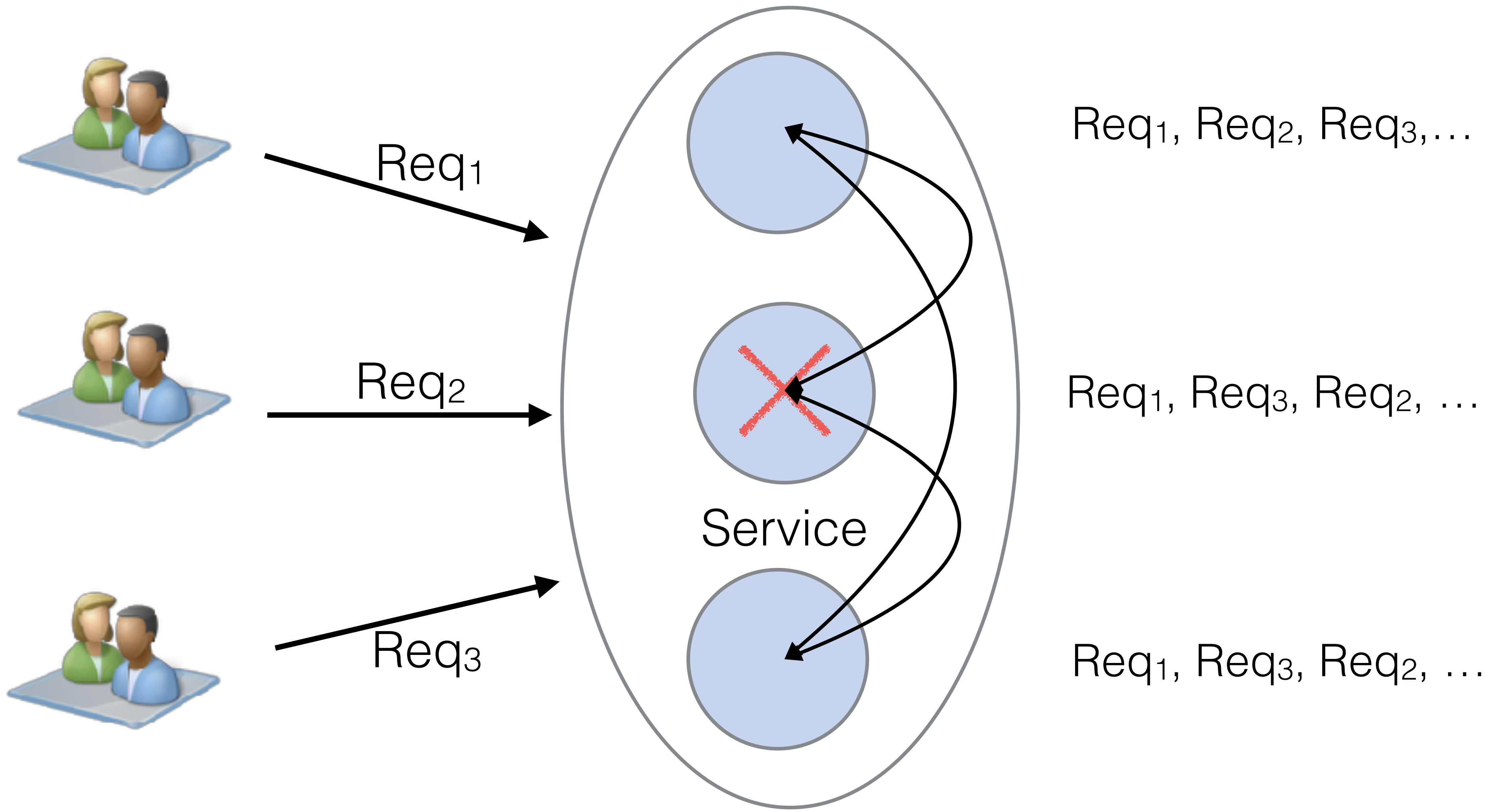
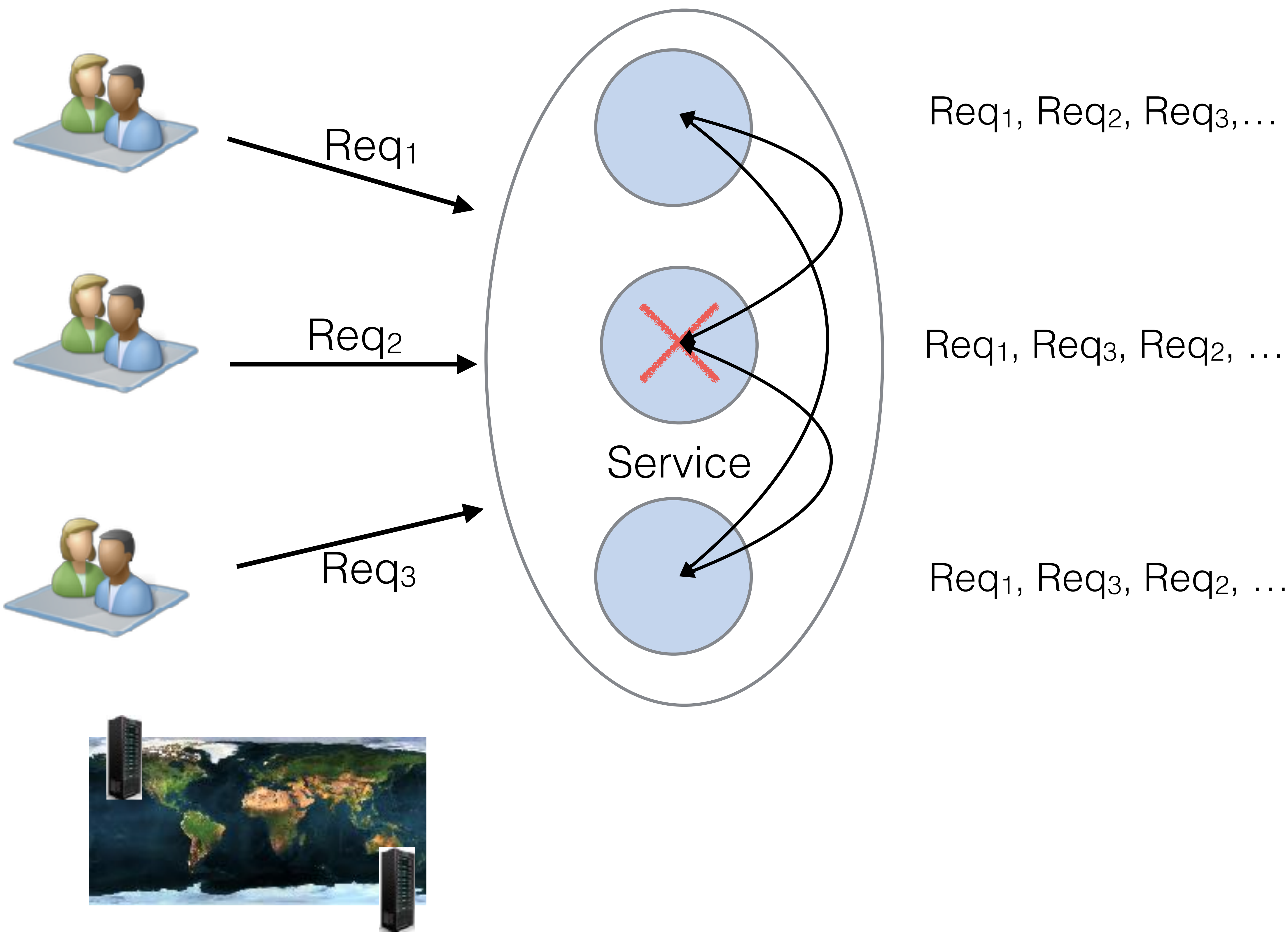
Replication Coordination Analysis and Synthesis

Farzin Houshmand, Mohsen Lesani
University of California, Riverside

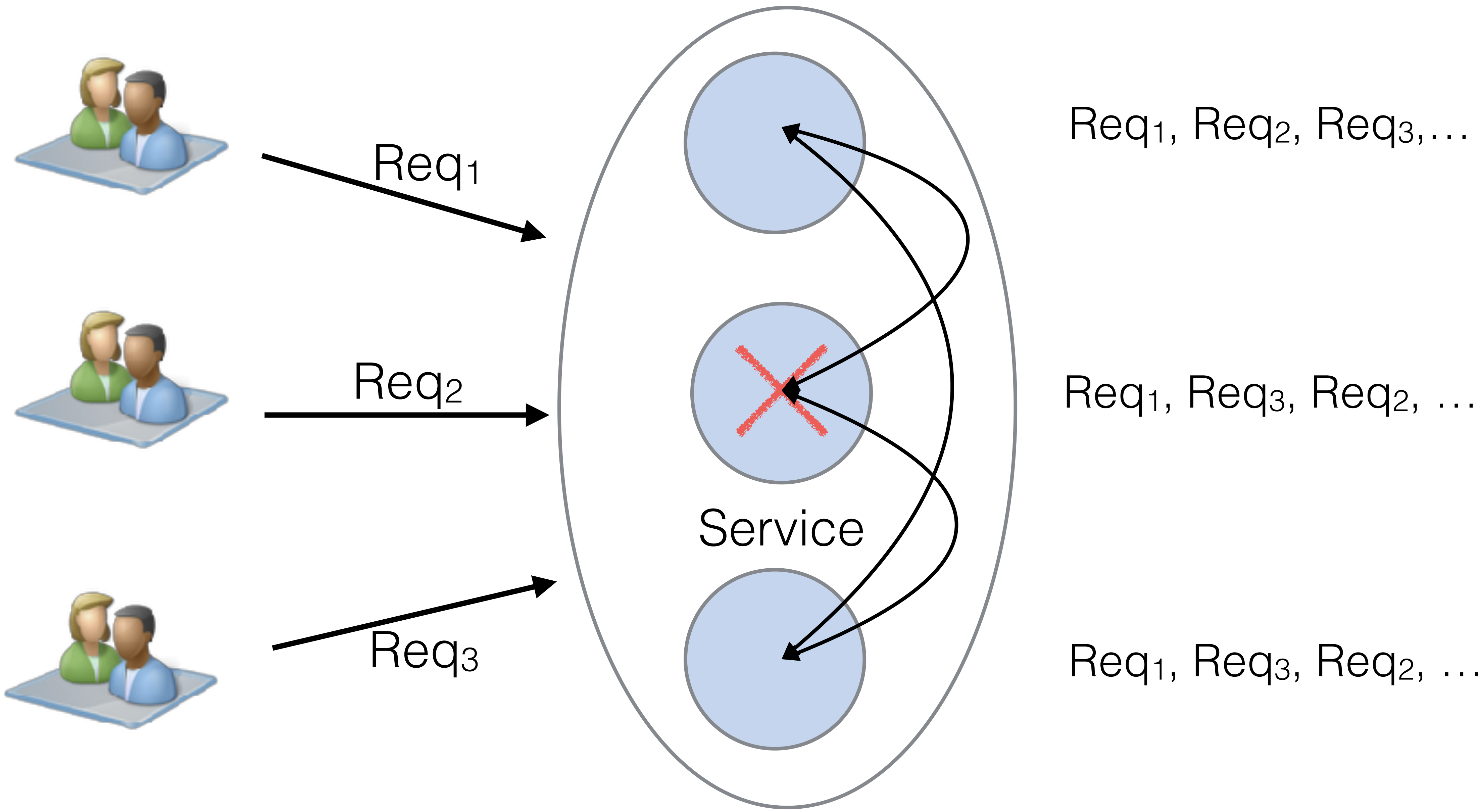
Replication



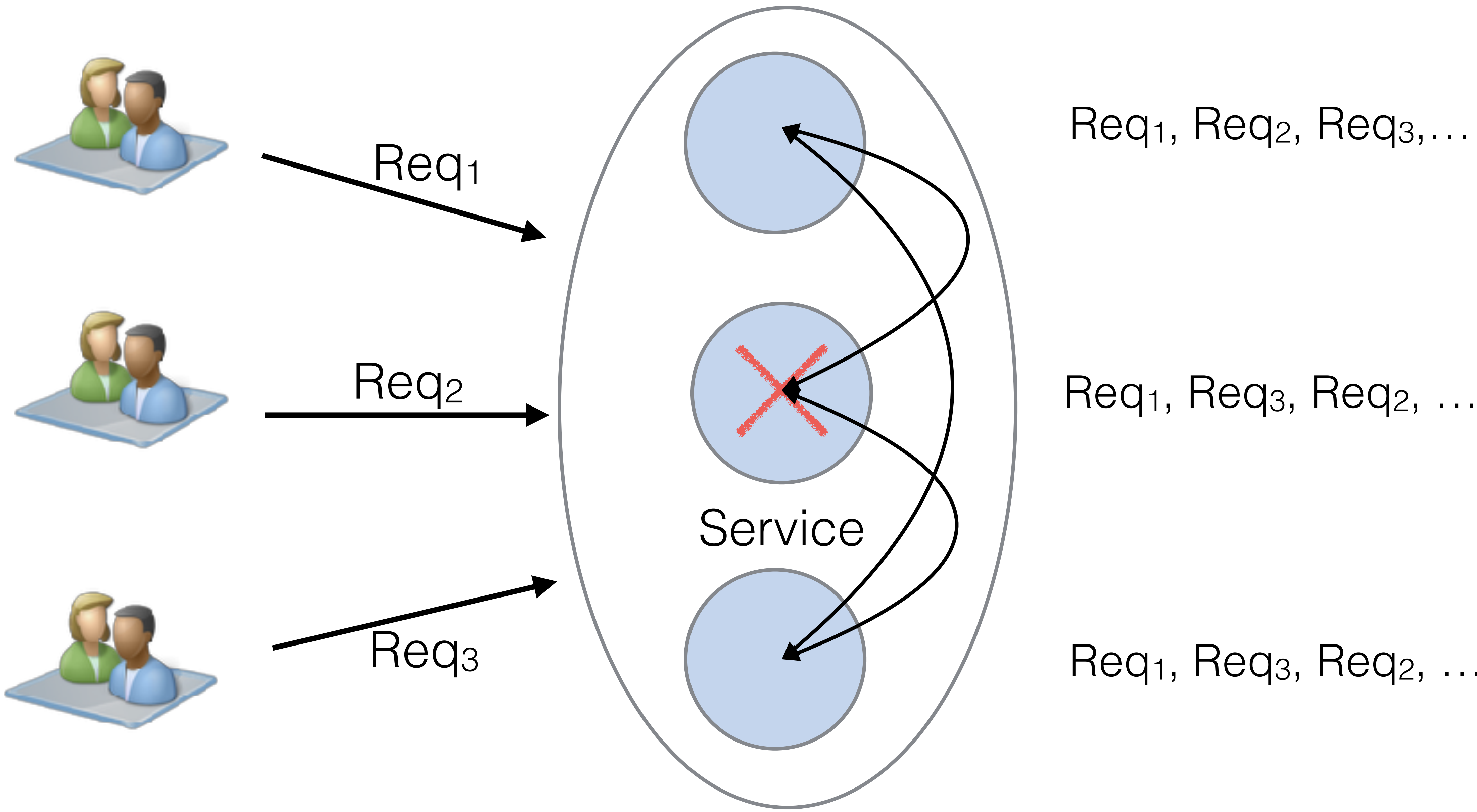
Replication



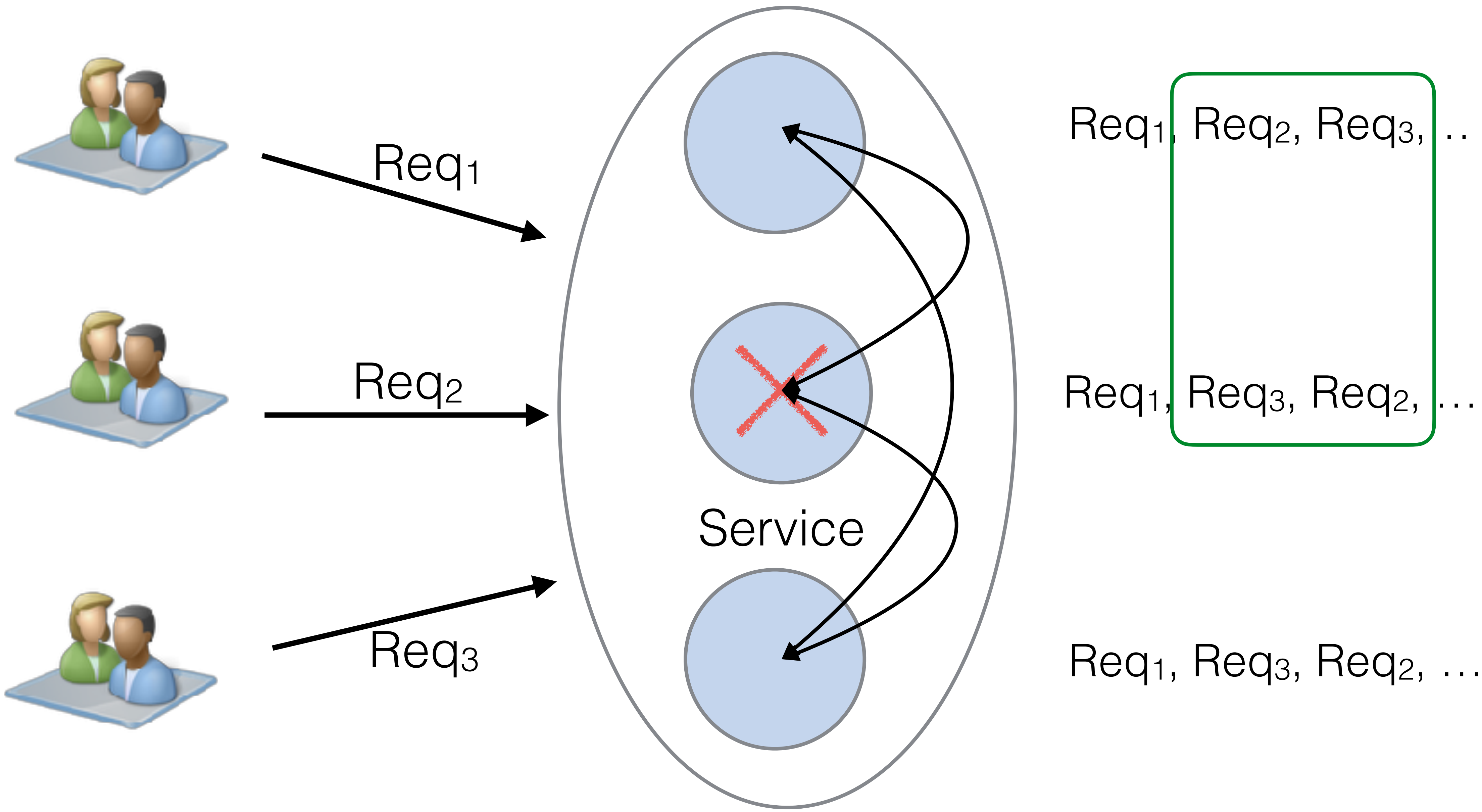
Replication

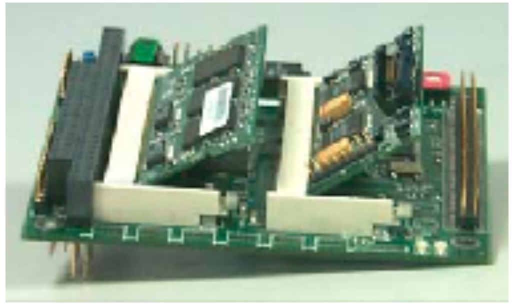


Replication



Replication





Consistency vs Responsiveness and Availability

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

Sequential Consistency

Consistency vs Responsiveness and Availability

Consistency

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

Sequential Consistency

amazon DynamoDB — SOSP'07
cassandra — OSR'10

Eventual Consistency

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

Consistency and Integrity

Consistency and Integrity

- What users need is integrity and not consistency. **Consistency** is a means to **Integrity**.

Consistency and Integrity

- What users need is integrity and not consistency. **Consistency** is a means to **Integrity**.
- Bank Account. Integrity: Non-negative balance.

Consistency and Integrity

- What users need is integrity and not consistency. **Consistency** is a means to **Integrity**.
- Bank Account. Integrity: Non-negative balance.
- Deposit
  No synchronization
  No dependency

Consistency and Integrity

- What users need is integrity and not consistency. **Consistency** is a means to **Integrity**.

- Bank Account. Integrity: Non-negative balance.

- Deposit
  No synchronization
  No dependency

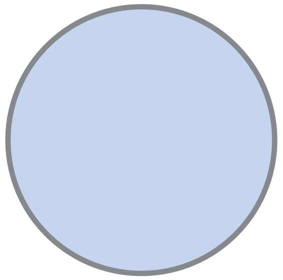
- Withdraw
  Synchronization with withdraw
  Dependent on preceding deposits

State of the Art

Facilitating the consistency choices

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

Hamsaz: Coordination-avoiding Replicated Object Synthesis

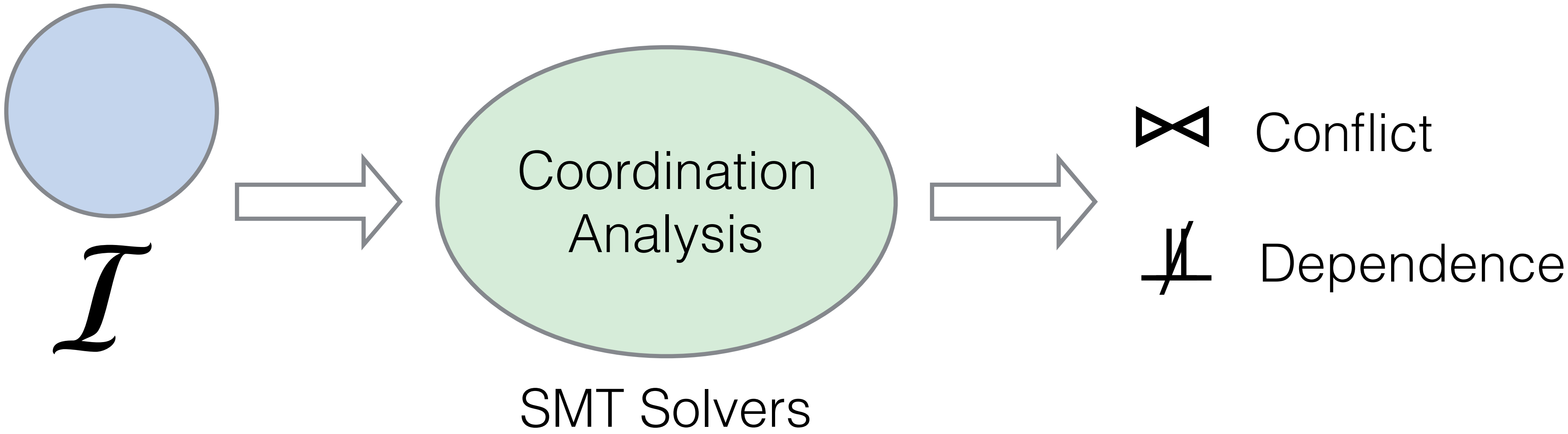


Object

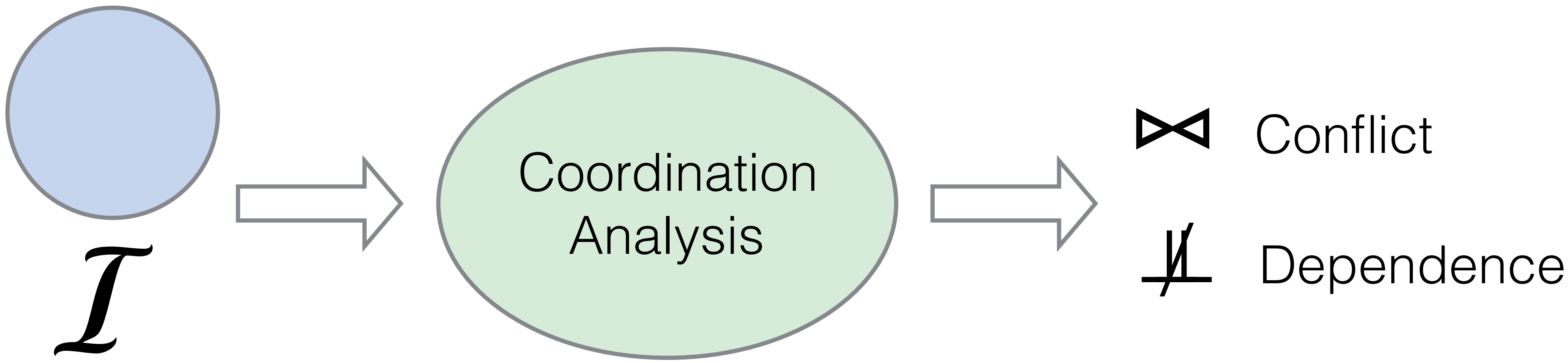
$\mathcal{I}$ Integrity Property

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

Hamsaz: Coordination-avoiding Replicated Object Synthesis

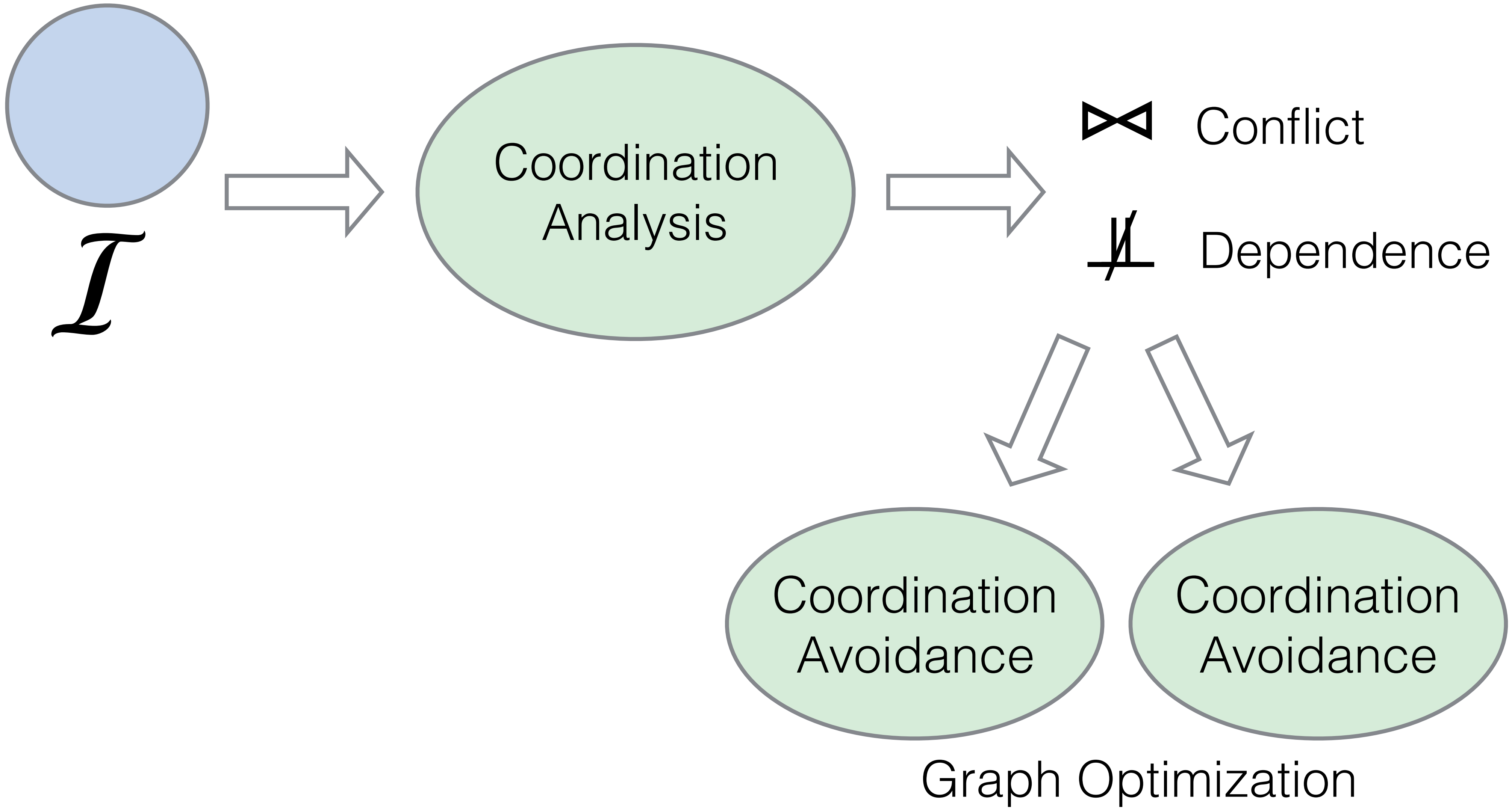


Hamsaz: Coordination-avoiding Replicated Object Synthesis

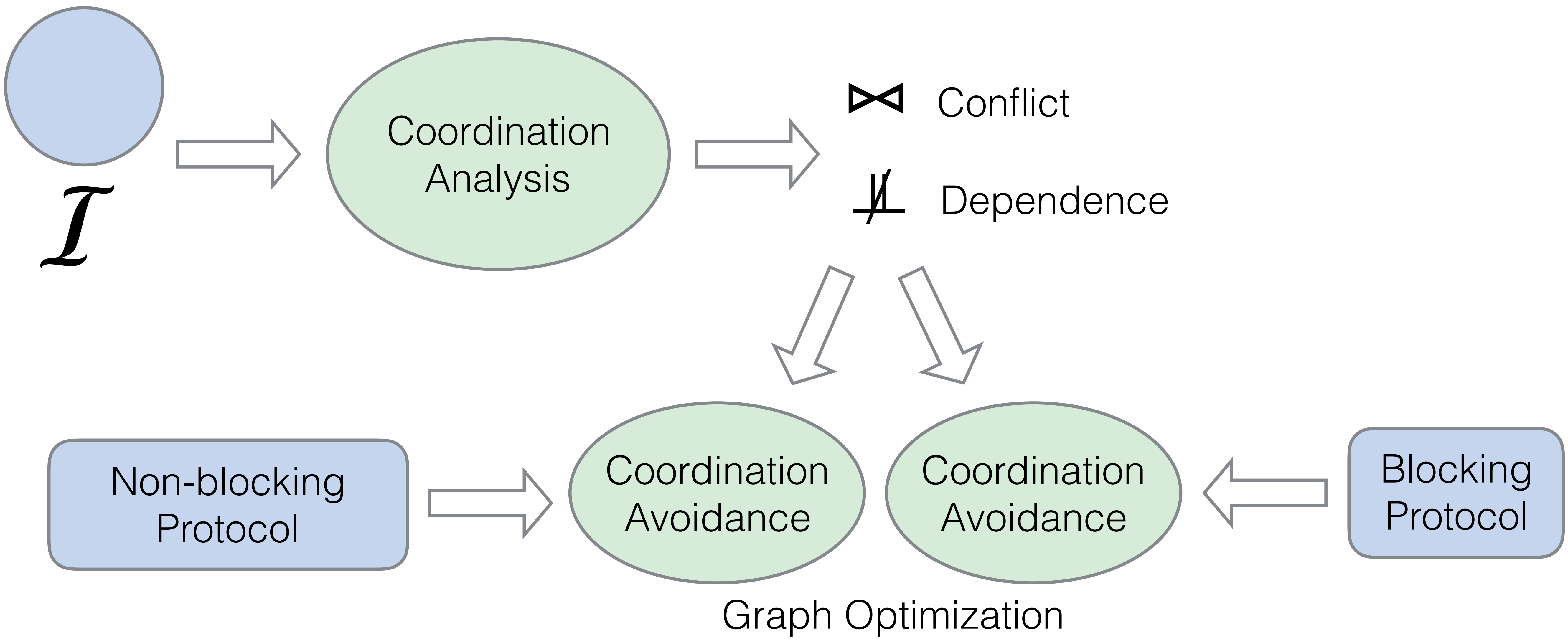


Well-coordination:
Synchronization between conflicting
Causality between dependent
Theorem:
Well-coordination is sufficient for
integrity and convergence

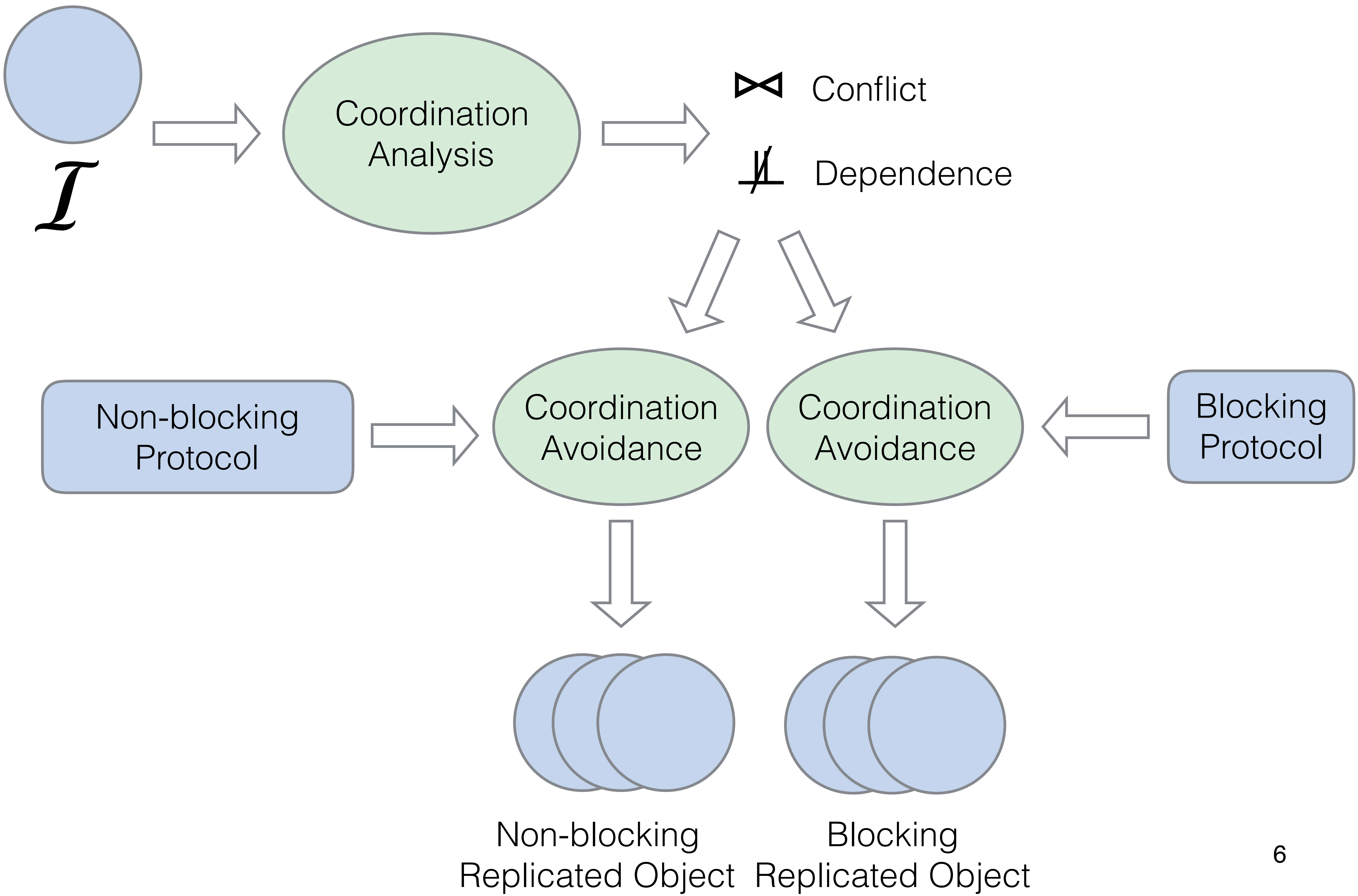
Hamsaz: Coordination-avoiding Replicated Object Synthesis



Hamsaz: Coordination-avoiding Replicated Object Synthesis



Hamsaz: Coordination-avoiding Replicated Object Synthesis



Example Specification

$$\langle \Sigma, \mathcal{I}, \mathcal{M} \rangle$$

Example Specification

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

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

Example Specification

Class Courseware

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

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

Example Specification

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

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

Example Specification

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

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

Example Specification

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

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

Example Specification

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

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

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

Example Specification

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) ∧

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 Specification

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) ∧

refIntegrity(es, ecid, cs, cid)

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

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

⟨guard, update, retv⟩

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

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

Example Specification

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

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

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

Example Specification

Class Courseware

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

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

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

Example Specification

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

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

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

Example Specification

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

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

$\text{refIntegrity}(es, \text{esid}, ss, \text{sid}) \wedge$

$\text{refIntegrity}(es, \text{ecid}, cs, \text{cid})$

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

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

⟨guard, update, retv⟩

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

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

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

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

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

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

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

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

Example Specification

Class Courseware

$\quad$ let Student := Set $\langle \text{sid}\colon \text{SId}\rangle$ in

$\quad$ let Course := Set $\langle \text{cid}\colon \text{CId}\rangle$ in

$\quad$ let Enrolment :=

$\quad\quad$ Set $\langle \text{esid}\colon \text{SId}, \text{ecid}\colon \text{CId}\rangle$ in

$\quad$ Σ := Student $\times$ Course $\times$ Enrolment

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

$\quad\quad$ $\text{refIntegrity}(es, \text{esid}, ss, \text{sid}) \wedge$

$\quad\quad$ $\text{refIntegrity}(es, \text{ecid}, cs, \text{cid})$

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

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

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

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

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

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

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

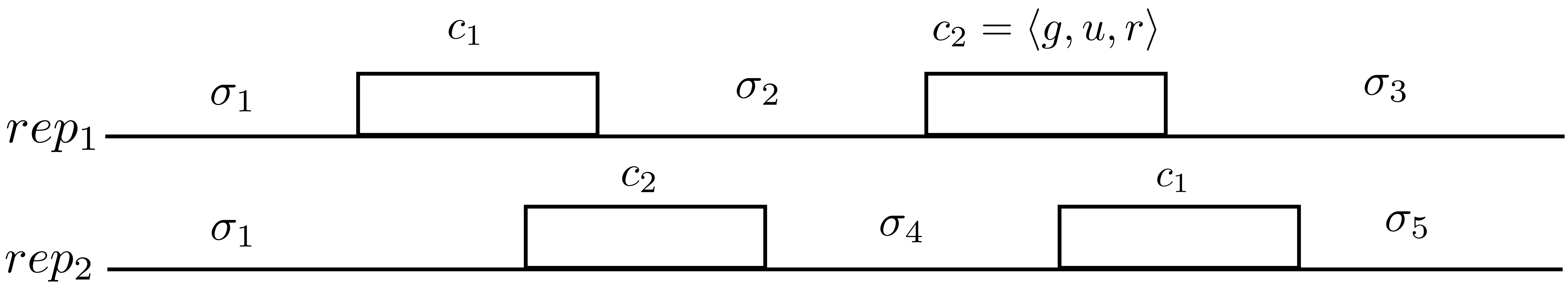
$\quad$ $\text{deleteCourse}(c) := \lambda\, \langle ss, cs, es\rangle.$

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

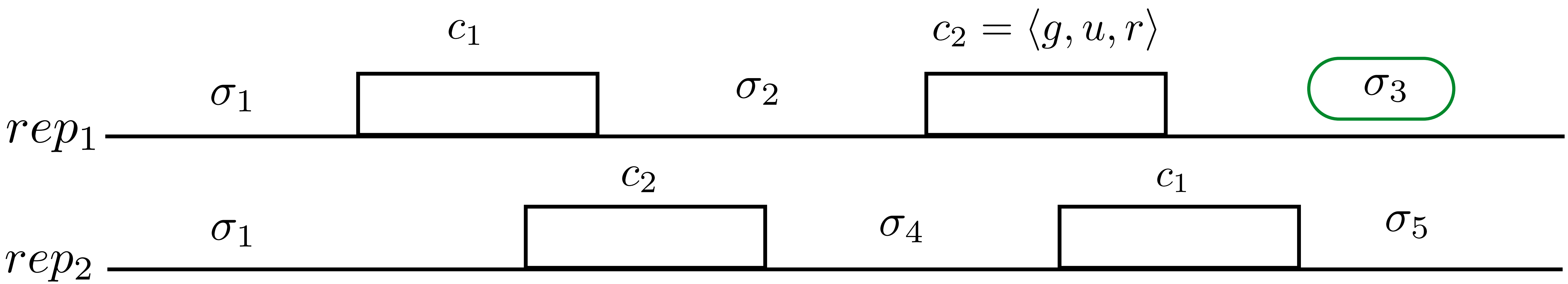
$\quad$ $\text{query} := \lambda\, \sigma.\, \langle \mathbb{T},\quad \sigma,\quad \sigma\rangle$

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

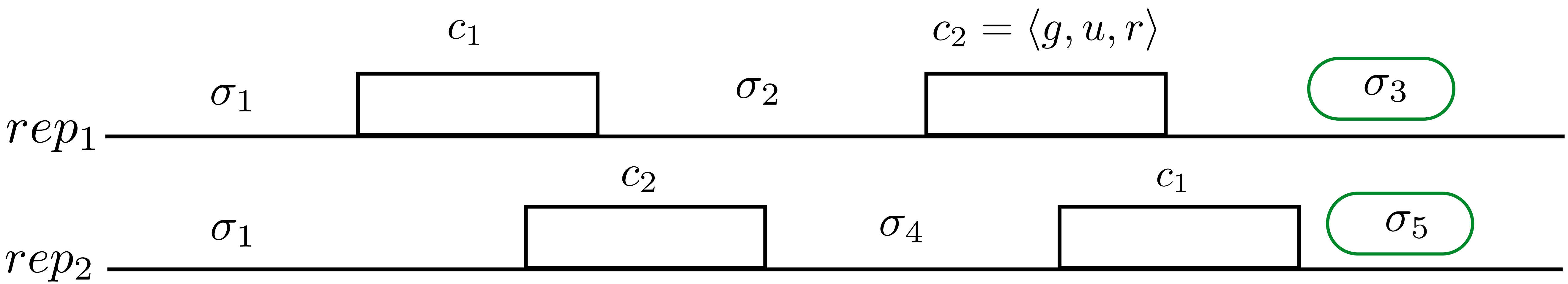
Convergence



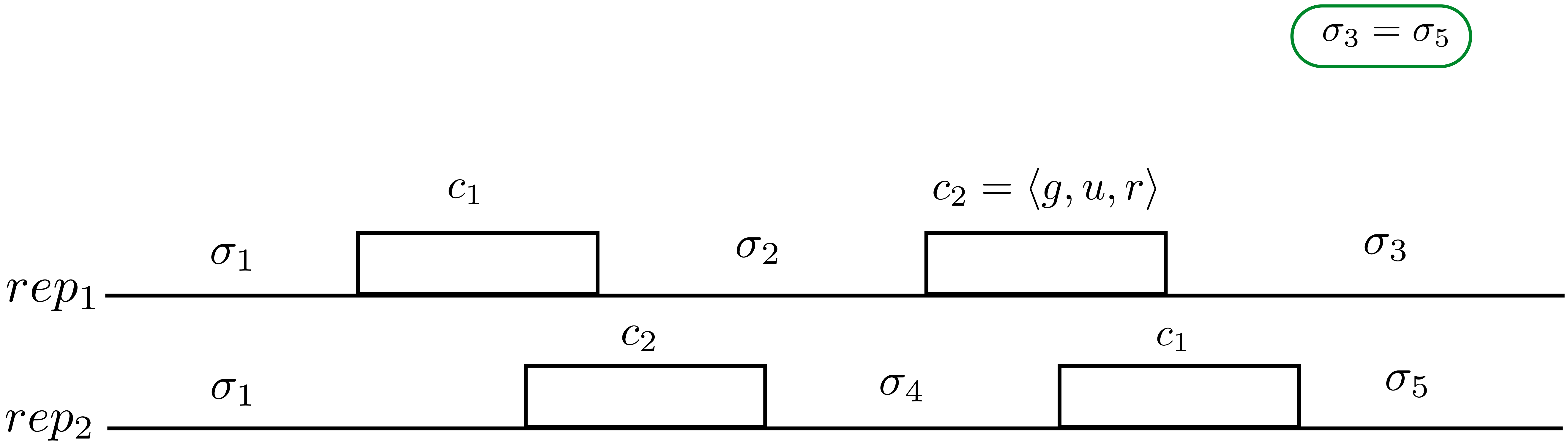
Convergence



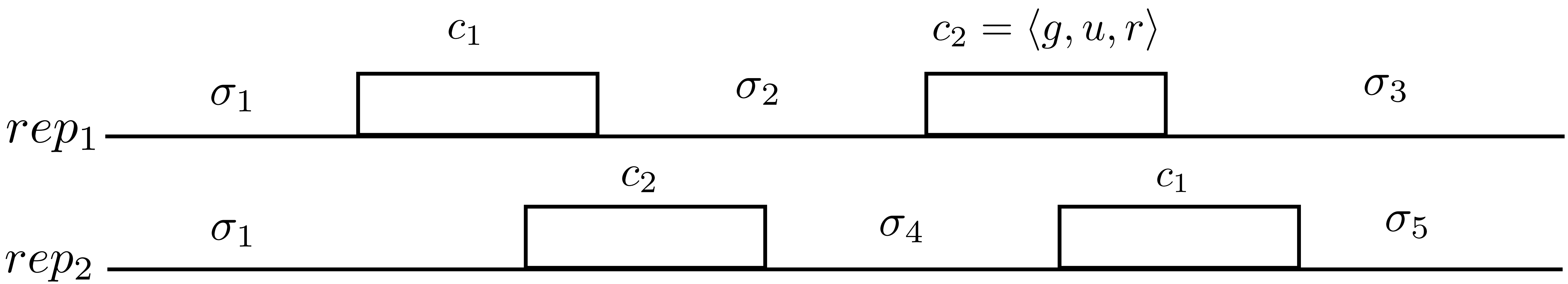
Convergence



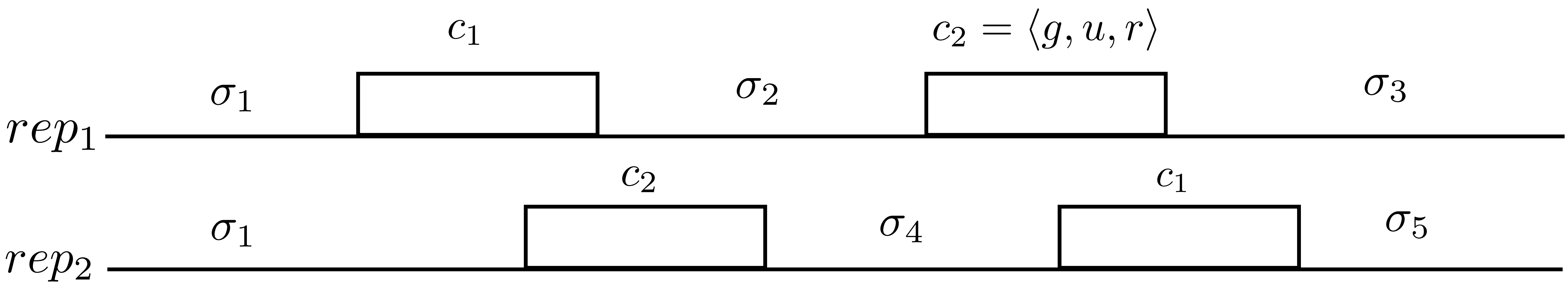
Convergence



Convergence

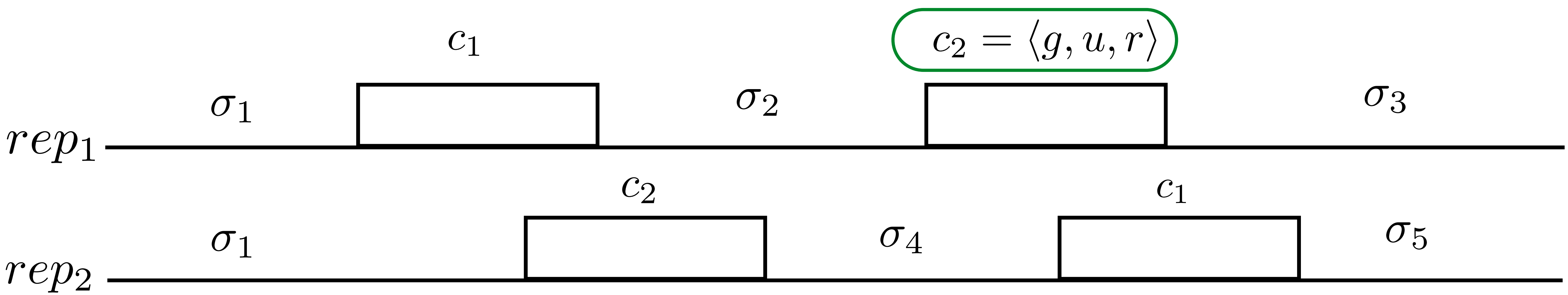


Consistency

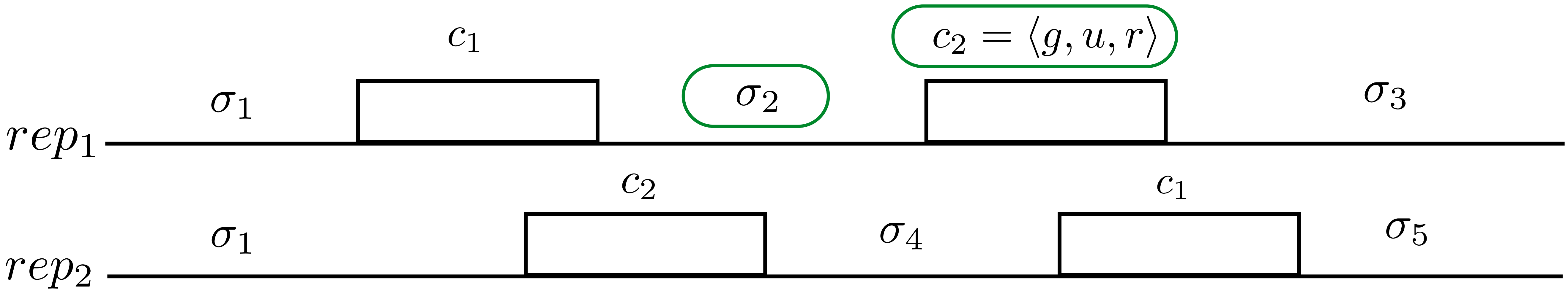


Convergence and Consistency

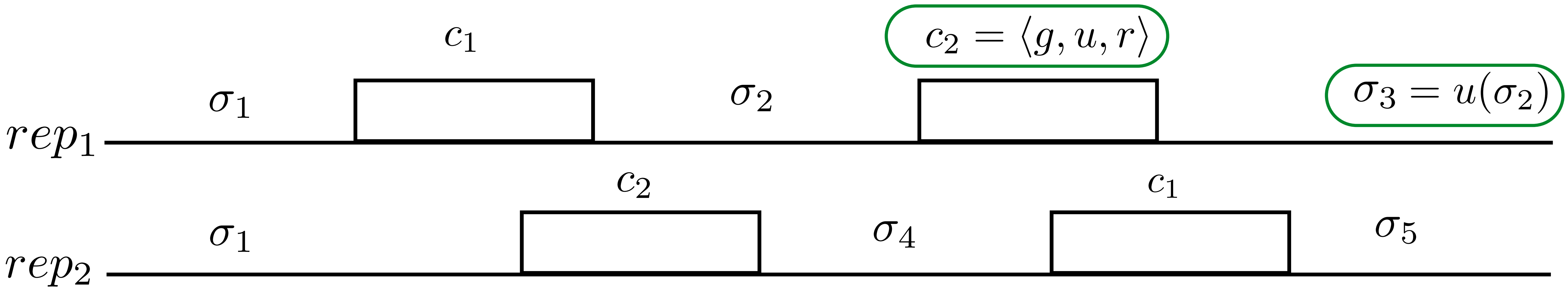
Consistency



Consistency



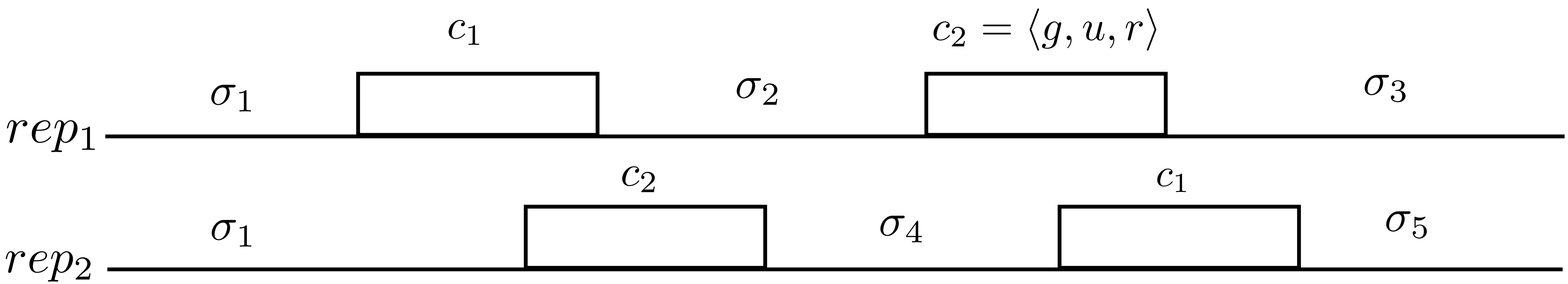
Consistency



Convergence and Consistency

Consistency

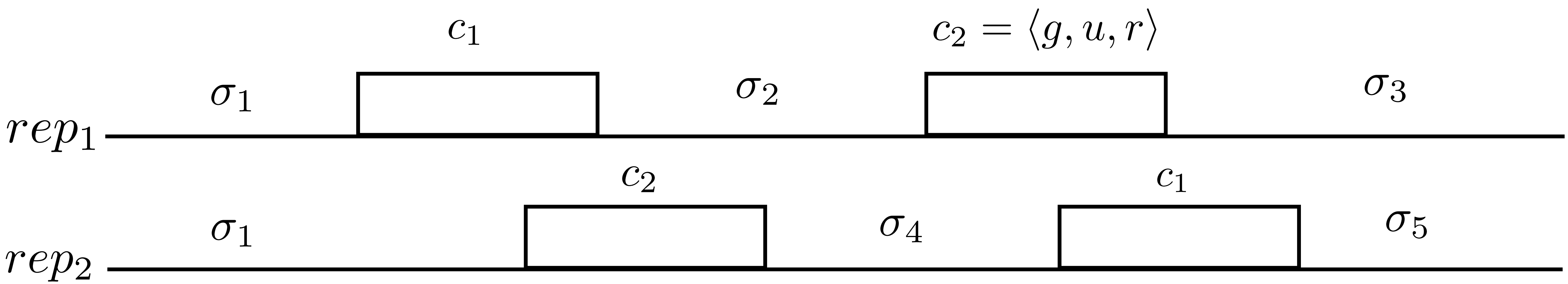
$\mathcal{C}(\sigma_2, c_2) =$
$g(\sigma_2) \wedge$
$\mathcal{I}(\sigma_2)$



Convergence and Consistency

Consistency

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

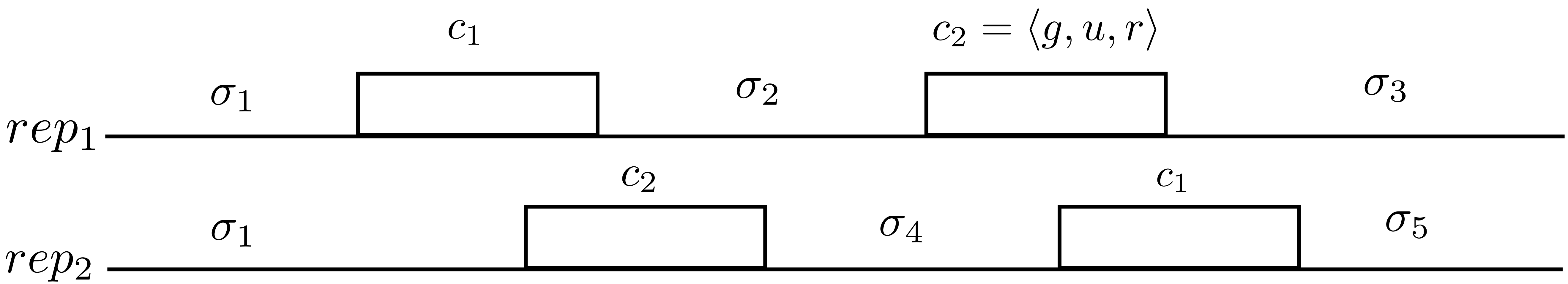


Convergence and Consistency

Consistency

Permissibility

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



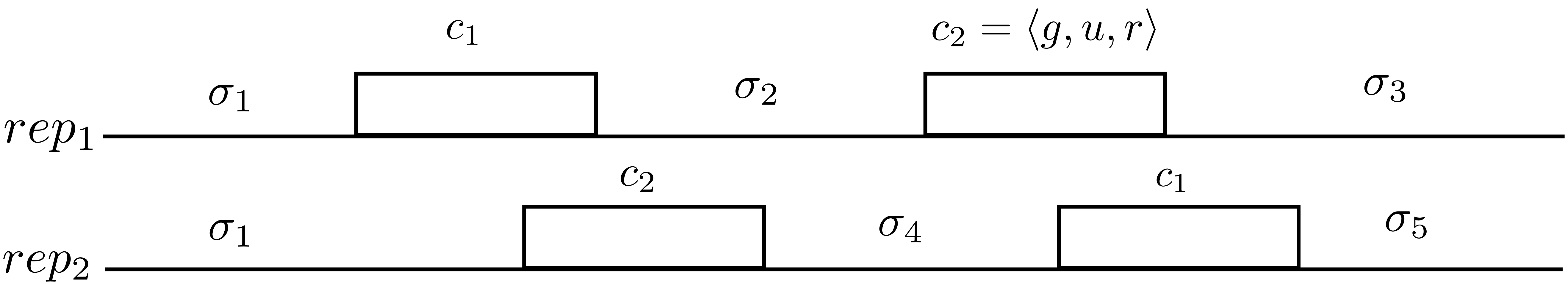
Convergence and Consistency

Consistency

Permissibility

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

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

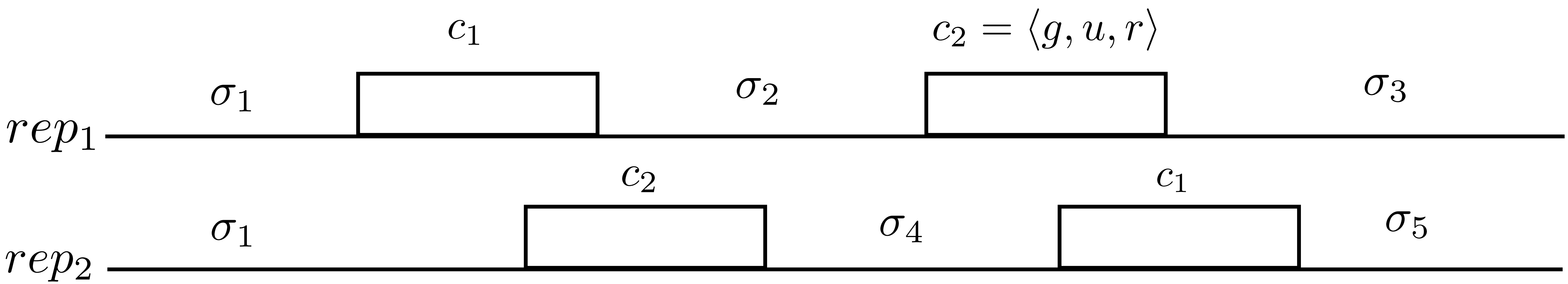


Consistency

Permissibility

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

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

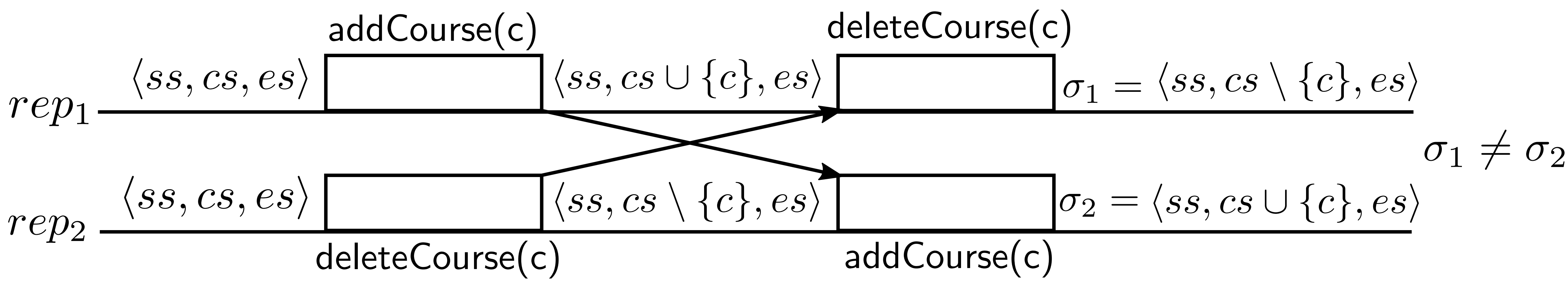


Conflict

1. $\mathcal{S}$-commute
2. $\mathcal{P}$-concur

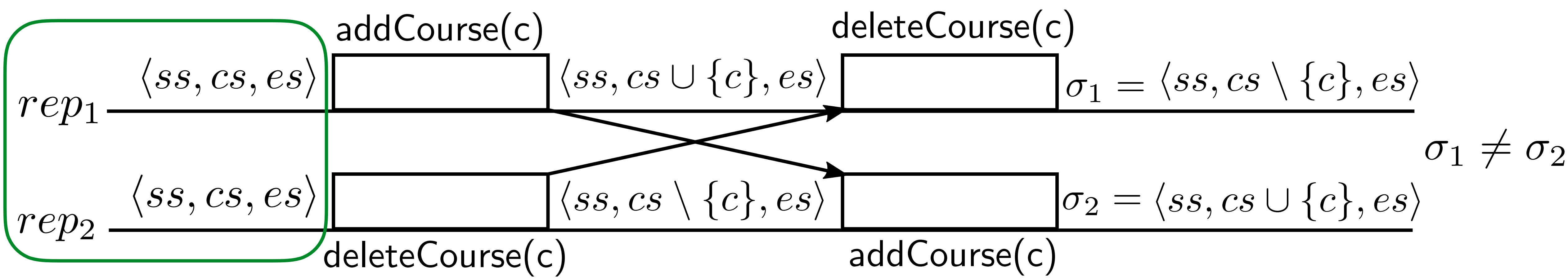
1 State-Conflict

$\mathcal{S}$-conflict

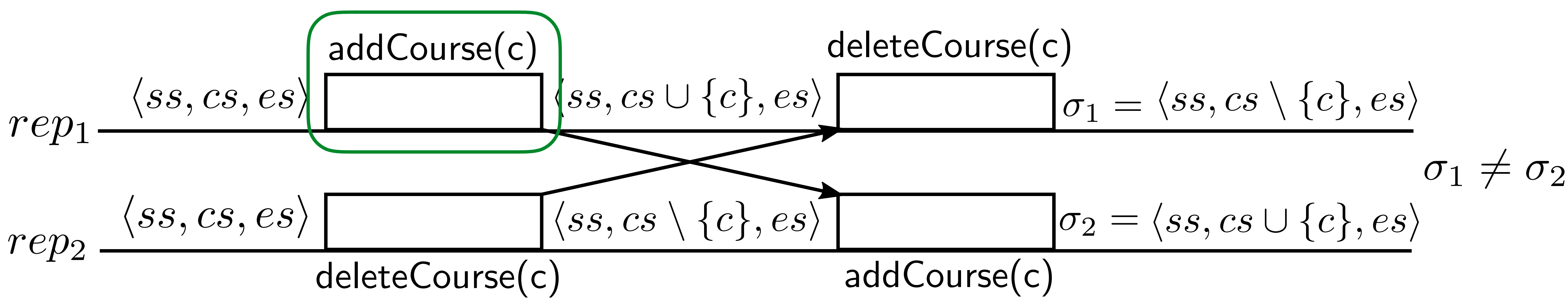


1 State-Conflict

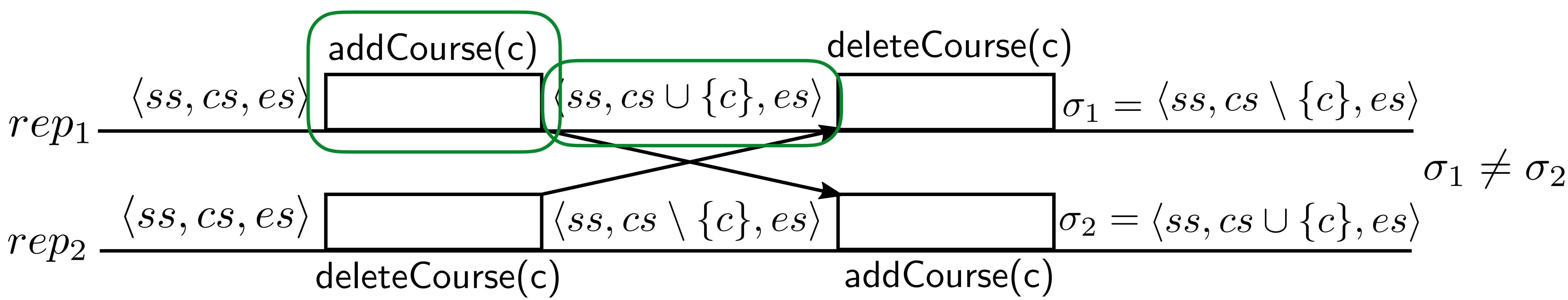
$\mathcal{S}$-conflict



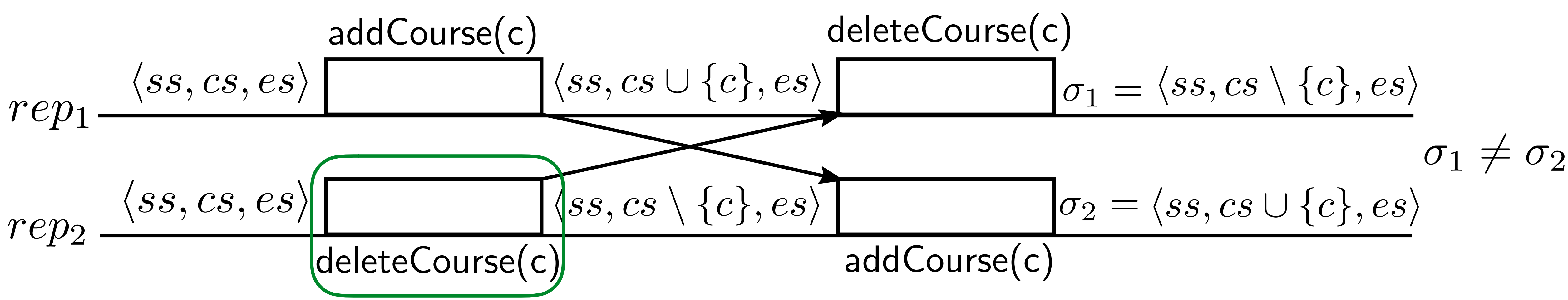
$\mathcal{S}$-conflict



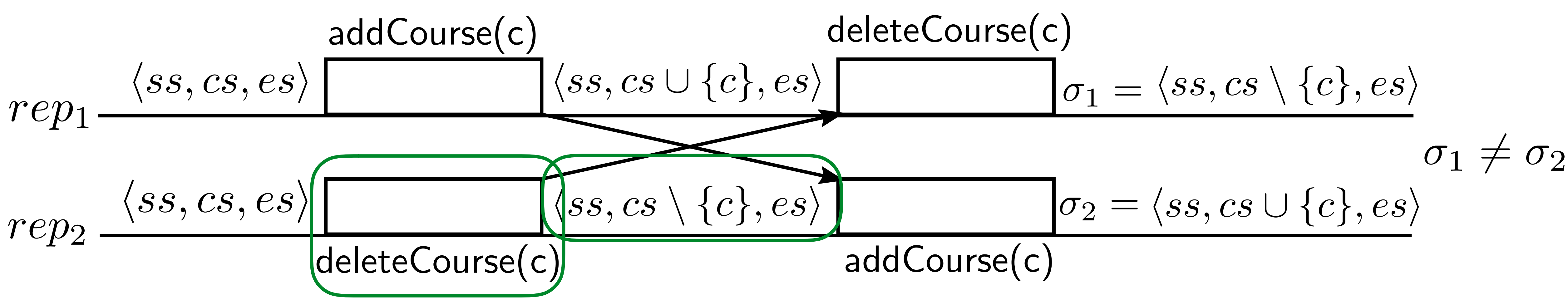
$\mathcal{S}$-conflict



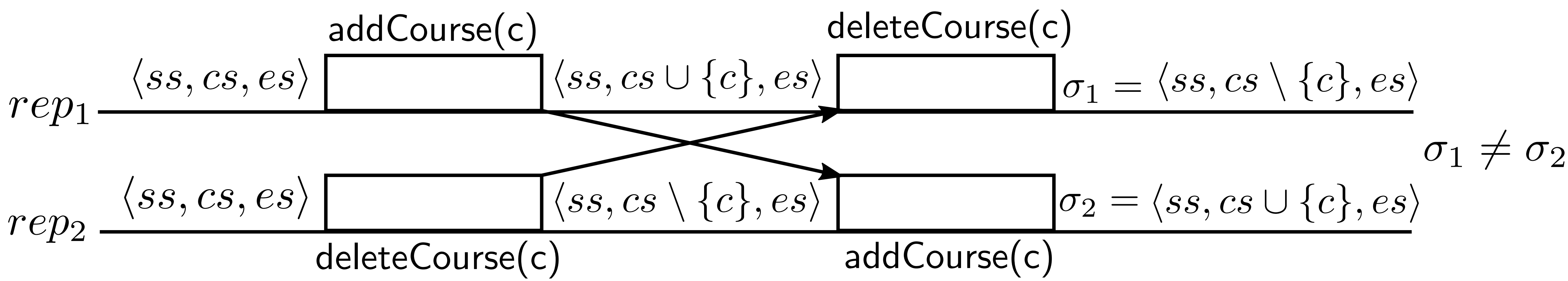
$\mathcal{S}$-conflict



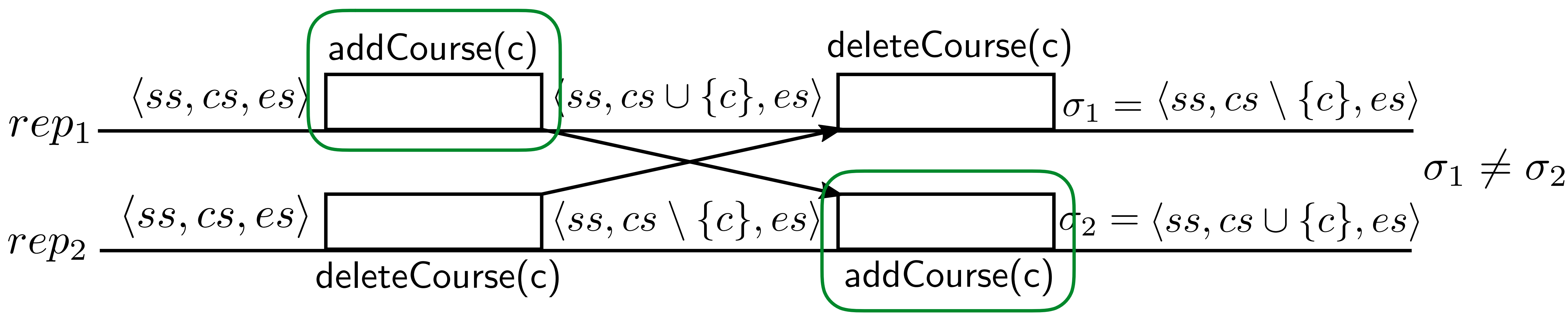
$\mathcal{S}$-conflict



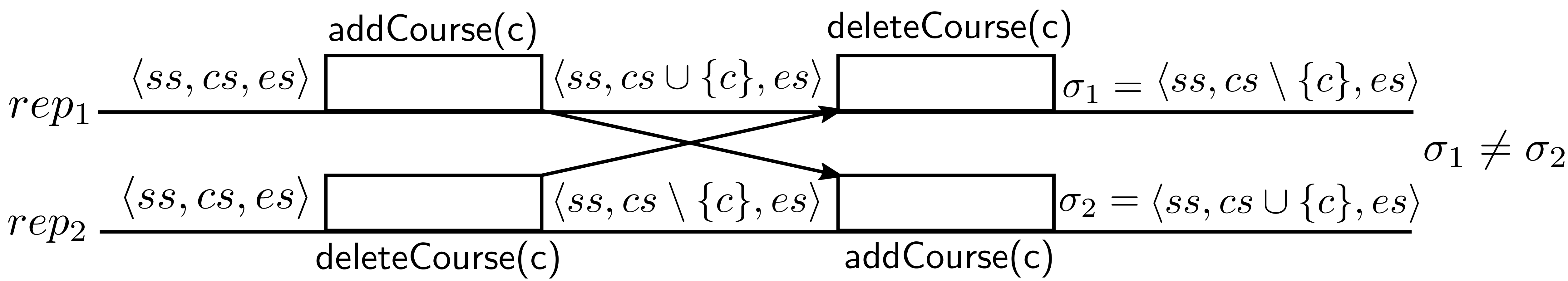
$\mathcal{S}$-conflict



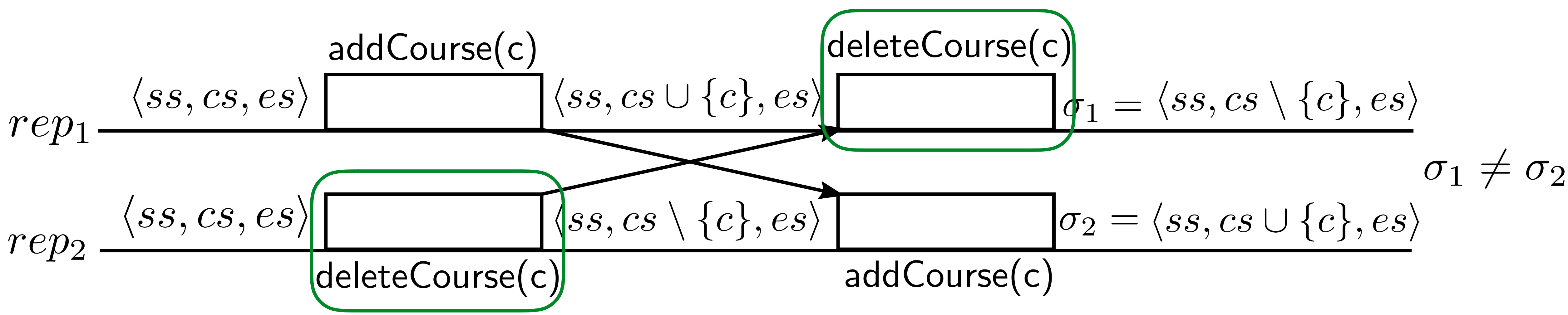
$\mathcal{S}$-conflict



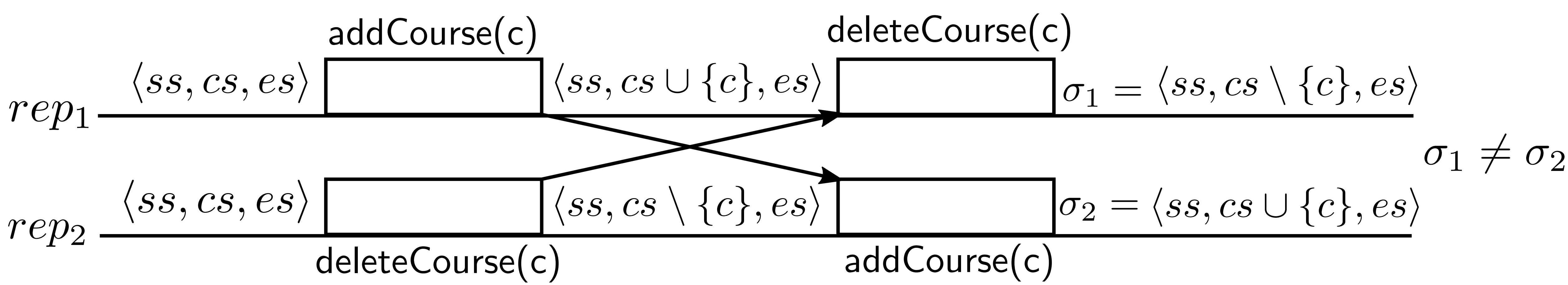
$\mathcal{S}$-conflict



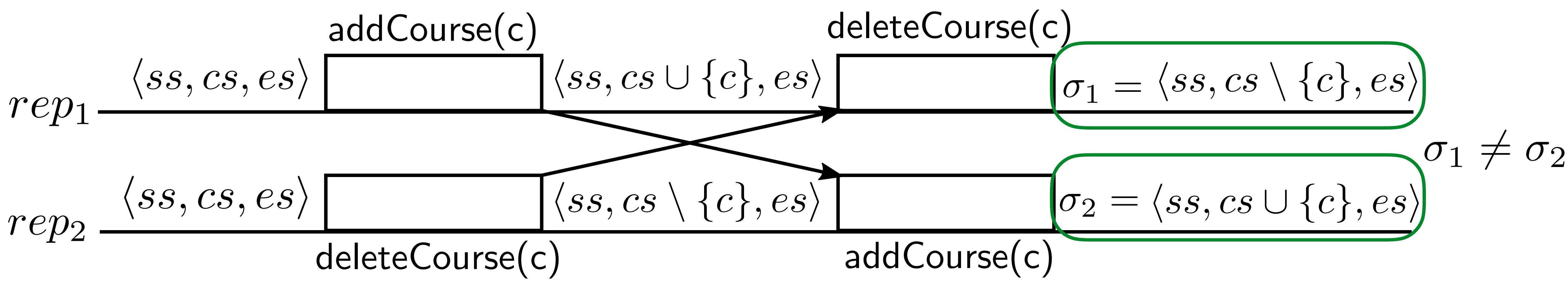
$\mathcal{S}$-conflict



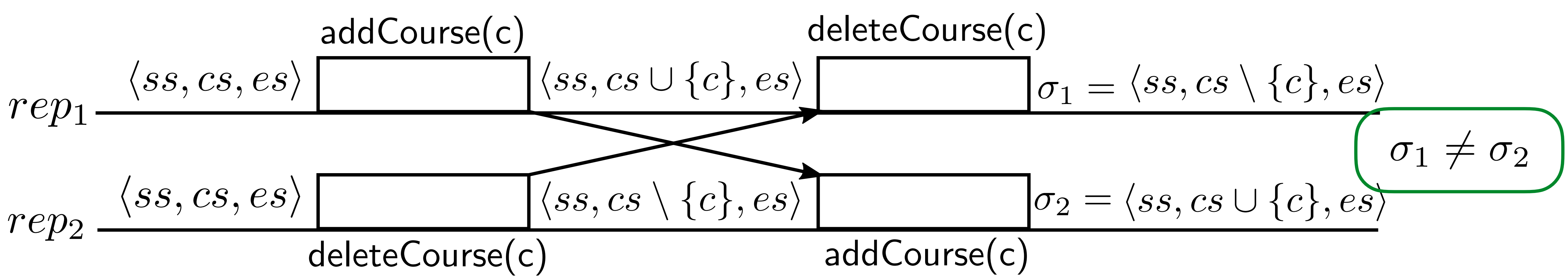
$\mathcal{S}$-conflict



$\mathcal{S}$-conflict

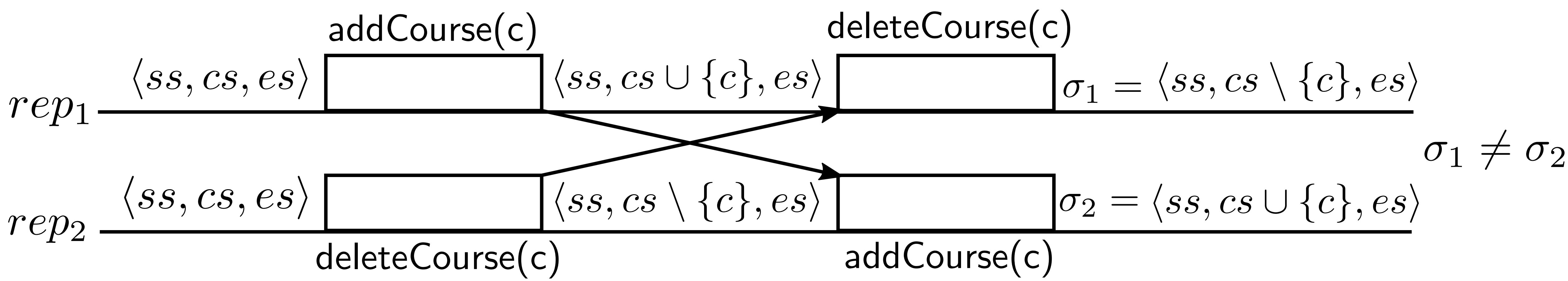


$\mathcal{S}$-conflict

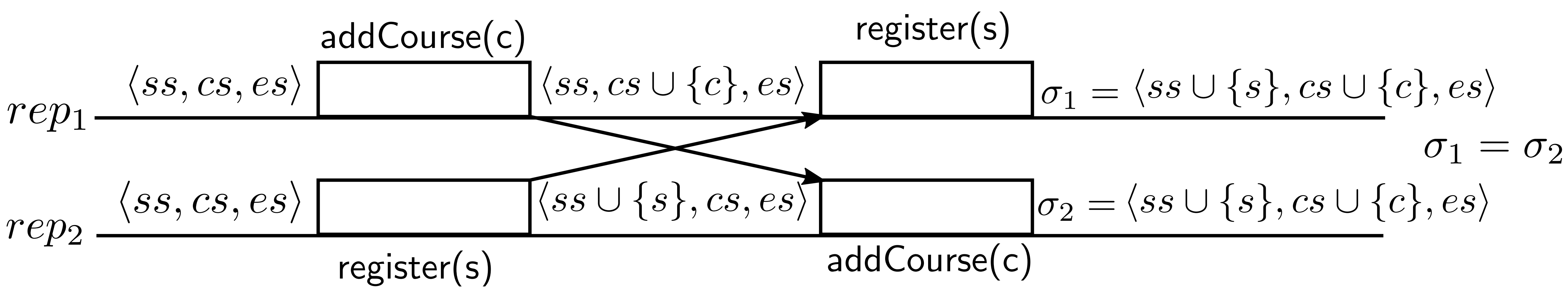


1 State-Conflict

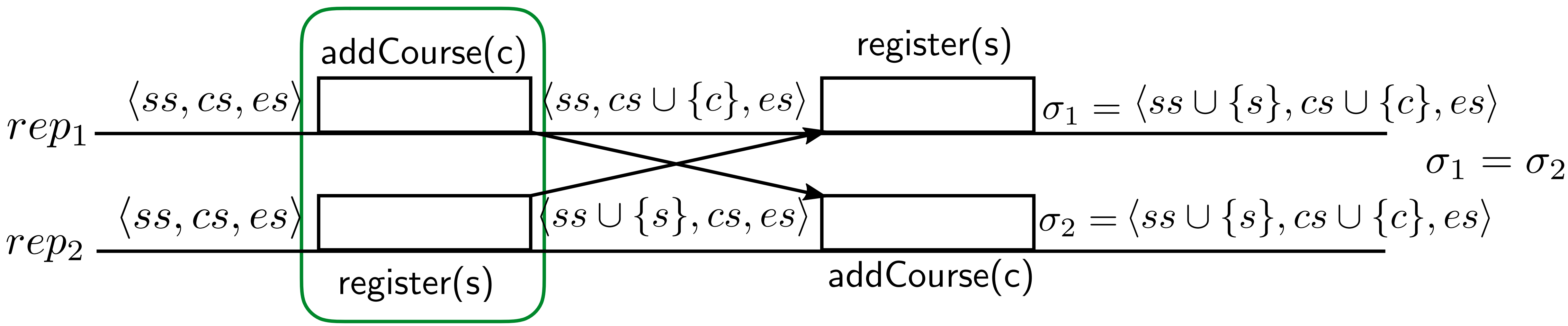
$\mathcal{S}$-conflict



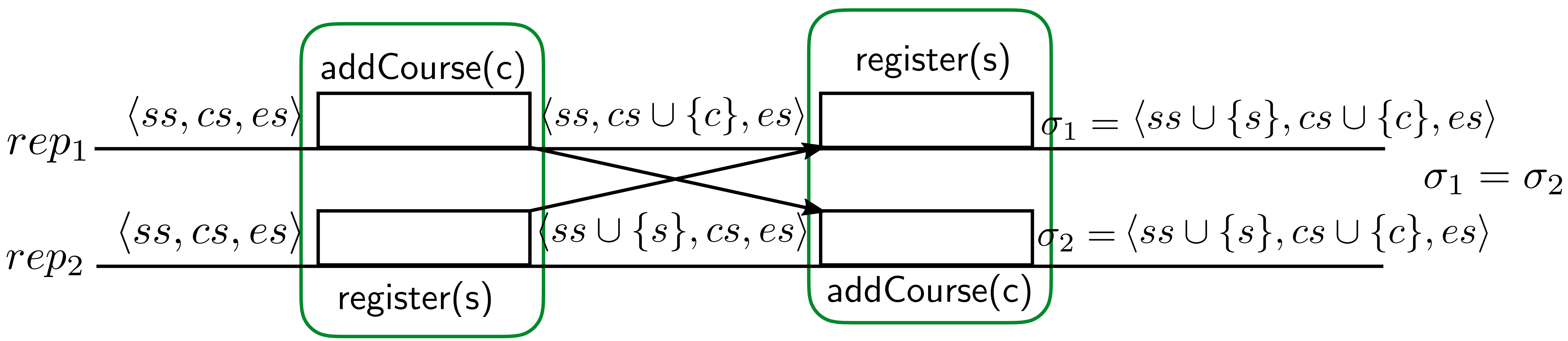
$\mathcal{S}$-commute



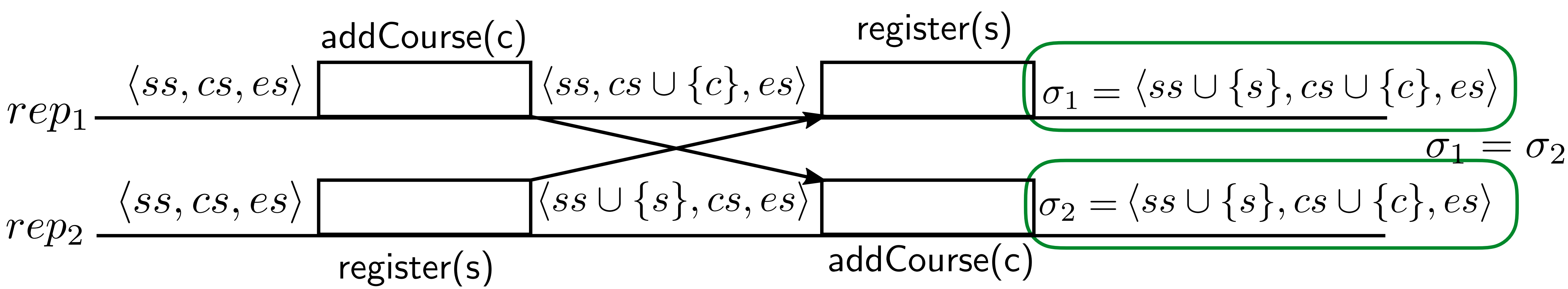
$\mathcal{S}$-commute



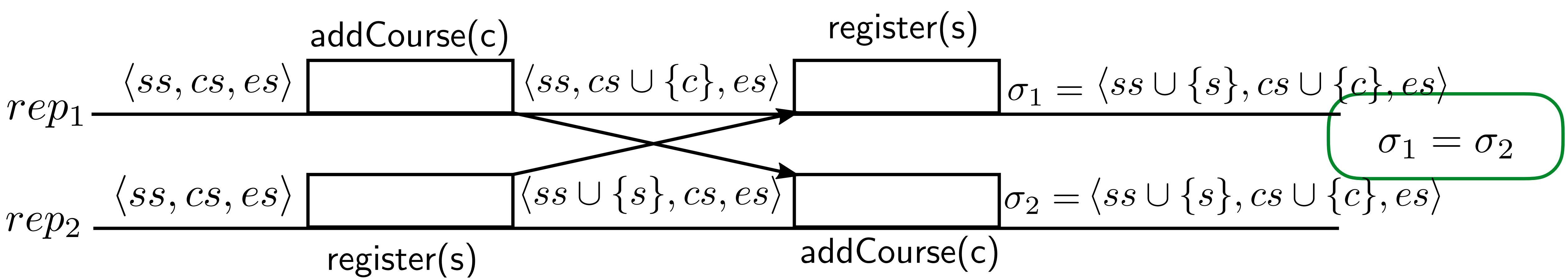
$\mathcal{S}$-commute



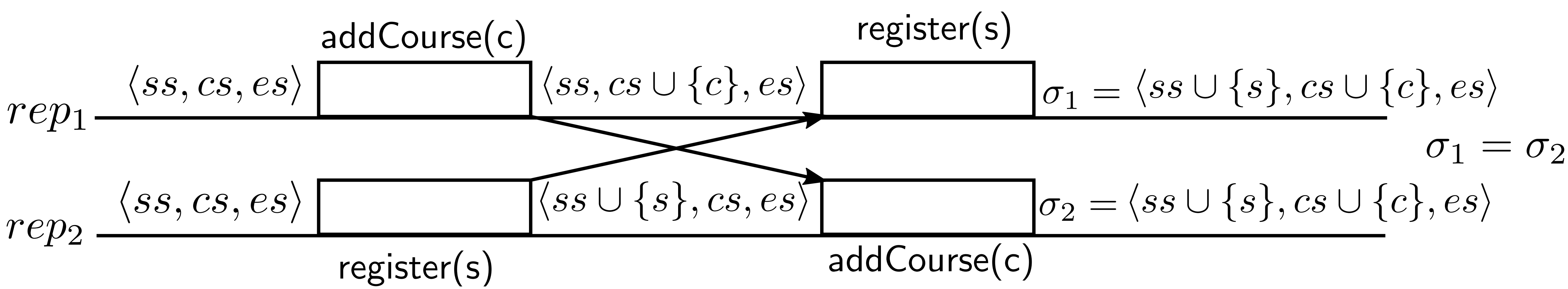
$\mathcal{S}$-commute



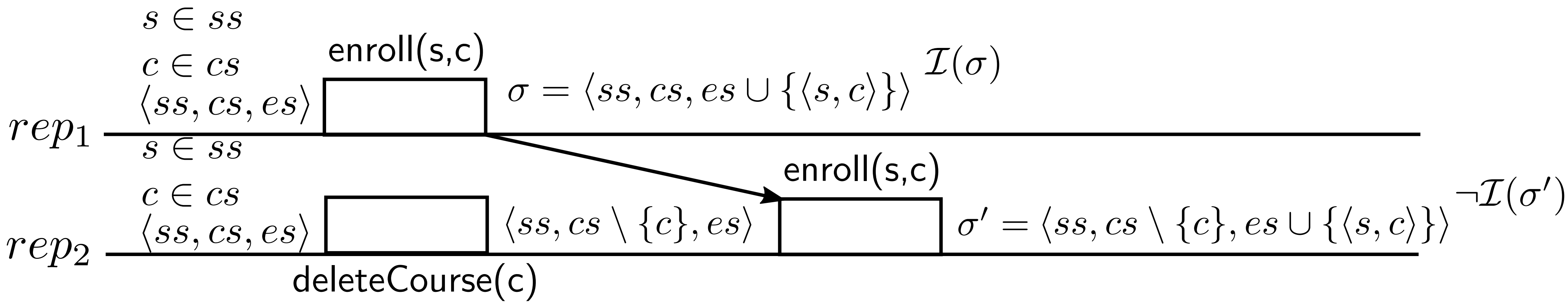
$\mathcal{S}$-commute



$\mathcal{S}$-commute

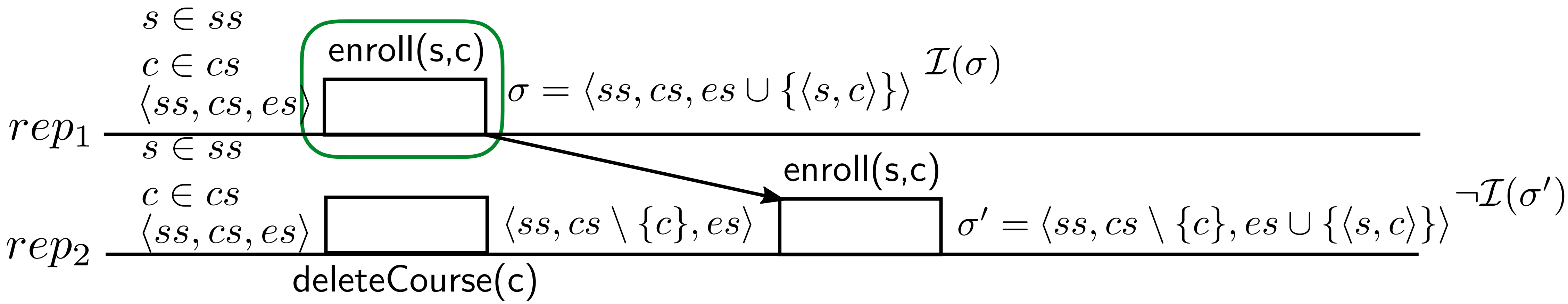


$\mathcal{P}$-conflict

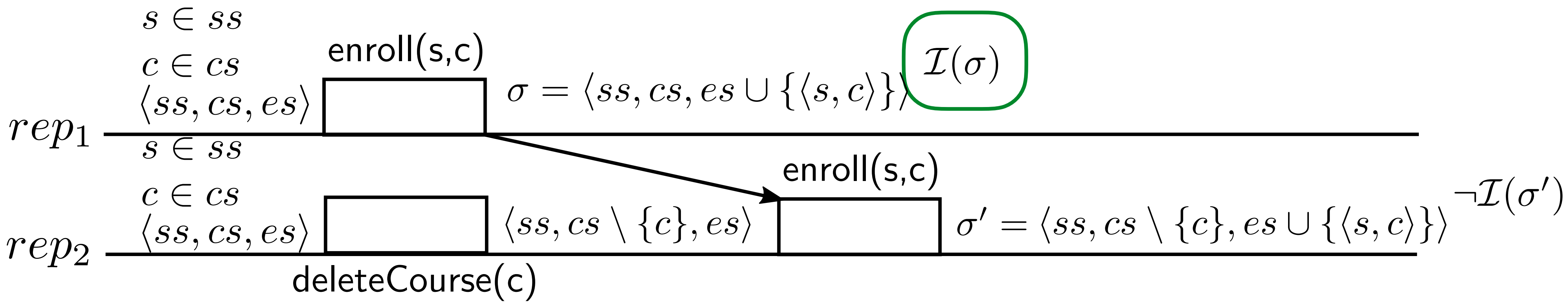


2 Permissible-Conflict

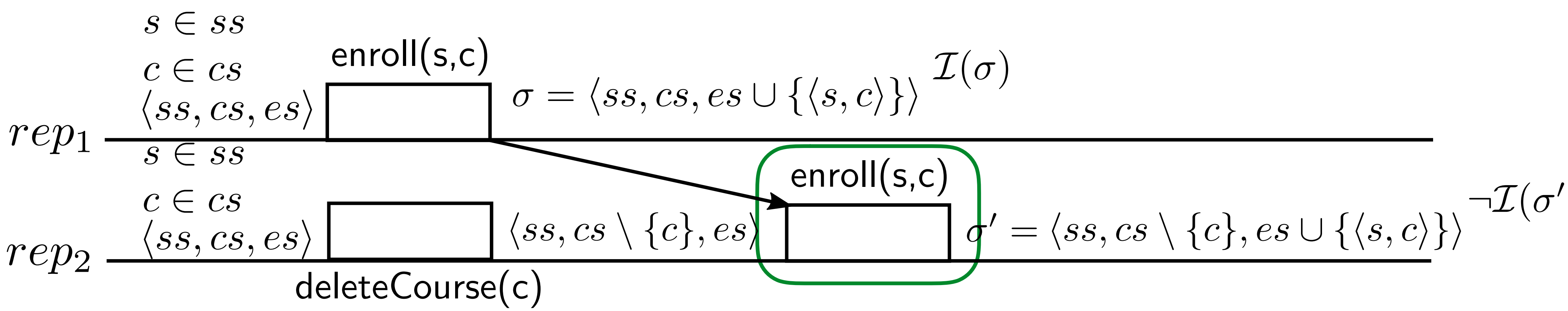
$\mathcal{P}$-conflict



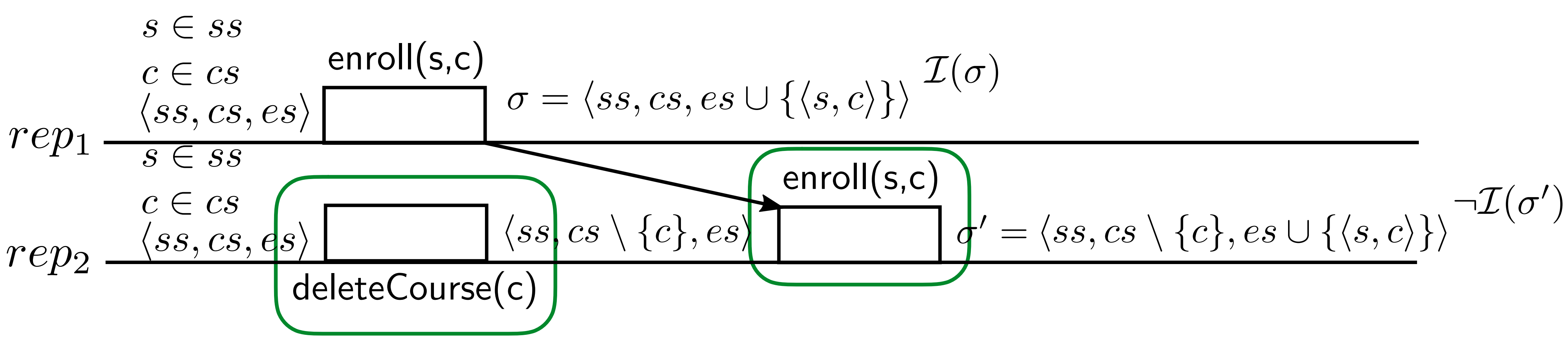
$\mathcal{P}$-conflict



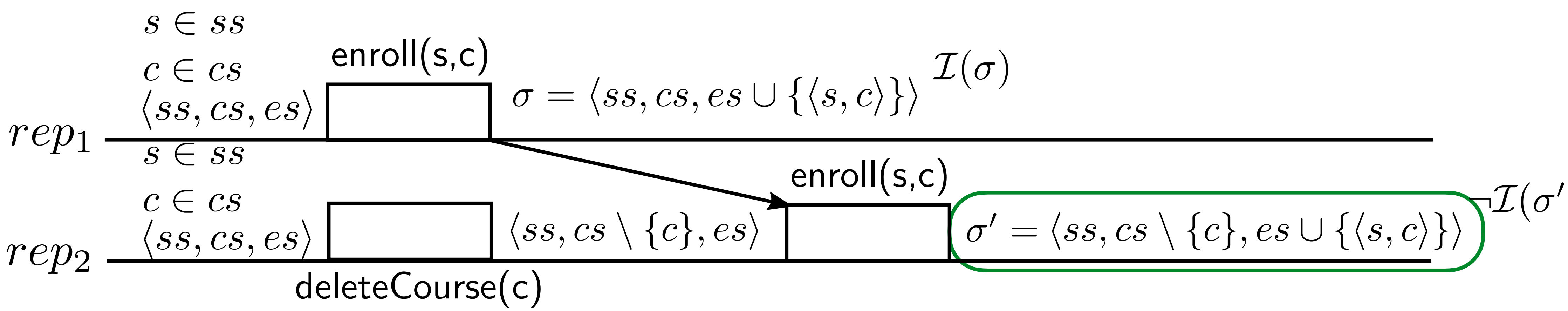
$\mathcal{P}$-conflict



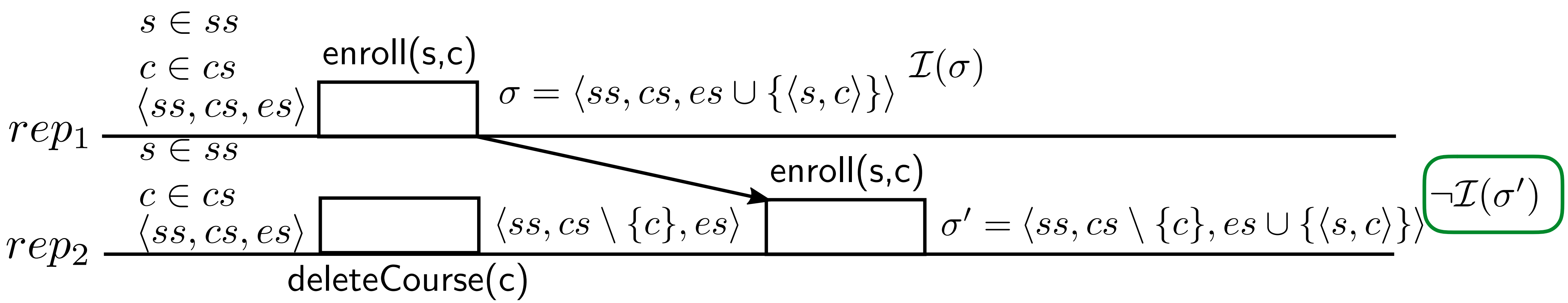
$\mathcal{P}$-conflict



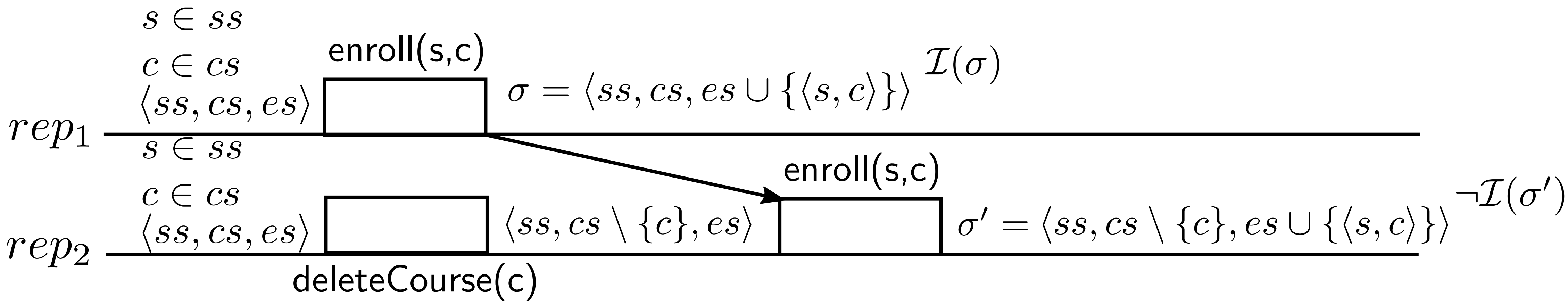
$\mathcal{P}$-conflict



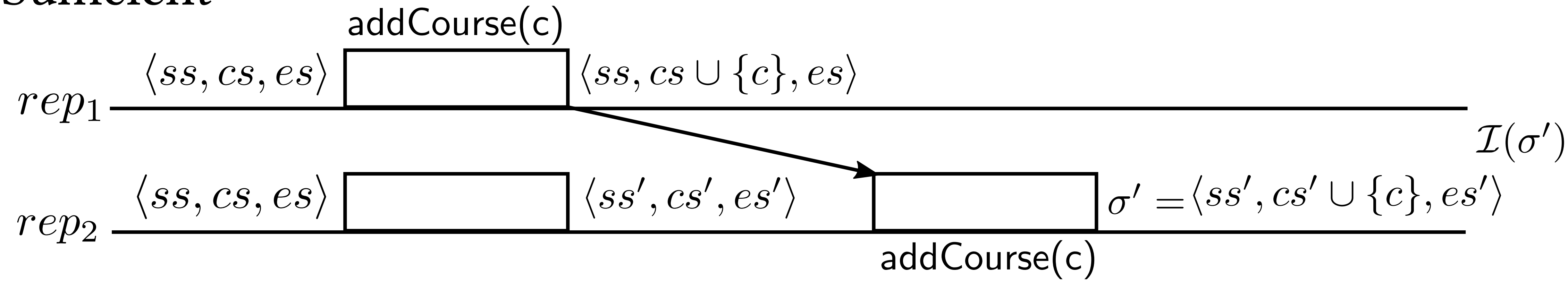
$\mathcal{P}$-conflict



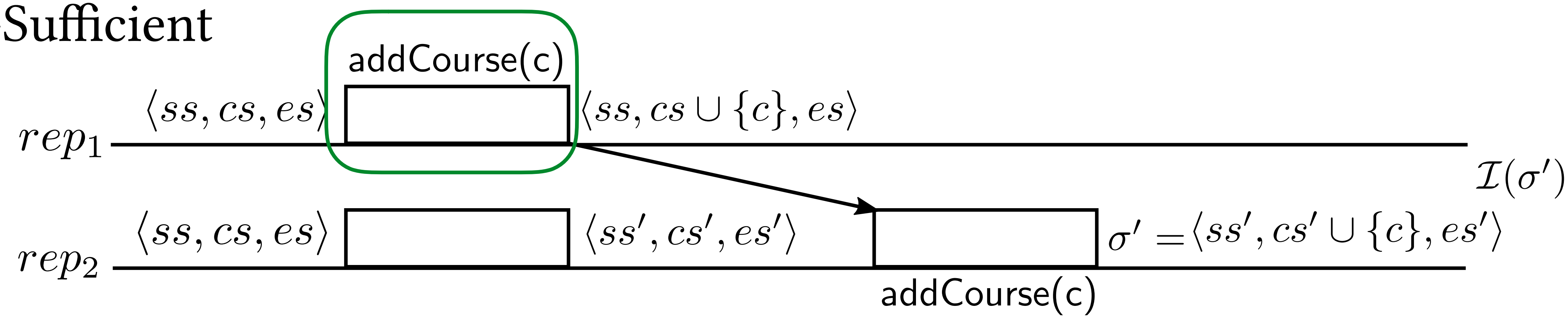
$\mathcal{P}$-conflict



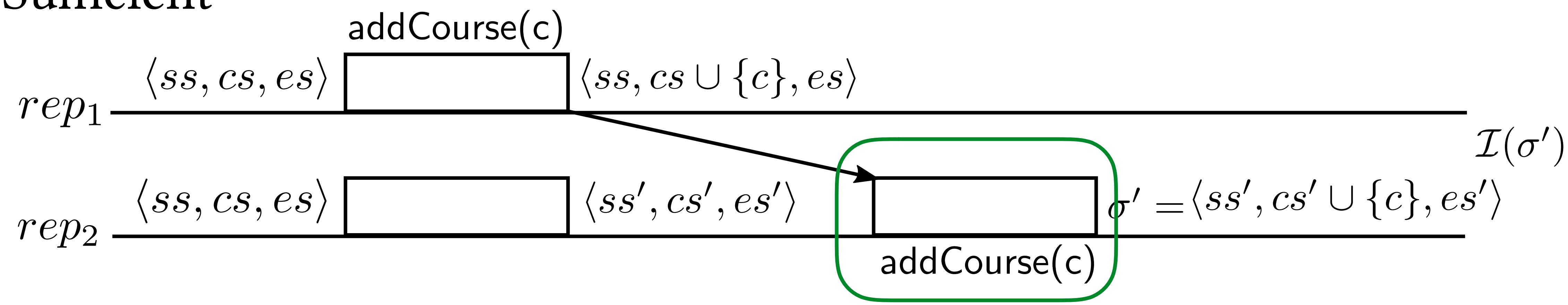
$\mathcal{I}$-Sufficient



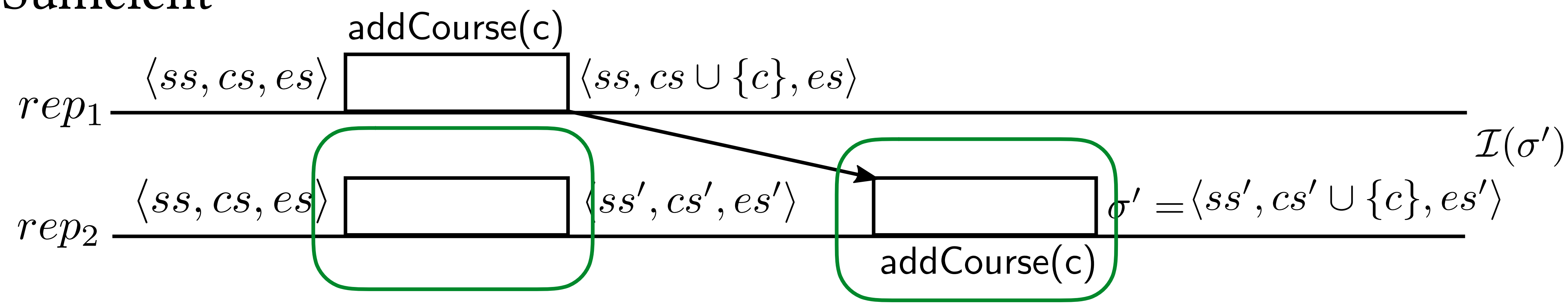
$\mathcal{I}$-Sufficient



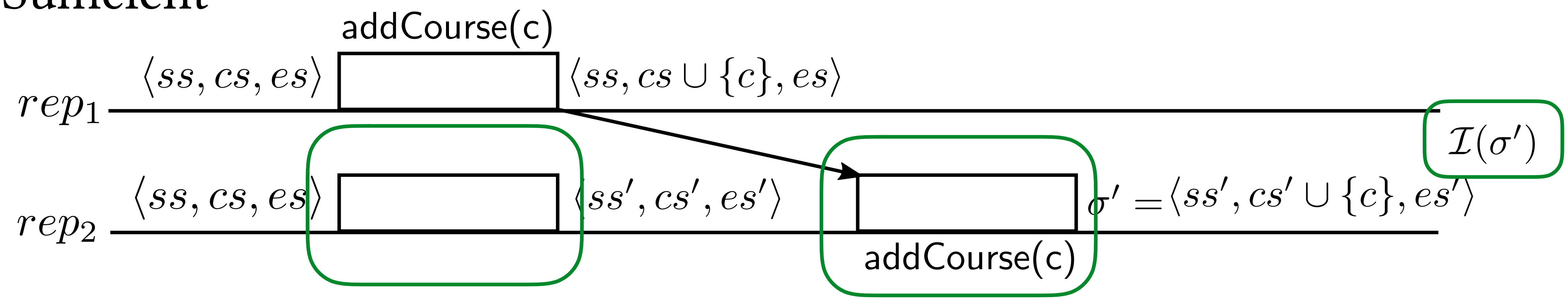
$\mathcal{I}$-Sufficient



$\mathcal{I}$-Sufficient

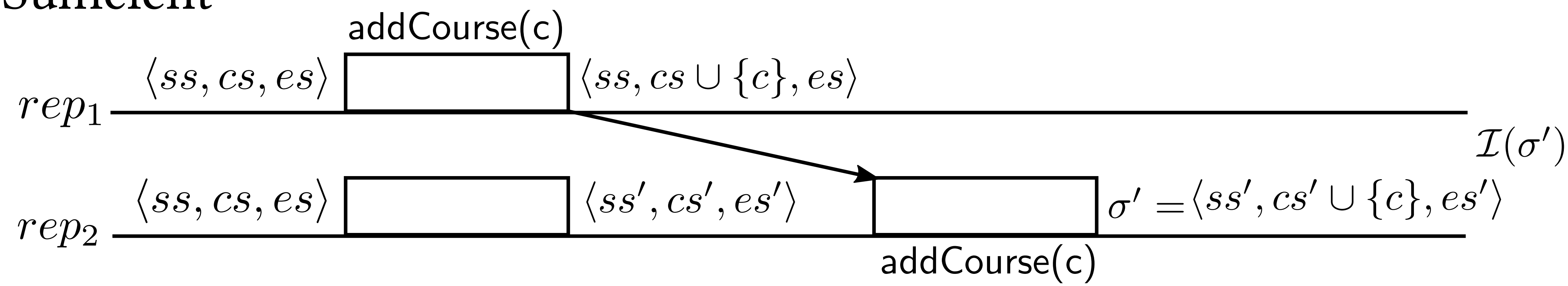


$\mathcal{I}$-Sufficient

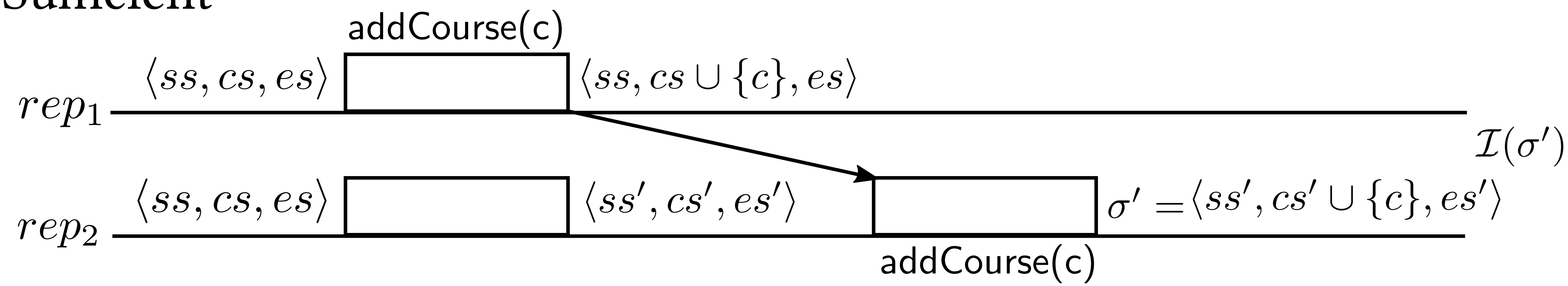


2 Permissible-Concur

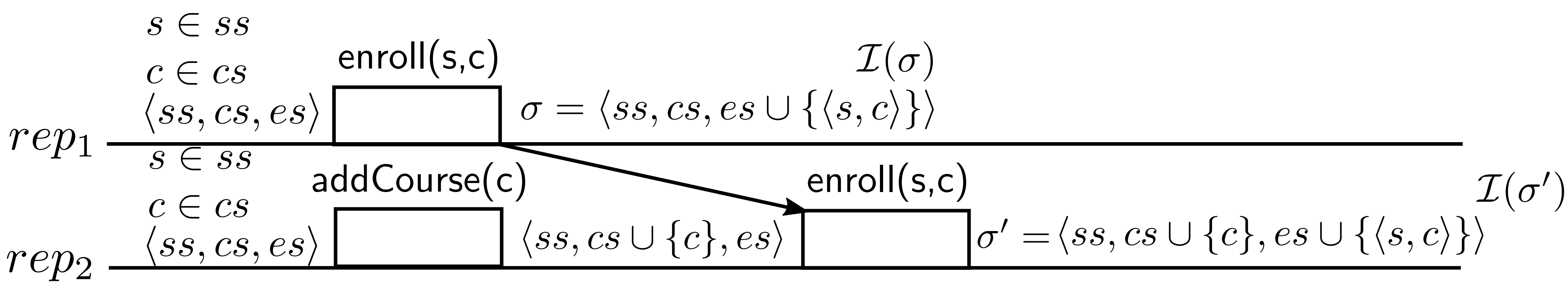
$\mathcal{I}$-Sufficient



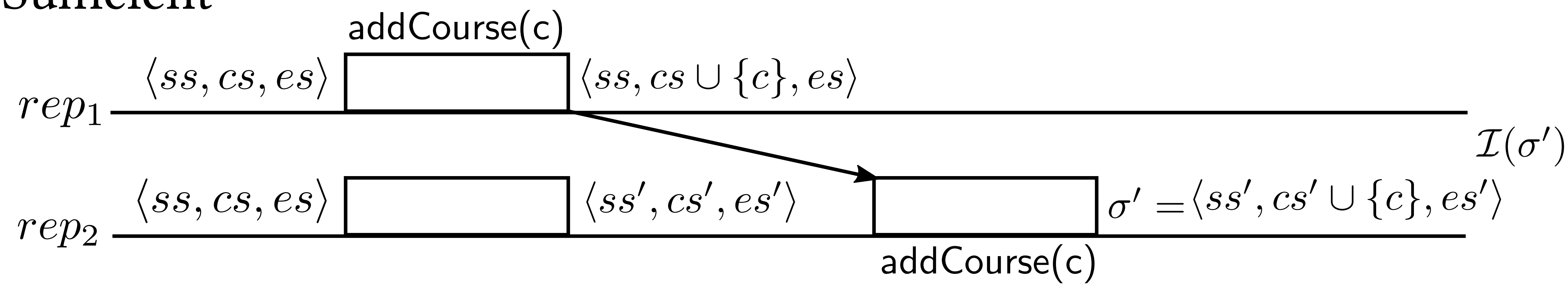
$\mathcal{I}$-Sufficient



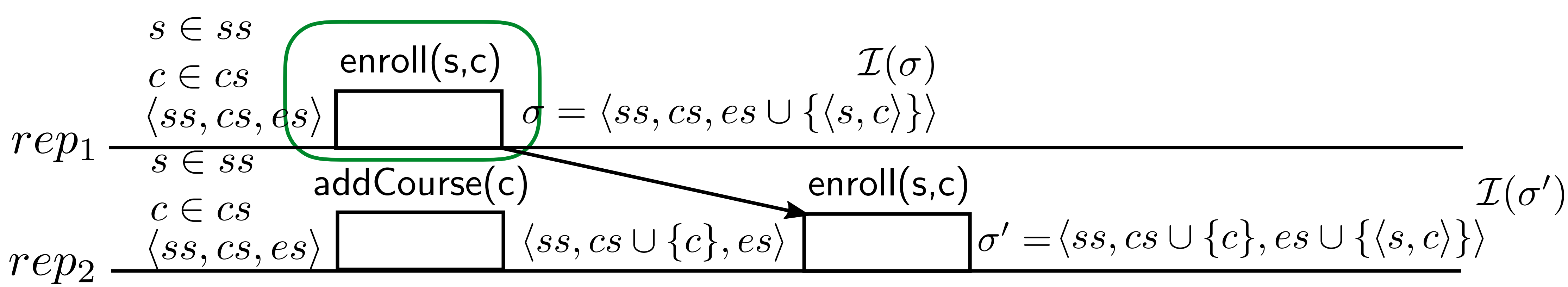
$\mathcal{P}$-R-Commutativity



$\mathcal{I}$-Sufficient

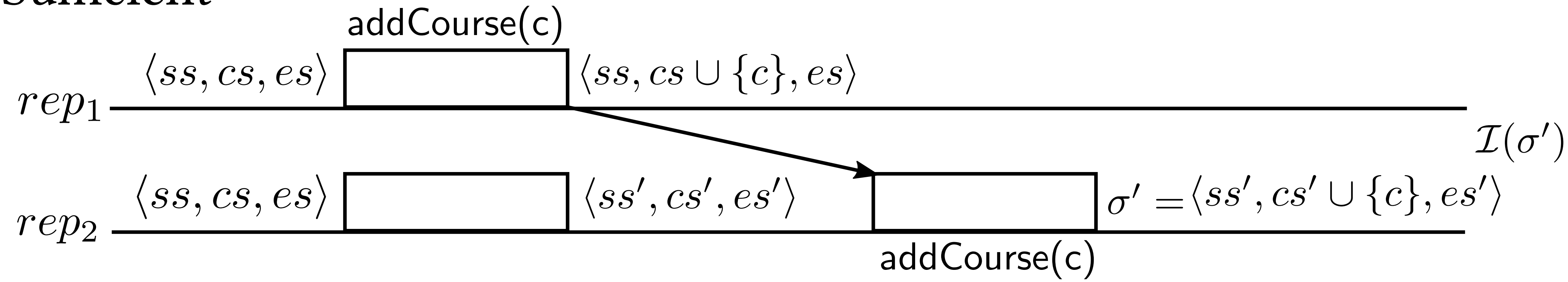


$\mathcal{P}$-R-Commutativity

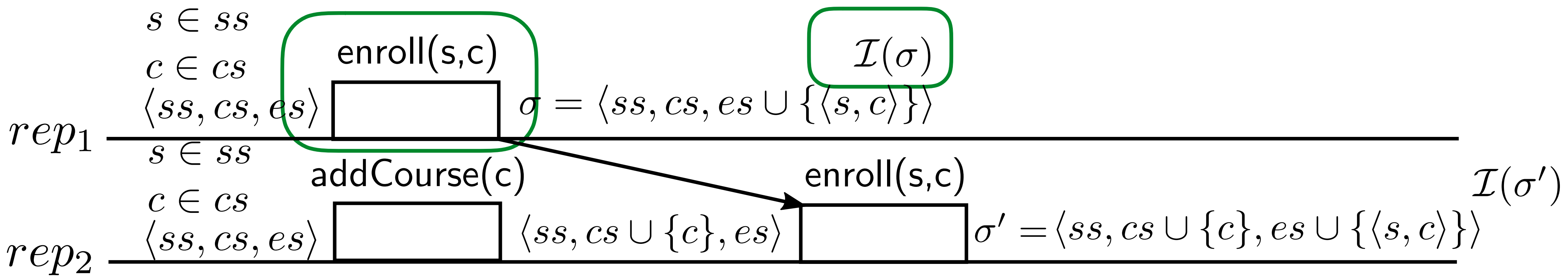


2 Permissible-Concur

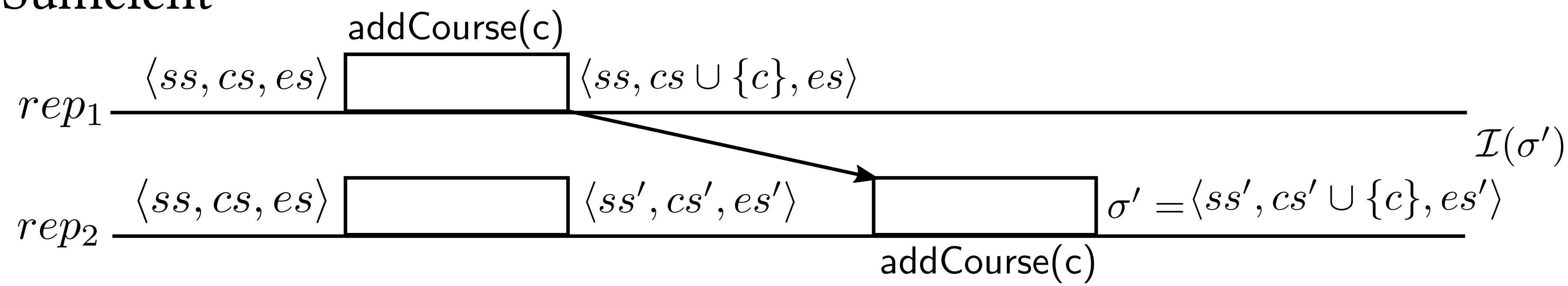
$\mathcal{I}$-Sufficient



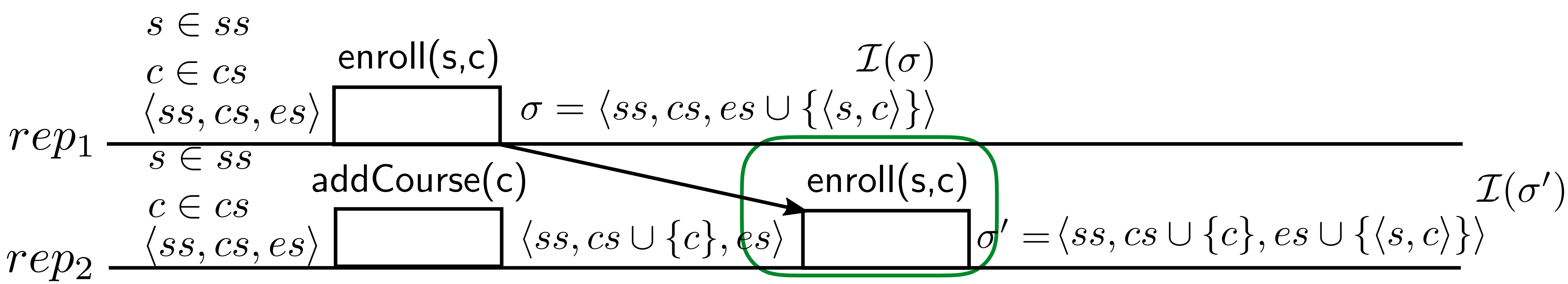
$\mathcal{P}$-R-Commutativity



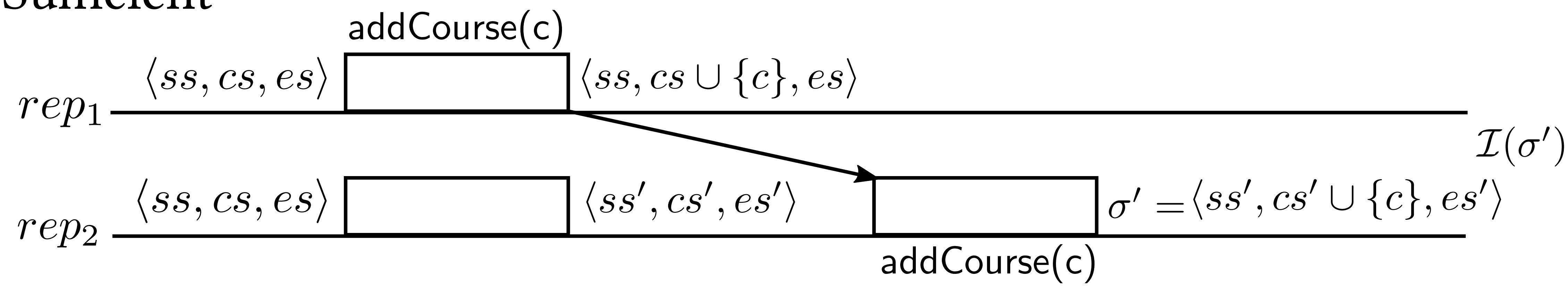
$\mathcal{I}$-Sufficient



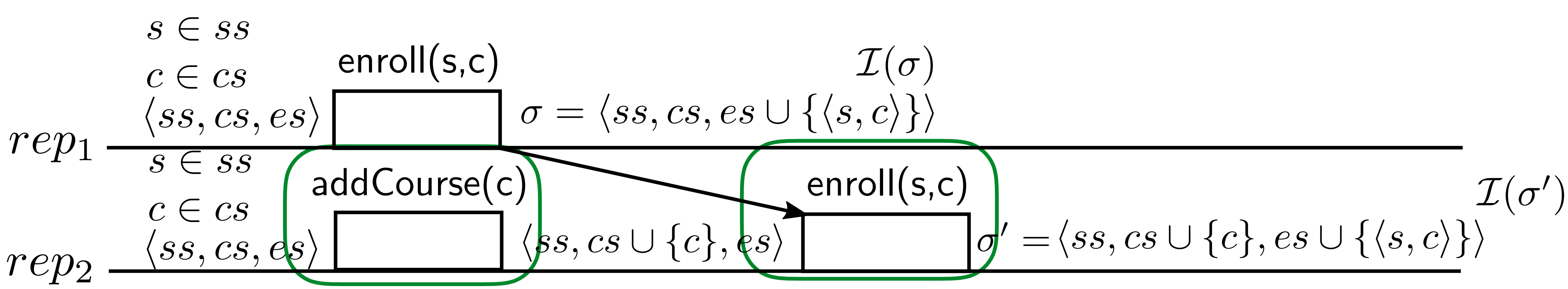
$\mathcal{P}$-R-Commutativity



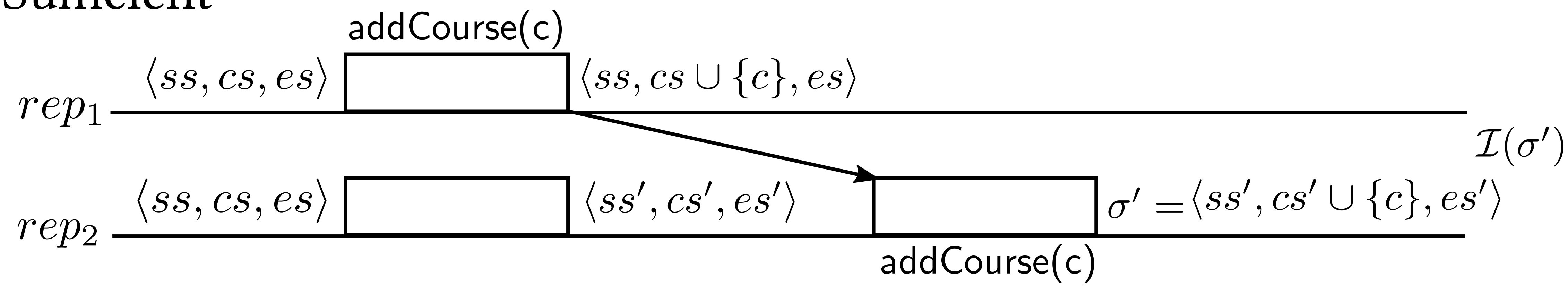
$\mathcal{I}$-Sufficient



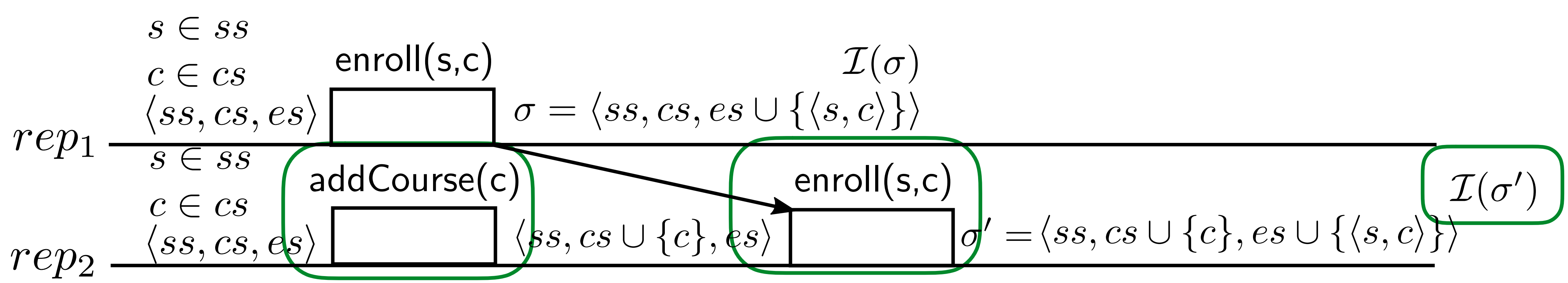
$\mathcal{P}$-R-Commutativity



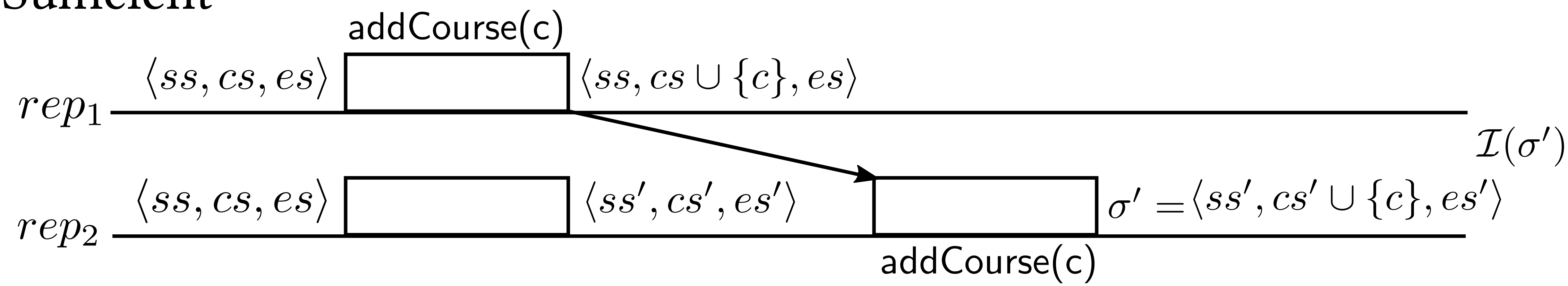
$\mathcal{I}$-Sufficient



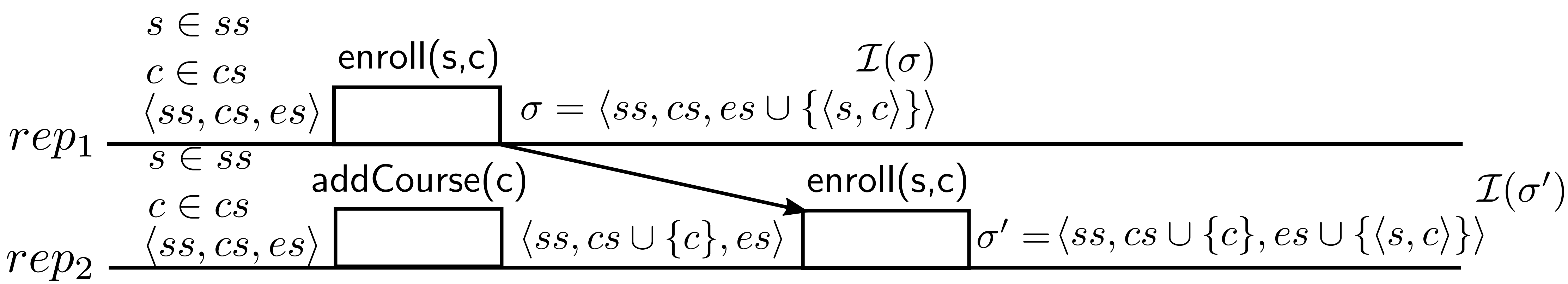
$\mathcal{P}$-R-Commutativity

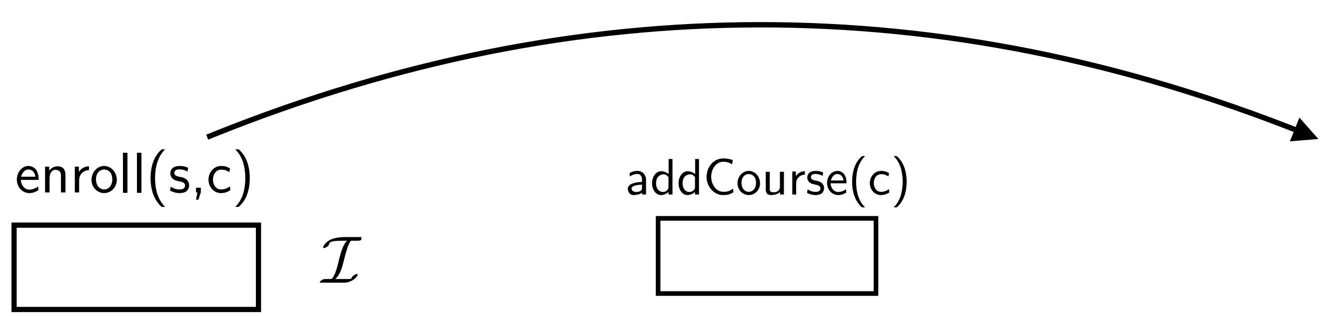


$\mathcal{I}$-Sufficient

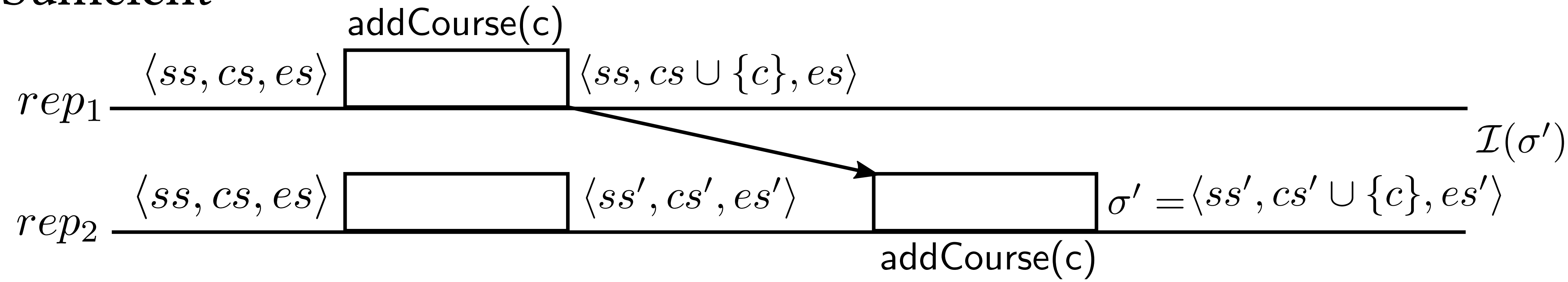


$\mathcal{P}$-R-Commutativity

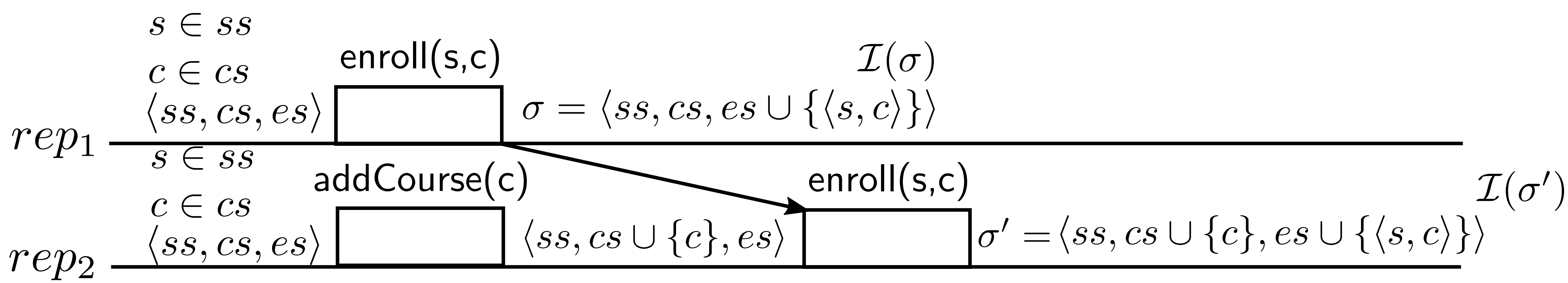


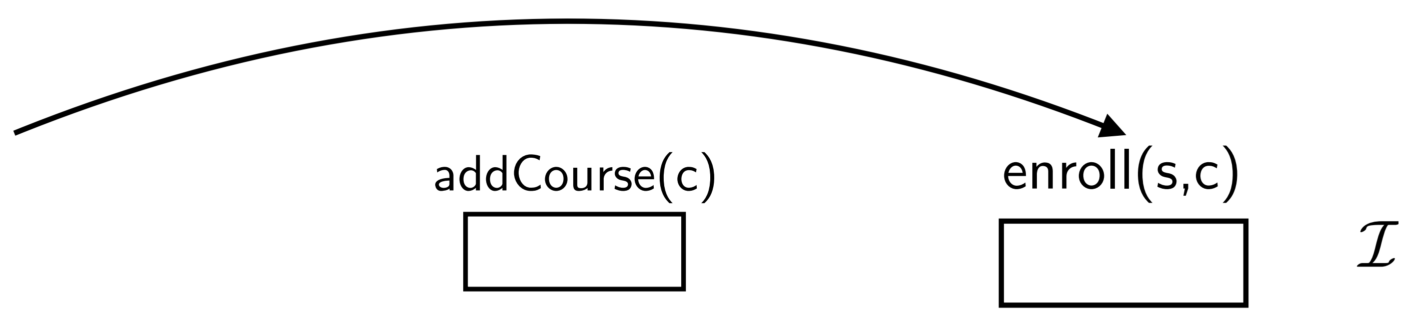


$\mathcal{I}$-Sufficient



$\mathcal{P}$-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	✓	✓	✓	✓	✓

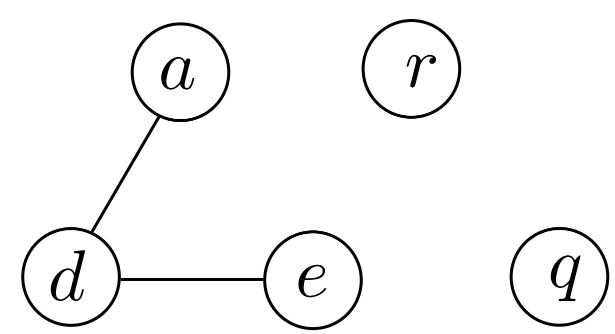
Concur

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

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

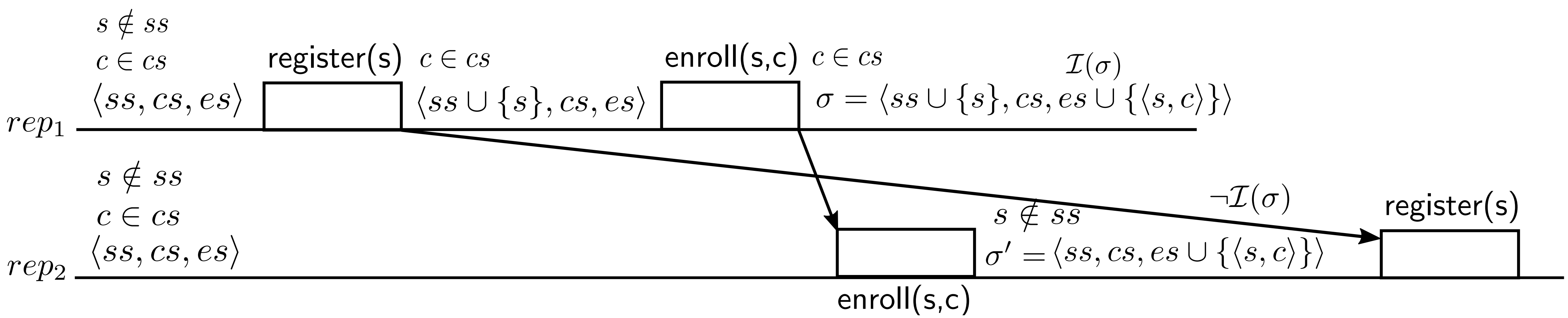
Conflict

$\neg$ Concur

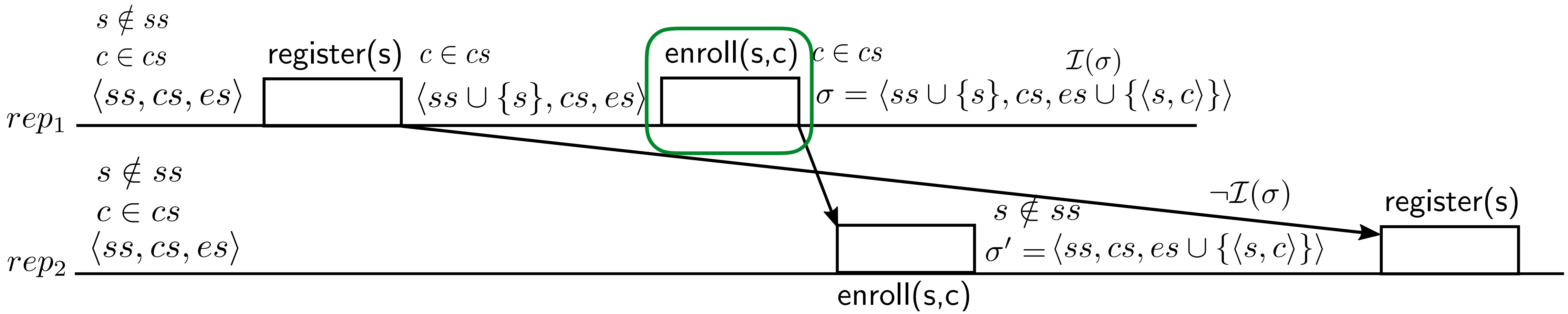


Dependence

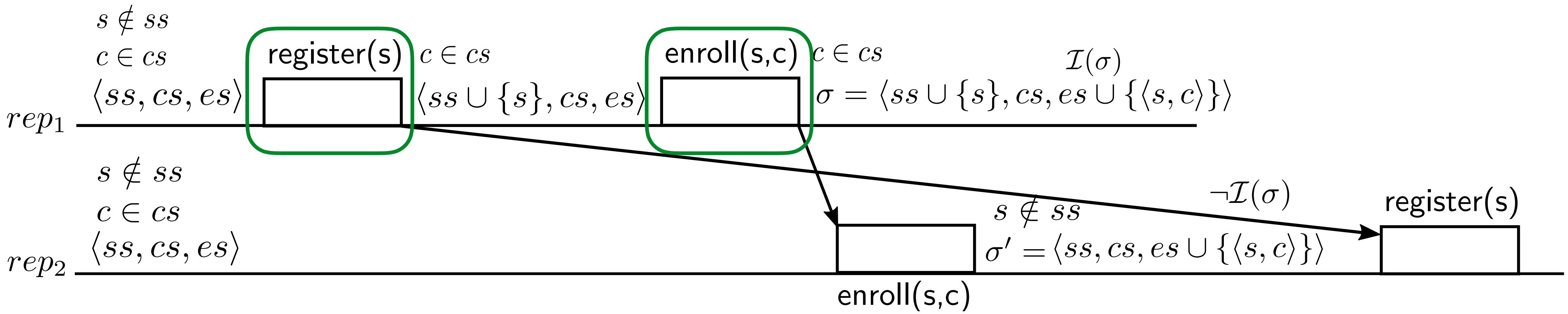
Dependence



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

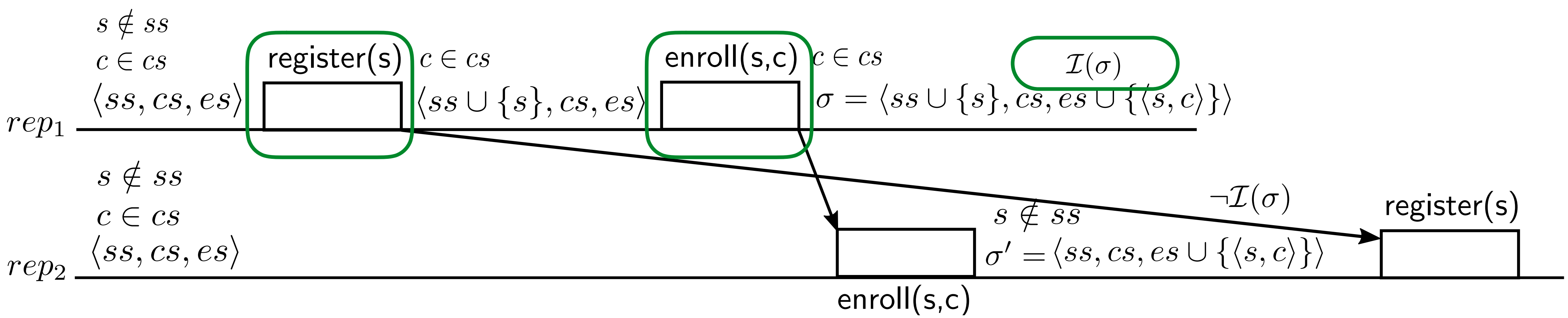
Dependence



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

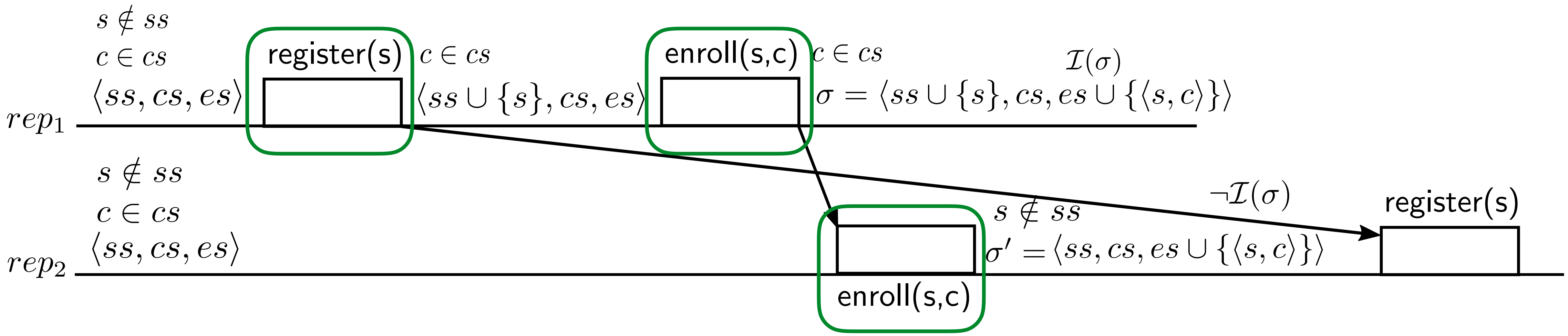
Dependence

Dependence

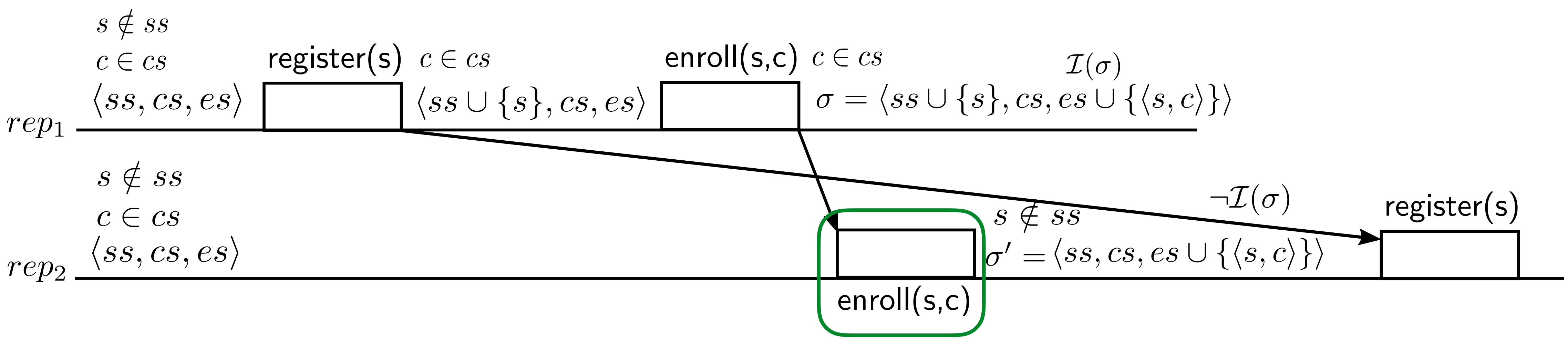


Dependence

Dependence

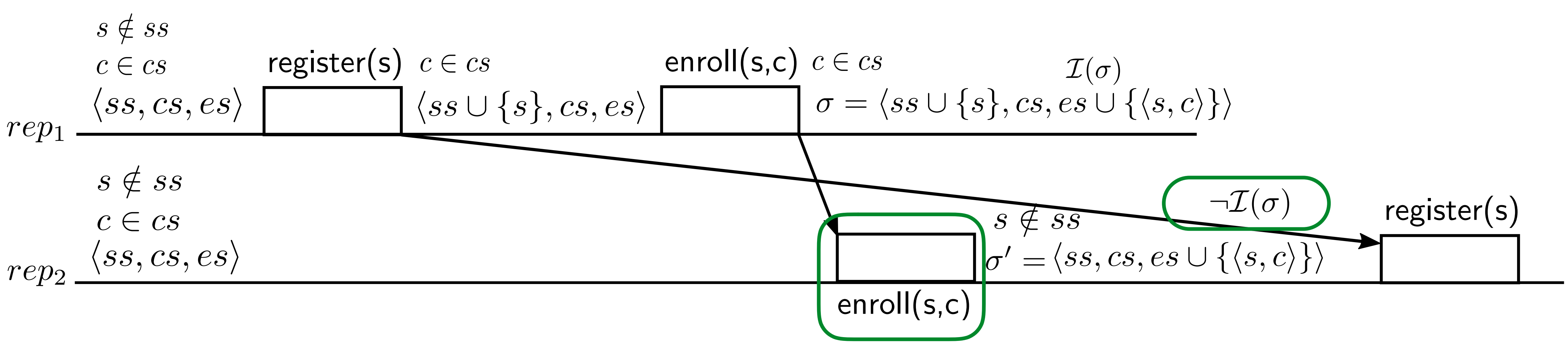


Dependence


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

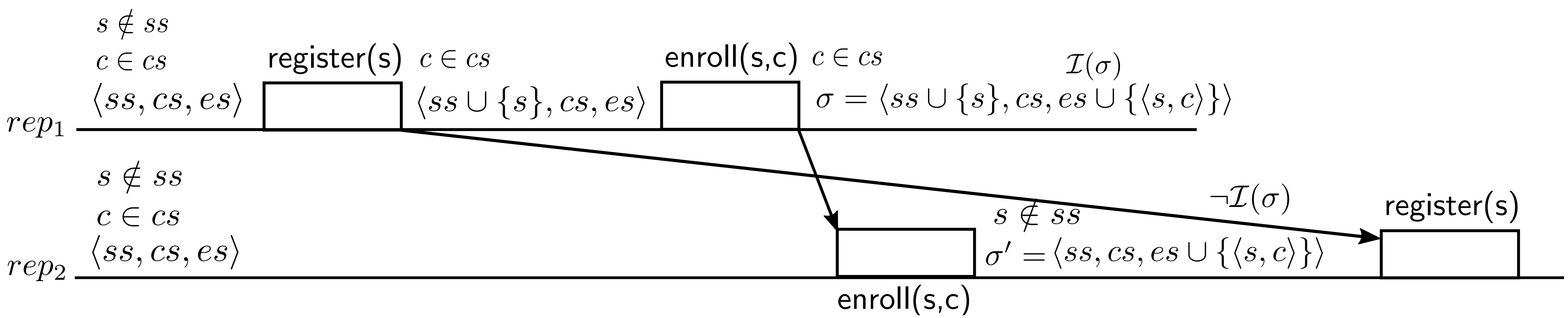
Dependence

Dependence



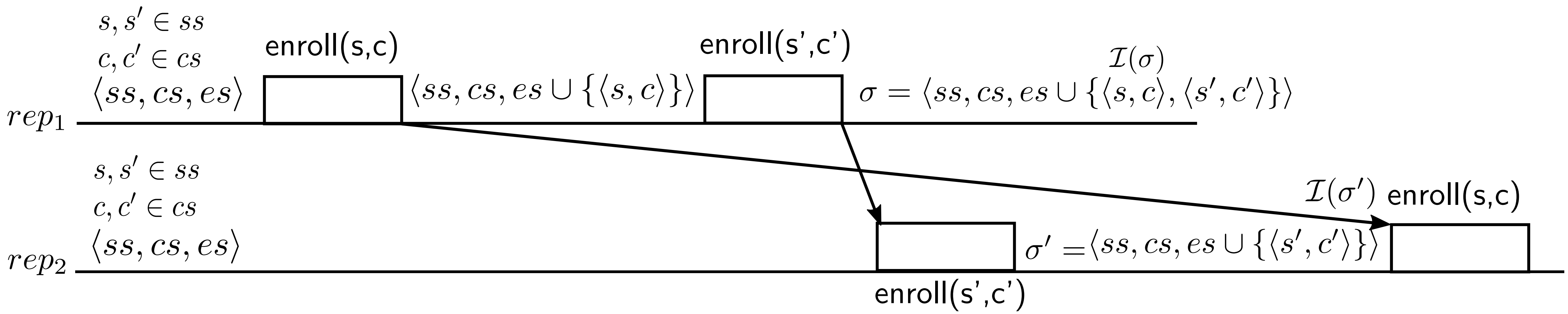
Dependence

Dependence



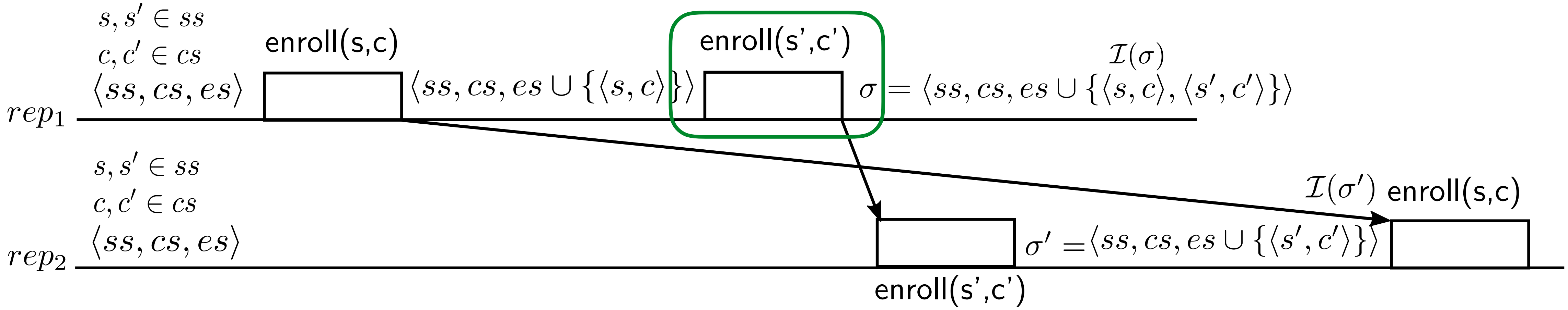
Independence

$\mathcal{P}$-L-commute



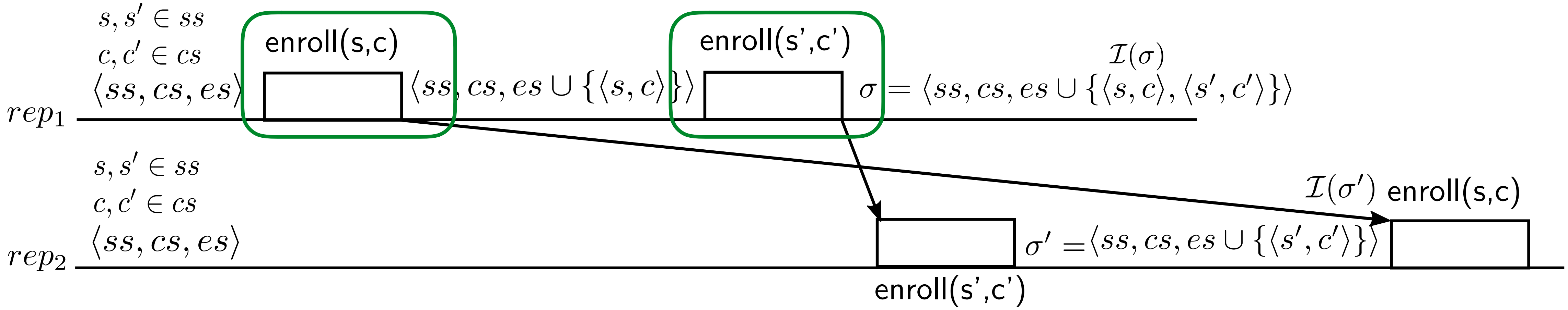
Independence

$\mathcal{P}$-L-commute



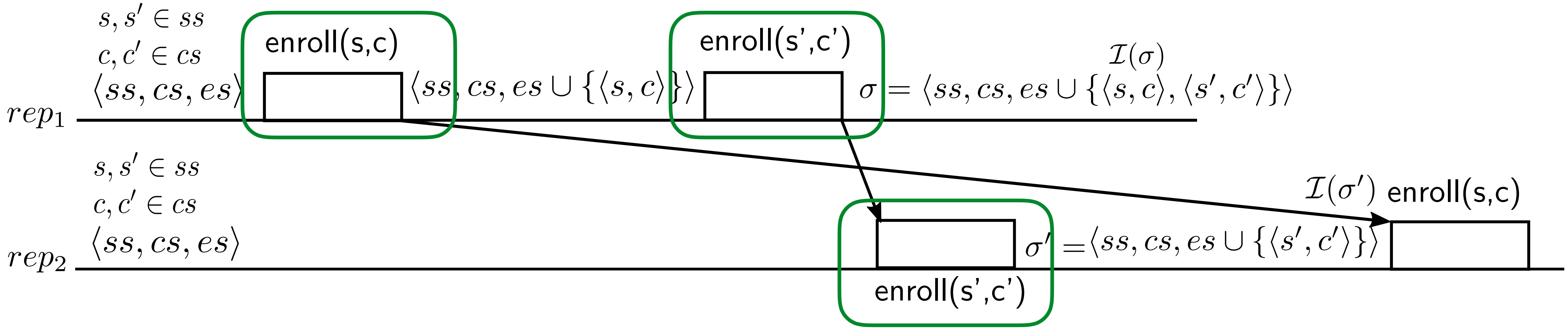
Independence

$\mathcal{P}$-L-commute



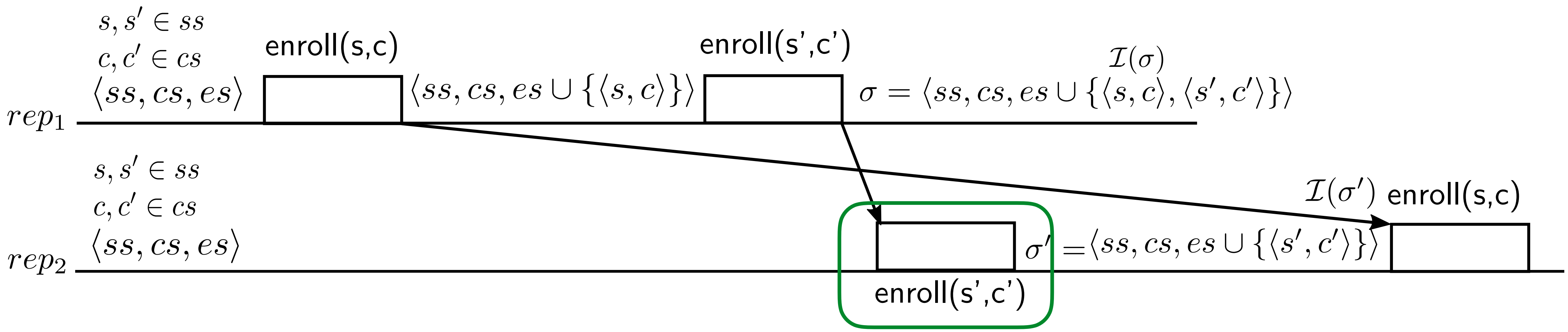
Independence

$\mathcal{P}$-L-commute



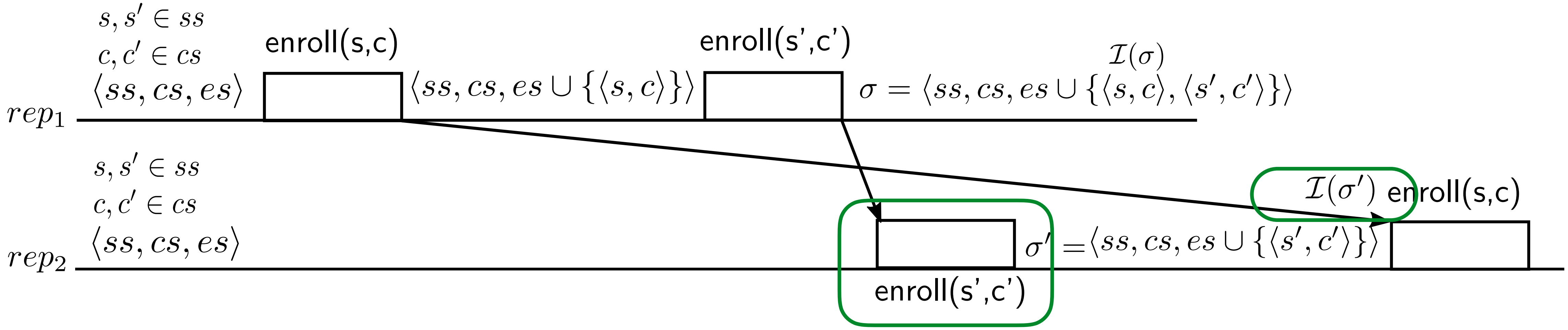
Independence

$\mathcal{P}$-L-commute



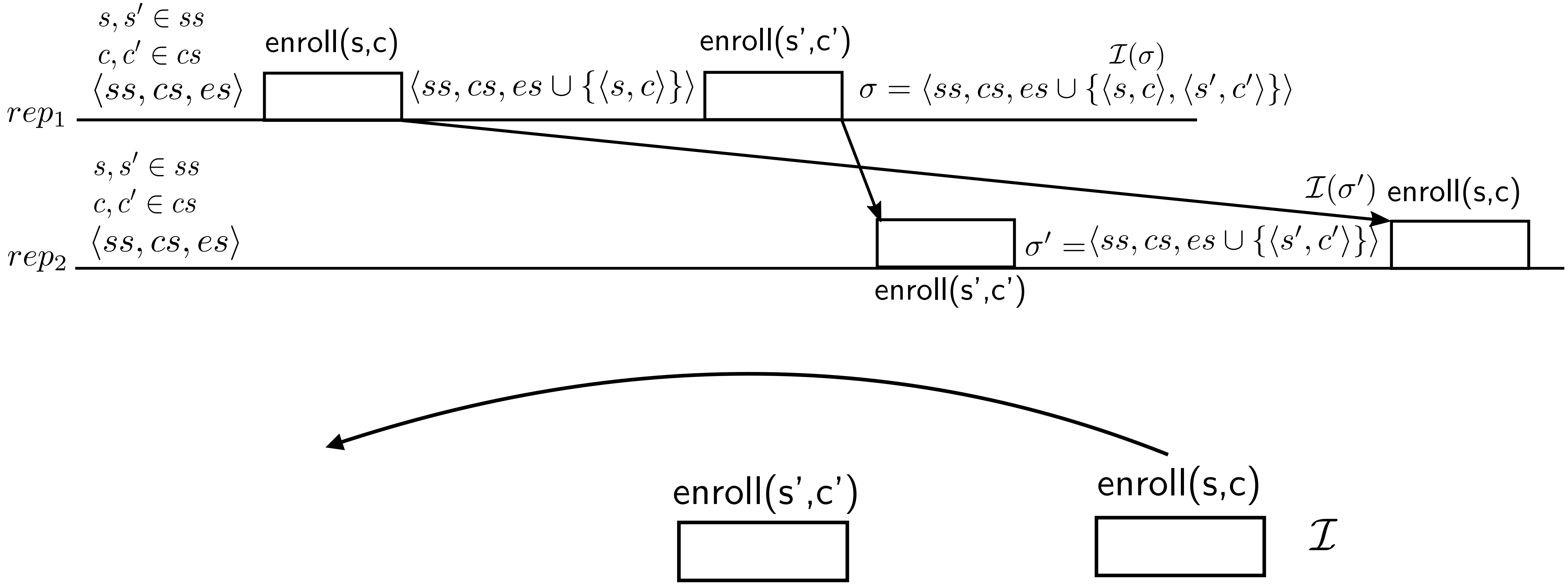
Independence

$\mathcal{P}$-L-commute



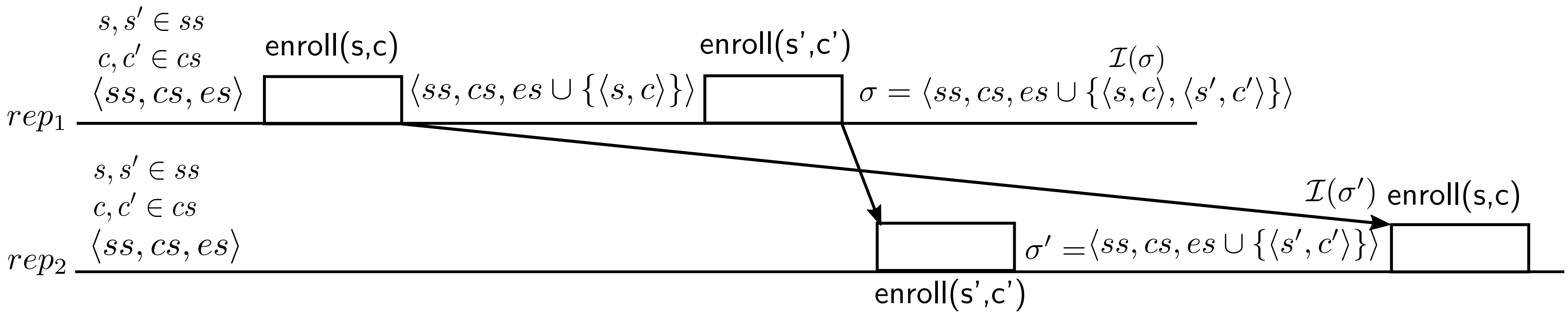
Independence

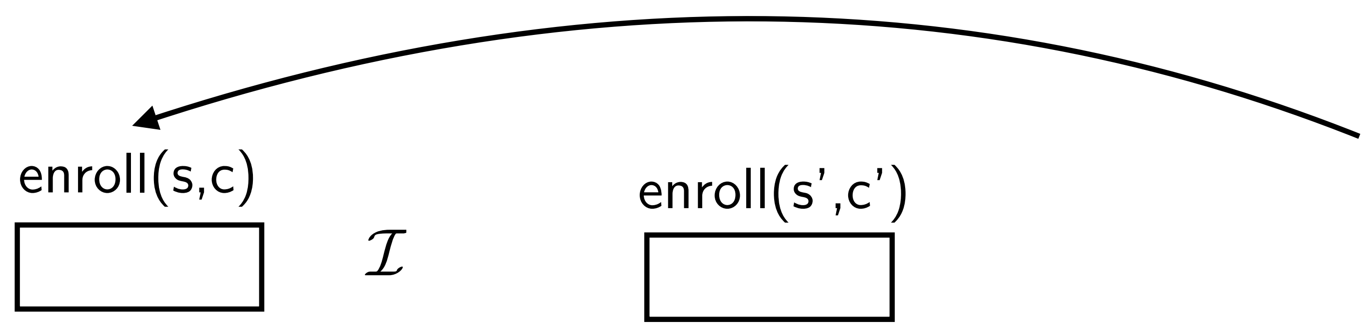
$\mathcal{P}$-L-commute



Independence

$\mathcal{P}$-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

Dependence

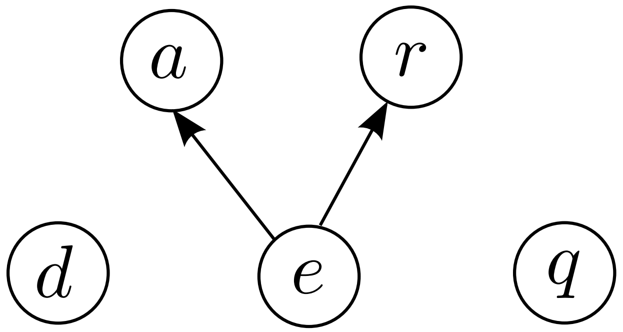
Independent

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

Dependent

$\neg$ Independent

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

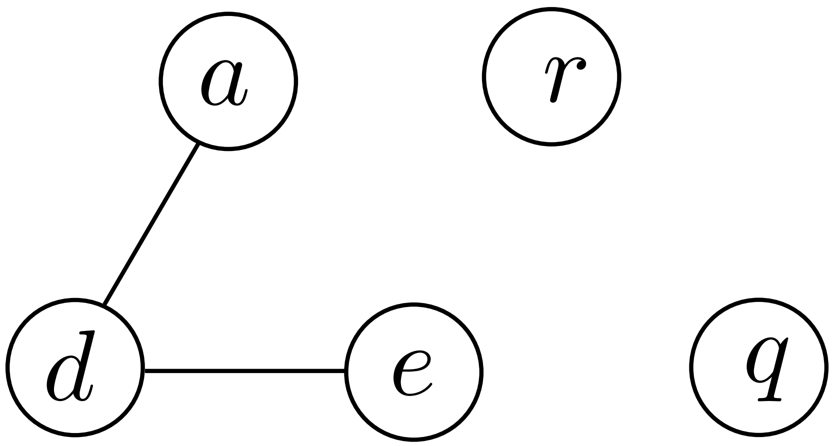


Well-coordination

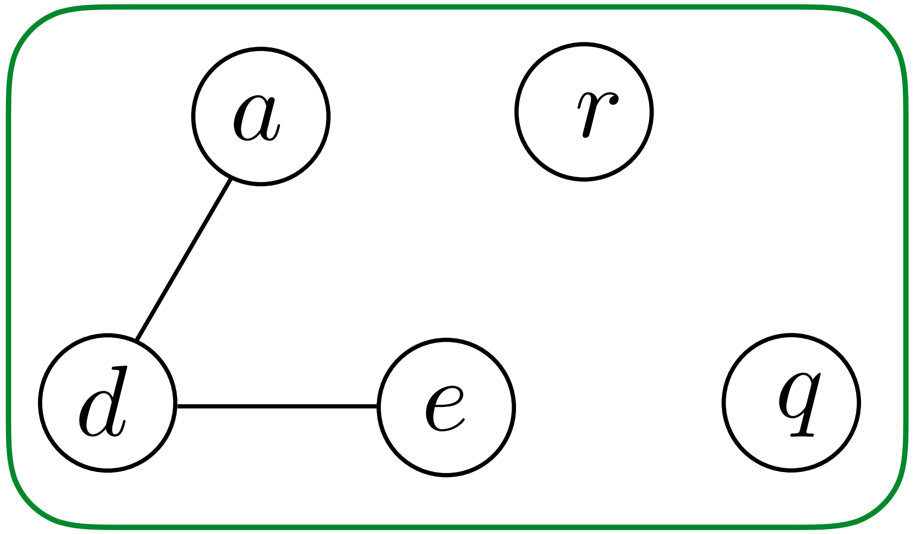
- Well-coordination
  - Locally Permissible
  - Conflict-Synchronizing
  - Dependency-preserving

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

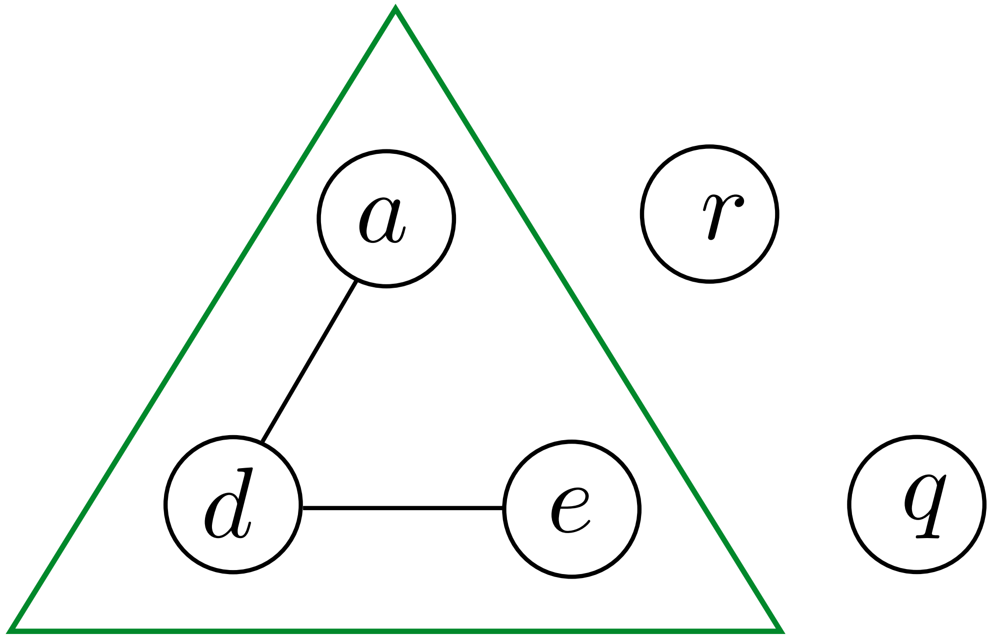
Synchronization

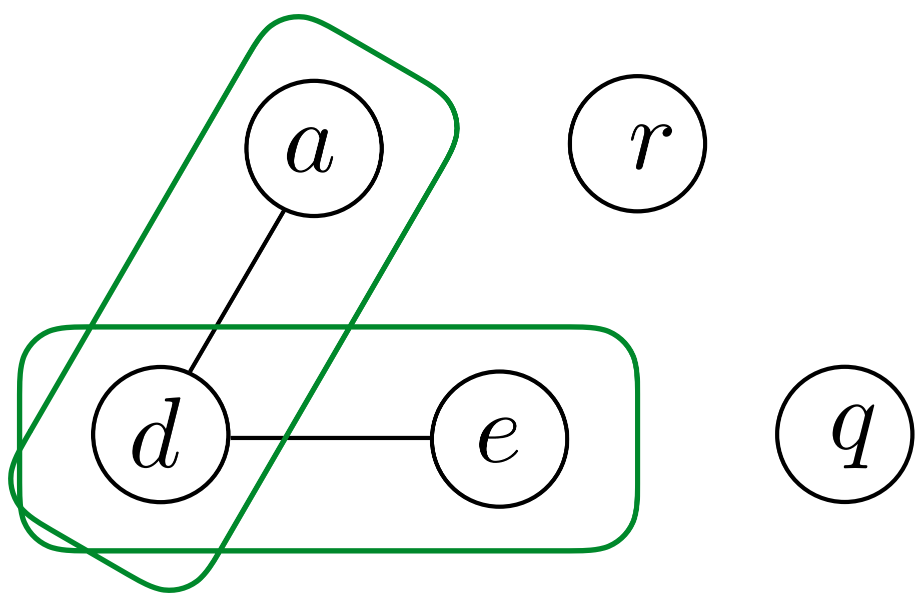


Synchronization



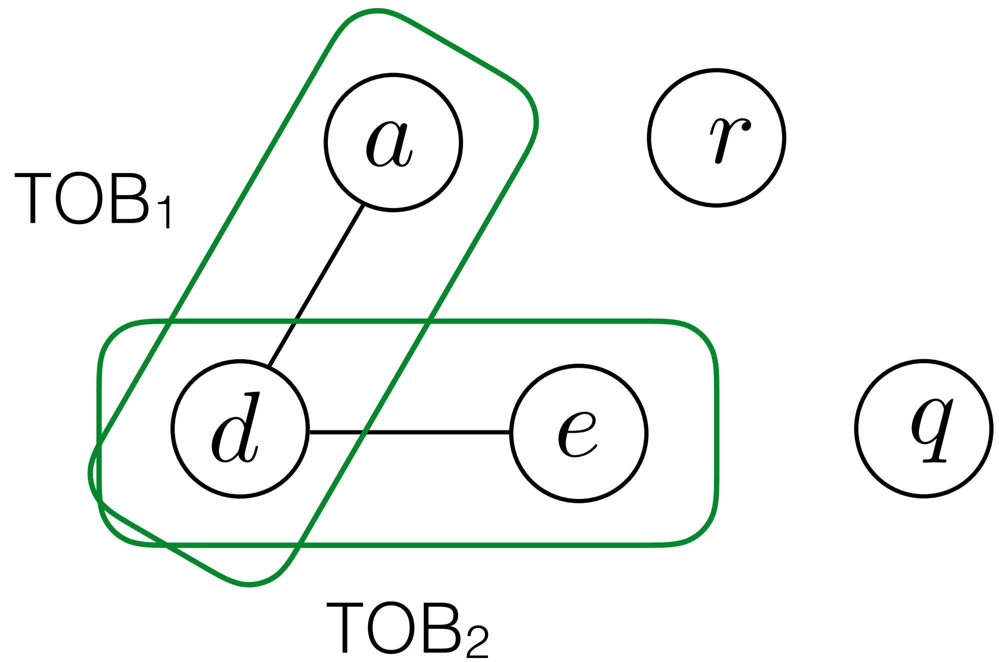
Synchronization





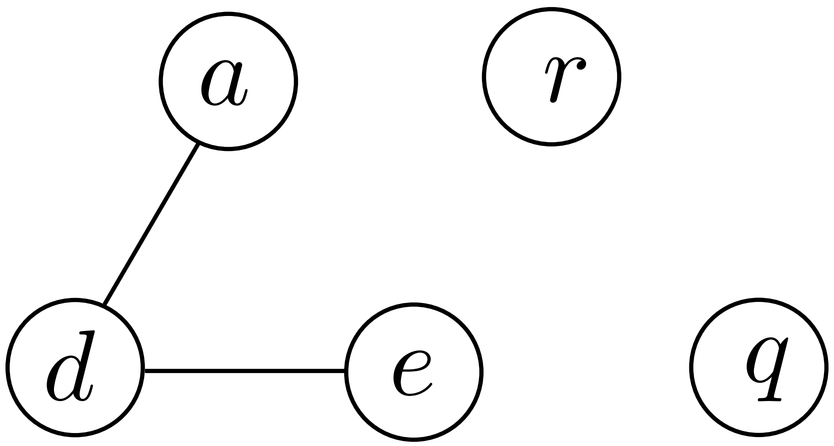
Maximal Cliques

Synchronization

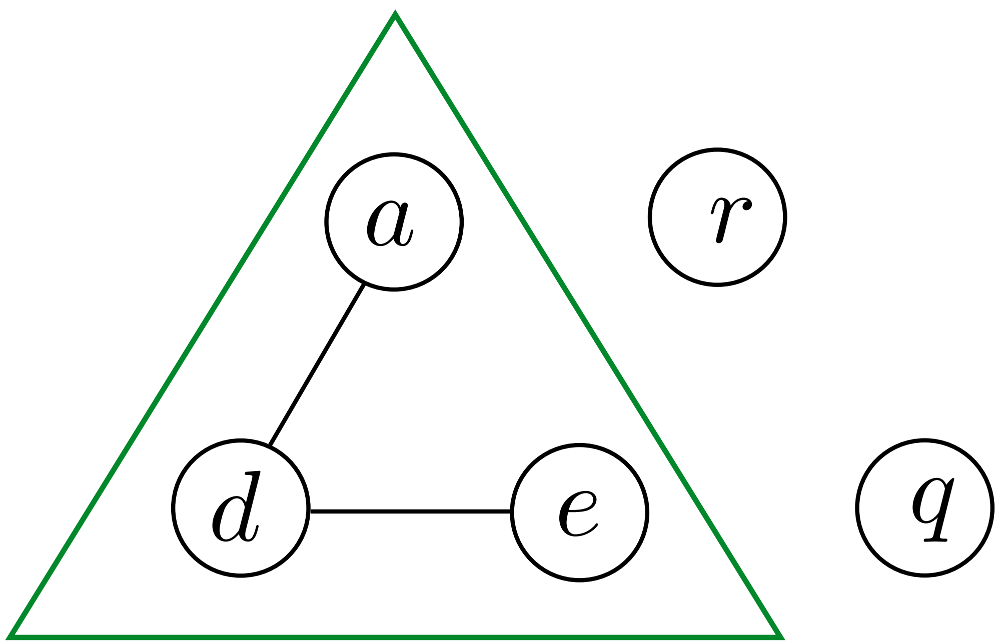


Maximal Cliques

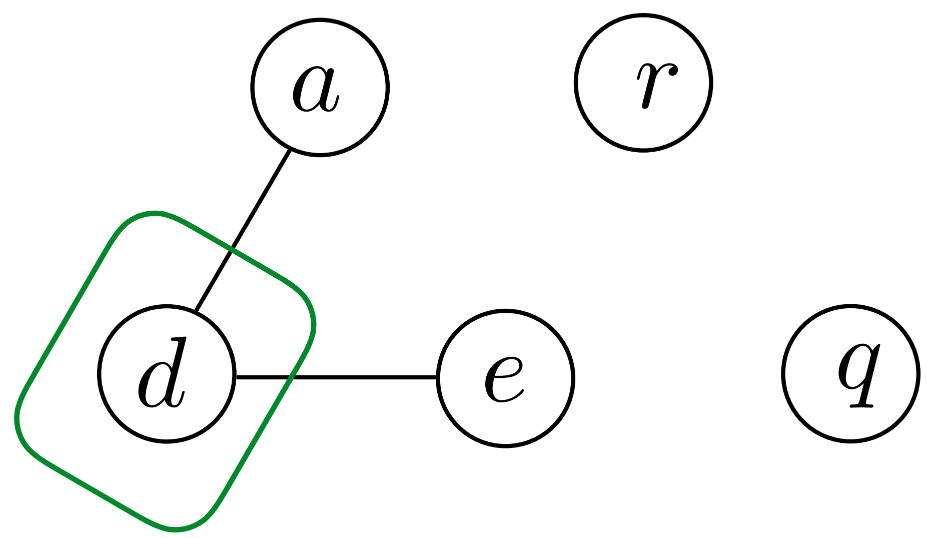
Asymmetric Synchronization



Asymmetric Synchronization

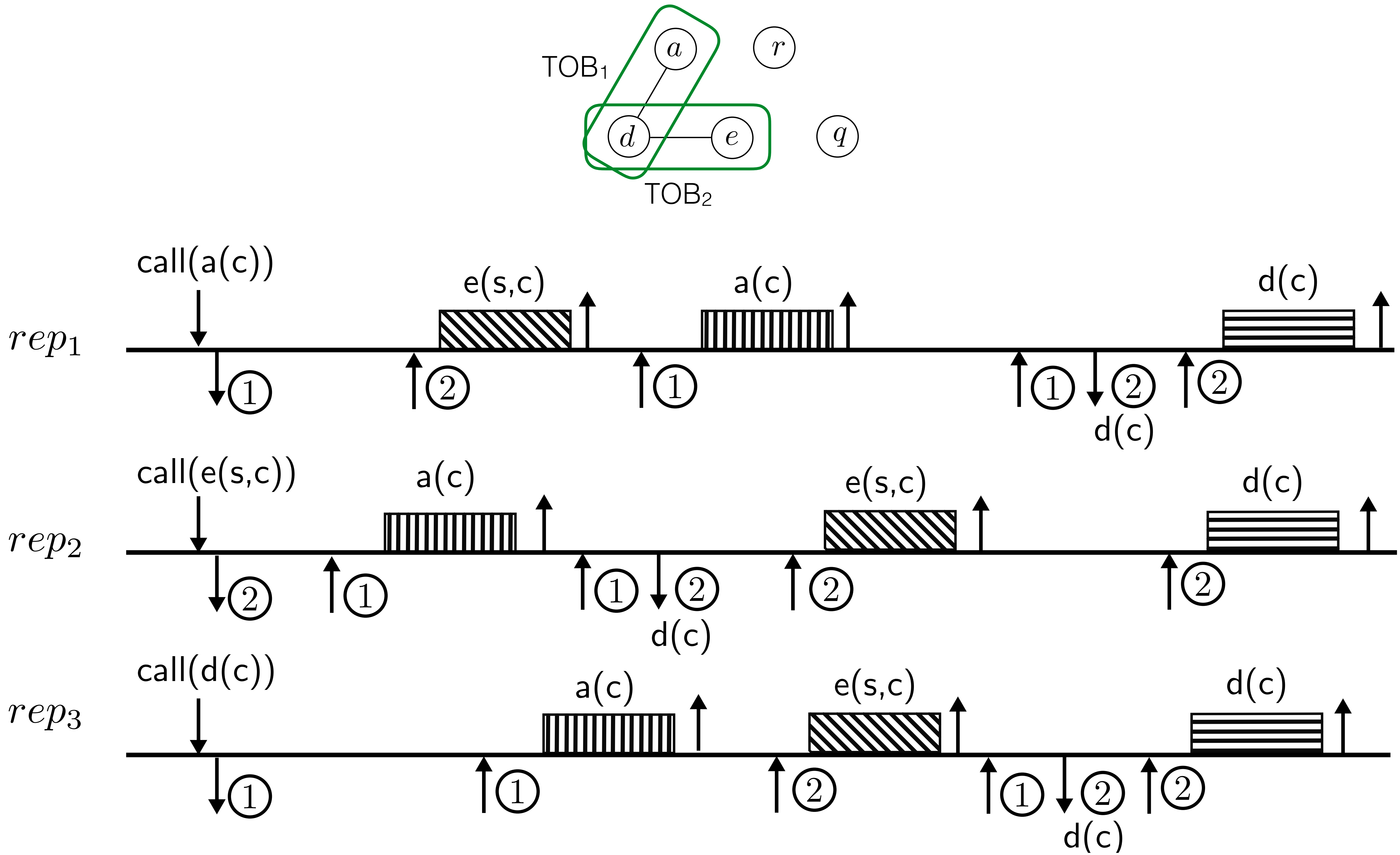


Asymmetric Synchronization

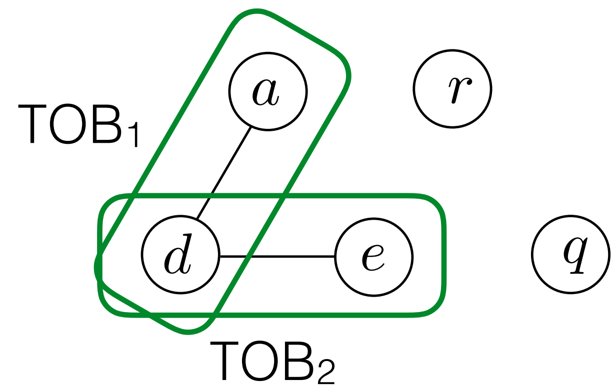


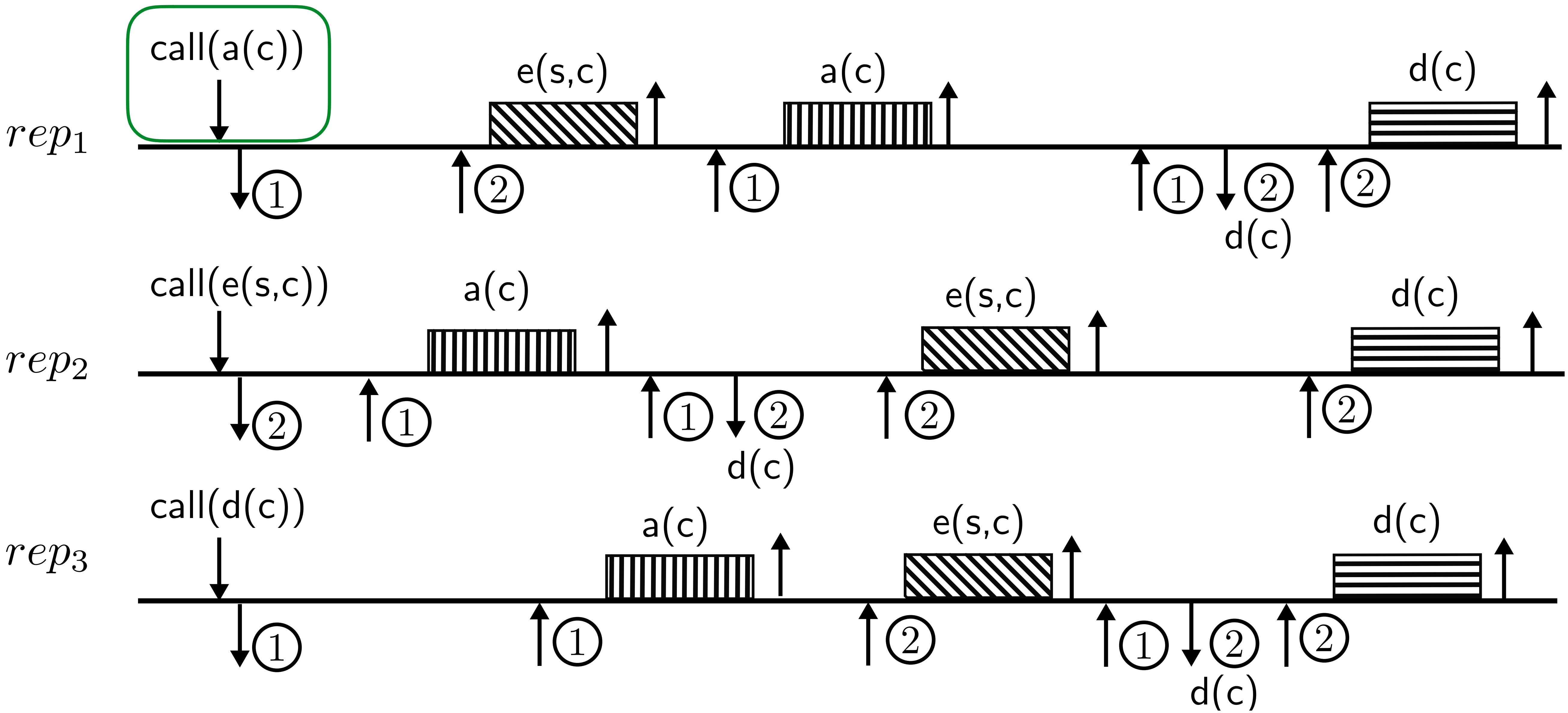
Minimum Vertex Cover

Non-blocking Protocol

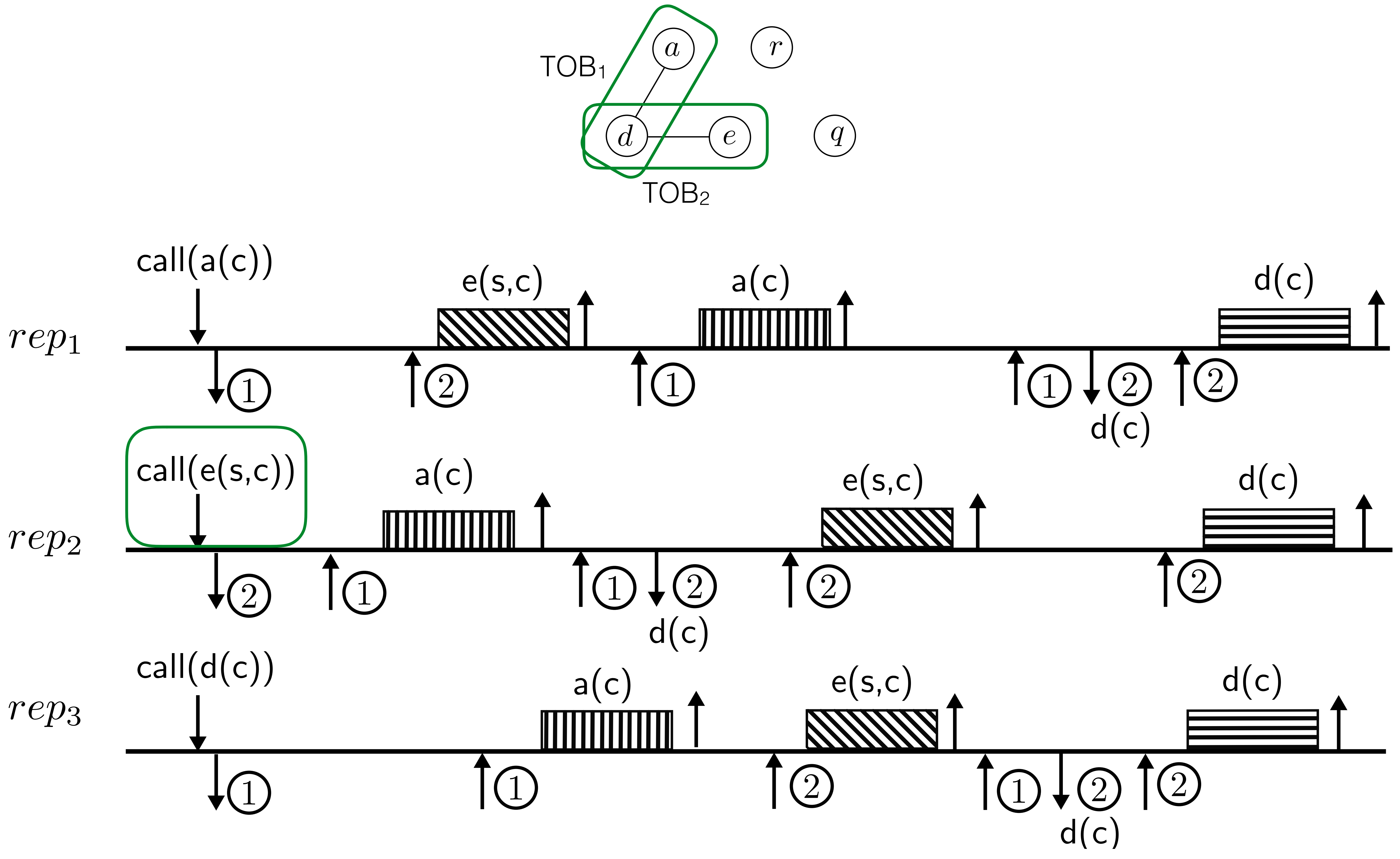


Non-blocking Protocol

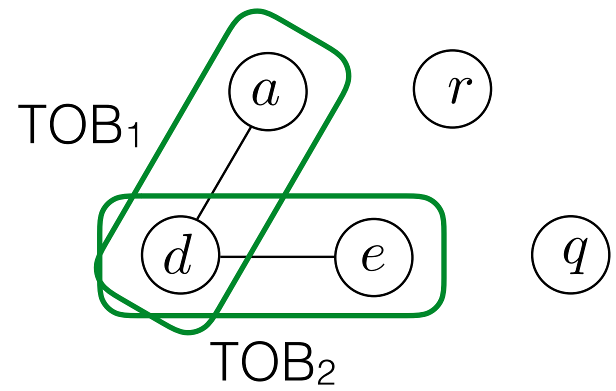


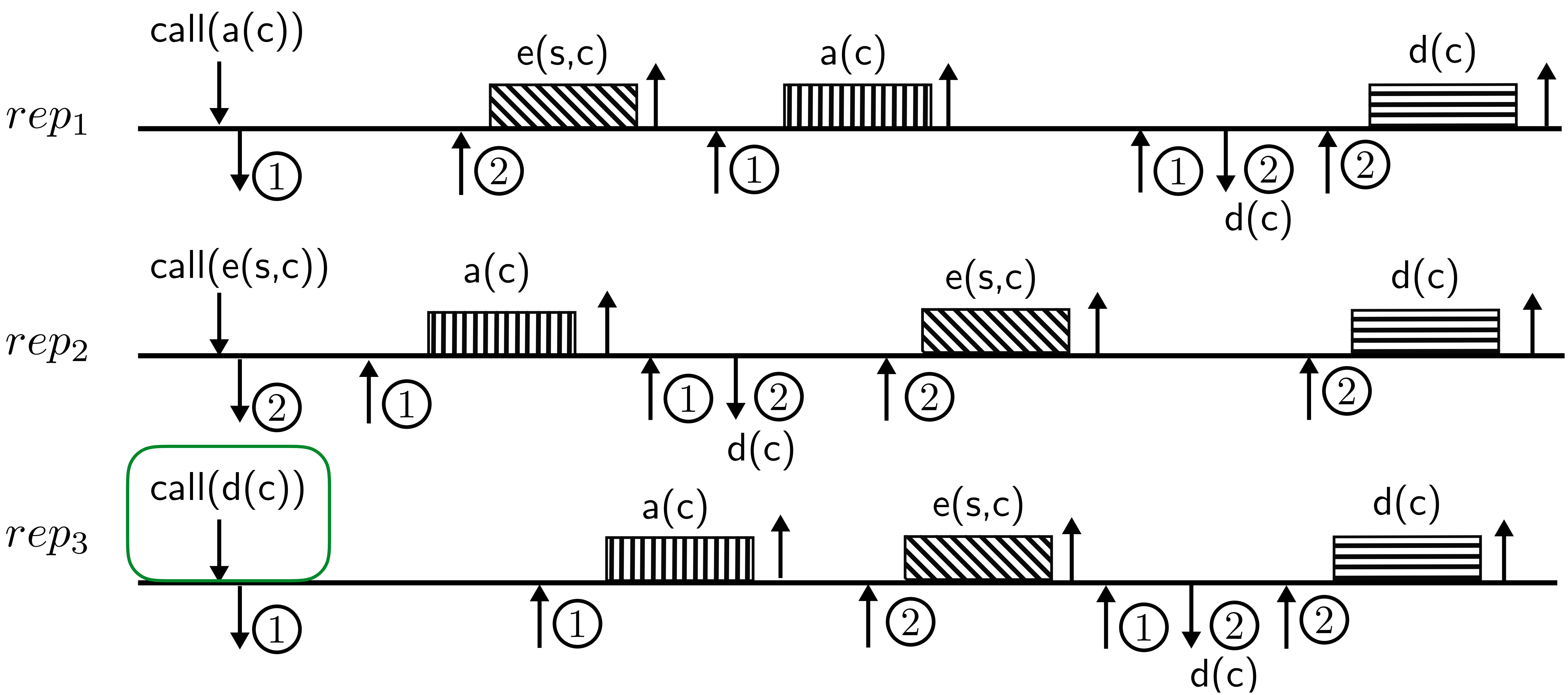


Non-blocking Protocol

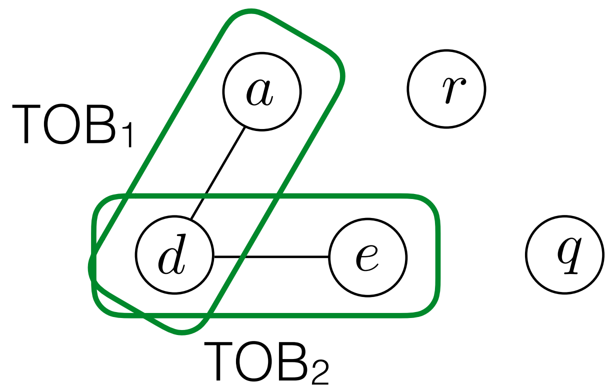


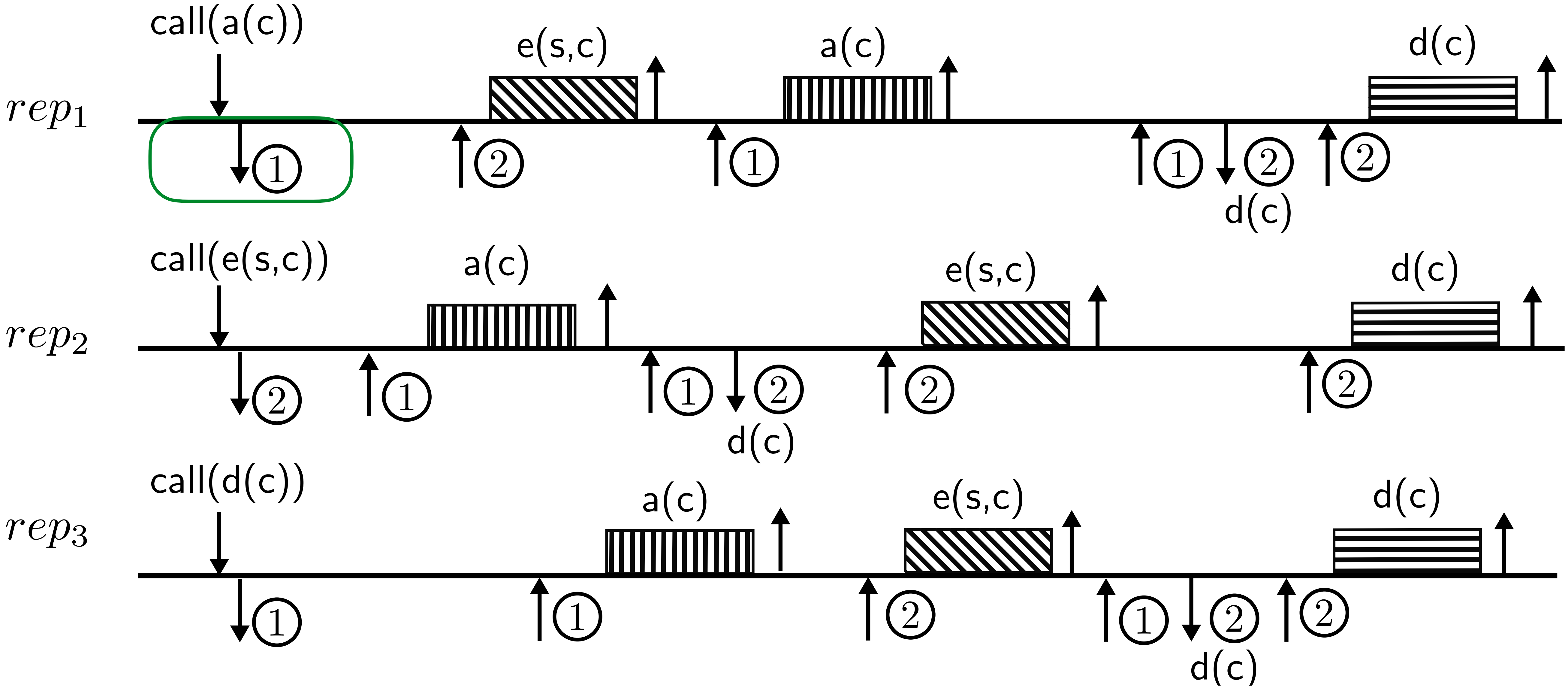
Non-blocking Protocol



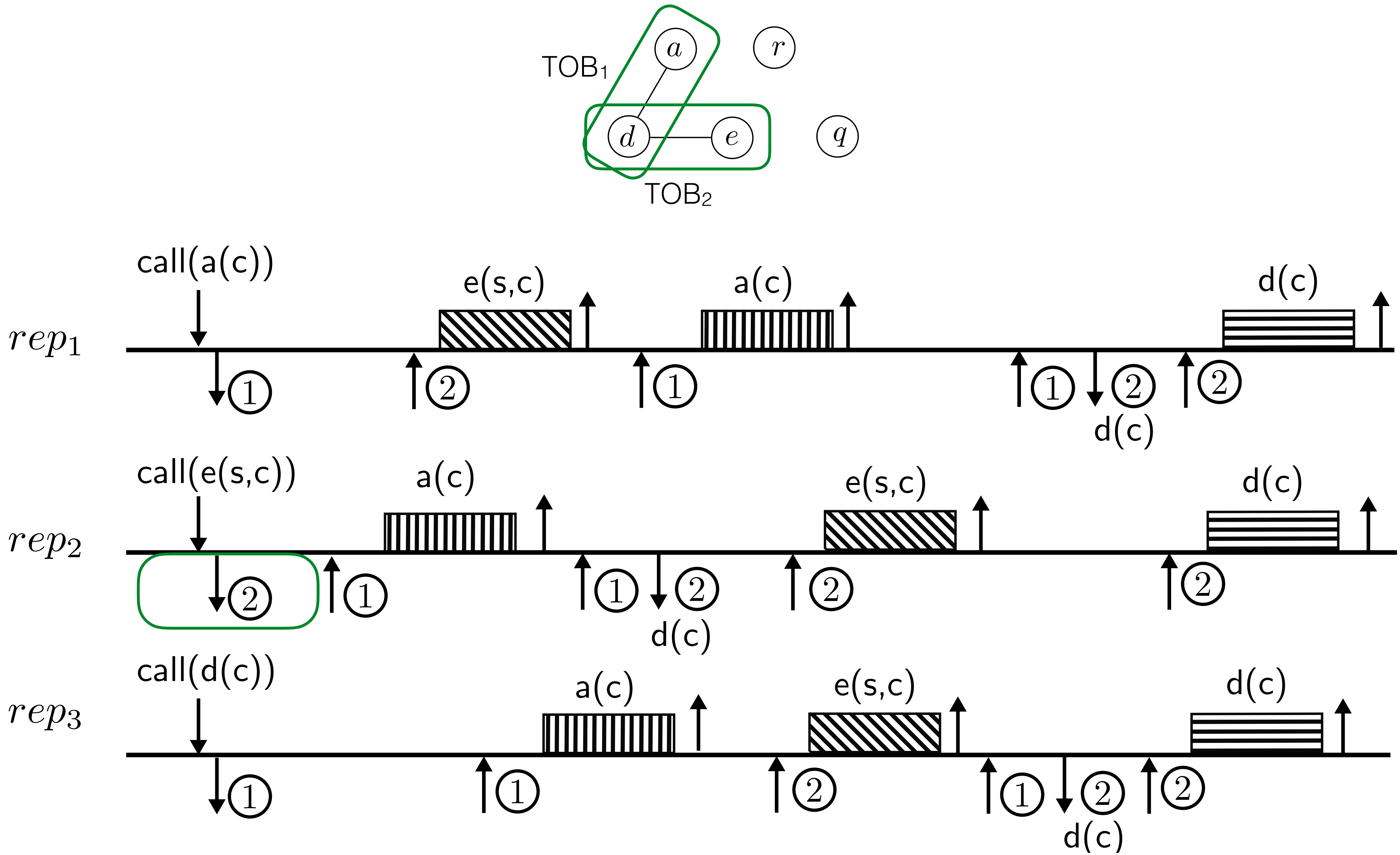


Non-blocking Protocol

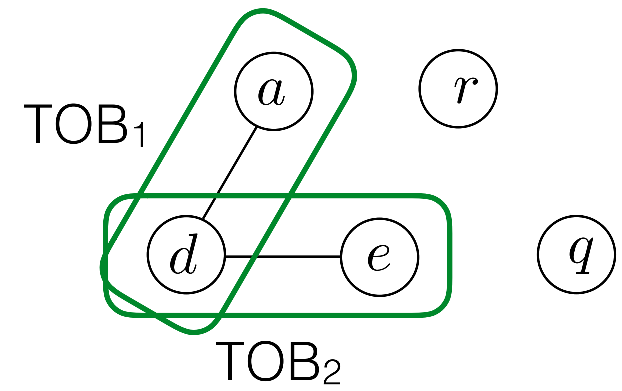


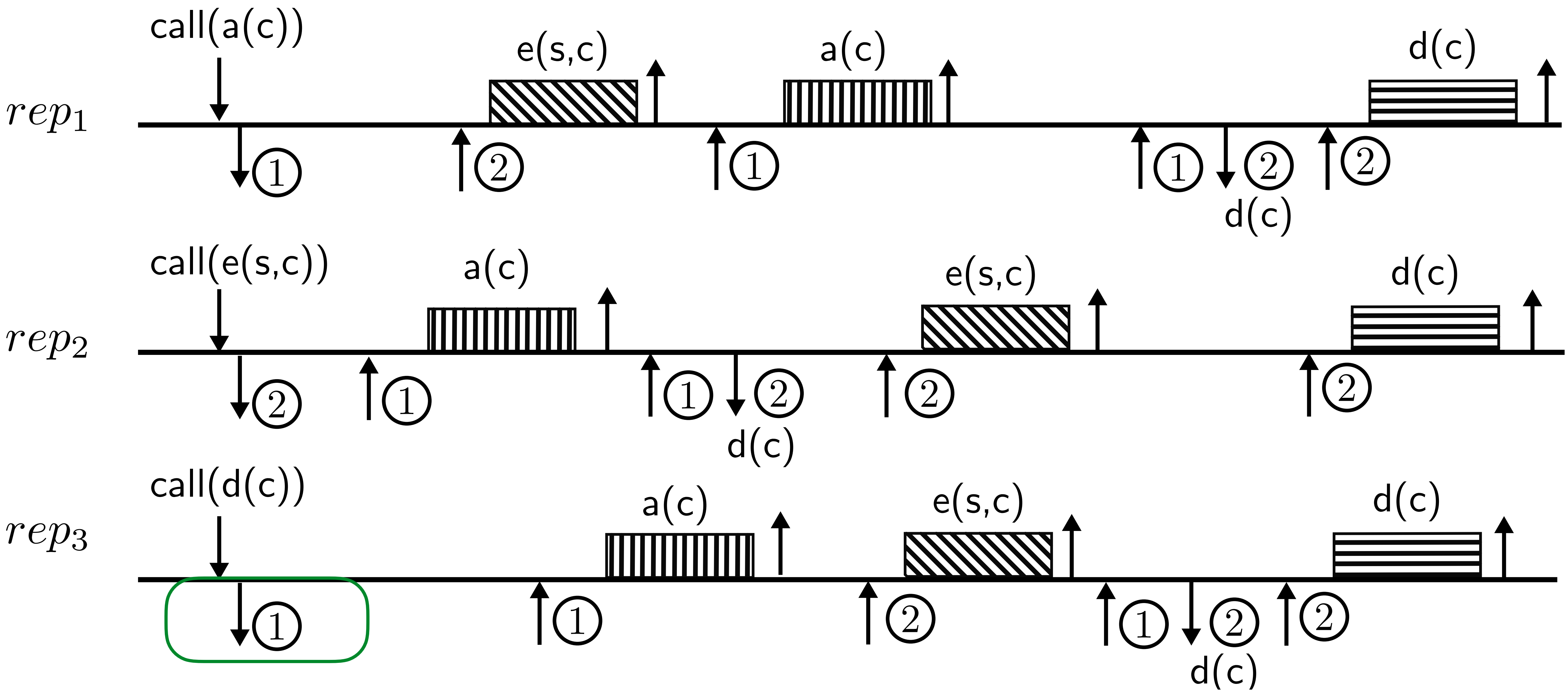


Non-blocking Protocol

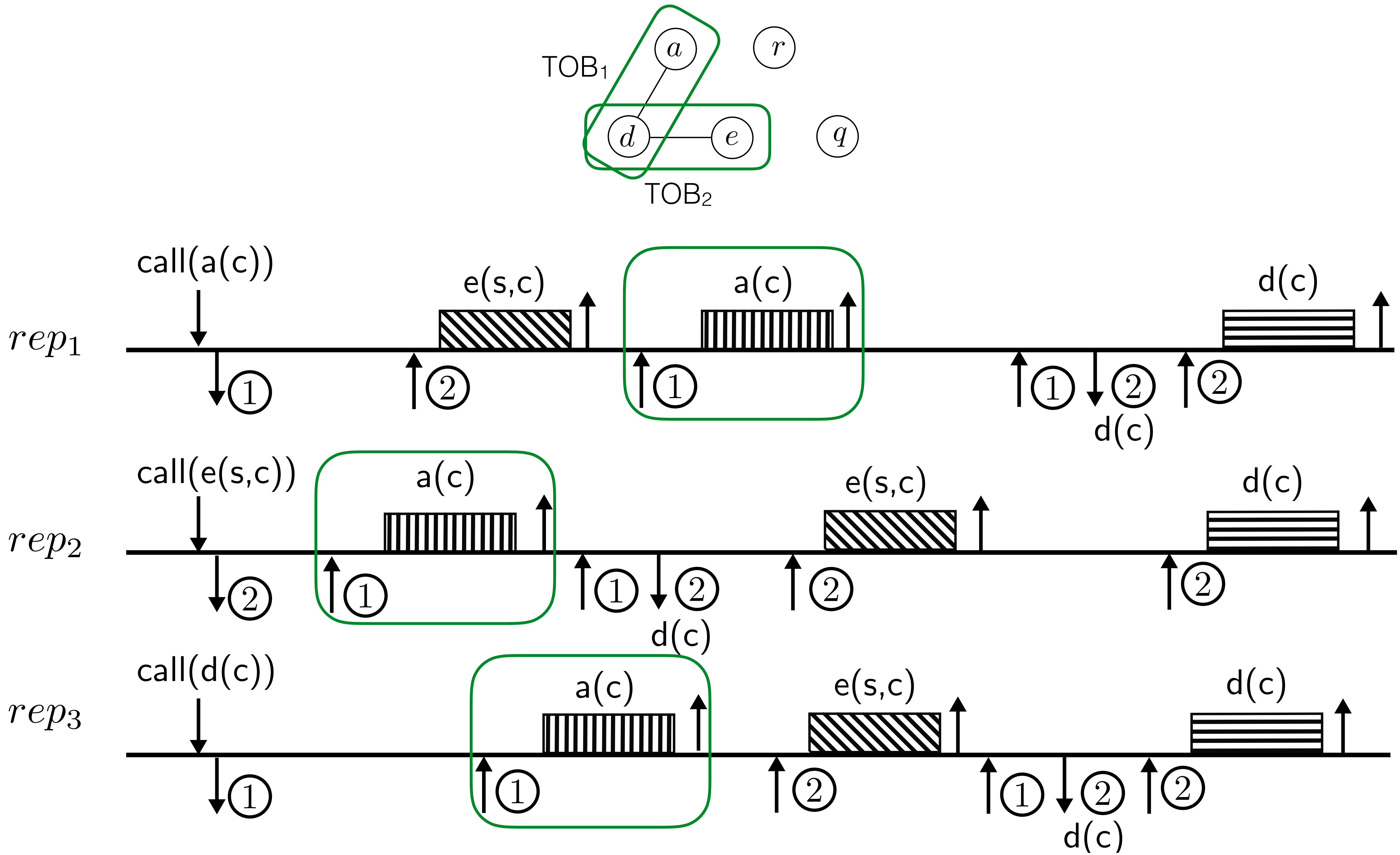


Non-blocking Protocol

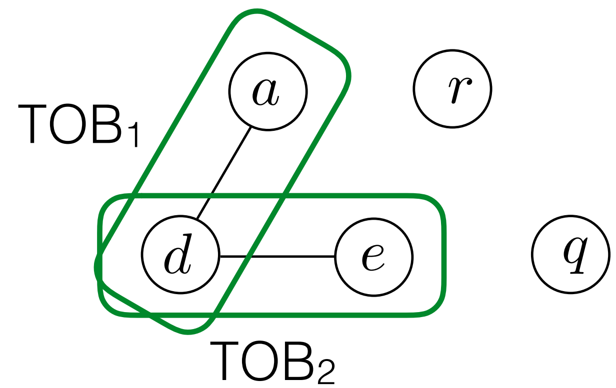


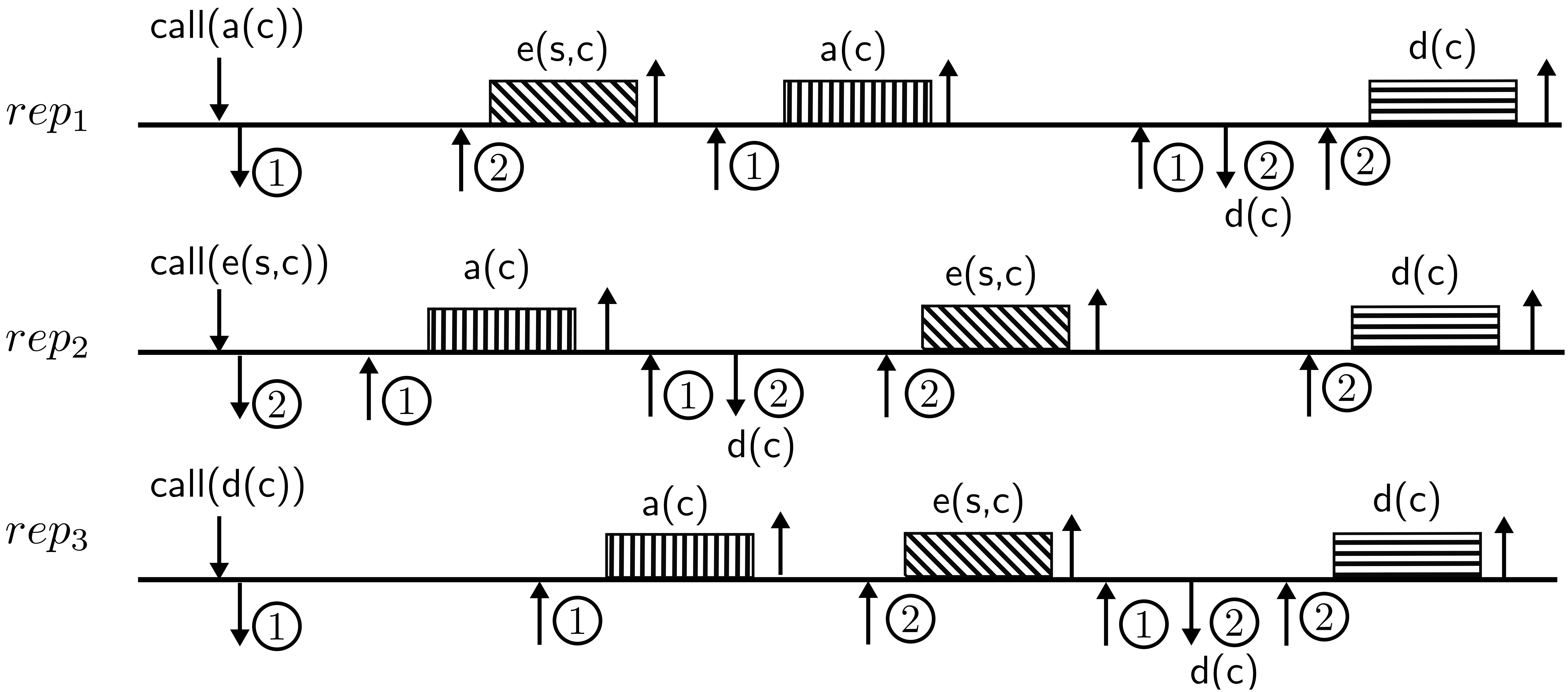


Non-blocking Protocol

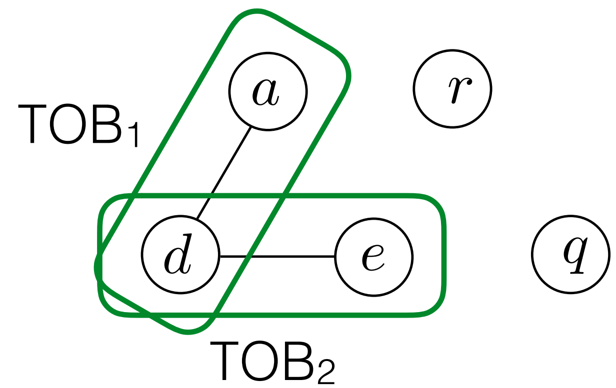


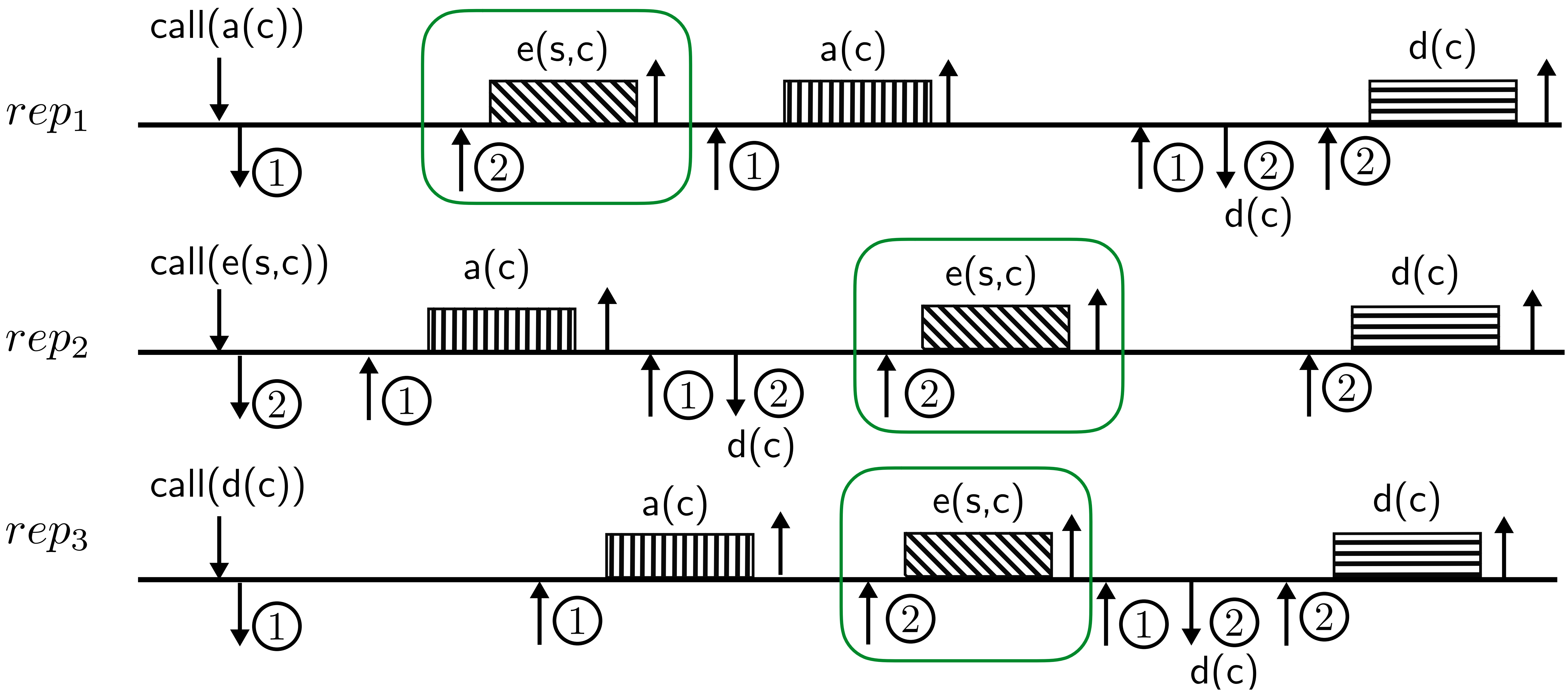
Non-blocking Protocol



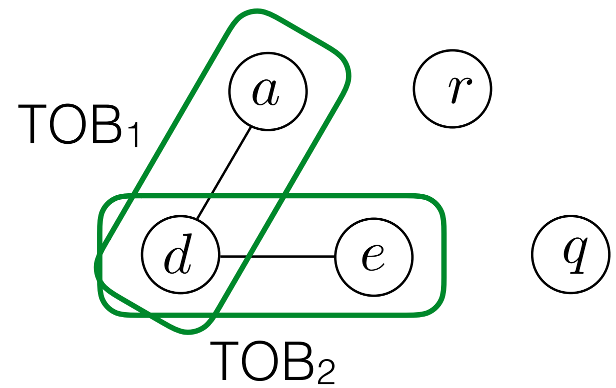


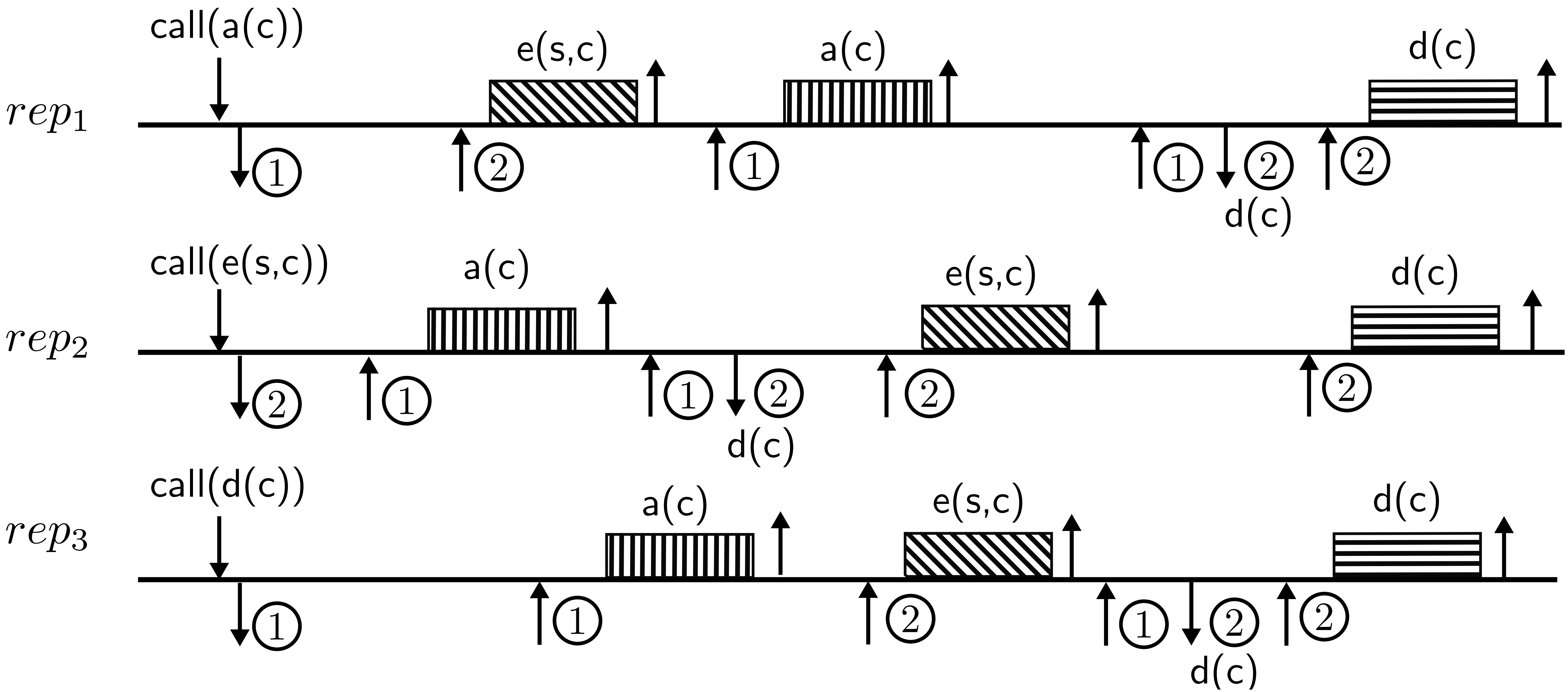
Non-blocking Protocol



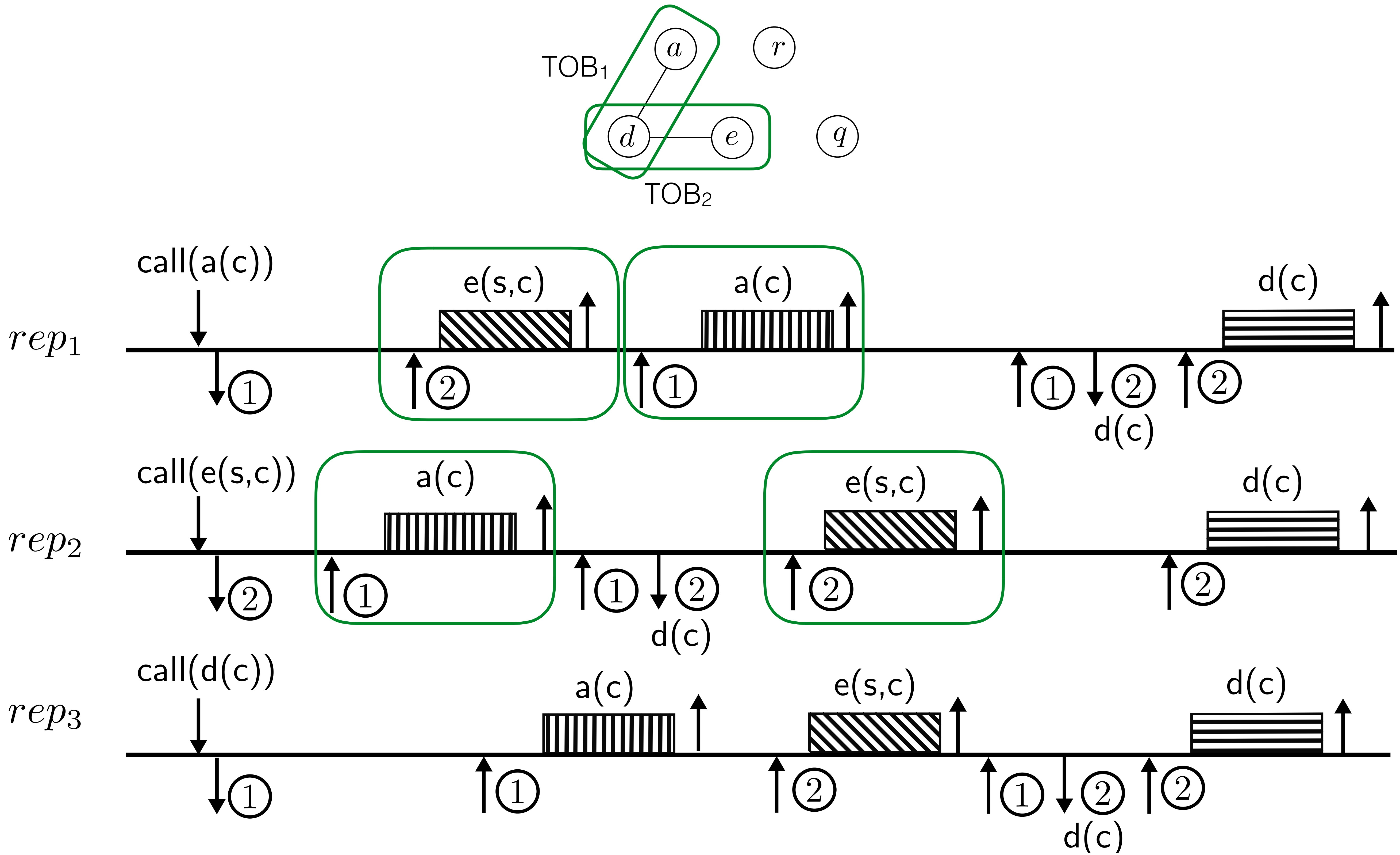


Non-blocking Protocol

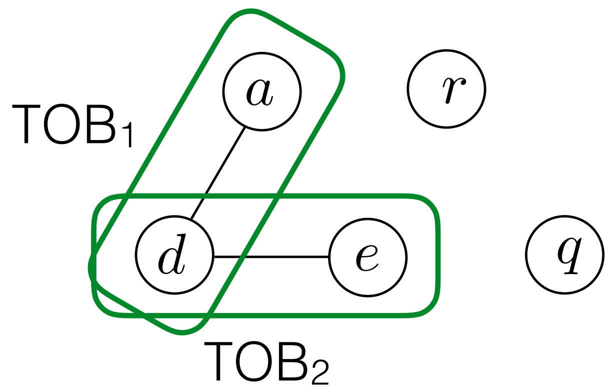


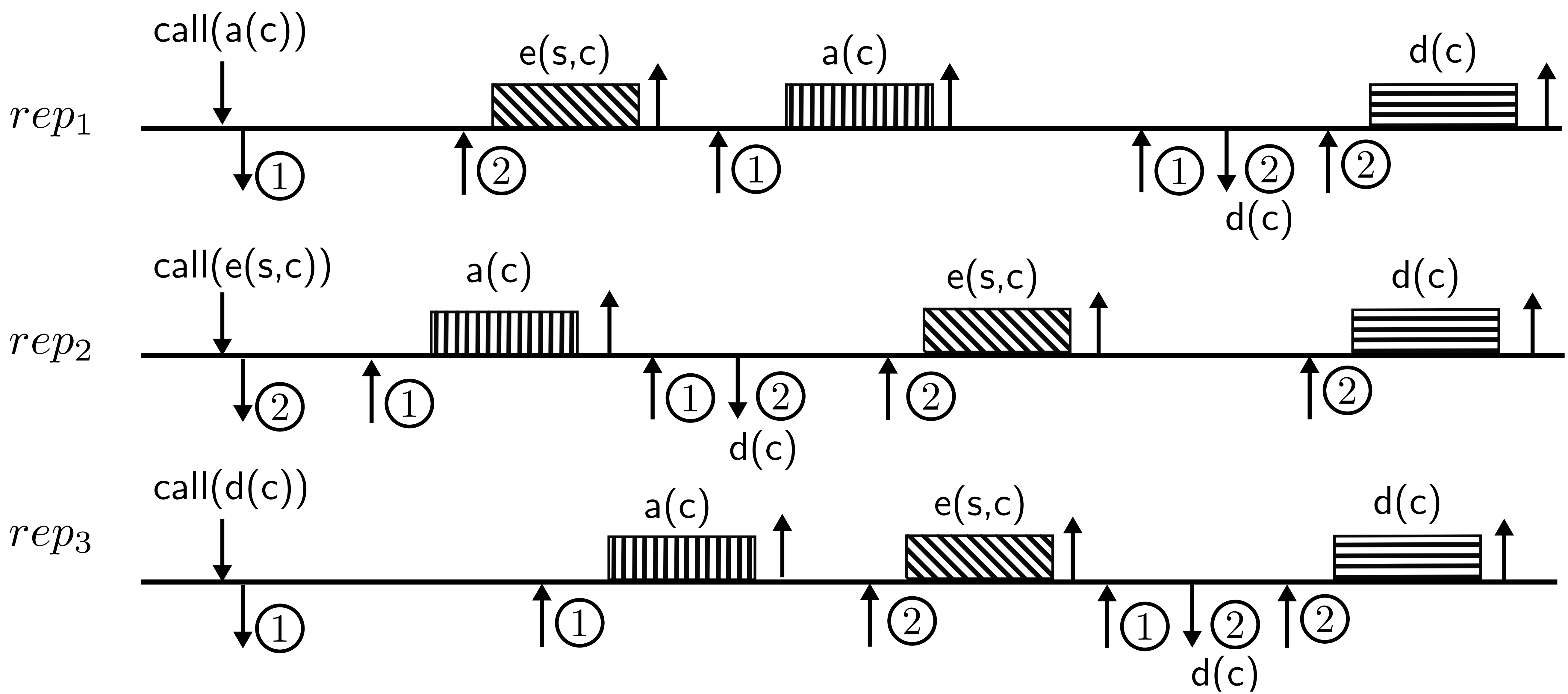


Non-blocking Protocol

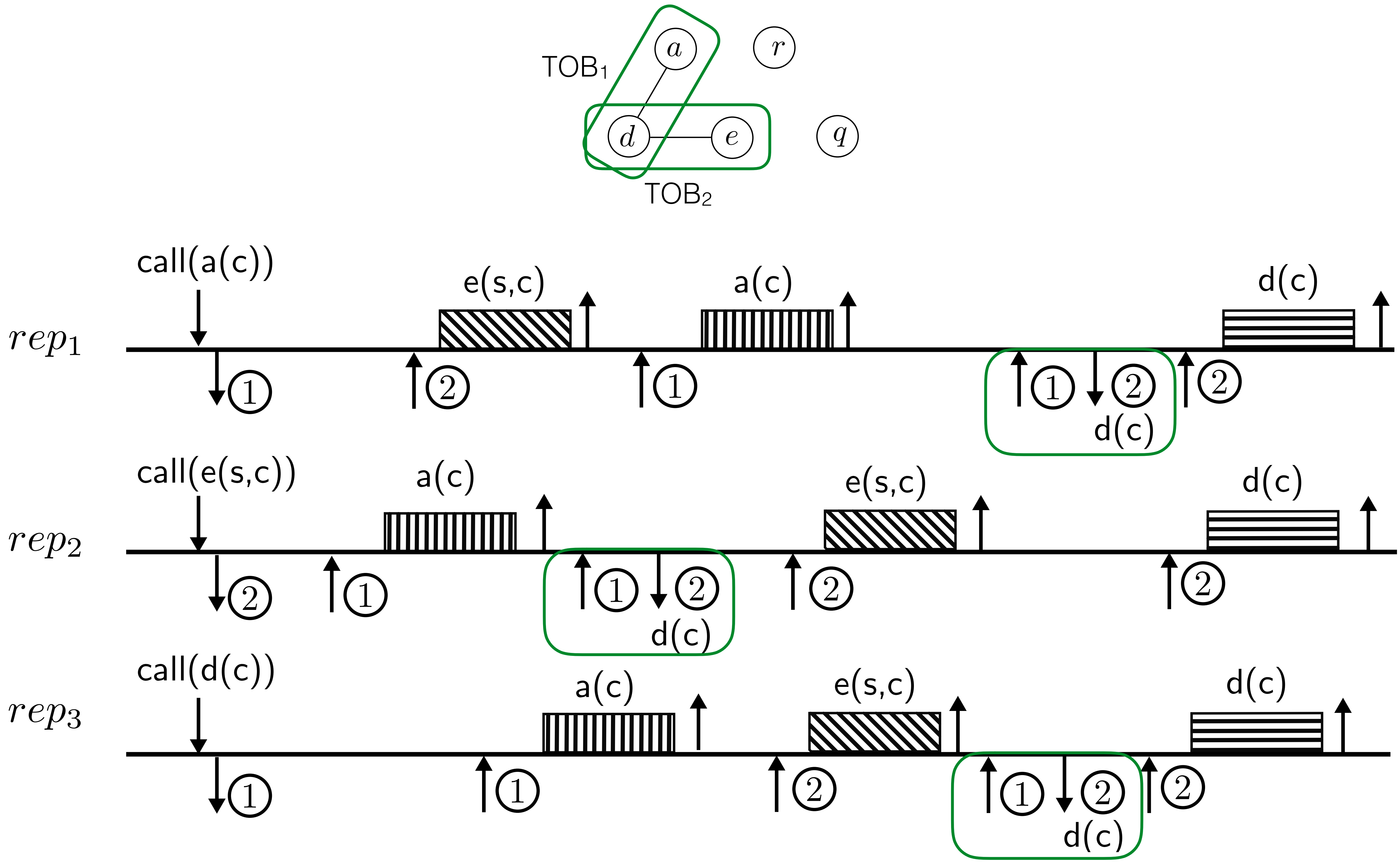


Non-blocking Protocol

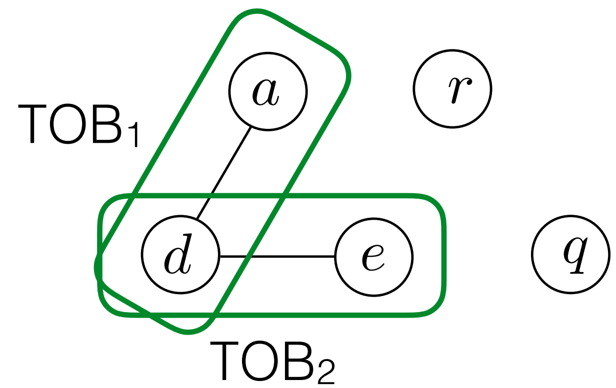


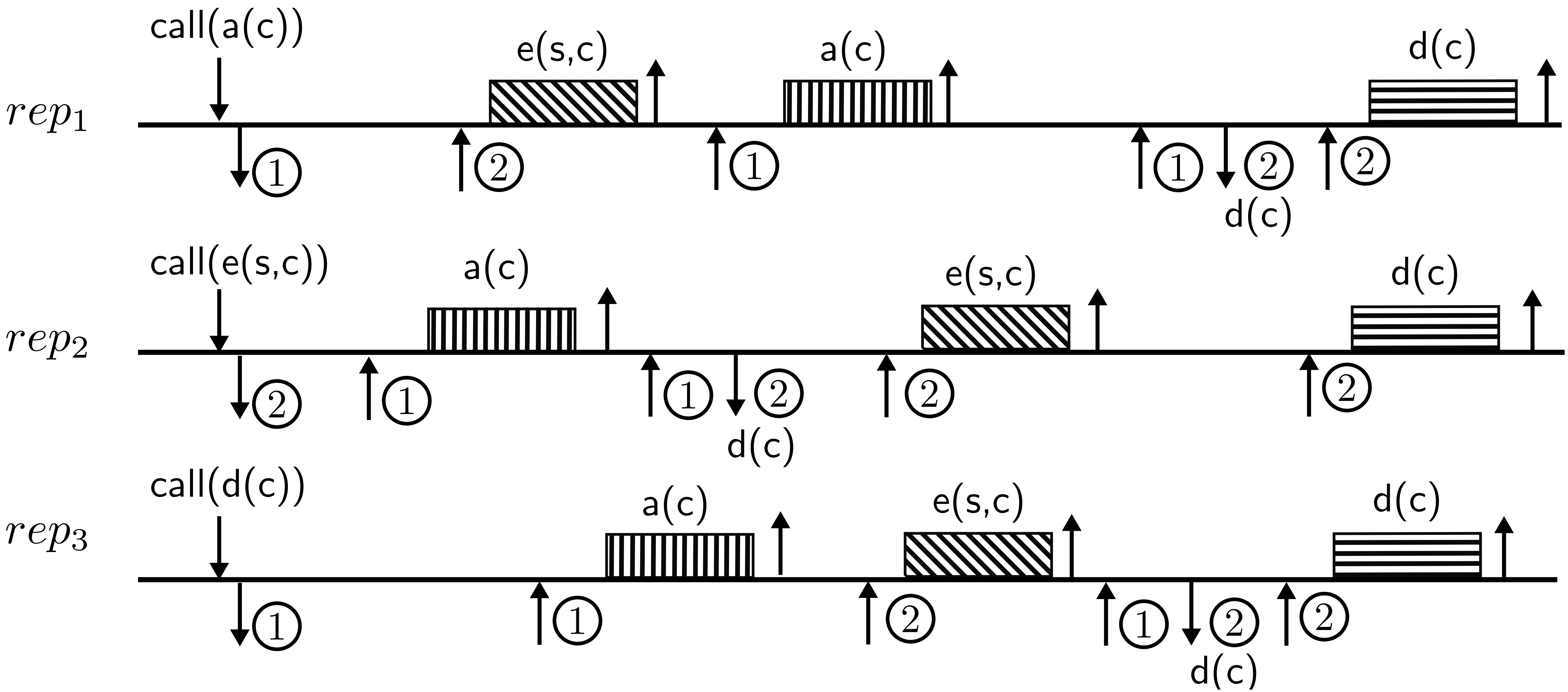


Non-blocking Protocol

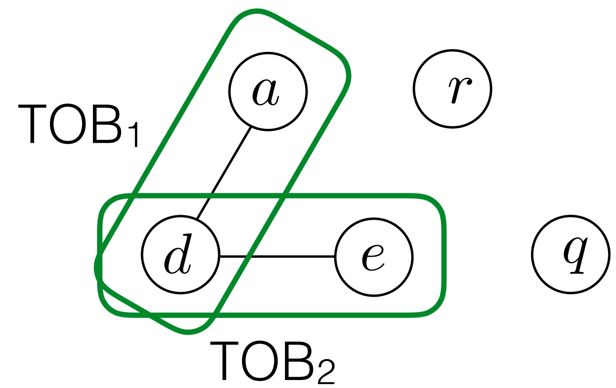


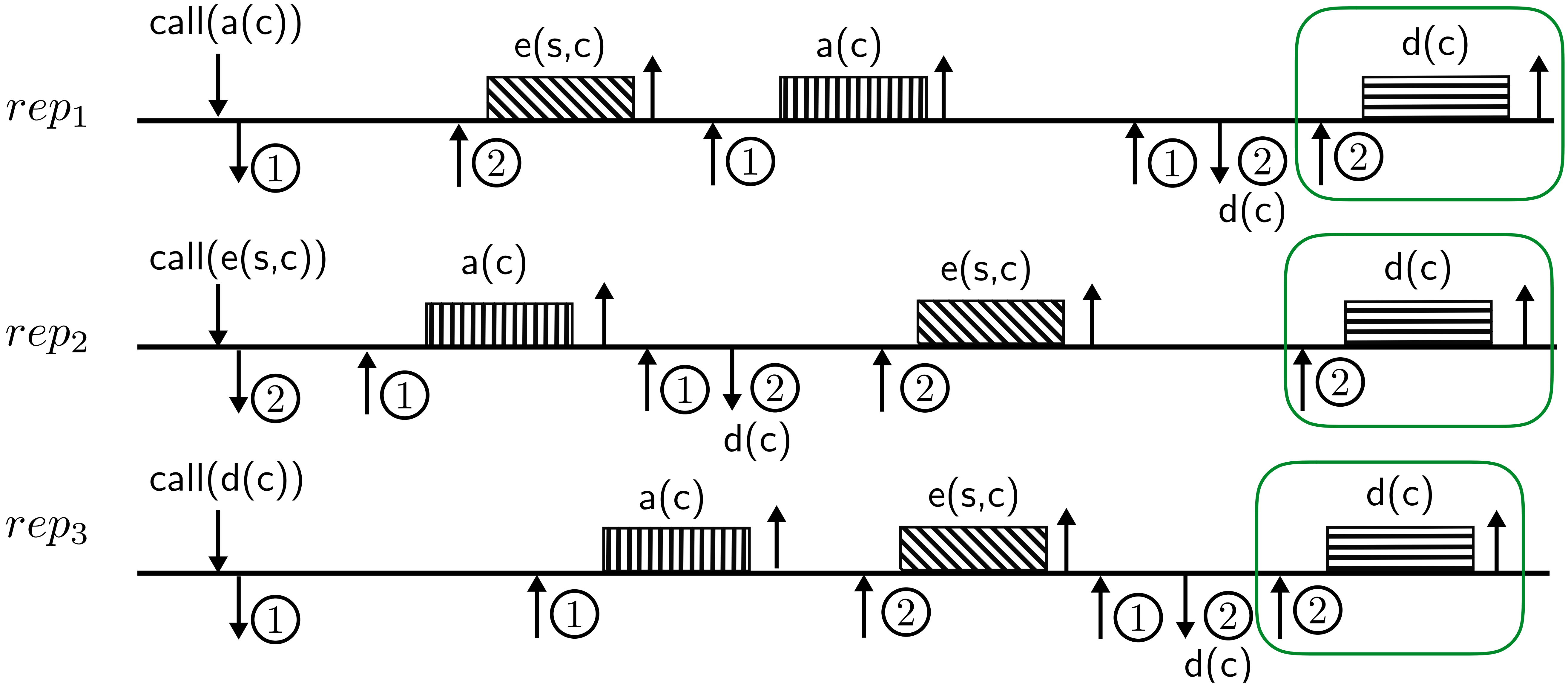
Non-blocking Protocol



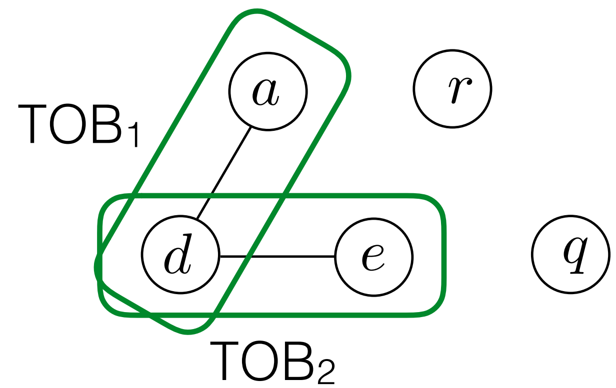


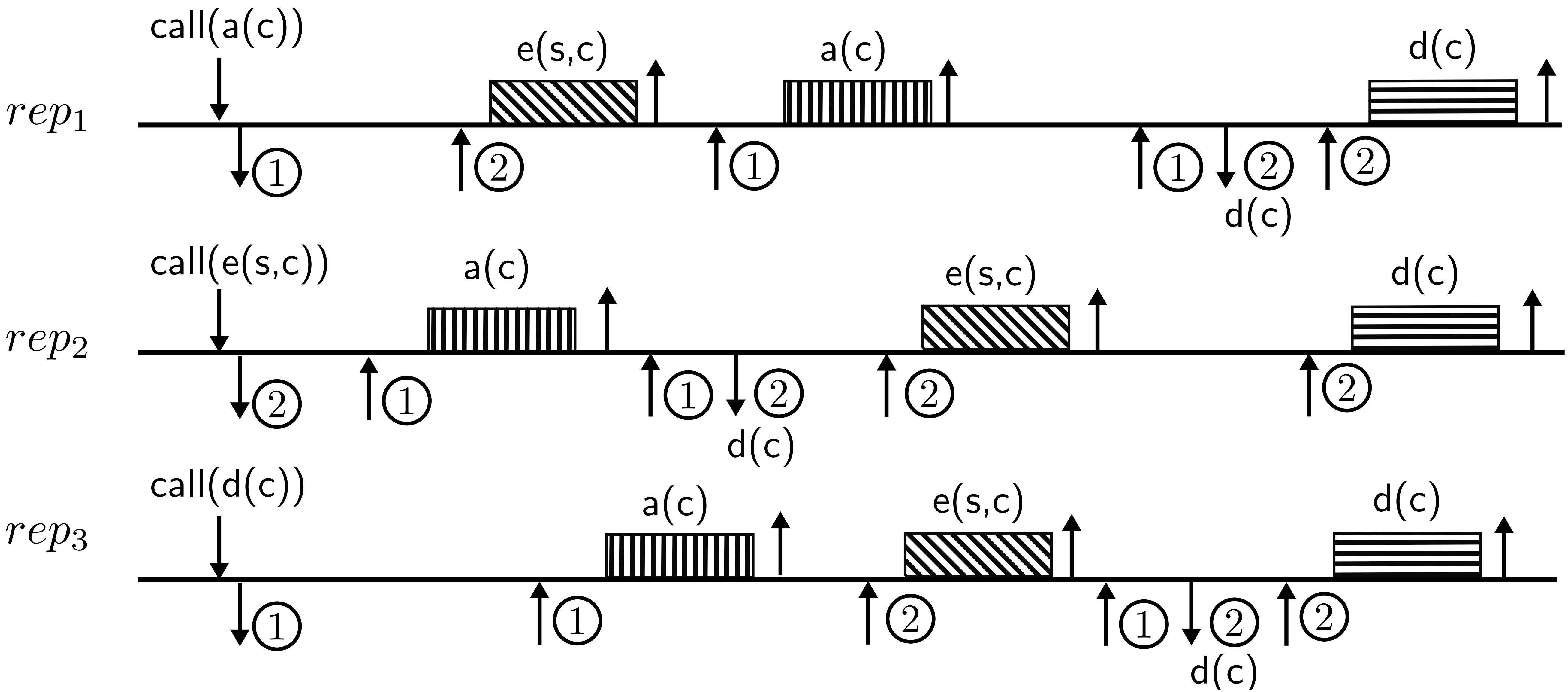
Non-blocking Protocol



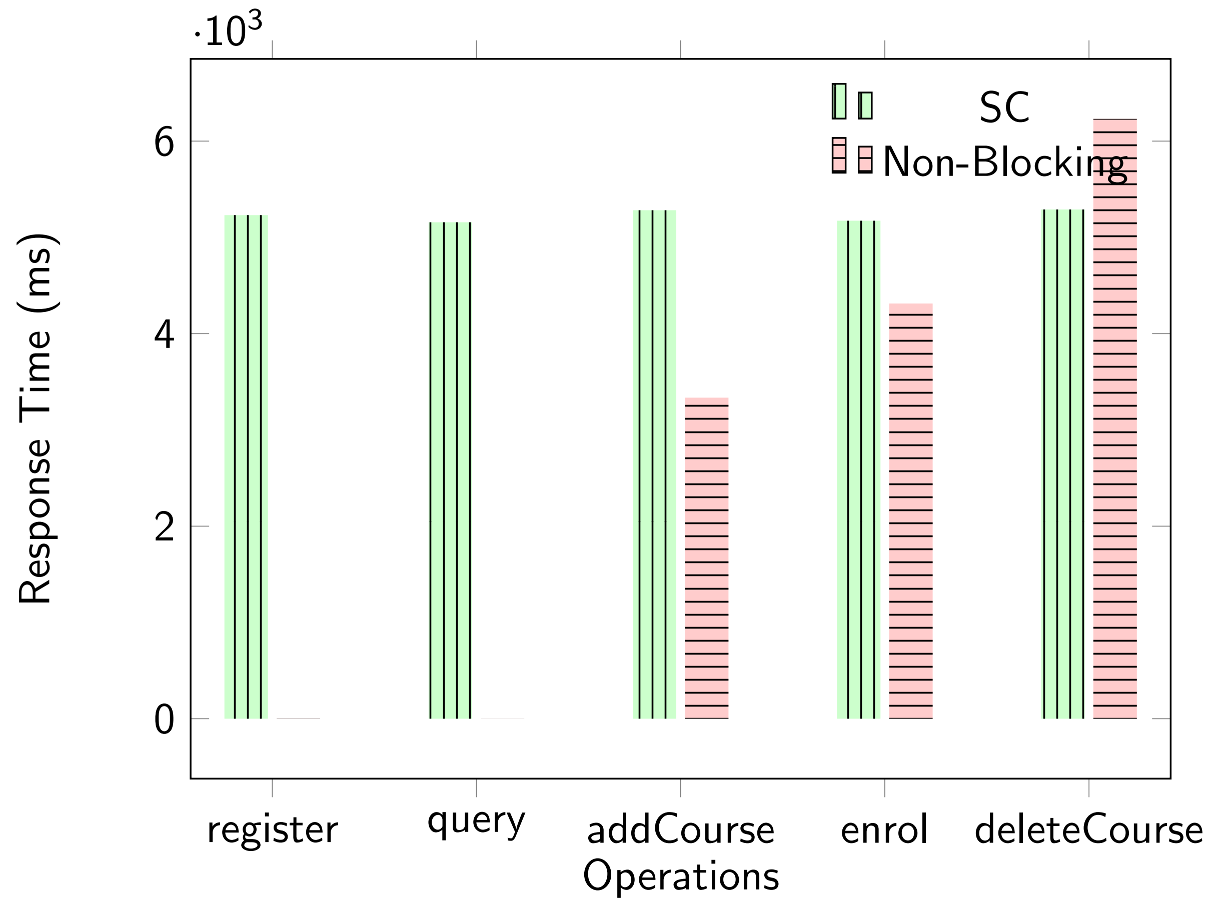


Non-blocking Protocol





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.

Hamsaz

- Synthesis of replicated objects
  that preserve integrity and convergence and minimize coordination
- Reduction of coordination minimization to classical graph optimization
- Well-coordination, a sufficient condition for correctness
- Protocols that implement well-coordination.

Replication Coordination Analysis and Synthesis

Farzin Houshmand, Mohsen Lesani
University of California, Riverside