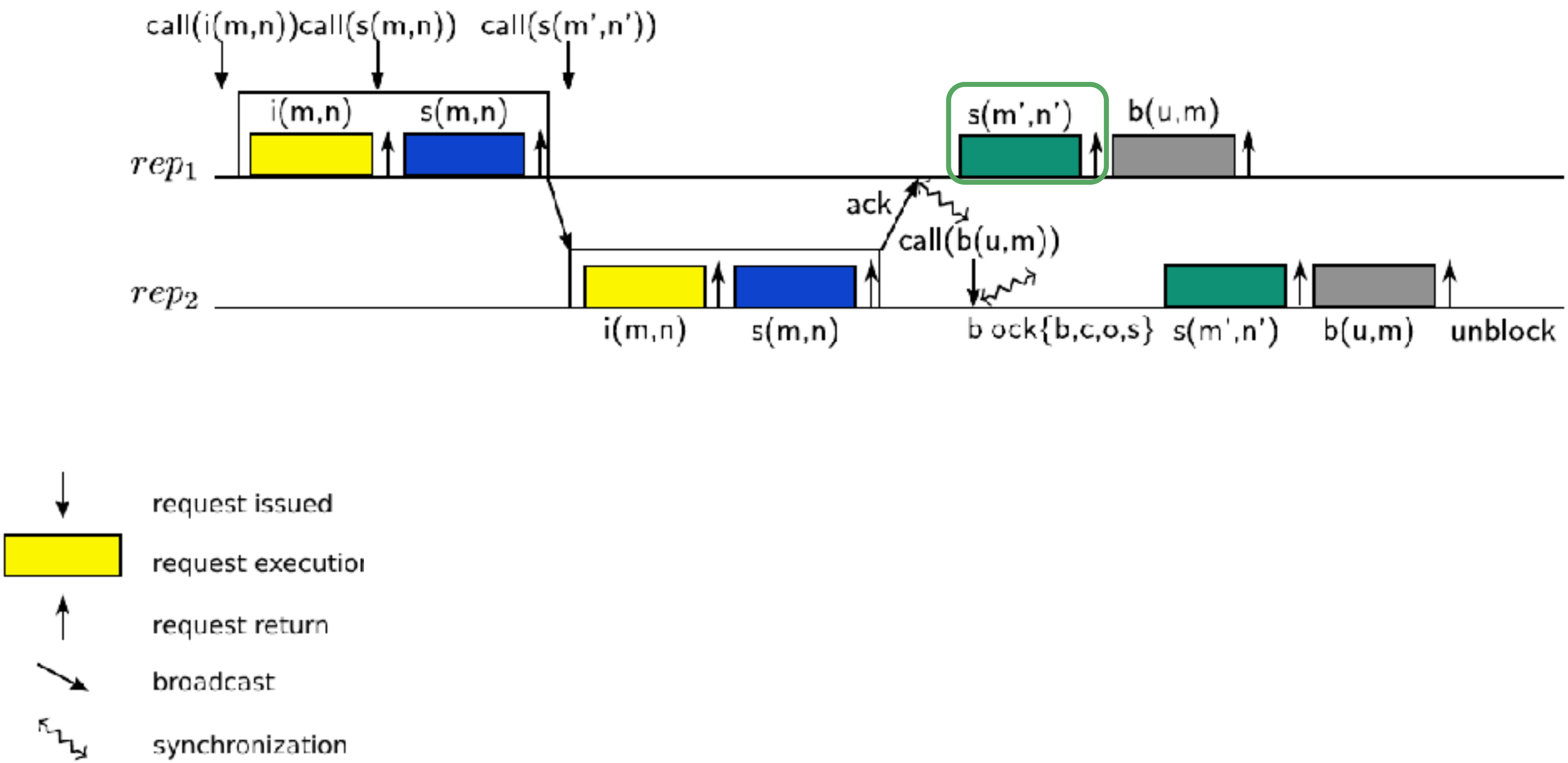

request issued
request executioi
request return
broadcast
synchronization

Protocol



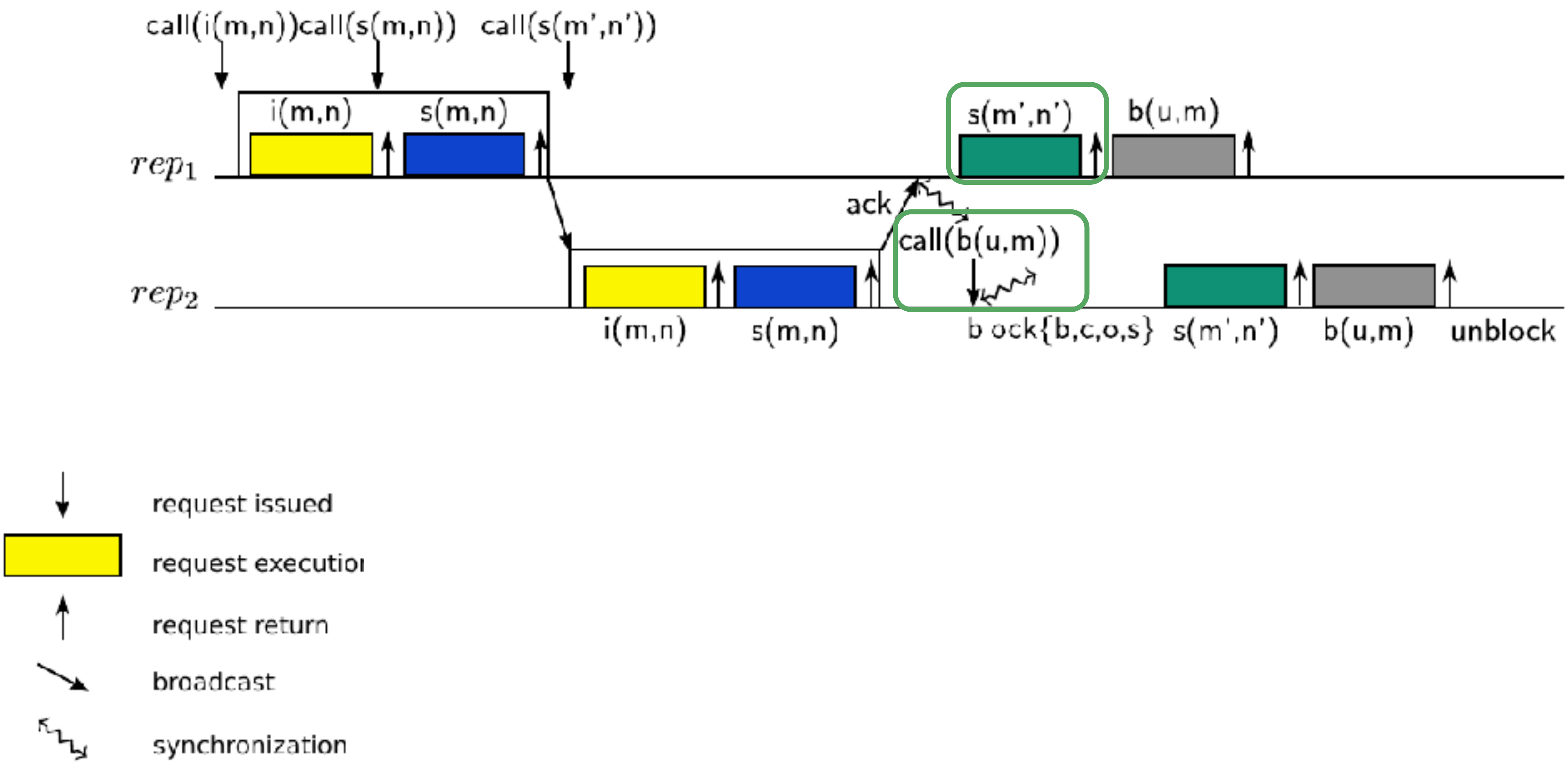
call(i(m,n))
call(s(m,n))
call(s(m',n'))
i(m,n)
s(m,n)
rep1
s(m',n')
b(u,m)
ack
call(b(u,m))
rep2
i(m,n)
s(m,n)
b ock{b,c,o,s}
s(m',n')
b(u,m)
unblock
request issued
request executio
request return
broadcast
synchronization

Protocol



call(i(m,n))call(s(m,n))
call(s(m',n'))
i(m,n)
s(m,n)
rep_1
s(m',n')
b(u,m)
ack
call(b(u,m))
rep_2
i(m,n)
s(m,n)
b ock{b,c,o,s}
s(m',n')
b(u,m)
unblock
request issued
request executioi
request return
broadcast
synchronization

Protocol

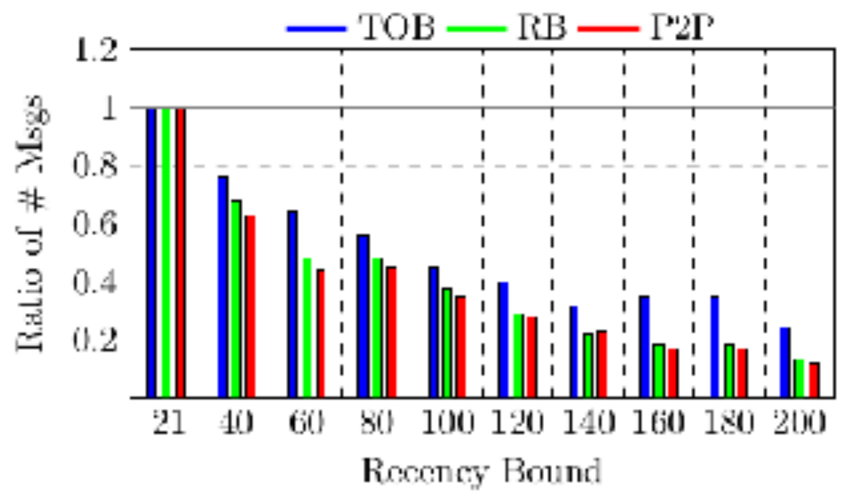


call(i(m,n))
call(s(m,n))
call(s(m',n'))
i(m,n)
s(m,n)
s(m',n')
b(u,m)
rep1
ack
call(b(u,m))
rep2
i(m,n)
s(m,n)
b ock{b,c,o,s}
s(m',n')
b(u,m)
unblock
request issued
request executioi
request return
broadcast
synchronization

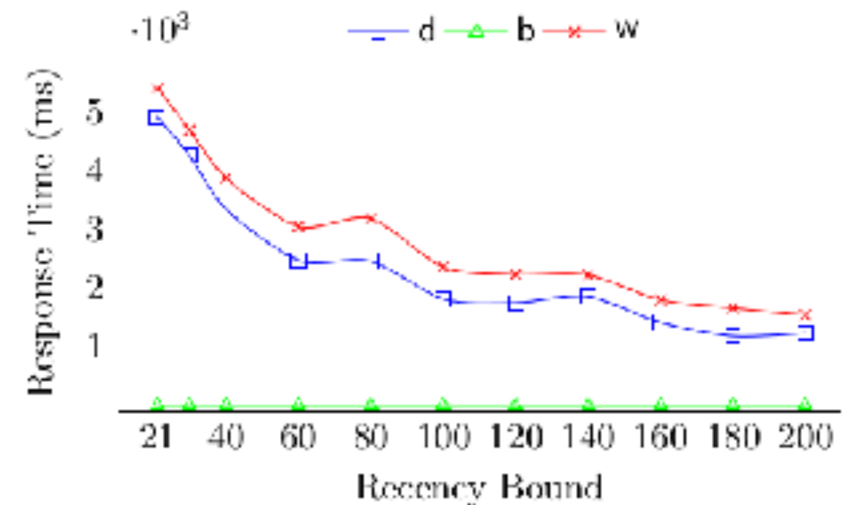
Experimental Results

Bank account

(a)

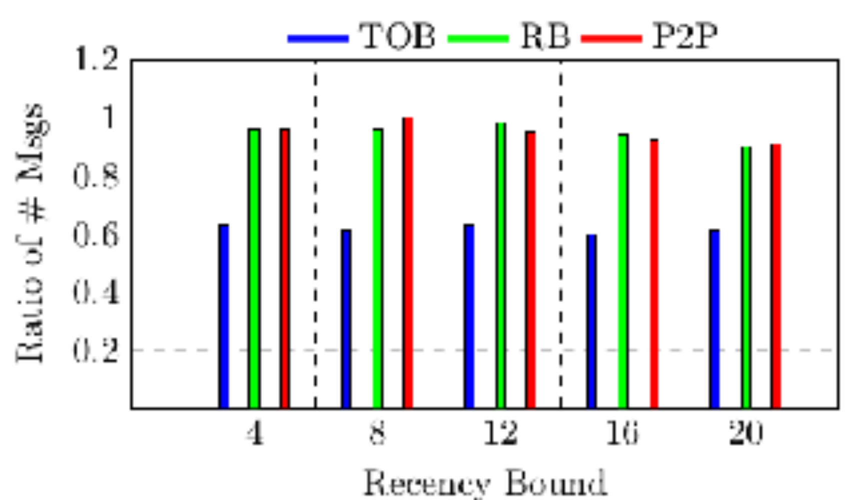


(b)

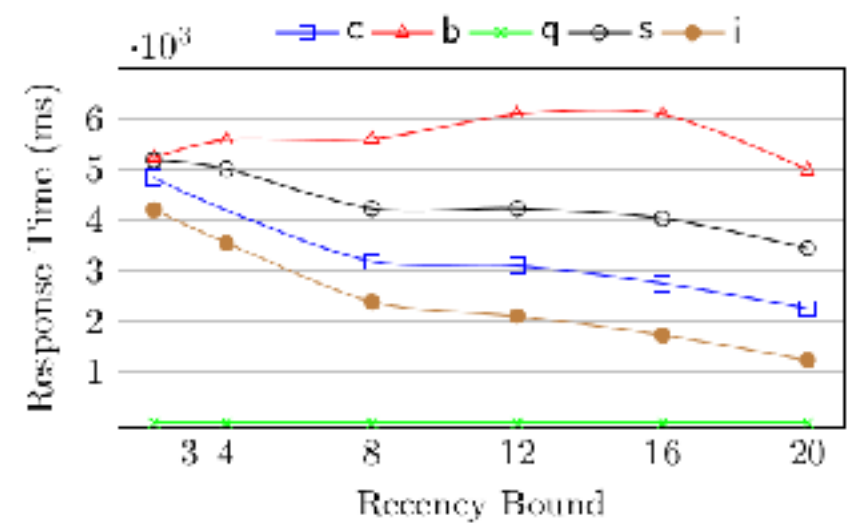


Movie booking

(c)



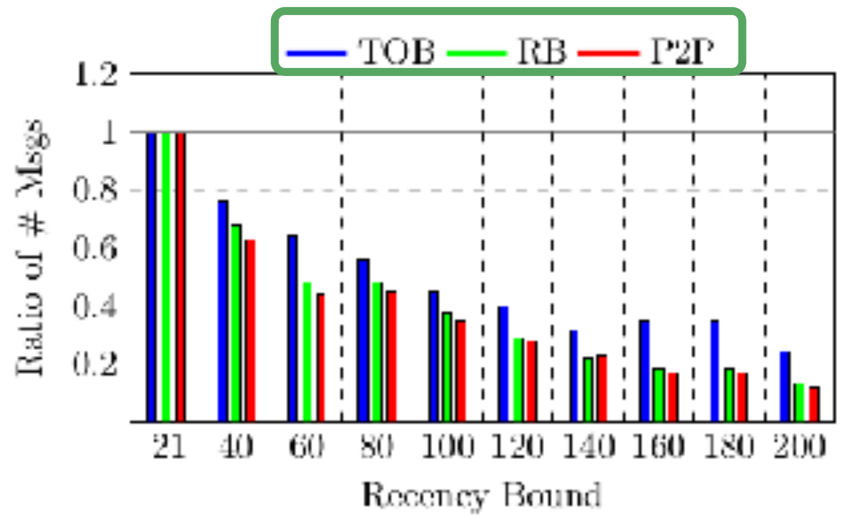
(d)



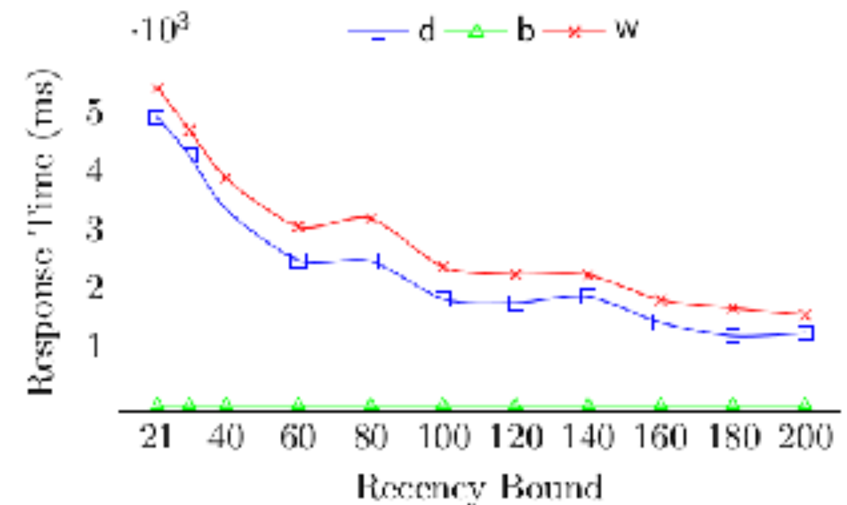
Experimental Results

Bank account

(a)

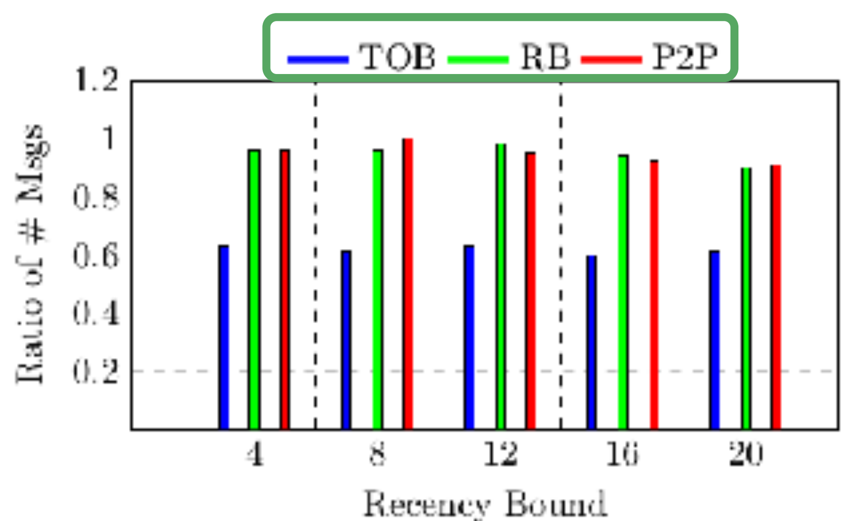


(b)

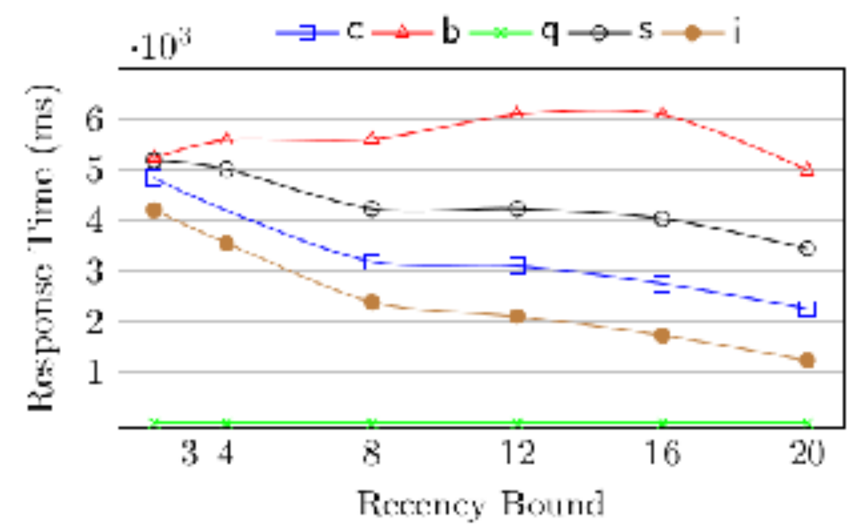


Movie booking

(c)

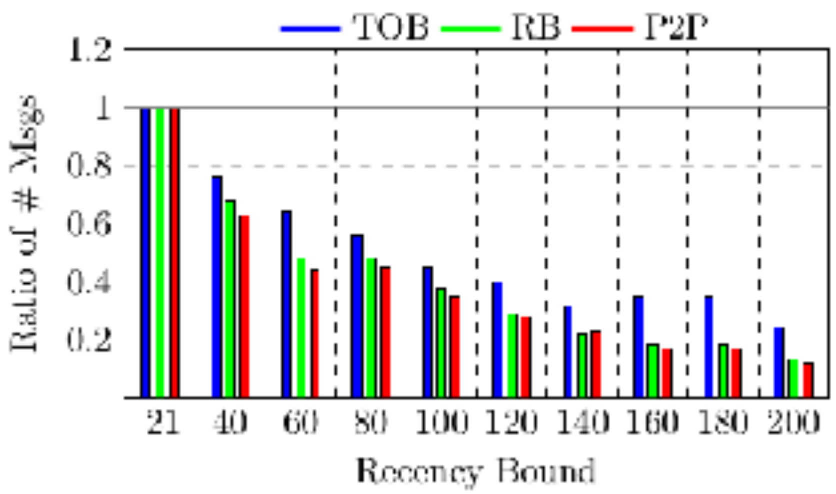


(d)

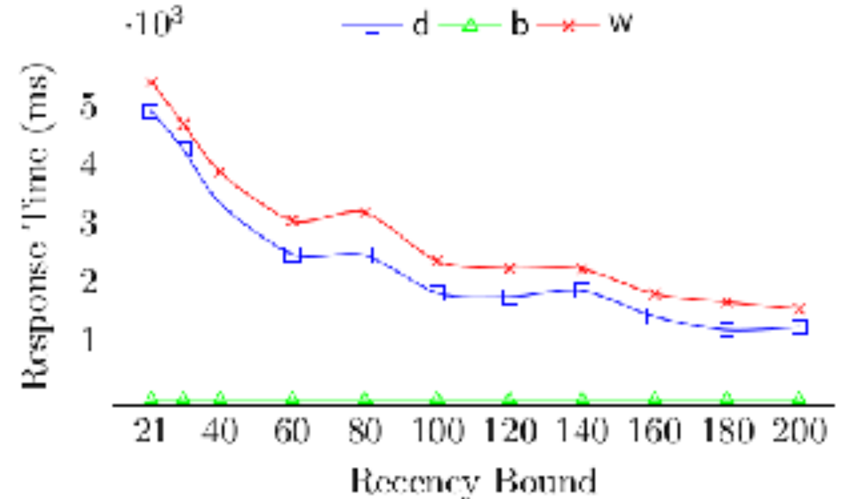


Experimental Results

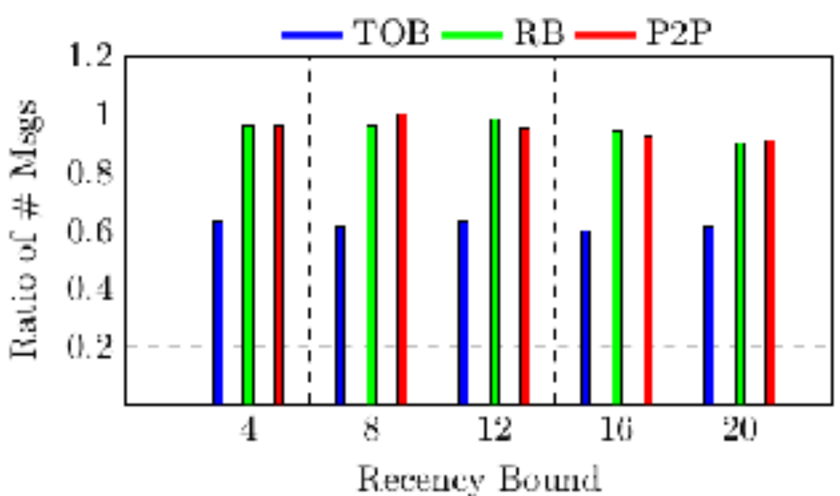
Bank account (a)



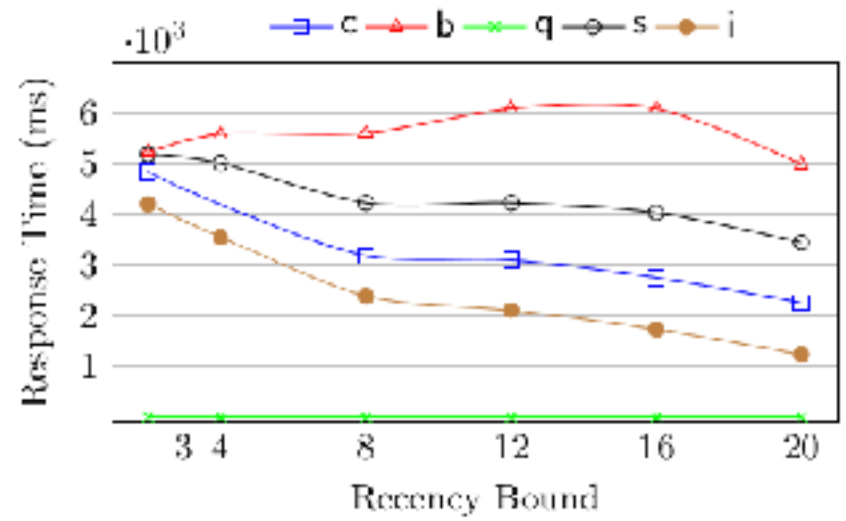
(b)



Movie booking (c)

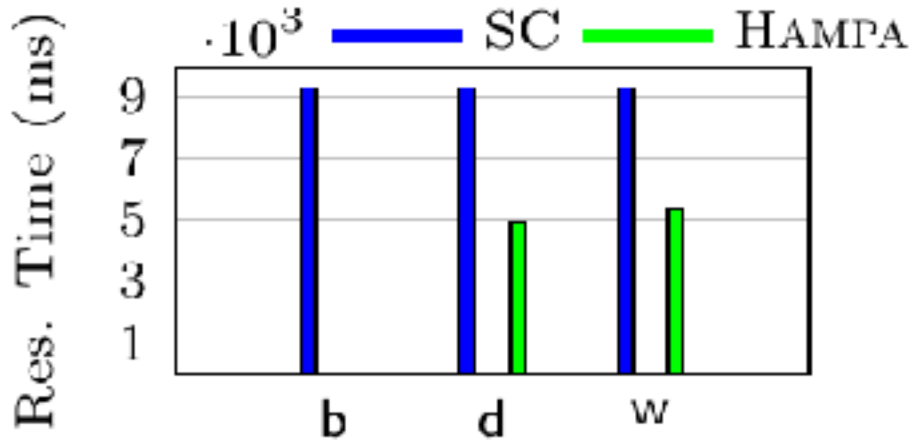


(d)

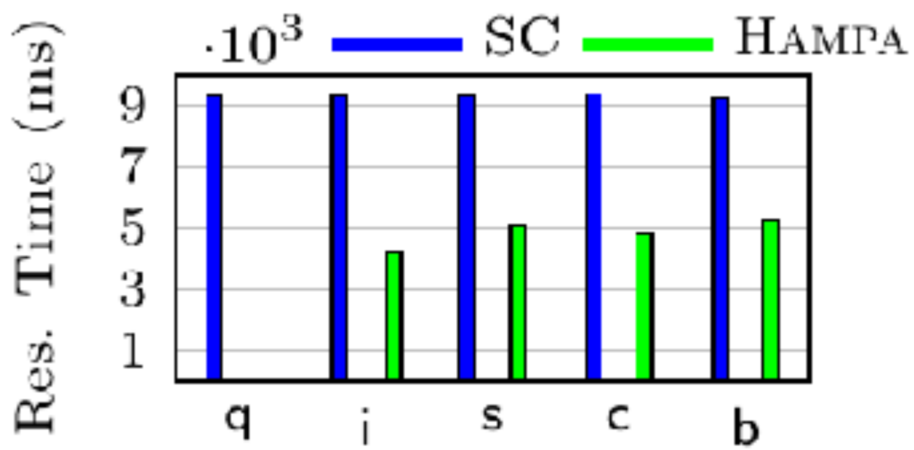


Experimental Results

Bank account



Movie booking



State of the art

	Convergence	Integrity	Recency	Synchronization avoidance	Communication avoidance
Strong consistency	✓	✓	✓	✗	✗
Eventual consistency / CRDT	✓	✗	✗	✓	✗
Sieve, Indigo, CISE, Hamsaz, Soteria	✓	✓	✗	✓	✗
TACT, TRAPP, FRACT, PBS	✓	✗	✓	✗	✓
Hampa	✓	✓	✓	✓	✓

State of the art

	Convergence	Integrity	Recency	Synchronization avoidance	Communication avoidance
Strong consistency	✓	✓	✓	✗	✗
Eventual consistency / CRDT	✓	✗	✗	✓	✗
Sieve, Indigo, CISE, Hamsaz, Soteria	✓	✓	✗	✓	✗
TACT, TRAPP, FRACT, PBS	✓	✗	✓	✗	✓
Hampa	✓	✓	✓	✓	✓

UCR | Computer Science and Engineering

State of the art

	Convergence	Integrity	Recency	Synchronization avoidance	Communication avoidance
Strong consistency	✓	✓	✓	✗	✗
Eventual consistency / CRDT	✓	✗	✗	✓	✗
Sieve, Indigo, CISE, Hamsaz, Soteria	✓	✓	✗	✓	✗
TACT, TRAPP, FRACT, PBS	✓	✗	✓	✗	✓
Hampa	✓	✓	✓	✓	✓

UCR | Computer Science and Engineering

State of the art

	Convergence	Integrity	Recency	Synchronization avoidance	Communication avoidance
Strong consistency	✓	✓	✓	✗	✗
Eventual consistency / CRDT	✓	✗	✗	✓	✗
Sieve, Indigo, CISE, Hamsaz, Soteria	✓	✓	✗	✓	✗
TACT, TRAPP, FRACT, PBS	✓	✗	✓	✗	✓
Hampa	✓	✓	✓	✓	✓

UCR | Computer Science and Engineering

State of the art

	Convergence	Integrity	Recency	Synchronization avoidance	Communication avoidance
Strong consistency	✓	✓	✓	✗	✗
Eventual consistency / CRDT	✓	✗	✗	✓	✗
Sieve, Indigo, CISE, Hamsaz, Soteria	✓	✓	✗	✓	✗
TACT, TRAPP, FRACT, PBS	✓	✗	✓	✗	✓
Hampa	✓	✓	✓	✓	✓

State of the art

	Convergence	Integrity	Recency	Synchronization avoidance	Communication avoidance
Strong consistency	✓	✓	✓	✗	✗
Eventual consistency / CRDT	✓	✗	✗	✓	✗
Sieve, Indigo, CISE, Hamsaz, Soteria	✓	✓	✗	✓	✗
TACT, TRAPP, FRACT, PBS	✓	✗	✓	✗	✓
Hampa	✓	✓	✓	✓	✓

State of the art

	Convergence	Integrity	Recency	Synchronization avoidance	Communication avoidance
Strong consistency	✓	✓	✓	✗	✗
Eventual consistency / CRDT	✓	✗	✗	✓	✗
Sieve, Indigo, CISE, Hamsaz, Soteria	✓	✓	✗	✓	✗
TACT, TRAPP, FRACT, PBS	✓	✗	✓	✗	✓
Hampa	✓	✓	✓	✓	✓

State of the art

	Convergence	Integrity	Recency	Synchronization avoidance	Communication avoidance
Strong consistency	✓	✓	✓	✗	✗
Eventual consistency / CRDT	✓	✗	✗	✓	✗
Sieve, Indigo, CISE, Hamsaz, Soteria	✓	✓	✗	✓	✗
TACT, TRAPP, FRACT, PBS	✓	✗	✓	✗	✓
Hampa	✓	✓	✓	✓	✓

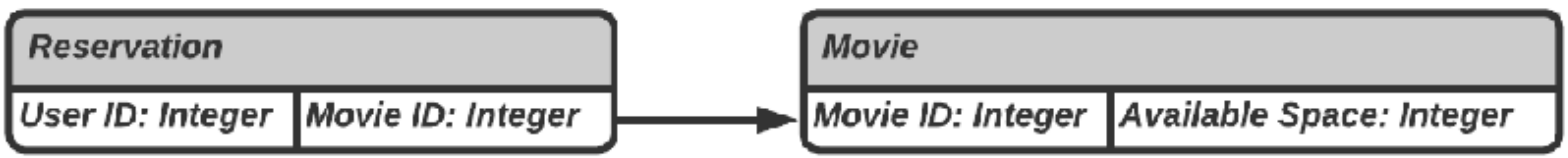
State of the art

	Convergence	Integrity	Recency	Synchronization avoidance	Communication avoidance
Strong consistency	✓	✓	✓	✗	✗
Eventual consistency / CRDT	✓	✗	✗	✓	✗
Sieve, Indigo, CISE, Hamsaz, Soteria	✓	✓	✗	✓	✗
TACT, TRAPP, FRACT, PBS	✓	✗	✓	✗	✓
Hampa	✓	✓	✓	✓	✓

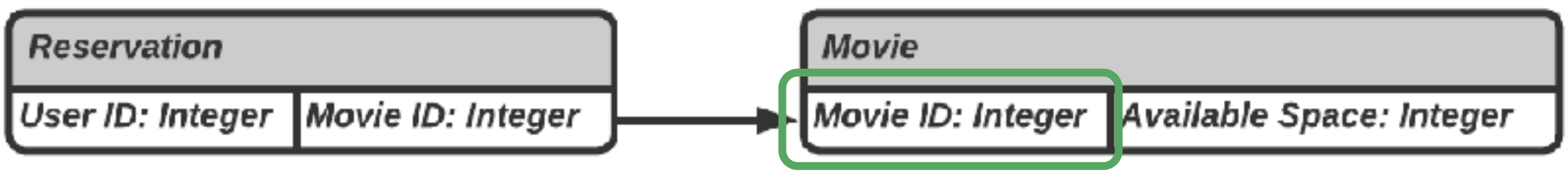
State of the art

	Convergence	Integrity	Recency	Synchronization avoidance	Communication avoidance
Strong consistency	✓	✓	✓	✗	✗
Eventual consistency / CRDT	✓	✗	✗	✓	✗
Sieve, Indigo, CISE, Hamsaz, Soteria	✓	✓	✗	✓	✗
TACT, TRAPP, FRACT, PBS	✓	✗	✓	✗	✓
Hampa	✓	✓	✓	✓	✓

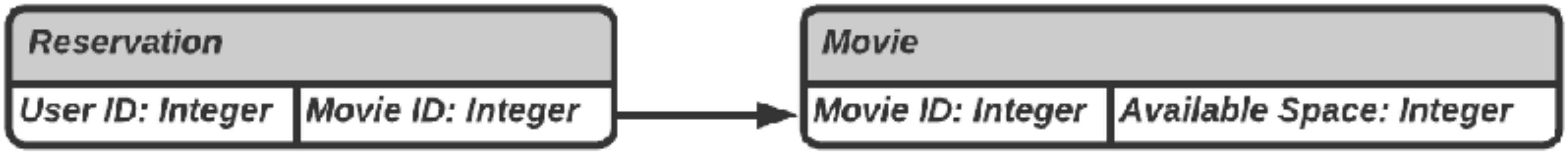
Movie use-case



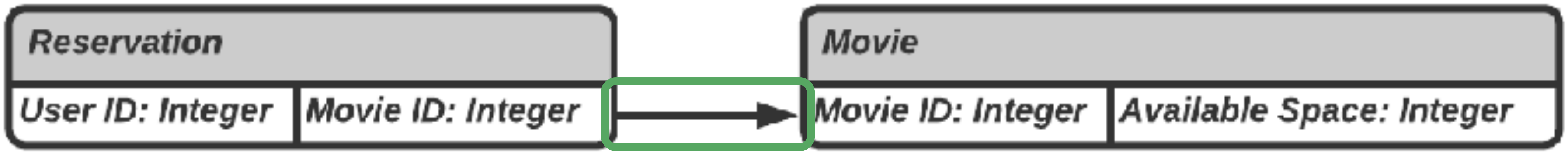
Movie use-case



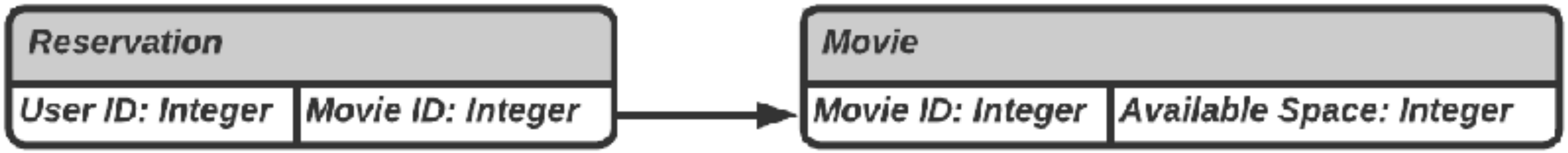
Movie use-case



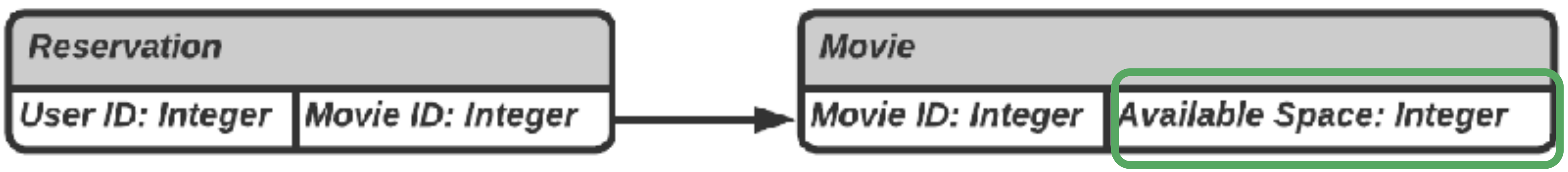
Movie use-case



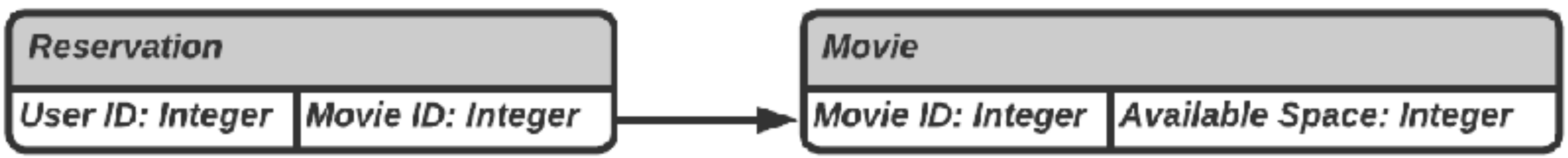
Movie use-case

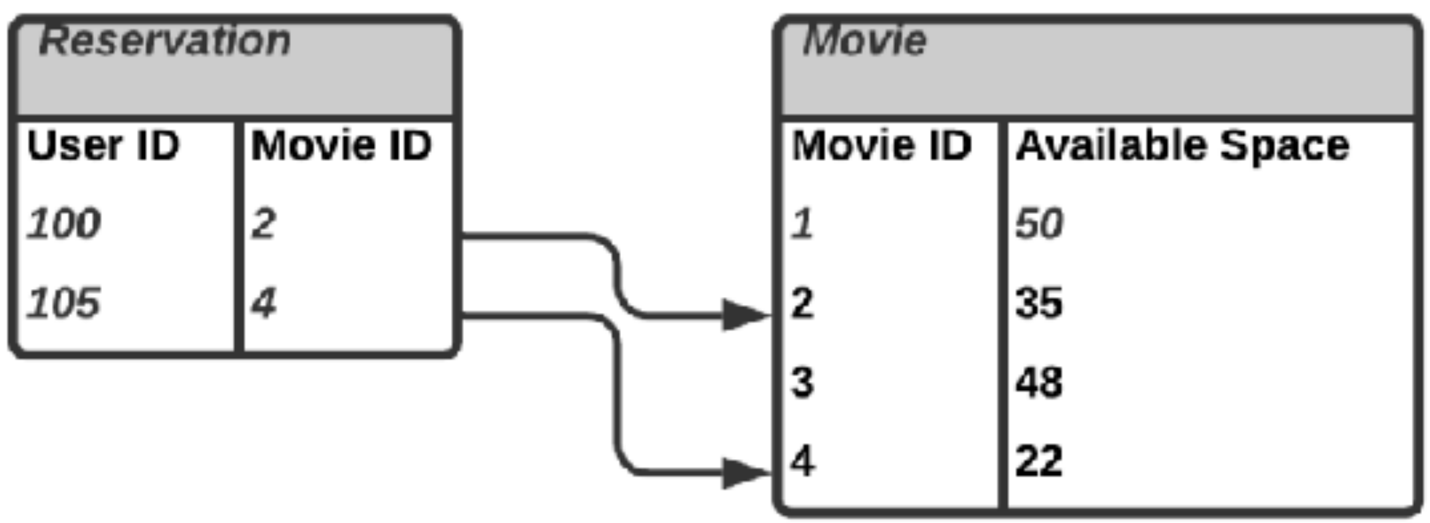


Movie use-case

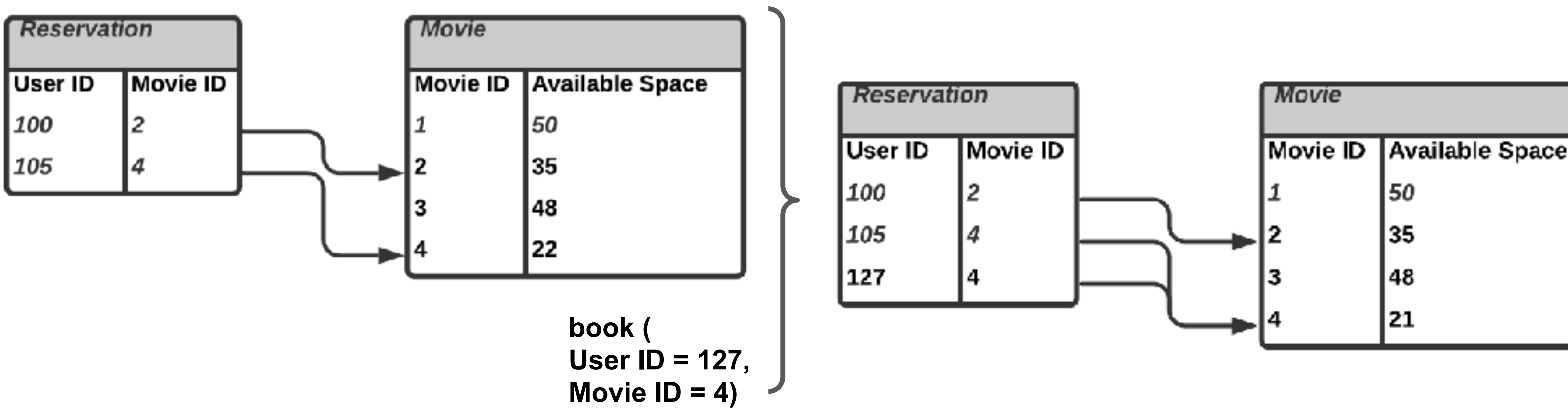


Movie use-case



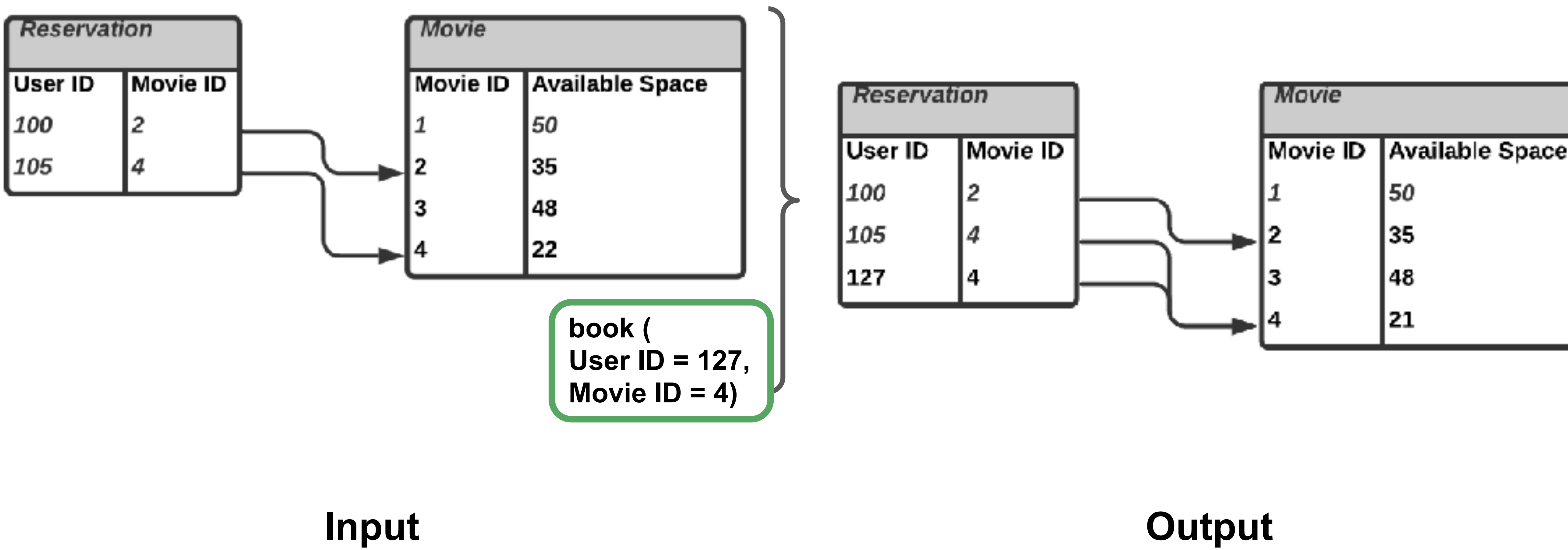


book method

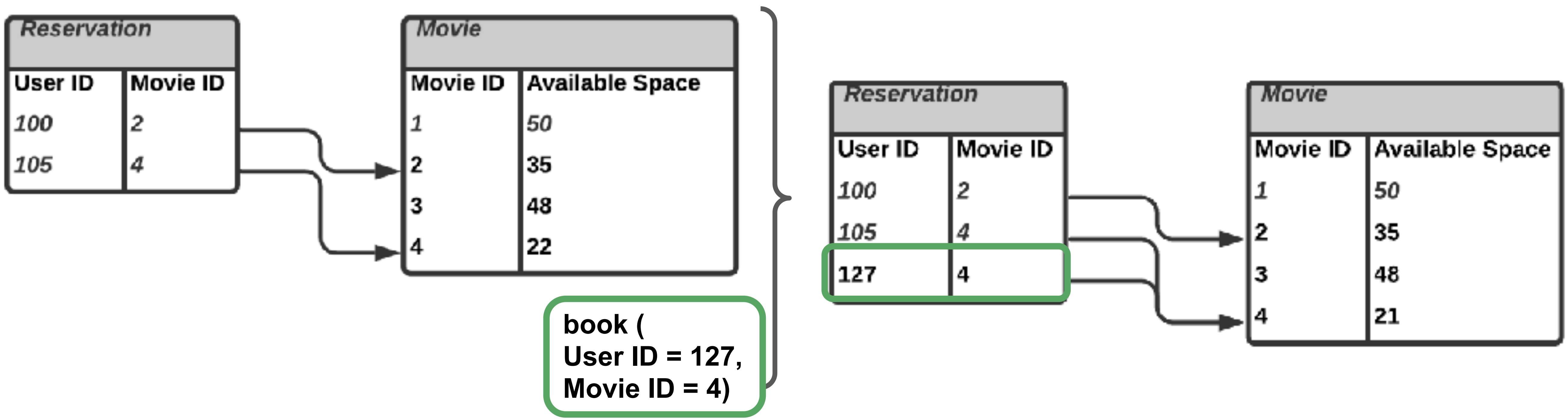


Input **Output**

book method

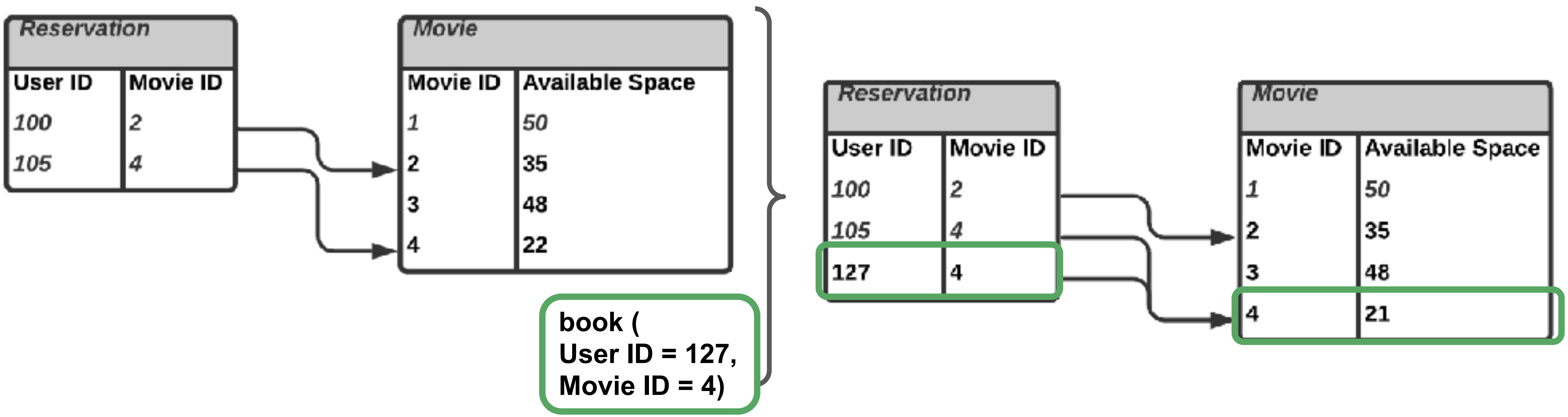

Reservation
User ID
Movie ID
100
2
105
4
Movie
Movie ID
Available Space
1
50
2
35
3
48
4
22
book (
User ID = 127,
Movie ID = 4)
Reservation
User ID
Movie ID
100
2
105
4
127
4
Movie
Movie ID
Available Space
1
50
2
35
3
48
4
21
Input
Output

book method



Input **Output**

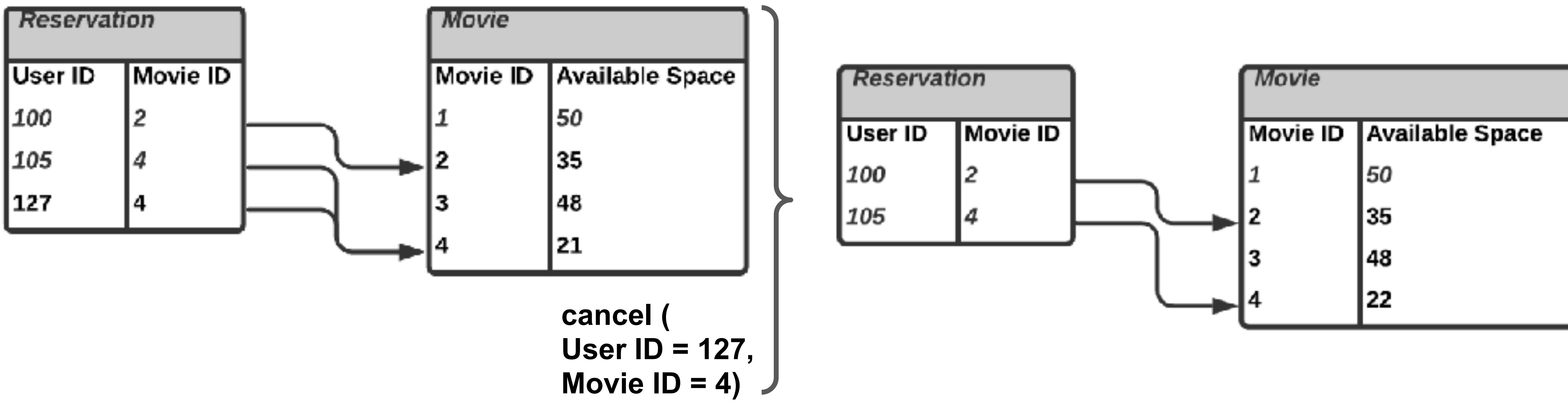
book method



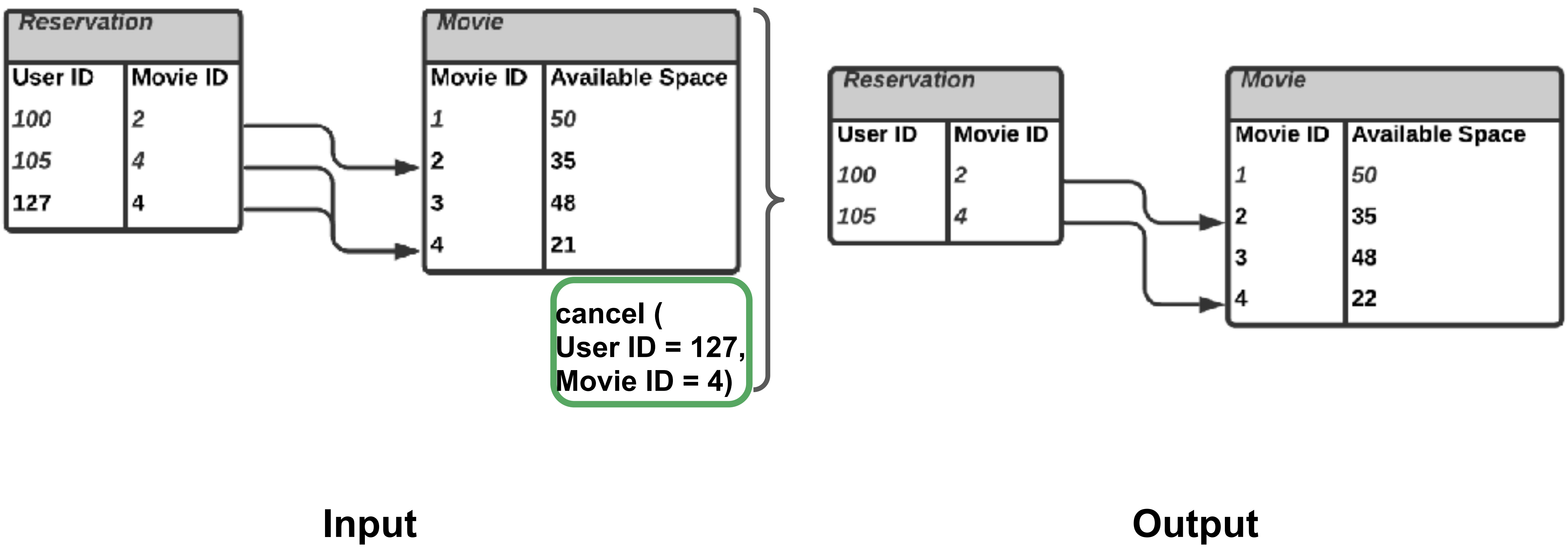
Input

Output

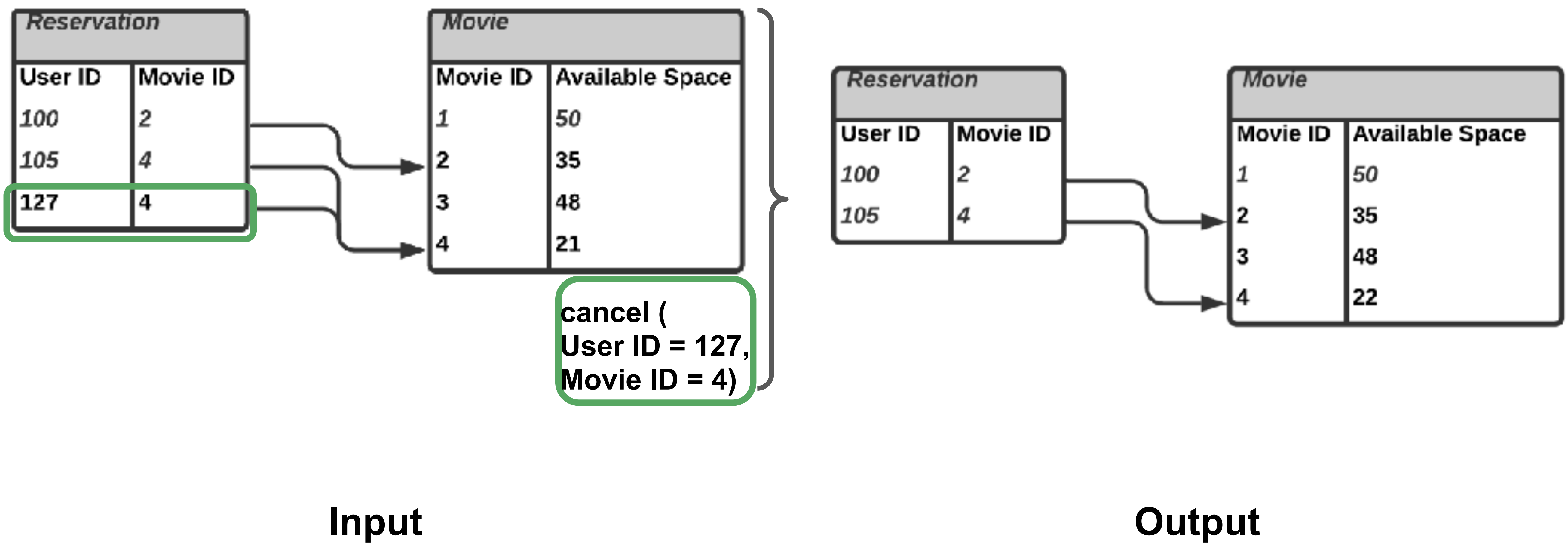
cancel method



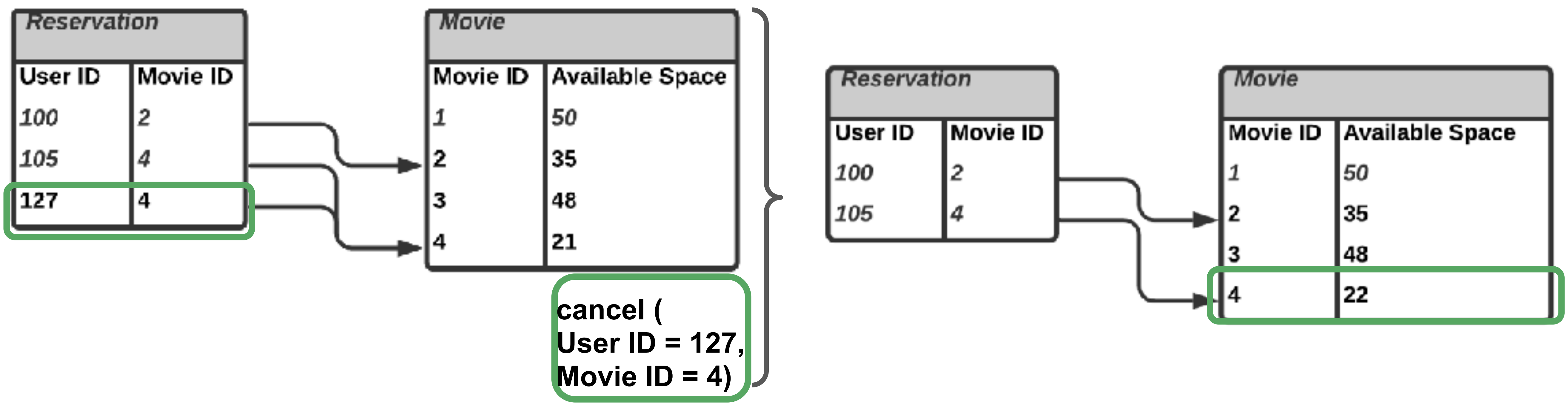
Input **Output**


Reservation
User ID
Movie ID
100
2
105
4
127
4
Movie
Movie ID
Available Space
1
50
2
35
3
48
4
21
cancel (
User ID = 127,
Movie ID = 4)
Reservation
User ID
Movie ID
100
2
105
4
Movie
Movie ID
Available Space
1
50
2
35
3
48
4
22
Input
Output

cancel method

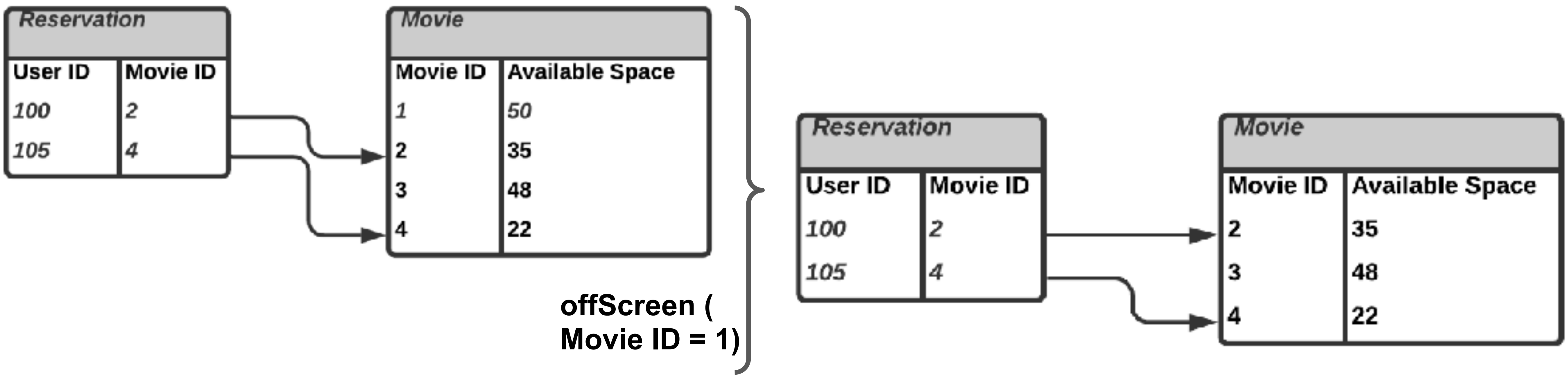


cancel method



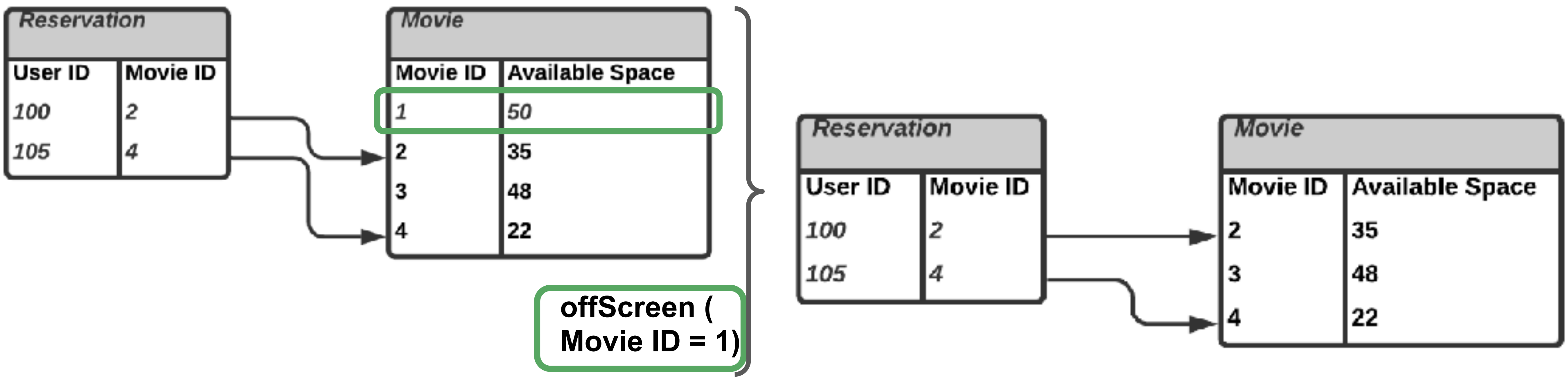
Input **Output**

offScreen method



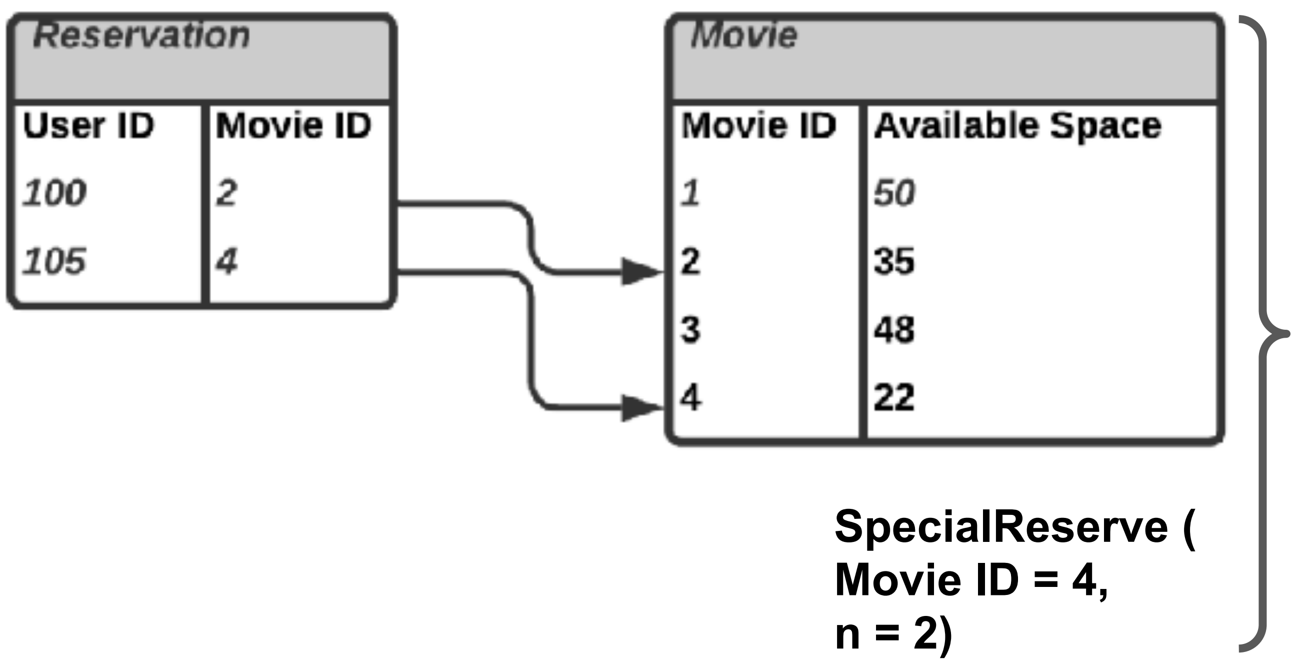
Input **Output**

offScreen method

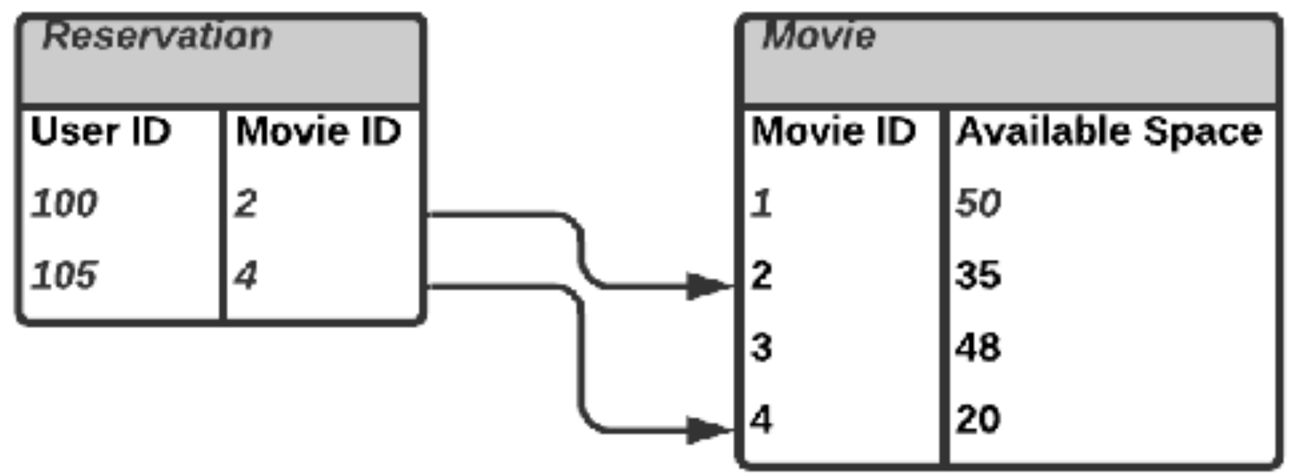


Input **Output**

specialReserve method

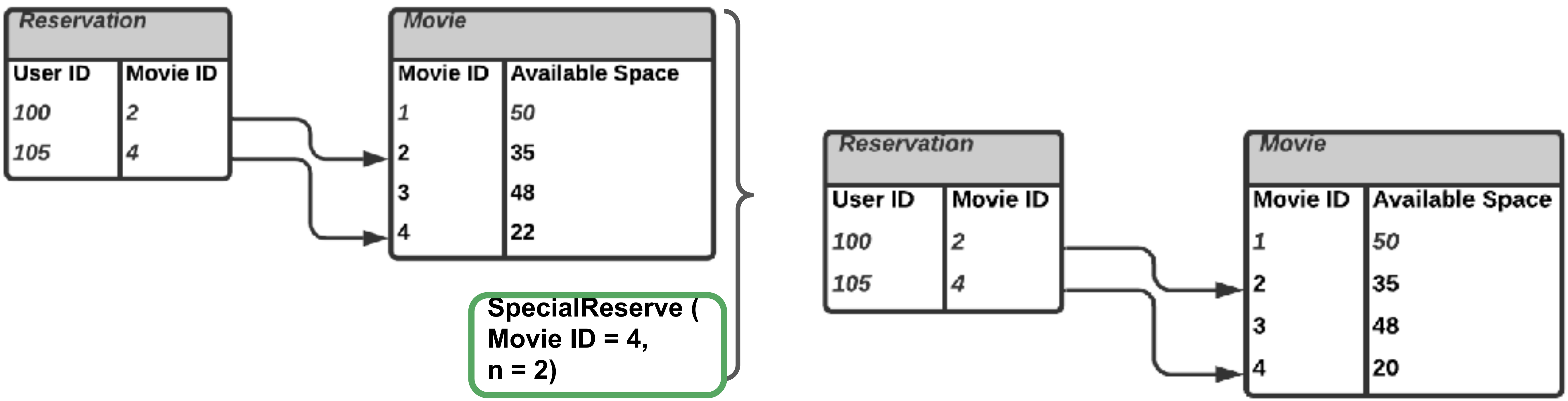


Input



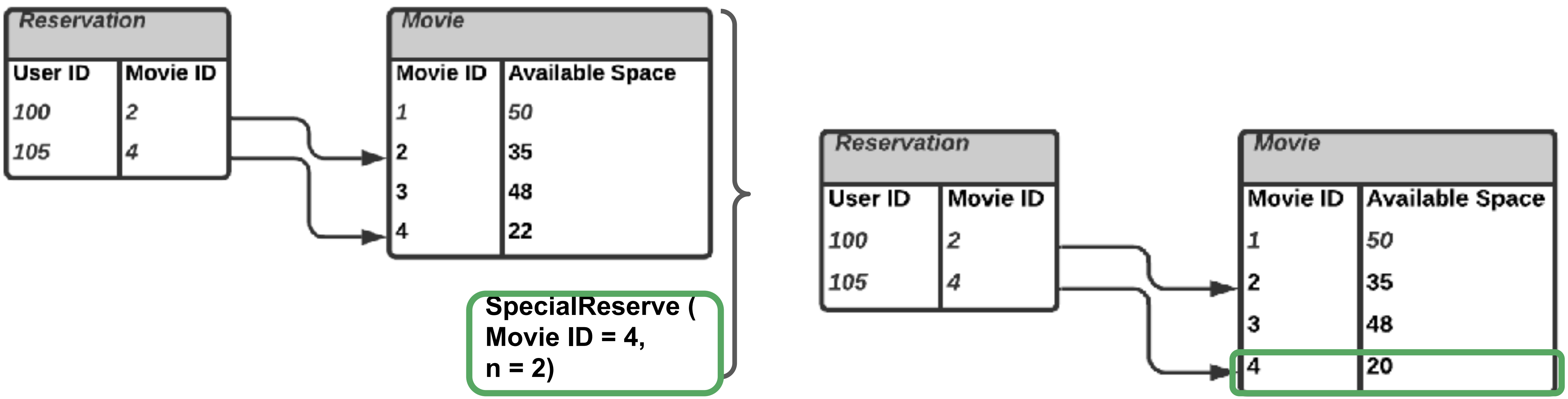
Output

specialReserve method



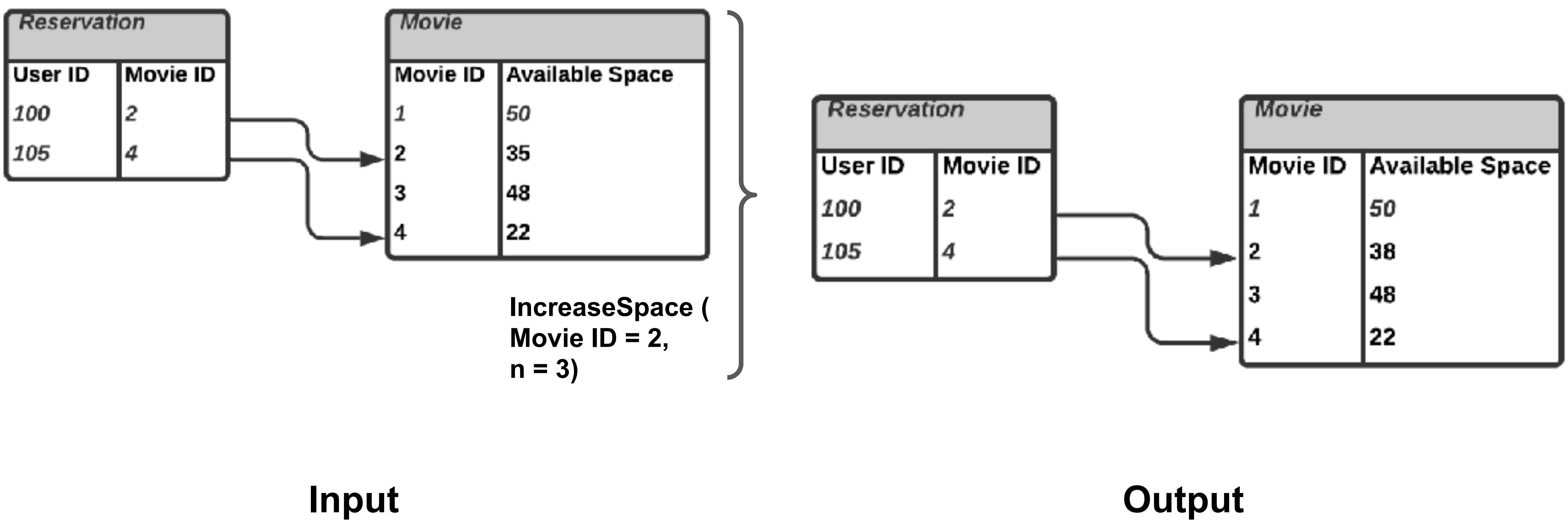
Input **Output**

specialReserve method

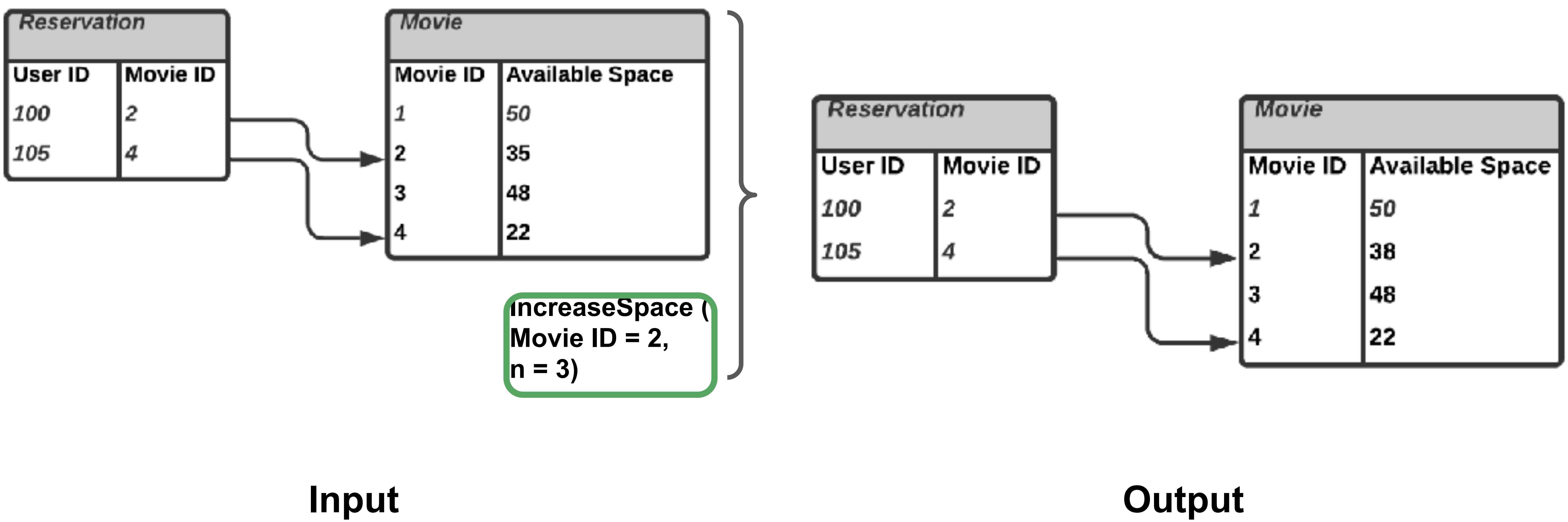


Input **Output**

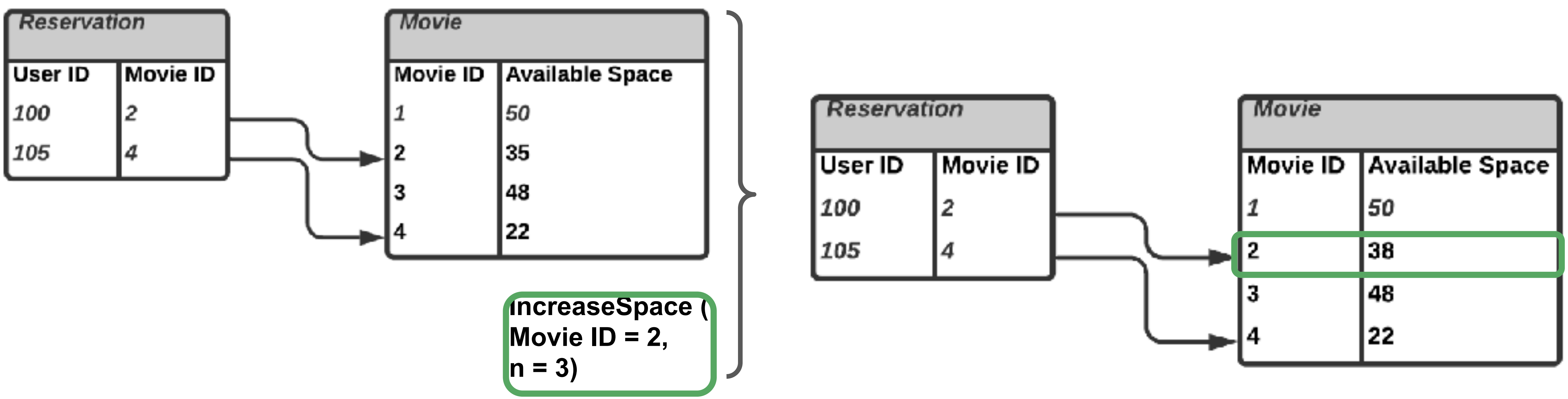
increaseSpace method



increaseSpace method



increaseSpace method



Input **Output**

State of the Art

Facilitating the consistency choices

- Require user-specified choices or annotations
- Crucially dependent on causal consistency as the weakest notion