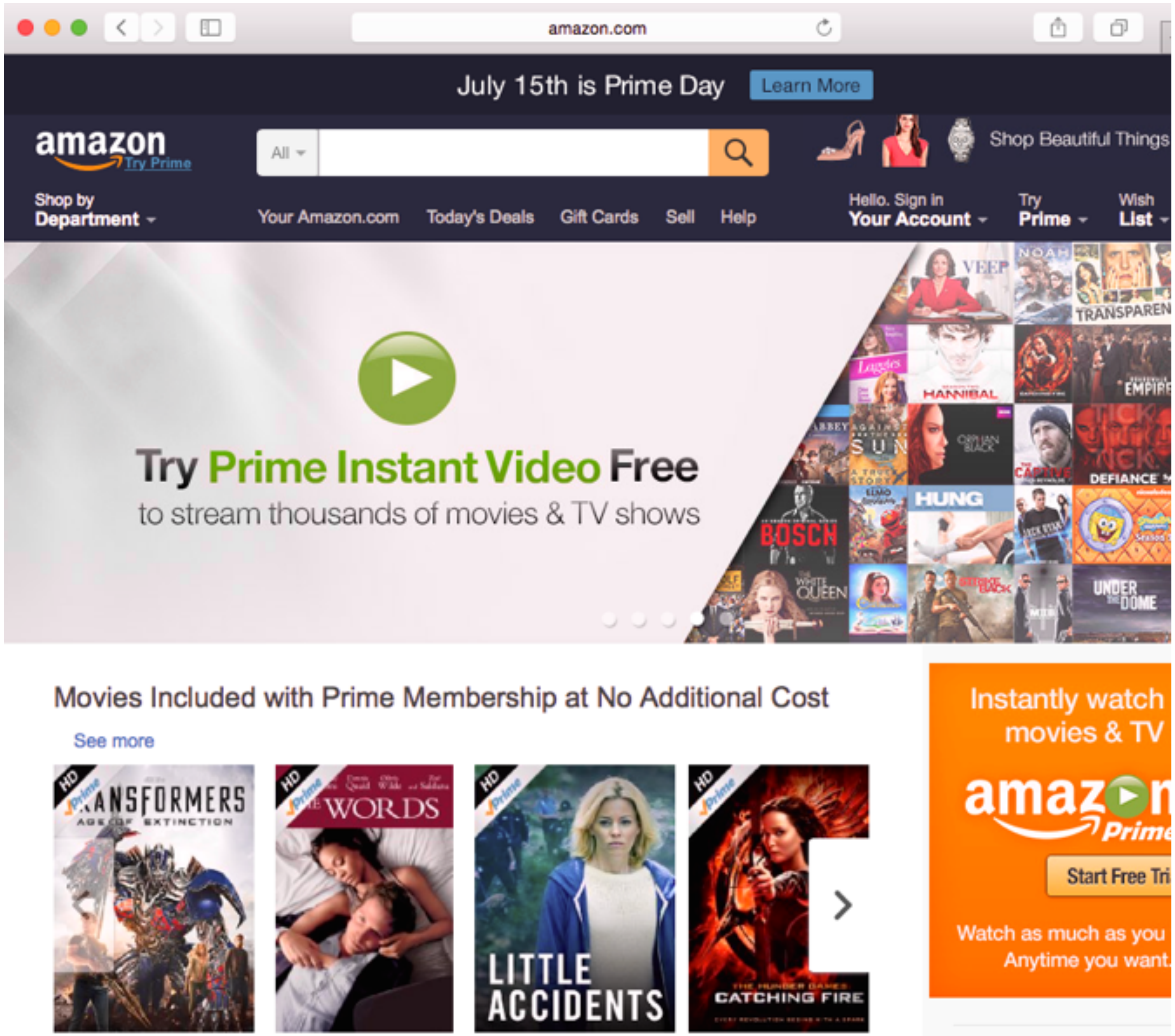
Reasoning about consistency choices in distributed systems

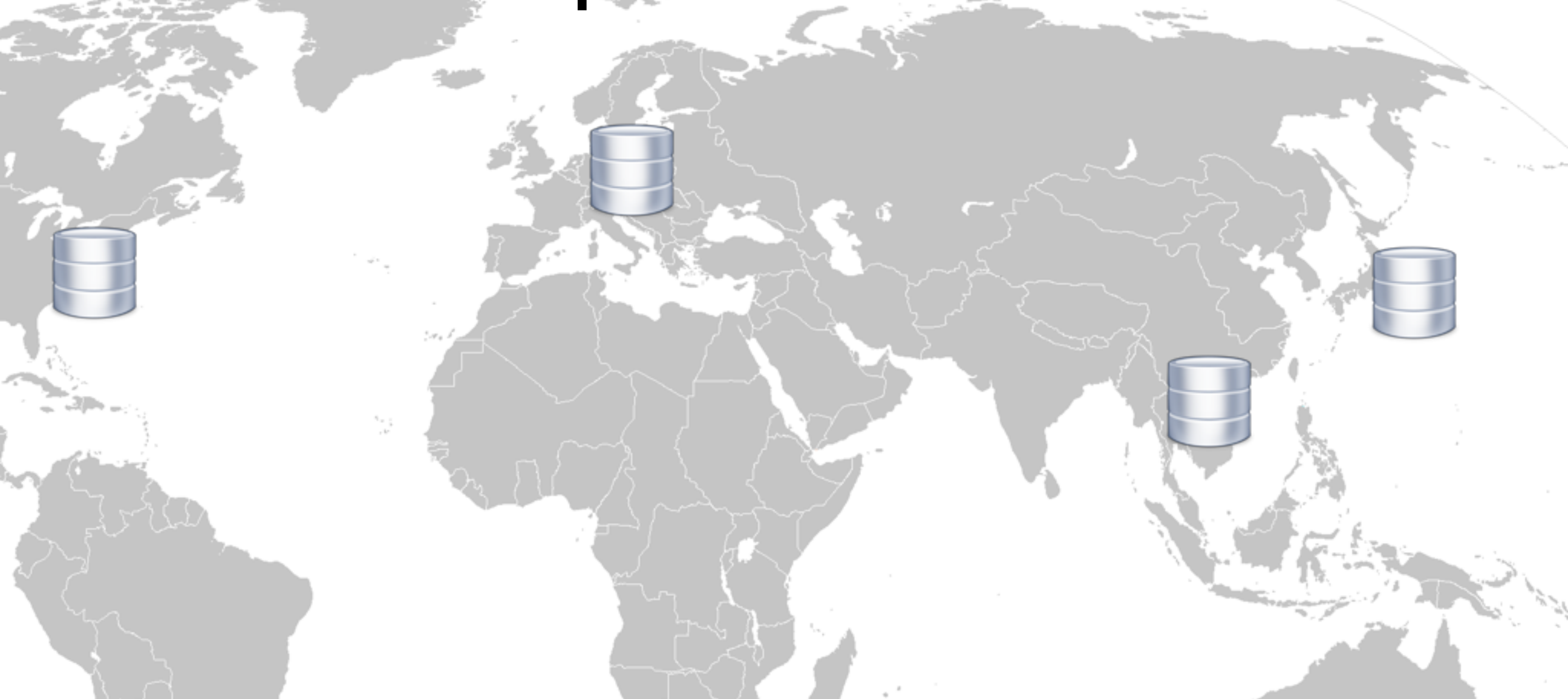
Hongseok Yang
University of Oxford

Joint work with Alexey Gotsman (IMDEA, Spain), Carla Ferreira (U Nova Lisboa), Mahsa Najafzadeh, Marc Shapiro (INRIA)

Global-scale Internet service

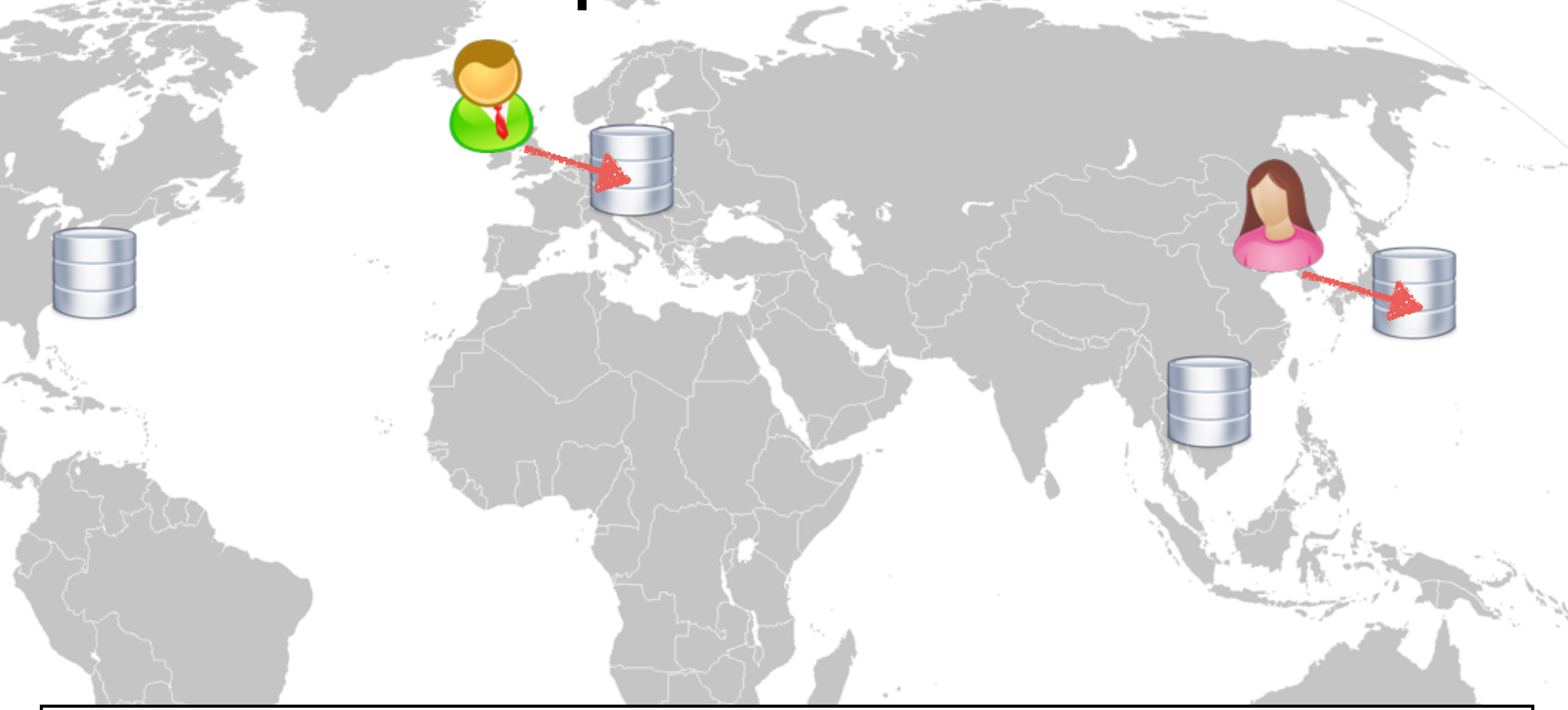


Geo-replicated databases



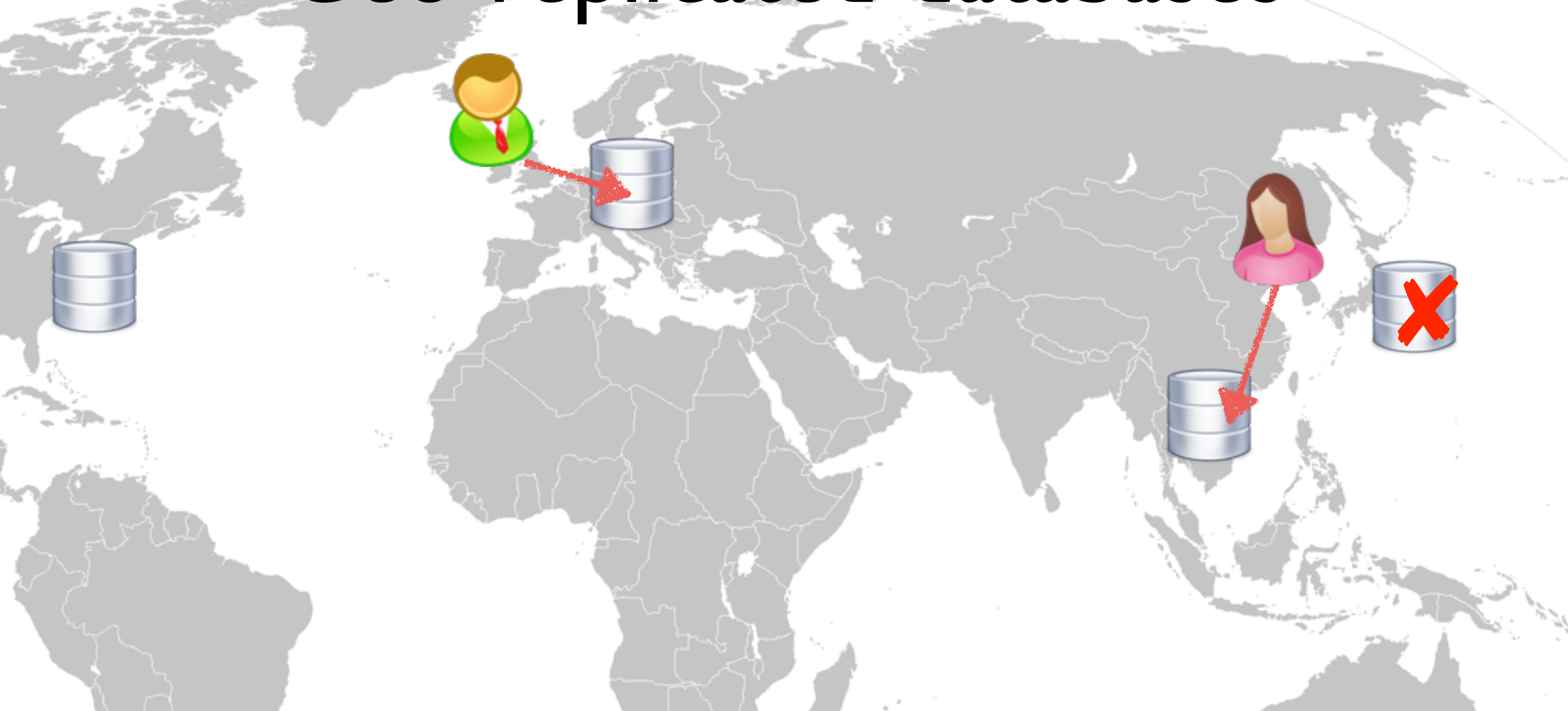
- Every data centre stores a complete replica of data
- Purpose: Minimising latency. Fault tolerance.

Geo-replicated databases



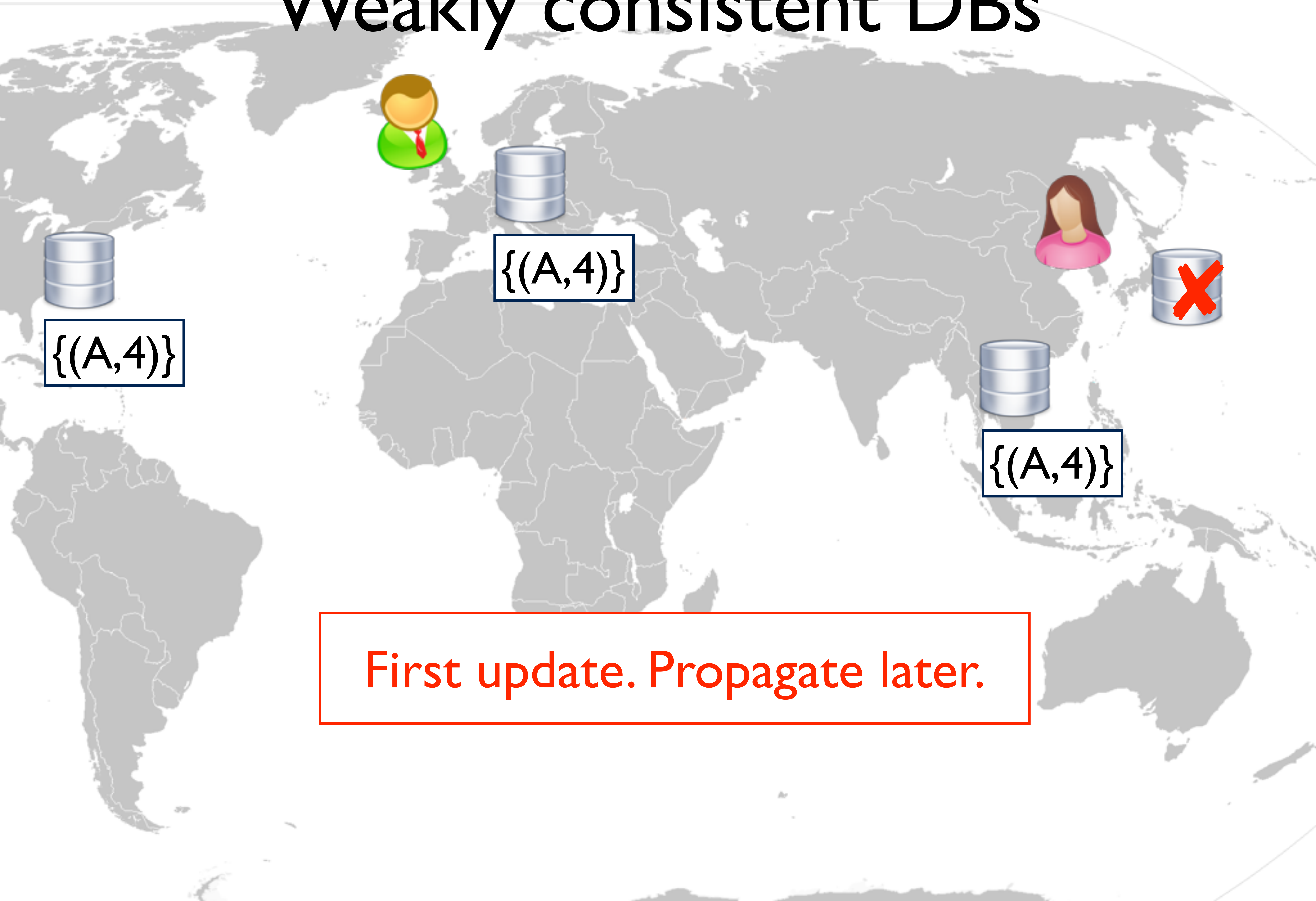
- Every data centre stores a complete replica of data
- Purpose: Minimising latency. Fault tolerance.

Geo-replicated databases

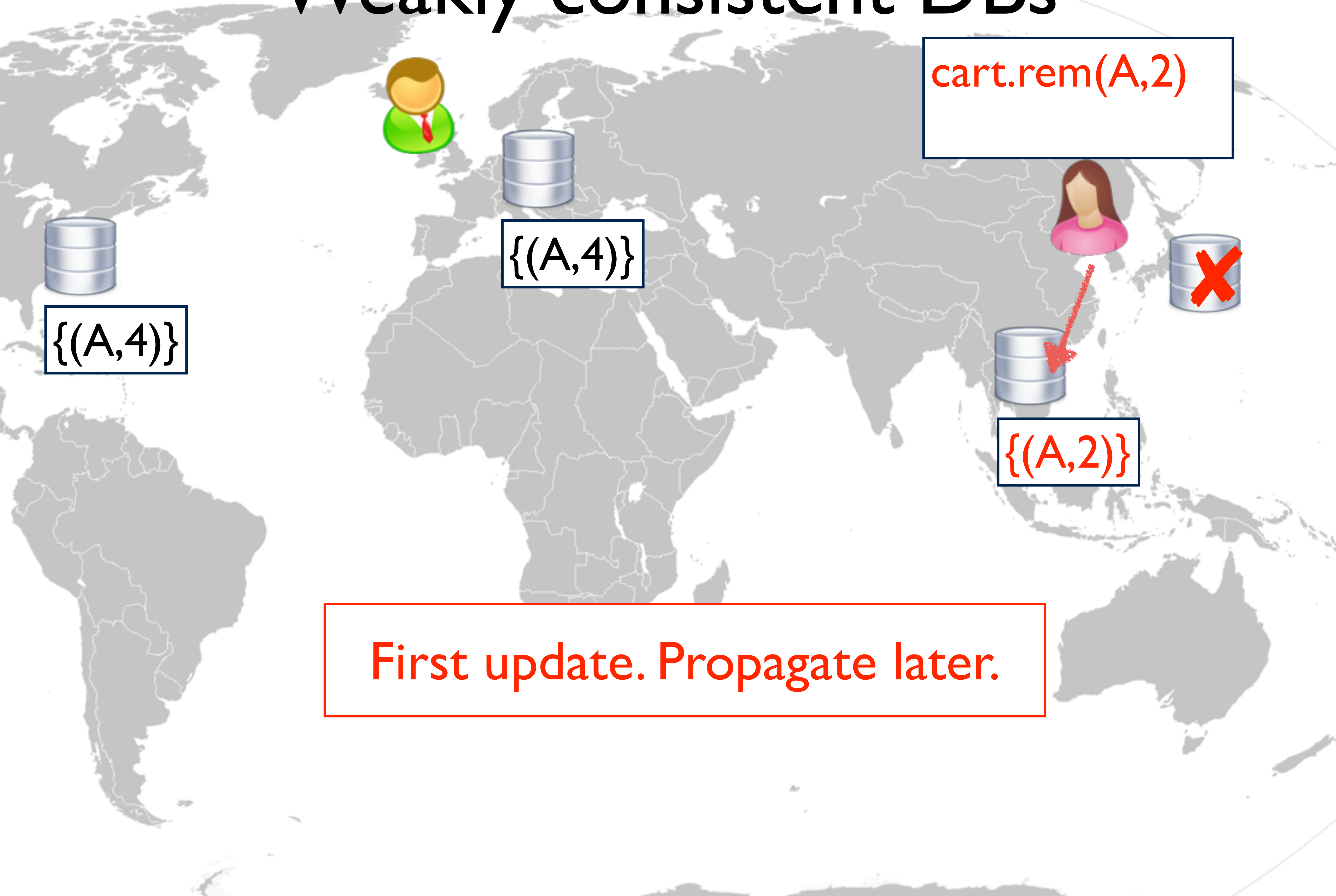


- Every data centre stores a complete replica of data
- Purpose: Minimising latency. Fault tolerance.

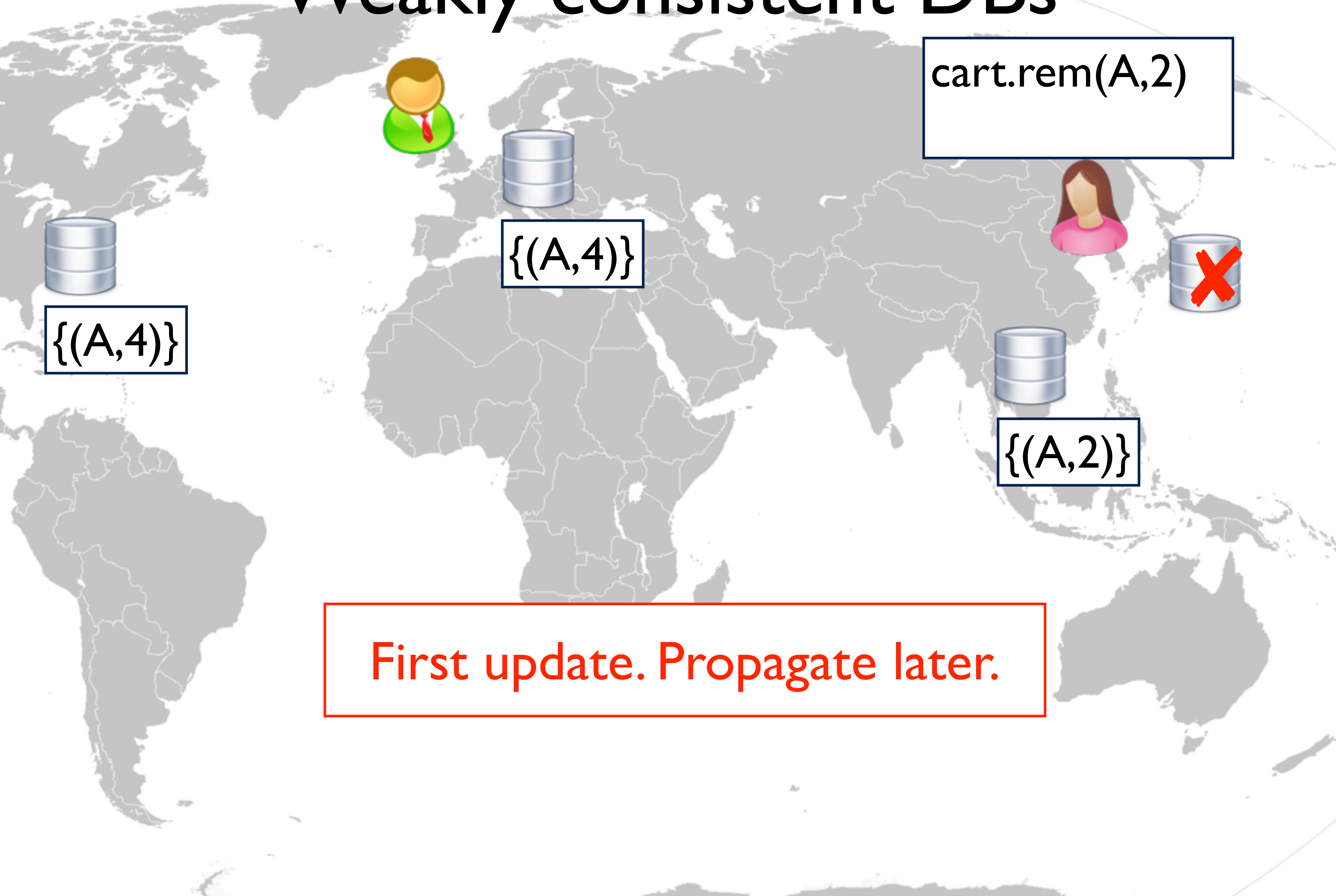
Weakly consistent DBs



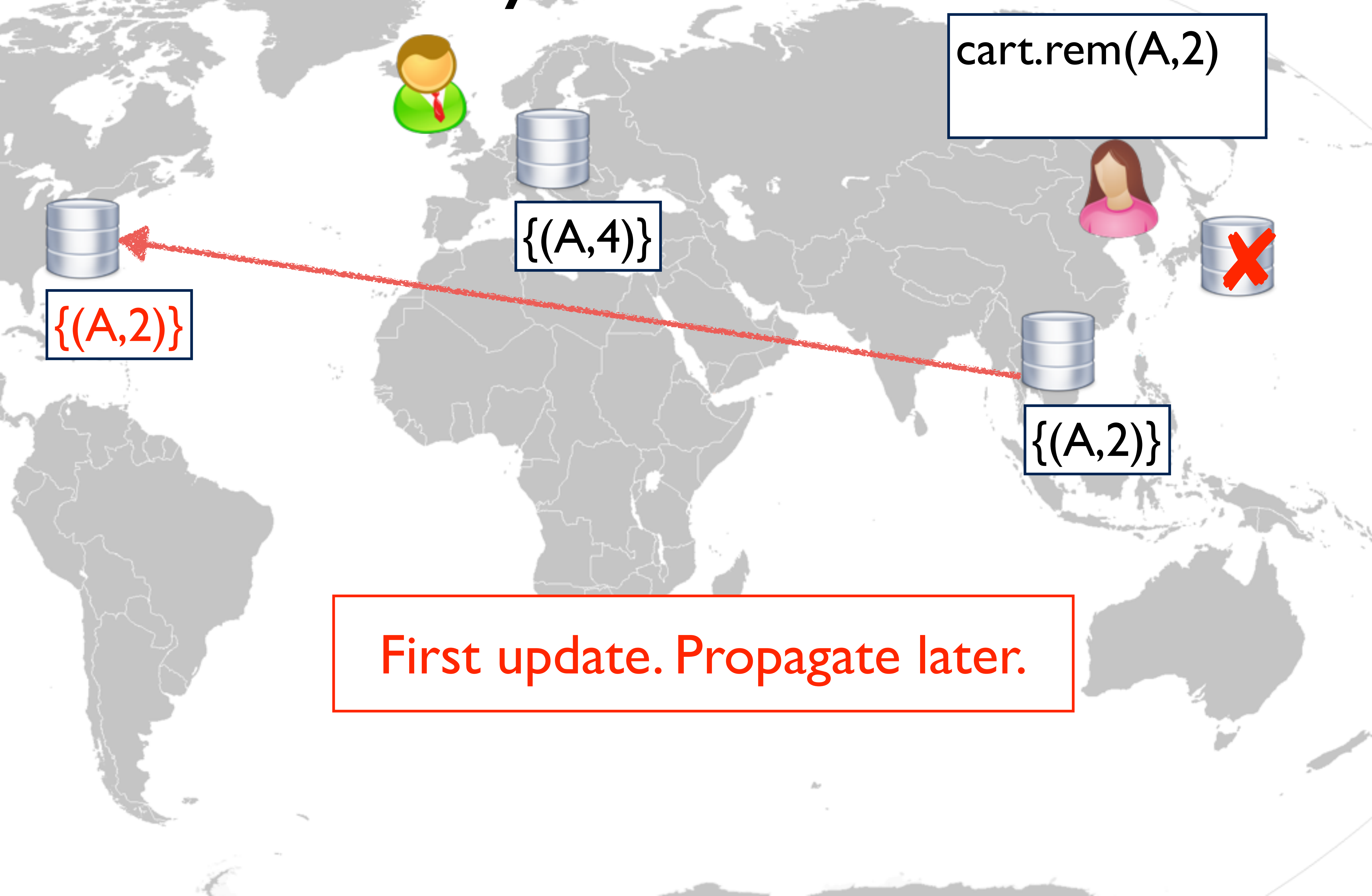
Weakly consistent DBs



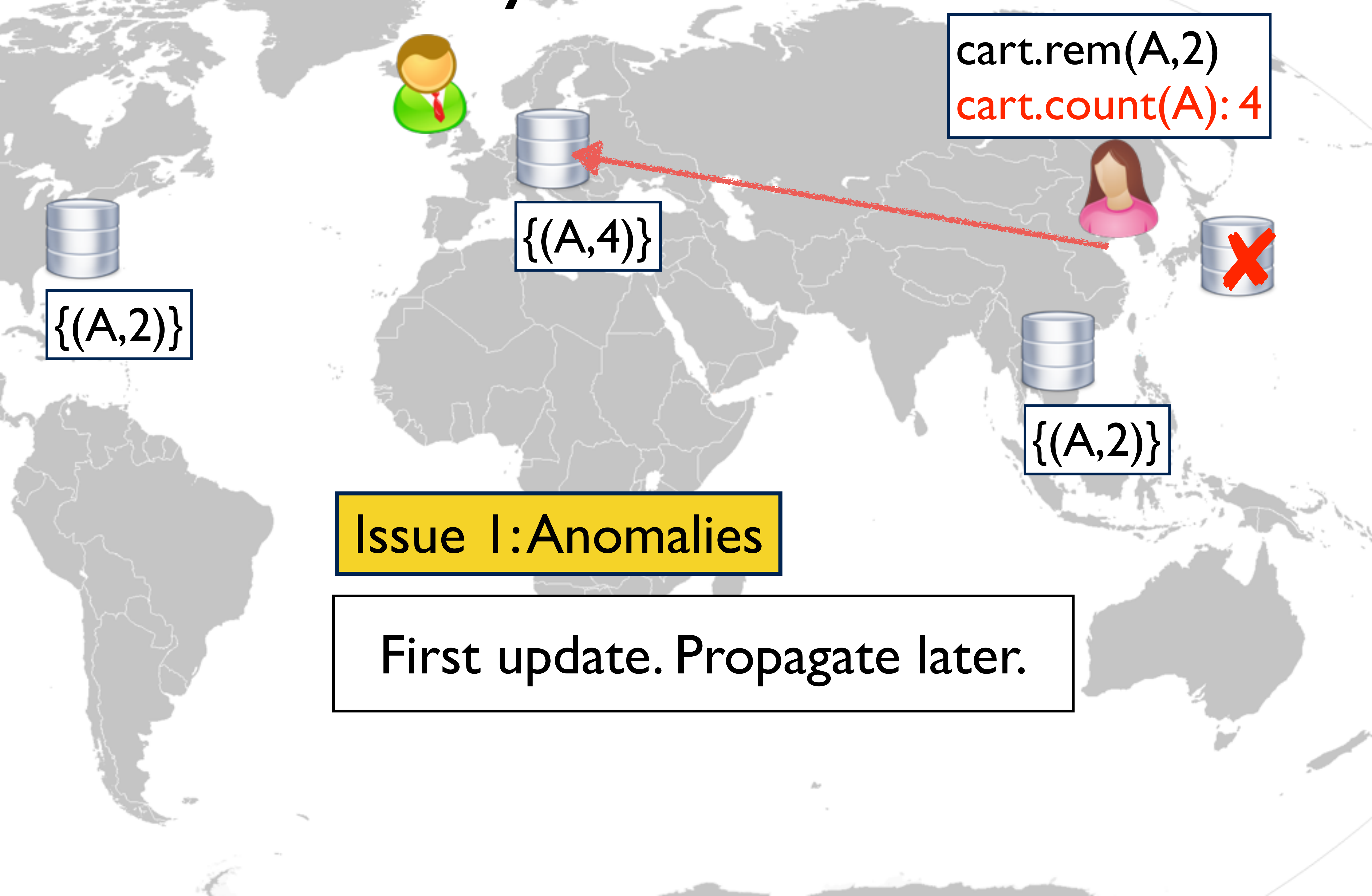
Weakly consistent DBs



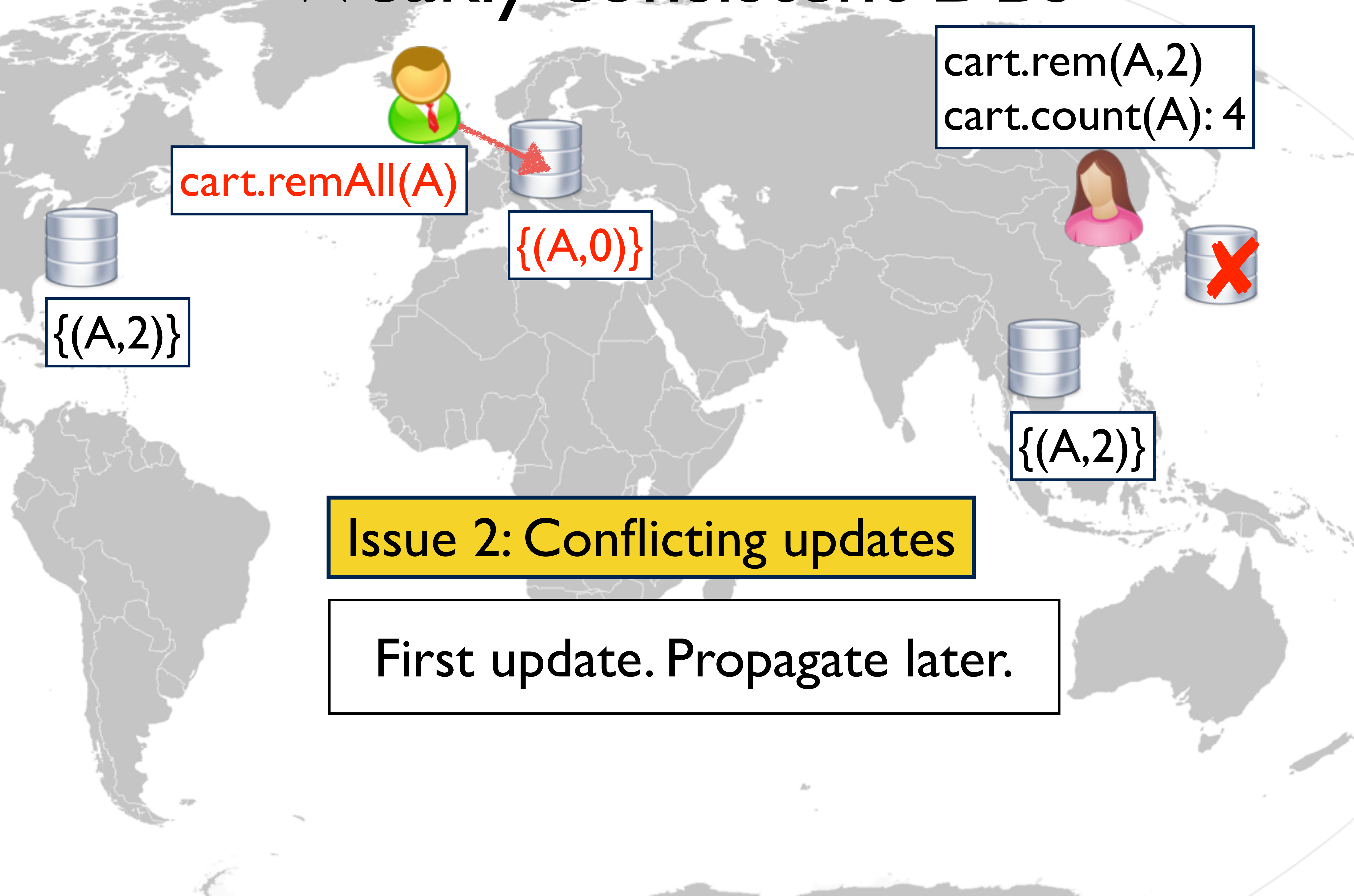
Weakly consistent DBs



Weakly consistent DBs



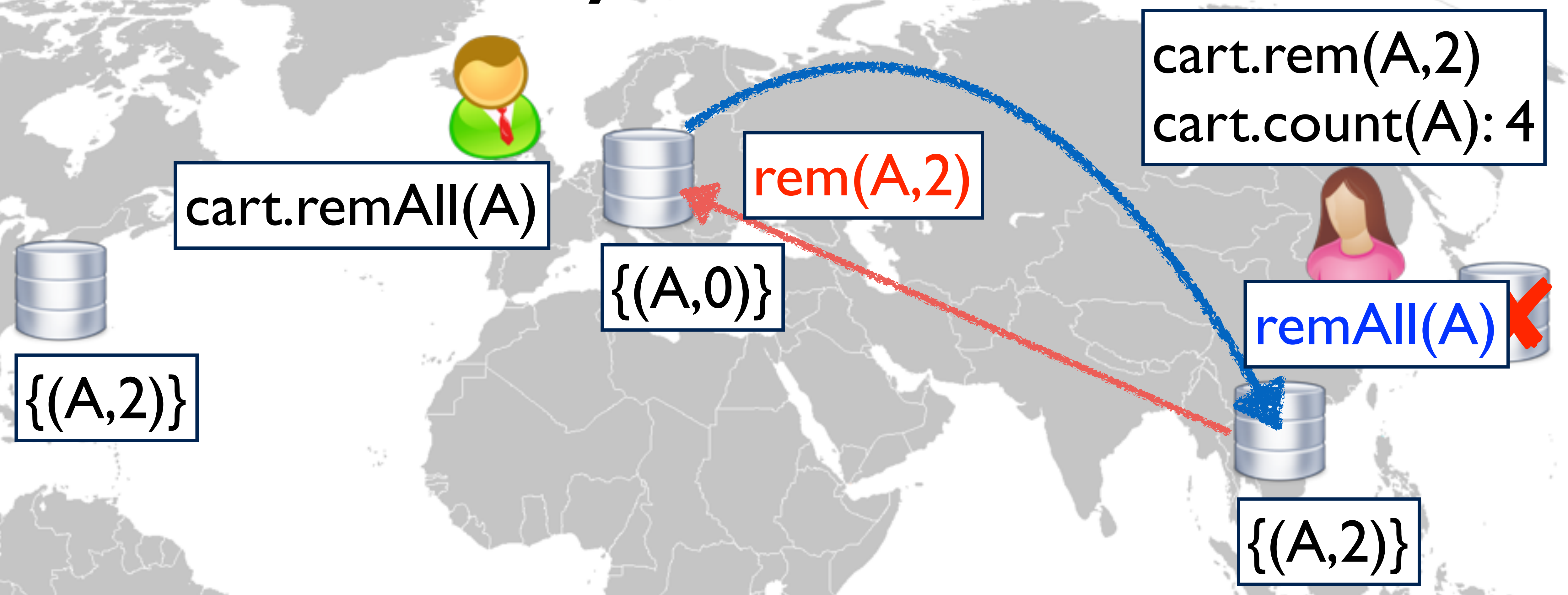
Weakly consistent DBs



Issue 2: Conflicting updates

First update. Propagate later.

Weakly consistent DBs



Issue 2: Conflicting updates

First update. Propagate later.

How to develop correct programs running on top of weakly consistent distributed databases?

How to develop correct programs running on top of weakly consistent distributed databases?

1. Strengthen consistency selectively.

How to develop correct programs running on top of weakly consistent distributed databases?

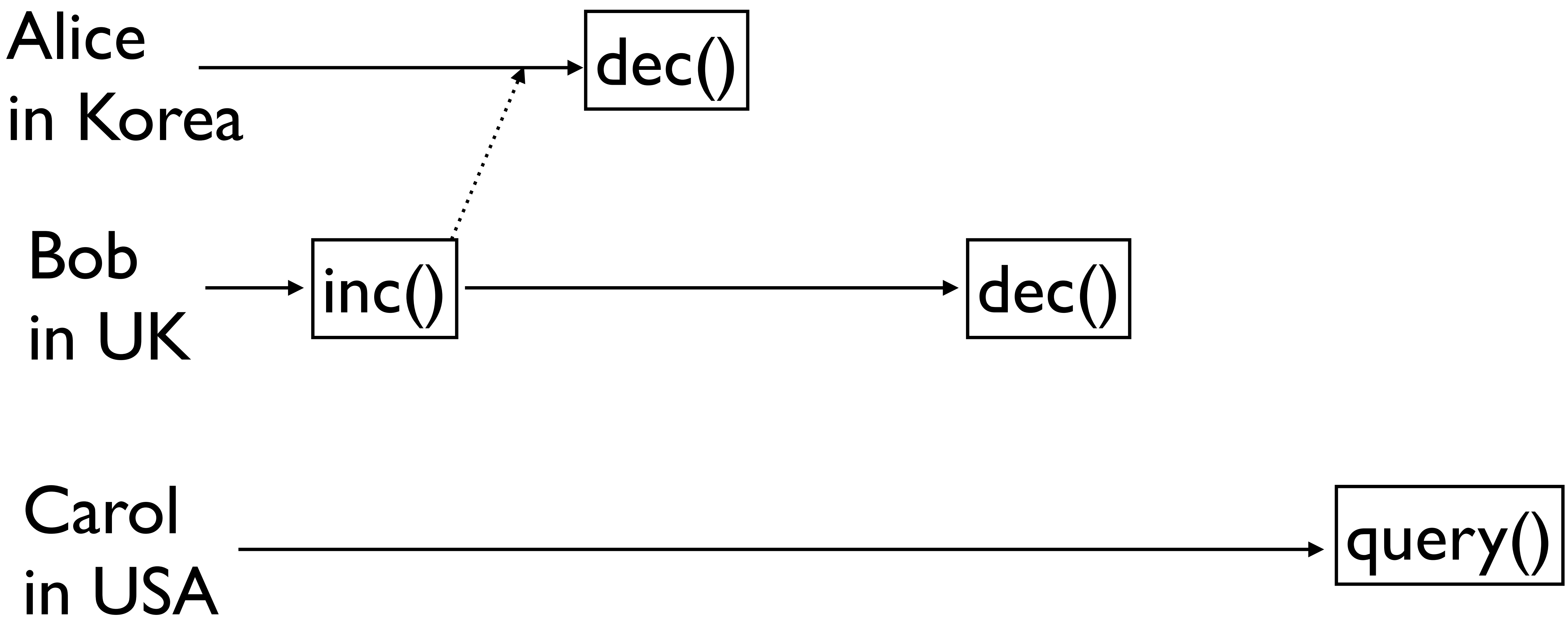
1. Strengthen consistency selectively.
2. Prove the correctness of a program.

Simple bank account

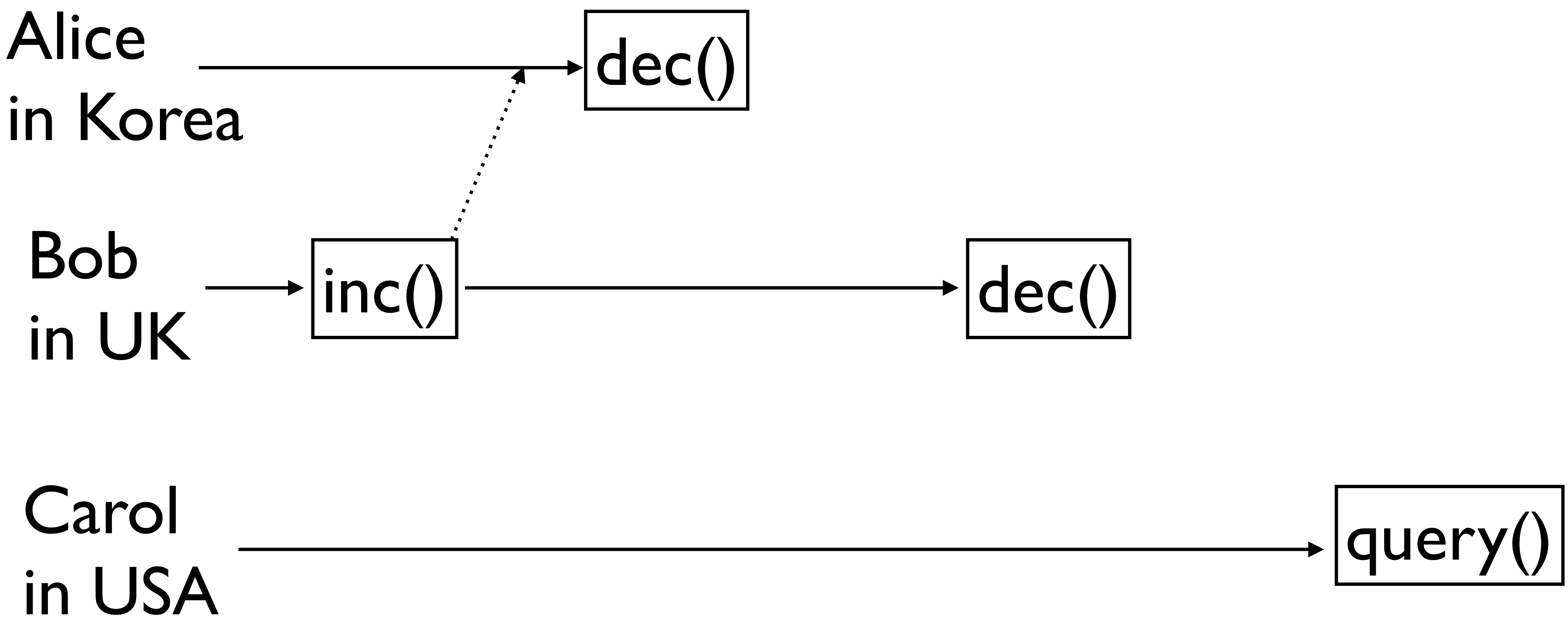
```
class account {
  // invariant: amount >= 0
  var amount = 0

  def query() = { return amount }

  def inc() = {
    amount = amount+1; return true
  }

  def dec() = {
    if (amount > 0) {
      amount = amount-1; return true
    }
    else { return false }
  }
}
```

Distributed bank account

```
class account {
  // invariant: amount >= 0
  var[dis] amount = 0

  def query() = { return (amount, (a)=>a) }

  def inc() = {
    amount = amount+1; return (true, (a)=>a+1)
  }

  def dec() = {
    if (amount > 0) {
      amount = amount-1; return (true, (a)=>a-1)
    }
    else { return (false, (a)=>a) }
  }
}
```

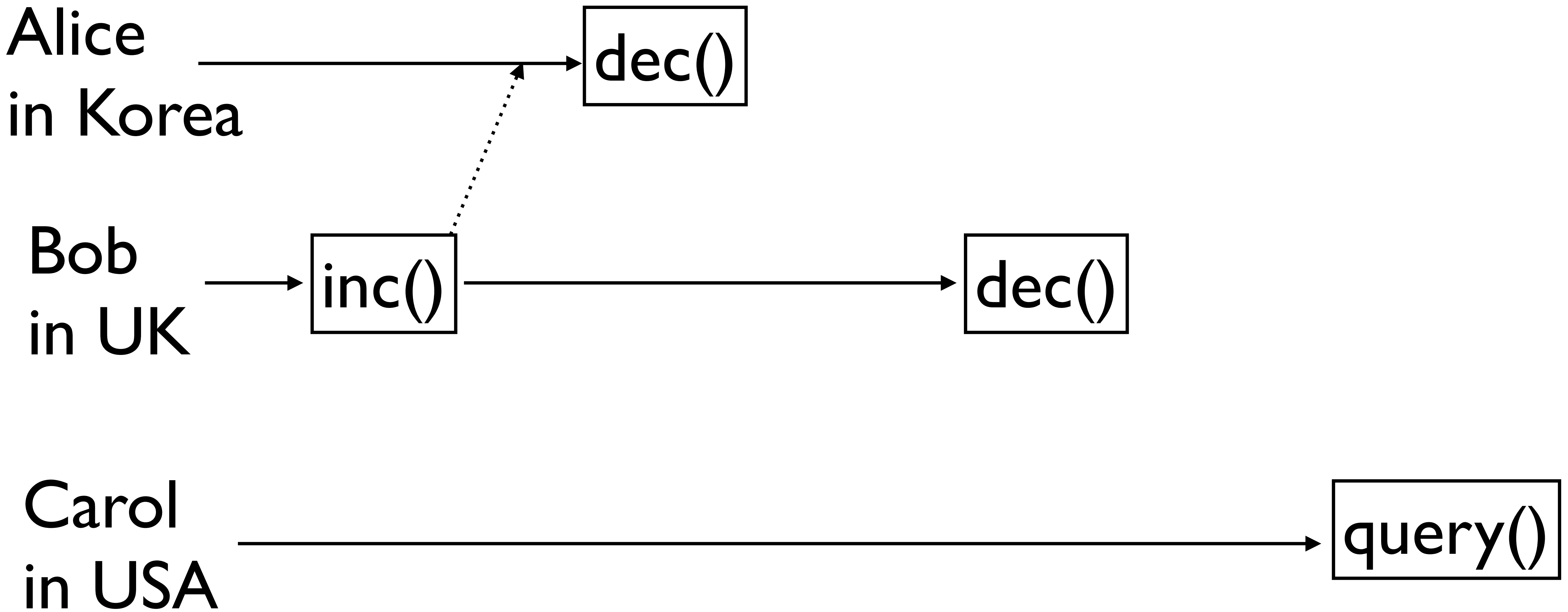


[Q] What cannot be the result of query()?
(a) 0 (b) -1 (c) -2 (d) all possible



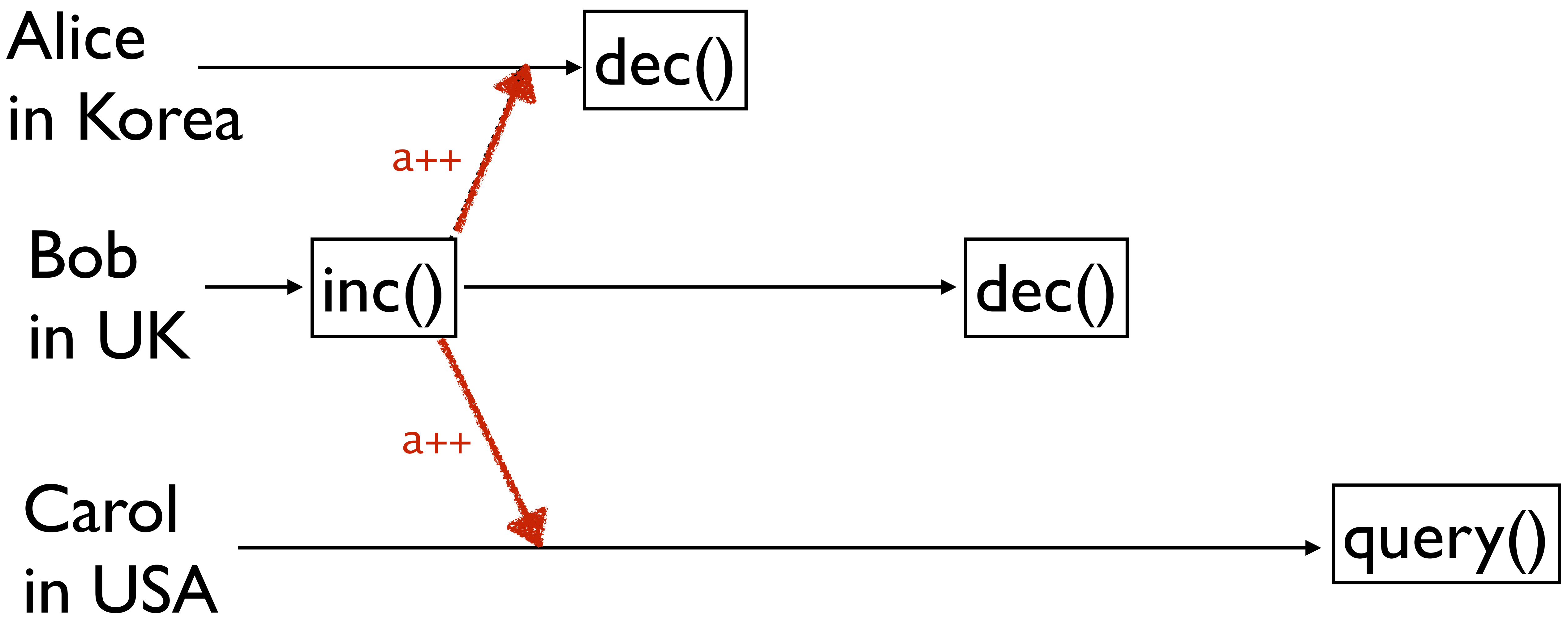
[Q] What cannot be the result of query()?

(a) 0 (b) -1 (c) -2 (d) all possible

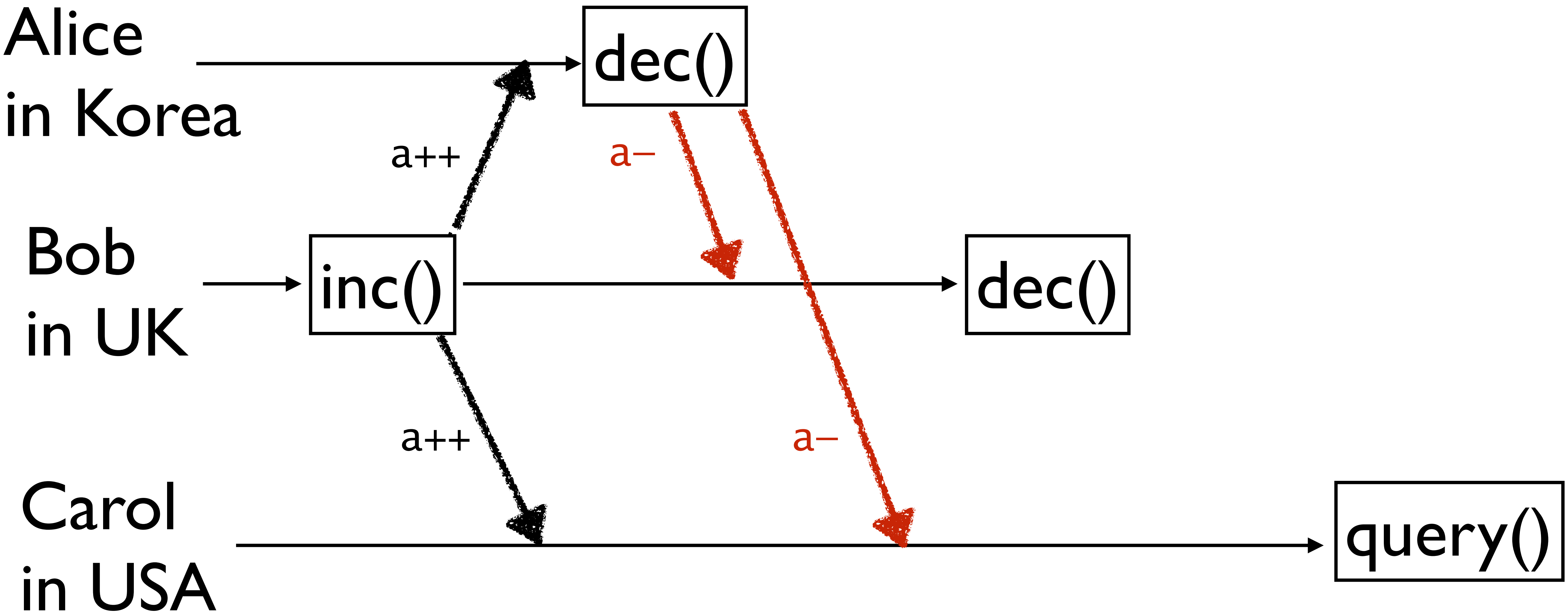


[Q] What cannot be the result of query()?

(a) 0 (b) -1 (c) -2 (d) all possible

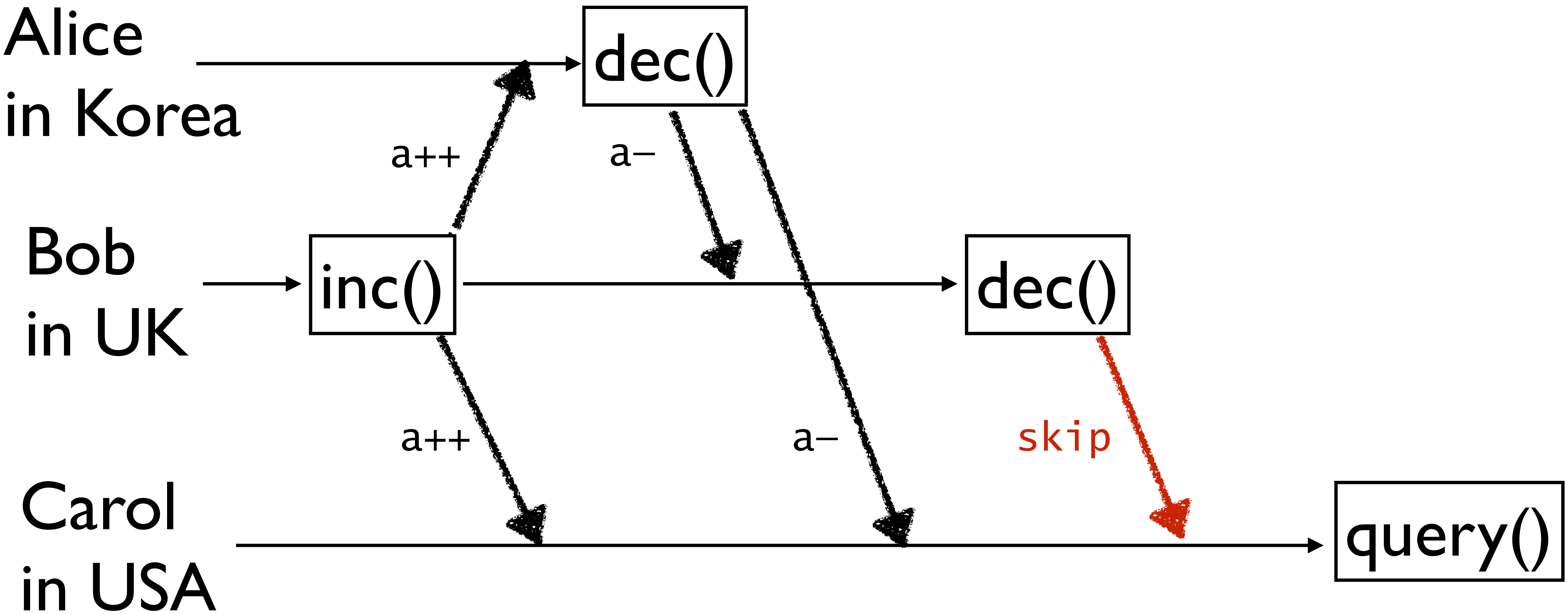


[Q] What cannot be the result of query()?
(a) 0 (b) -1 (c) -2 (d) all possible

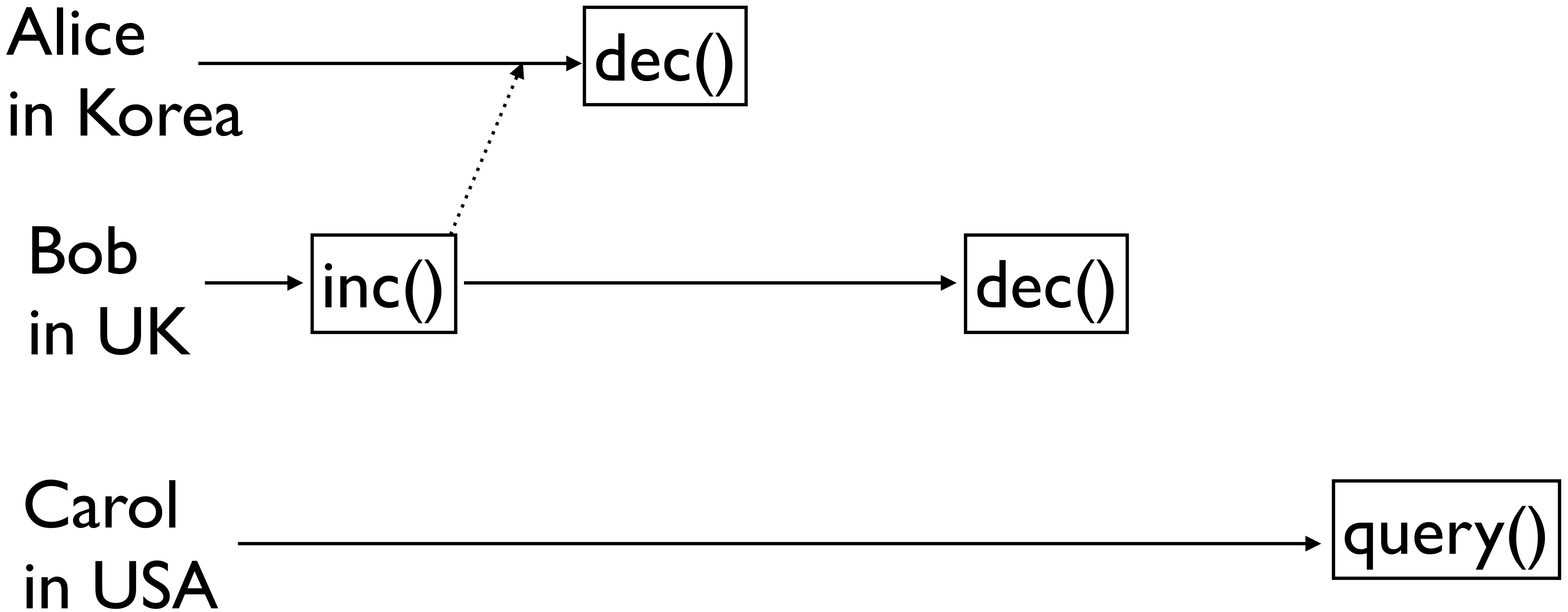


[Q] What cannot be the result of query()?

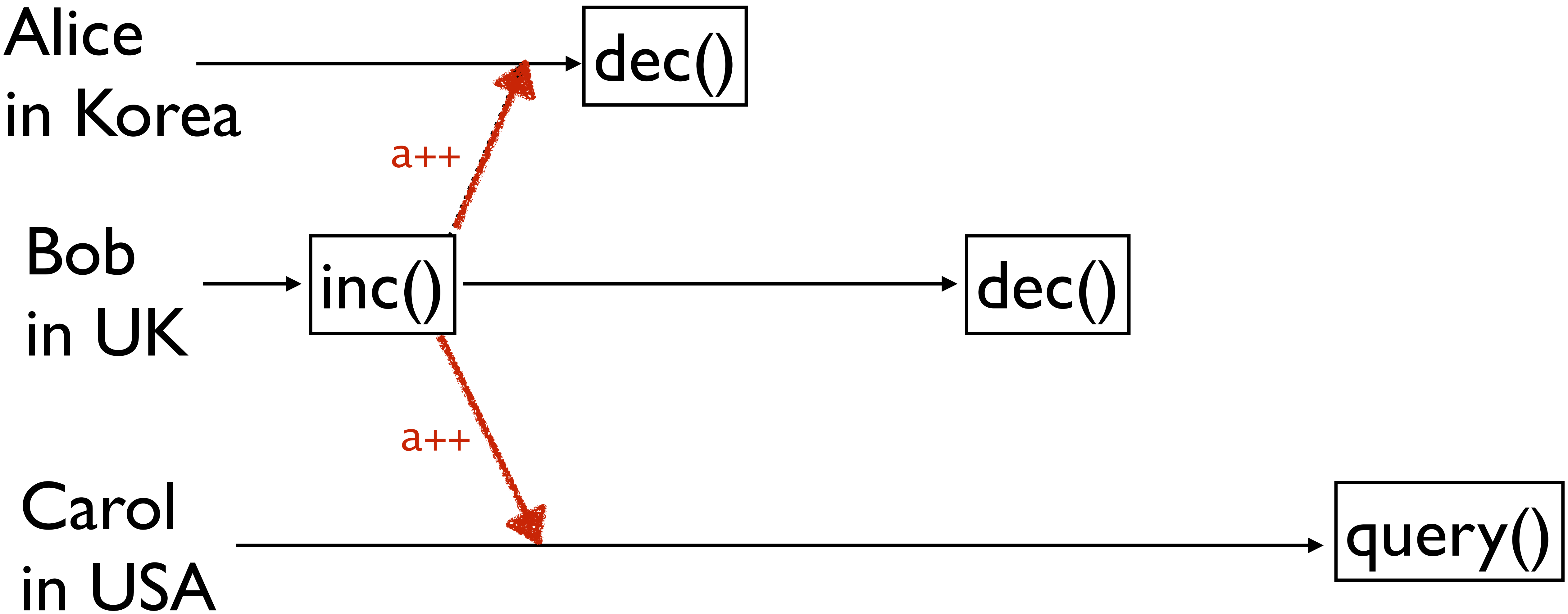
(a) 0 (b) -1 (c) -2 (d) all possible



[Q] What cannot be the result of query()?
(a) 0 (b) -1 (c) -2 (d) all possible

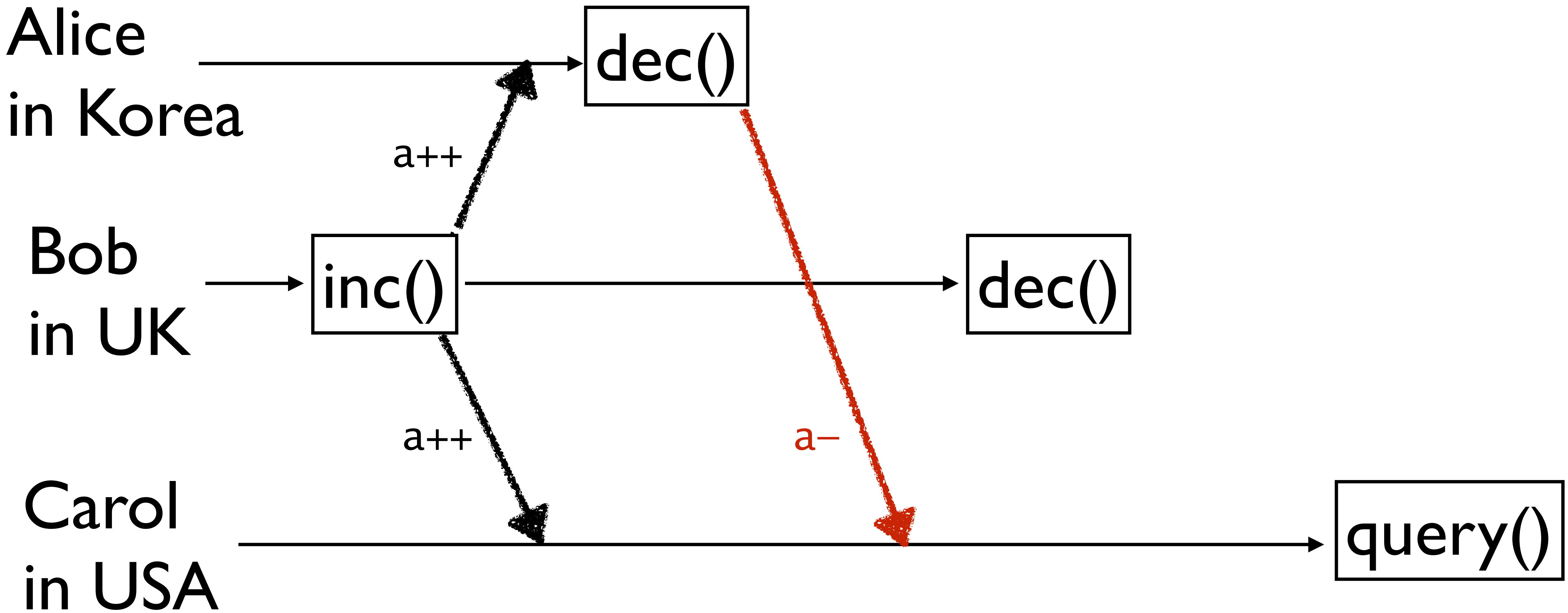


[Q] What cannot be the result of query()?
(a) 0 (b) -1 (c) -2 (d) all possible



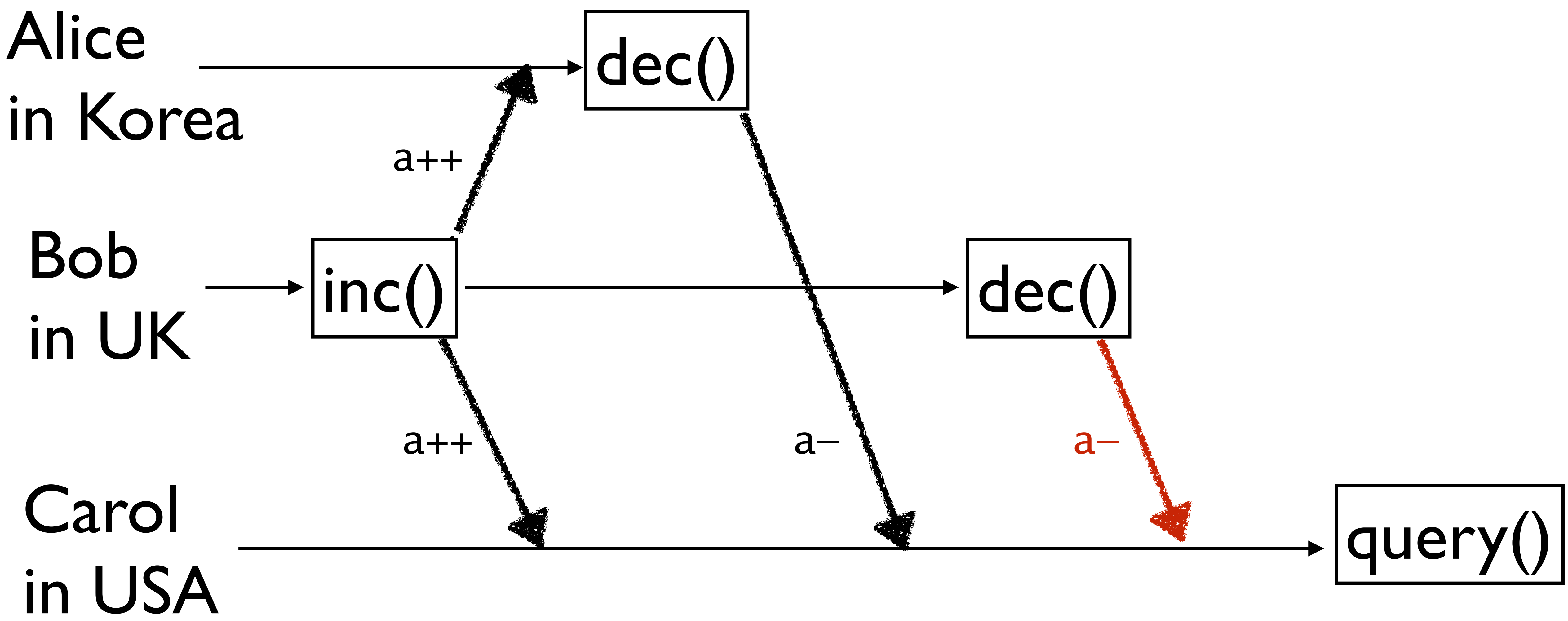
[Q] What cannot be the result of query()?

(a) 0 (b) -1 (c) -2 (d) all possible

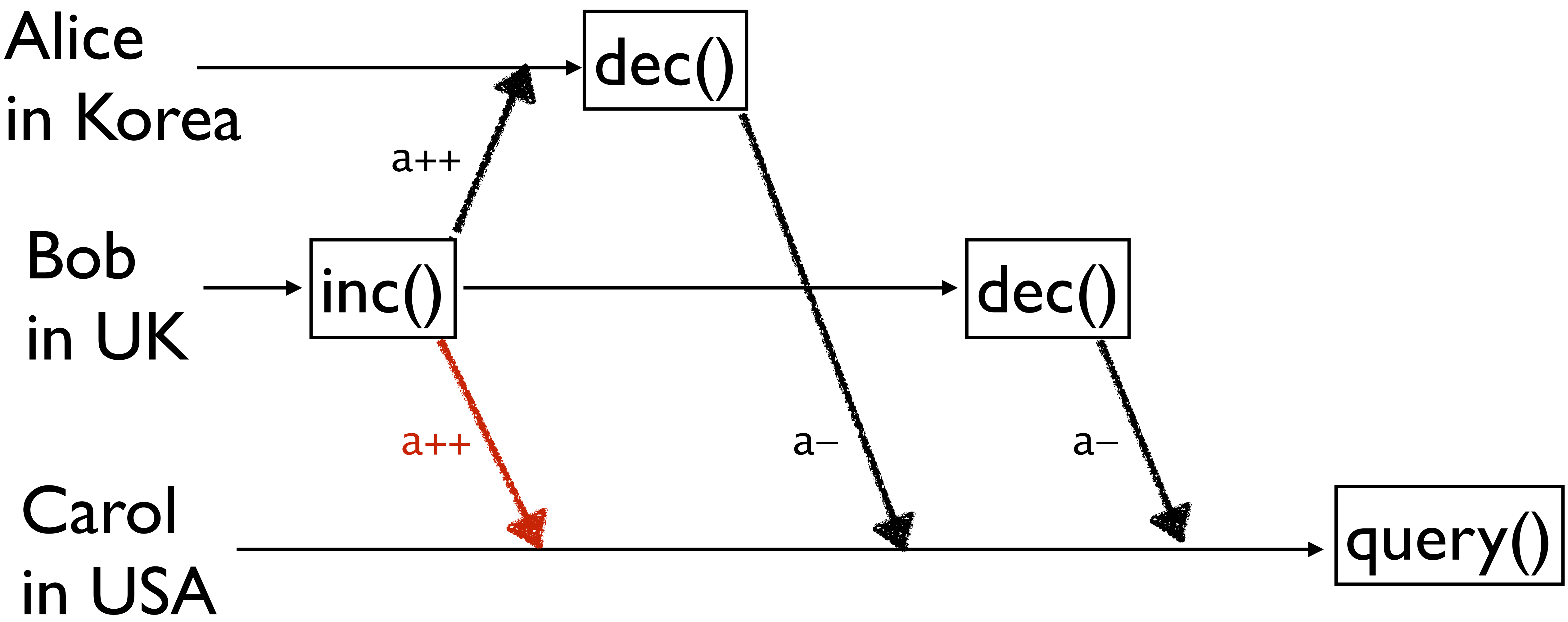


[Q] What cannot be the result of query()?

(a) 0 (b) -1 (c) -2 (d) all possible

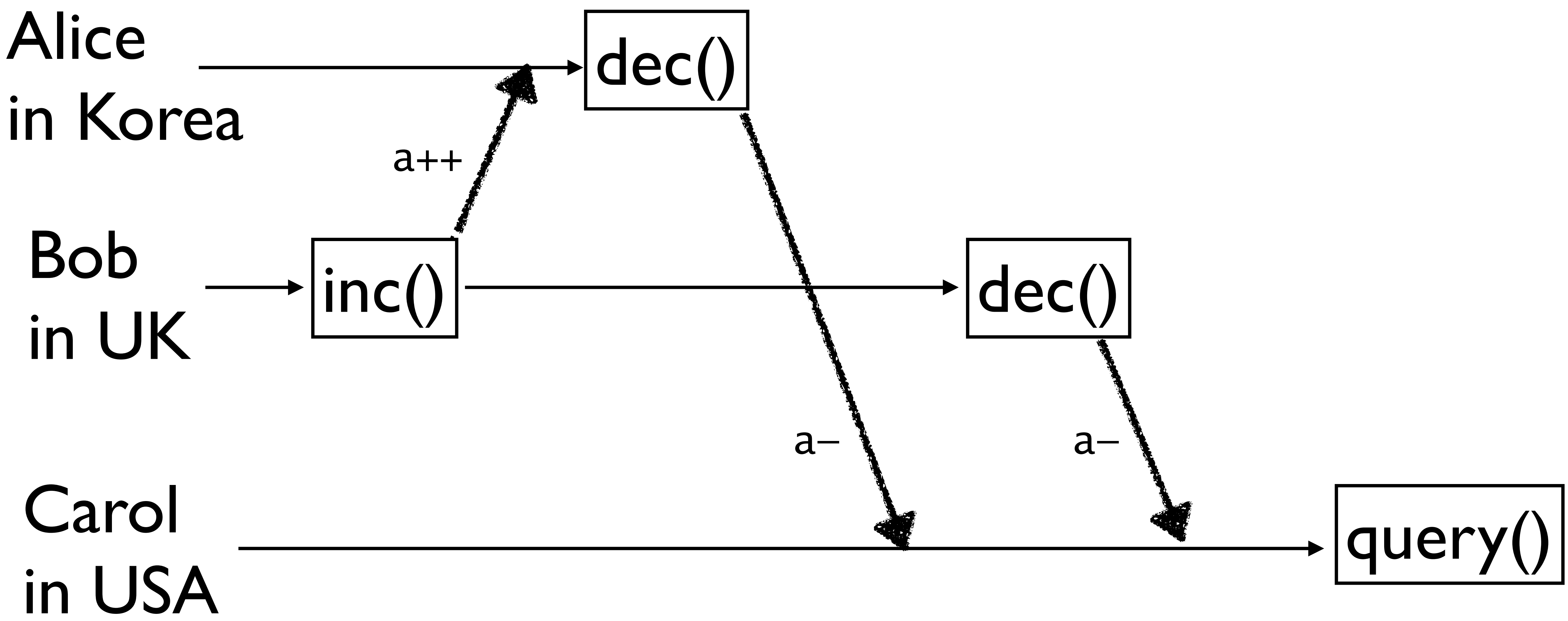


[Q] What cannot be the result of query()?
(a) 0 (b) -1 (c) -2 (d) all possible



[Q] What cannot be the result of query()?

(a) 0 (b) -1 (c) -2 (d) all possible



[Q] What cannot be the result of query()?

(a) 0 (b) -1 (c) -2 (d) all possible

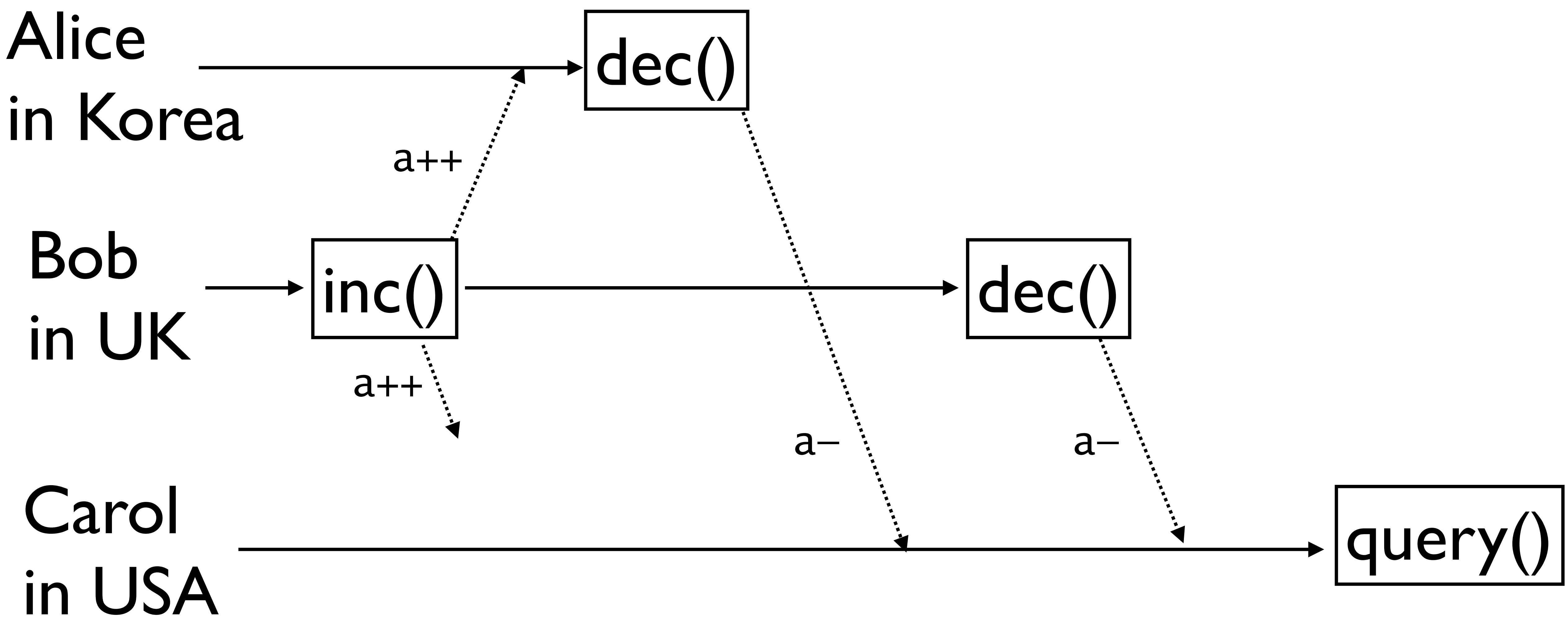
How to write correct prog.?

1. Strengthen consistency selectively.
2. Prove the correctness of a program.

Causal consistency

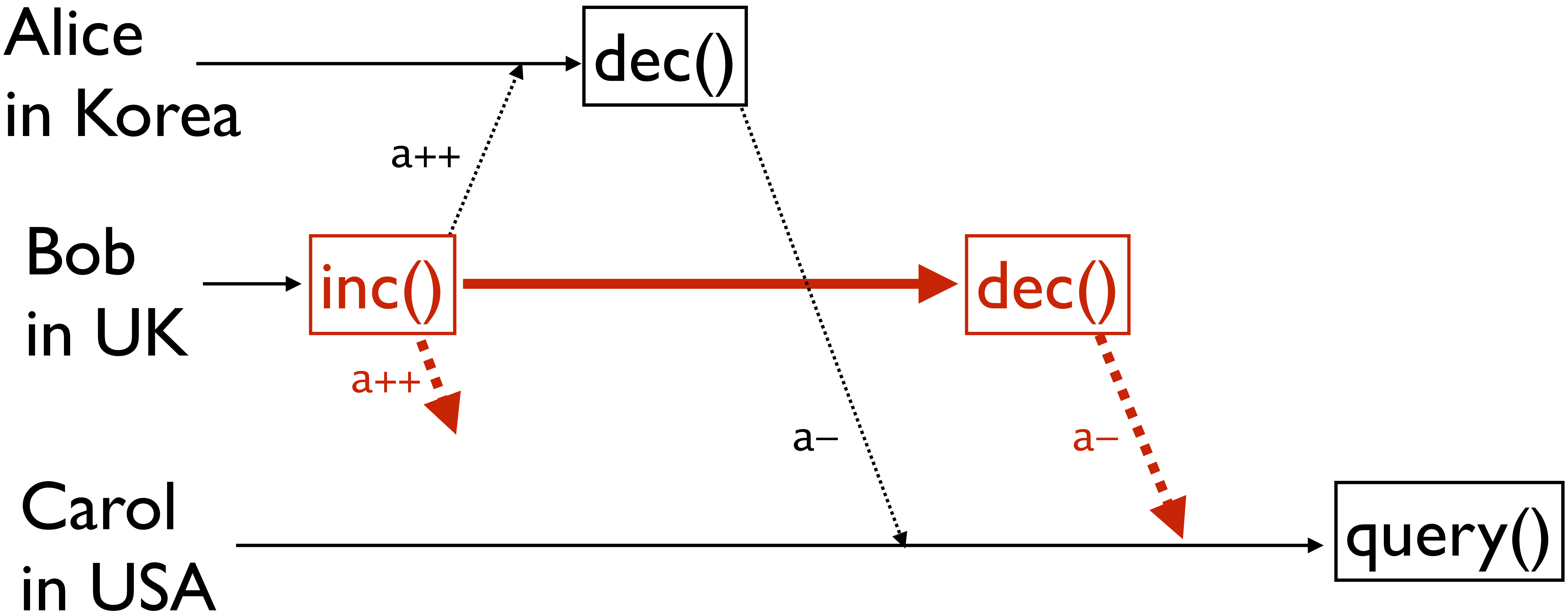
- Message delivery preserves the dependency of events.

Axiom: HB is transitive.



[Q] What cannot be the result of query()?

(a) 0 (b) -1 (c) -2 (d) all possible



Not causally consistent.

[Q] What cannot be the result of query()?
(a) 0 (b) -1 (c) -2 (d) all possible

```
use causality

class account {
  // invariant: amount >= 0
  var[dis] amount = 0

  def query() = { return (amount, (a)=>a) }

  def inc() = {
    amount = amount+1; return (true, (a)=>a+1)
  }

  def dec() = {
    if (amount > 0) {
      amount = amount-1; return (true, (a)=>a-1)
    }
    else { return (false, (a)=>a) }
  }
}
```

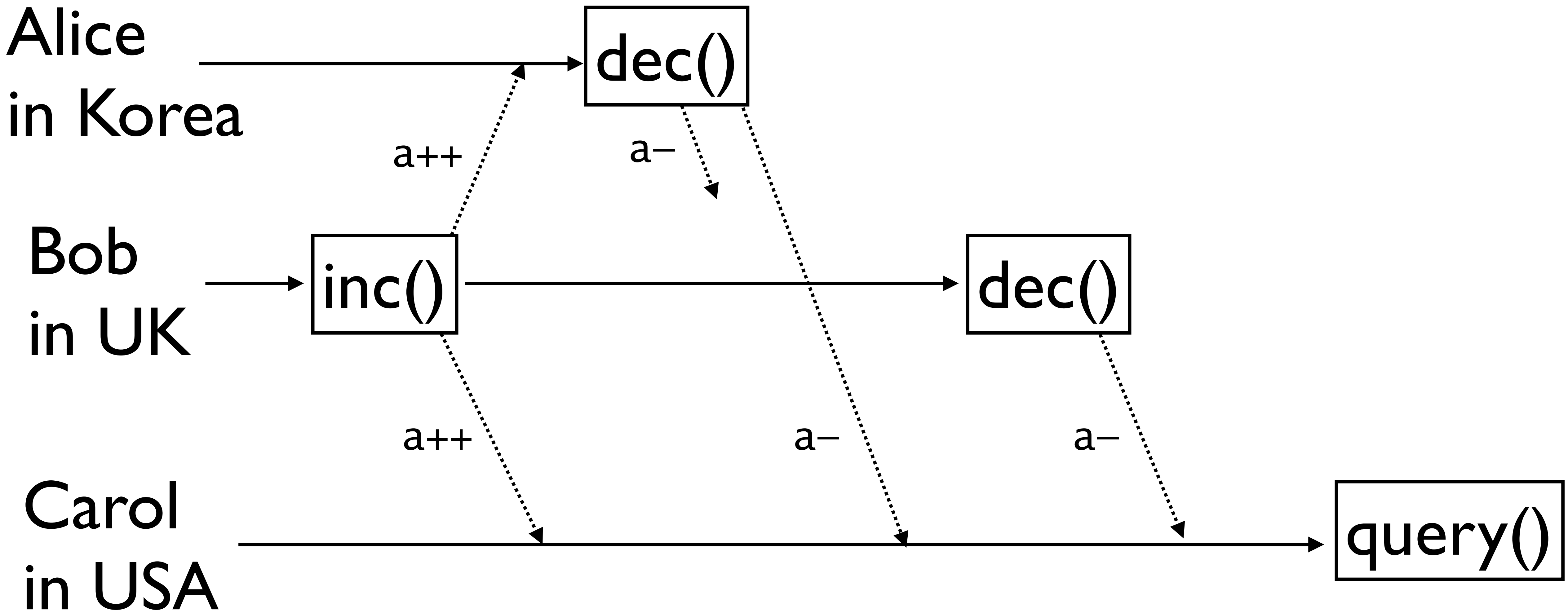
Token system

- $(\mathcal{T}, $💔$)$ where 💔 is a symmetric rel. on $\mathcal{T}$.
- Examples:

1. $\mathcal{T}$ = {lock}, 💔 = {(lock,lock)}
2. $\mathcal{T}$ = {rd,wr}, 💔 = {(rd,wr), (wr,wr), (wr,rd)}

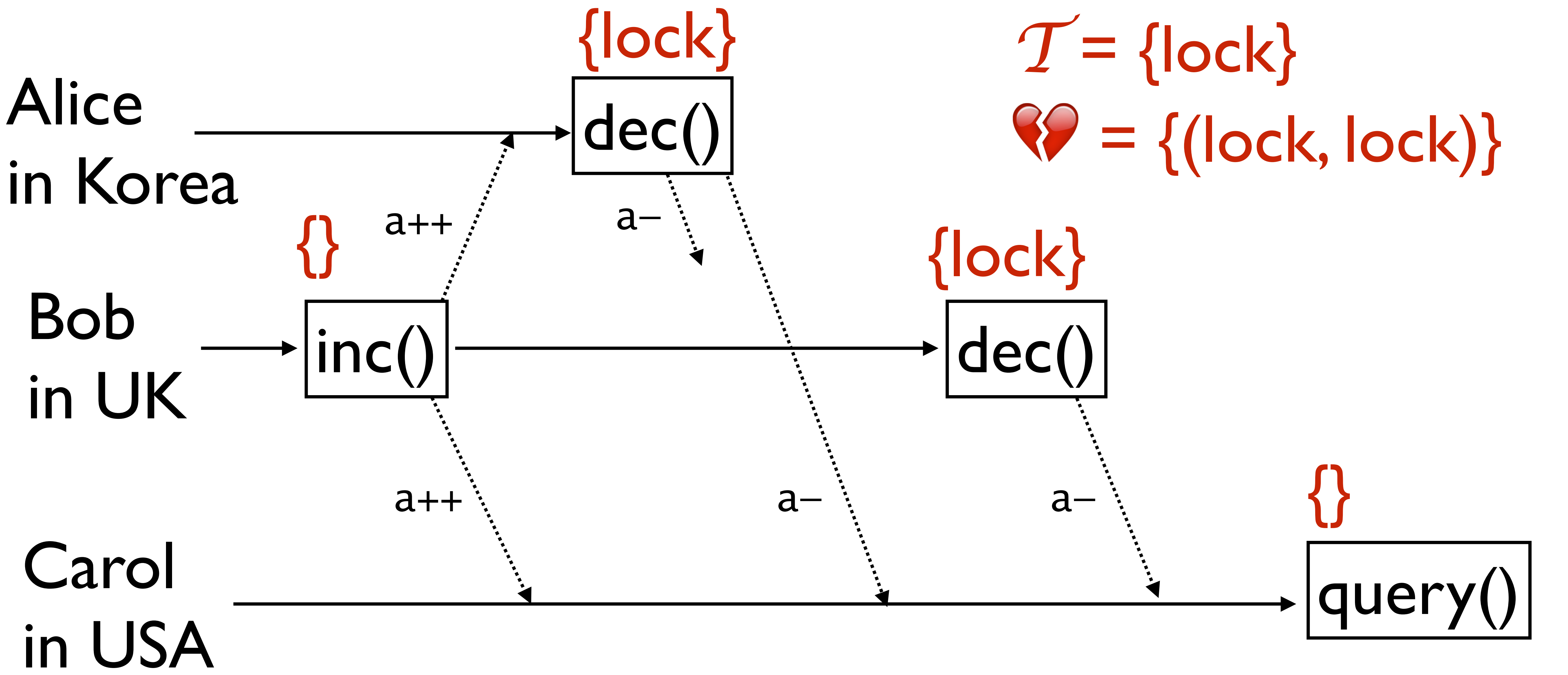
On-demand consistency using a token system ($\mathcal{T}$, 💔)

- Each operation acquires a set of tokens.
- Operations with conflicting tokens cannot be run concurrently.



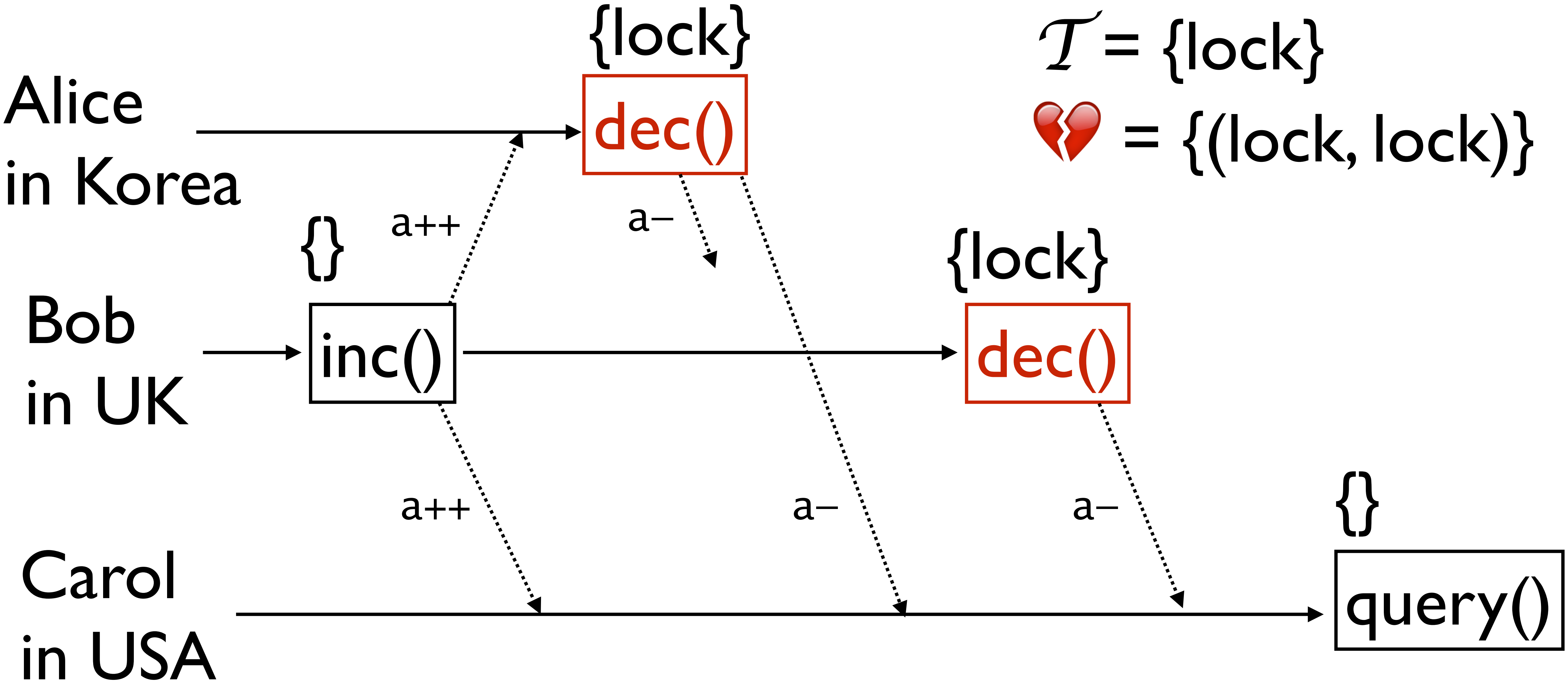
[Q] What cannot be the result of query()?

(a) 0 (b) -1 (c) -2 (d) all possible



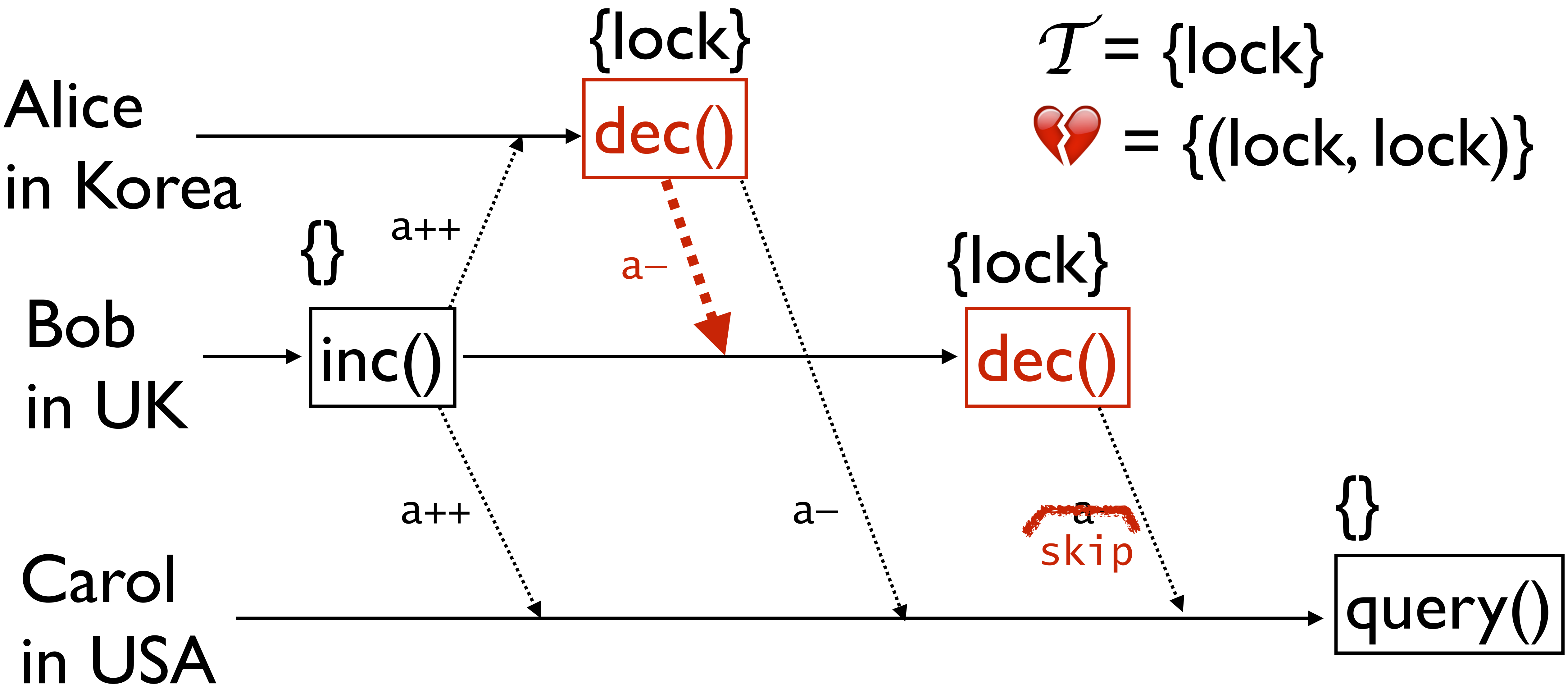
[Q] What cannot be the result of query()?

(a) 0 (b) -1 (c) -2 (d) all possible



[Q] What cannot be the result of query()?

(a) 0 (b) -1 (c) -2 (d) all possible



[Q] What cannot be the result of query()?

(a) 0 (b) -1 (c) -2 (d) all possible

```
use causality

class account {
  // invariant: amount >= 0
  var[dis] amount = 0
  use-token-system({lock},{(lock,lock)})

  def query() with {} =
  { return (amount, (a)=>a) }

  def inc() with {} = {
    amount = amount+1; return (true, (a)=>a+1)
  }

  def dec() with {lock} = {
    if (amount > 0) {
      amount = amount-1; return (true, (a)=>a-1)
    }
    else { return (false, (a)=>a) }
  }
}
```

How to write correct prog.?

1. Strengthen consistency selectively.
2. Prove the correctness of a program.

Our proof rule

- Based on rely-guarantee.
- Incorporates guarantees from causal and on-demand consistency.

To prove that I is an invariant

To prove that $\mathcal{I}$ is an invariant

$$\exists G_0 \in \mathcal{P}(\mathsf{State} \times \mathsf{State}), G \in \mathsf{Token} \to \mathcal{P}(\mathsf{State} \times \mathsf{State})$$

To prove that I is an invariant

$$\exists G_0 \in \mathcal{P}(\mathsf{State} \times \mathsf{State}), G \in \mathsf{Token} \rightarrow \mathcal{P}(\mathsf{State} \times \mathsf{State})$$

S1. $\sigma_{\mathsf{init}} \in I$

To prove that I is an invariant

$$\exists G_0 \in \mathcal{P}(\mathsf{State} \times \mathsf{State}), G \in \mathsf{Token} \to \mathcal{P}(\mathsf{State} \times \mathsf{State})$$

S1. $\sigma_{\mathsf{init}} \in I$

S2. $G_0(I) \subseteq I \land \forall \tau.\ G(\tau)(I) \subseteq I$

To prove that I is an invariant

$$\exists G_0 \in \mathcal{P}(\mathsf{State} \times \mathsf{State}), G \in \mathsf{Token} \to \mathcal{P}(\mathsf{State} \times \mathsf{State})$$

S1. $\sigma_{\mathsf{init}} \in I$

S2. $G_0(I) \subseteq I \wedge \forall \tau.\ G(\tau)(I) \subseteq I$

S3. $\forall o, \sigma, \sigma'.\ (\sigma \in I \wedge (\sigma, \sigma') \in (G_0 \cup G((\mathcal{F}_o^{\mathsf{tok}}(\sigma))^{\perp}))^*)$
$\implies (\sigma', \mathcal{F}_o^{\mathsf{eff}}(\sigma)(\sigma')) \in G_0 \cup G(\mathcal{F}_o^{\mathsf{tok}}(\sigma))$

To prove that I is an invariant

$$\text{S3. } \forall o, \sigma, \sigma'.\ (\sigma \in I \land (\sigma, \sigma') \in (G_0 \cup G((\mathcal{F}_o^{\mathsf{tok}}(\sigma))^{\perp}))^*) \implies (\sigma', \mathcal{F}_o^{\mathsf{eff}}(\sigma)(\sigma')) \in G_0 \cup G(\mathcal{F}_o^{\mathsf{tok}}(\sigma))$$

To prove that I is an invariant

$$\text{S3. } \forall o, \sigma, \sigma'. \ (\sigma \in I \land (\sigma, \sigma') \in (\quad \ldots \quad)^*) \implies (\sigma', \mathcal{F}_o^{\mathsf{eff}}(\sigma)(\sigma')) \in \quad \ldots$$

To prove that I is an invariant

$$\text{S3. } \forall o, \sigma, \sigma'.\ (\sigma \in I \wedge (\sigma, \sigma') \in (\quad \ldots \quad)^*) \implies (\sigma', \mathcal{F}_o^{\mathsf{eff}}(\sigma)(\sigma')) \in \quad \ldots$$

To prove that $\mathcal{I}$ is an invariant

$$\text{S3. } \forall o, \sigma, \sigma'.\ (\sigma \in I \wedge (\sigma, \sigma') \in (\quad \ldots \quad)^*)$$
$$\implies (\sigma', \mathcal{F}_o^{\mathsf{eff}}(\sigma)(\sigma')) \in \quad \ldots$$

To prove that I is an invariant

$$\text{S3. } \forall o, \sigma, \sigma'.\ (\sigma \in I \wedge (\sigma, \sigma') \in (\quad \ldots \quad)^*)$$
$$\implies \boxed{(\sigma', \mathcal{F}_o^{\text{eff}}(\sigma)(\sigma')) \in \quad \ldots}$$

To prove that I is an invariant

$$\text{S3. } \forall o, \sigma, \sigma'. \ (\sigma \in I \wedge (\sigma, \sigma') \in (\qquad \ldots \qquad)^*)$$
$$\implies (\sigma', \mathcal{F}_o^{\mathsf{eff}}(\sigma)(\sigma')) \in G_0 \cup G(\mathcal{F}_o^{\mathsf{tok}}(\sigma))$$

To prove that $\mathcal{I}$ is an invariant

$$\text{S3. } \forall o, \sigma, \sigma'.\ (\sigma \in I \wedge (\sigma, \sigma') \in (G_0 \cup G(\boxed{(\mathcal{F}_o^{\textsf{tok}}(\sigma))^{\perp}}))^*)$$
$$\implies (\sigma', \mathcal{F}_o^{\textsf{eff}}(\sigma)(\sigma')) \in G_0 \cup G(\mathcal{F}_o^{\textsf{tok}}(\sigma))$$

T$^{\perp}$ = {**τ** | ∄**τ**'∈T. (**τ**, **τ**') ∈ 💔}

$$\mathcal{I} = \{ \sigma \mid 0 \le \sigma \}$$
$$G_0 = \{(\sigma, \sigma') \mid \sigma \le \sigma'\}$$
$$G_1(\text{lock}) = \{(\sigma, \sigma') \mid 0 < \sigma \wedge \sigma' \le \sigma\}$$

$$\text{S3. } \forall o, \sigma, \sigma'. \ (\sigma \in I \wedge (\sigma, \sigma') \in (G_0 \cup G((\mathcal{F}_o^{\mathsf{tok}}(\sigma))^{\perp}))^*) \\ \implies (\sigma', \mathcal{F}_o^{\mathsf{eff}}(\sigma)(\sigma')) \in G_0 \cup G(\mathcal{F}_o^{\mathsf{tok}}(\sigma))$$

$$\mathcal{I} = \{ \sigma \mid 0 \leq \sigma \}$$
$$G_0 = \{(\sigma, \sigma') \mid \sigma \leq \sigma'\}$$
$$G_1(\text{lock}) = \{(\sigma, \sigma') \mid 0 < \sigma \wedge \sigma' \leq \sigma\}$$

$$\text{S3. } \forall o, \sigma, \sigma'.\ (\sigma \in I \wedge (\sigma, \sigma') \in (G_0 \cup G((\mathcal{F}_o^{\mathsf{tok}}(\sigma))^{\perp}))^*)$$

dec()

$$\implies (\sigma', \mathcal{F}_o^{\mathsf{eff}}(\sigma)(\sigma')) \in G_0 \cup G(\mathcal{F}_o^{\mathsf{tok}}(\sigma))$$

$\mathcal{I}$ = { σ | 0 ≤ σ }
G_0 = {(σ, σ') | σ ≤ σ'}
G_1(lock) = {(σ, σ') | 0 < σ ∧ σ' ≤ σ}

{}

$$\text{S3. } \forall o, \sigma, \sigma'.\ (\sigma \in I \wedge (\sigma, \sigma') \in (G_0 \cup G(((\mathcal{F}_o^{\mathsf{tok}}(\sigma))^{\perp}))^*)$$

dec()

$$\implies (\sigma', \mathcal{F}_o^{\mathsf{eff}}(\sigma)(\sigma')) \in G_0 \cup G(\mathcal{F}_o^{\mathsf{tok}}(\sigma))$$

{lock}

$I = \{ \sigma \mid 0 \leq \sigma \}$

$G_0 = \{(\sigma, \sigma') \mid \sigma \leq \sigma'\}$

$G_1(\text{lock}) = \{(\sigma, \sigma') \mid 0 < \sigma \wedge \sigma' \leq \sigma\}$

G_0^*

$\{\}$

$$\text{S3. } \forall o, \sigma, \sigma'.\ (\sigma \in I \wedge (\sigma, \sigma') \in (G_0 \cup G((\mathcal{F}_o^{\text{tok}}(\sigma))^{\perp}))^*)$$

dec()

$$\implies (\sigma', \mathcal{F}_o^{\text{eff}}(\sigma)(\sigma')) \in G_0 \cup G(\mathcal{F}_o^{\text{tok}}(\sigma))$$

$\{\text{lock}\}$

$G_0 \cup G_1(\text{lock})$

$\mathcal{I} = \{ \sigma \mid 0 \leq \sigma \}$

$G_0 = \{(\sigma, \sigma') \mid \sigma \leq \sigma'\}$

$G_1(\text{lock}) = \{(\sigma, \sigma') \mid 0 < \sigma \wedge \sigma' \leq \sigma\}$

G_0^*

$$\text{S3. } \forall \underline{o}, \sigma, \sigma'.\ (\sigma \in I \wedge (\sigma, \sigma') \in \boxed{})$$

dec()

$$\implies (\sigma', \mathcal{F}_o^{\mathsf{eff}}(\sigma)(\sigma')) \in \boxed{}$$

$G_0 \cup G_1(\text{lock})$

$$\mathcal{I} = \{ \sigma \mid 0 \le \sigma \}$$

$$G_0 = \{(\sigma, \sigma') \mid \sigma \le \sigma'\}$$

$$G_1(\text{lock}) = \{(\sigma, \sigma') \mid 0 < \sigma \wedge \sigma' \le \sigma\}$$

G_0^*

$$\text{S3. } \forall \underline{o}, \sigma, \sigma'. \; (\sigma \in I \wedge (\sigma, \sigma') \in \square)$$

dec()

$$\implies (\sigma', \underline{\mathcal{F}_o^{\text{eff}}(\sigma)(\sigma')}) \in \square$$

if $0 < \sigma$ then σ'-1 else σ'

$G_0 \cup G_1(\text{lock})$

What if no on-demand consistency?

$$\text{S3. } \forall o, \sigma, \sigma'.\ (\sigma \in I \wedge (\sigma, \sigma') \in (G_0 \cup G((\mathcal{F}_o^{\mathsf{tok}}(\sigma))^{\perp}))^*) \implies (\sigma', \mathcal{F}_o^{\mathsf{eff}}(\sigma)(\sigma')) \in G_0 \cup G(\mathcal{F}_o^{\mathsf{tok}}(\sigma))$$

What if no on-demand consistency?

G*

$$\text{S3. } \forall o, \sigma, \sigma'.\ (\sigma \in I \wedge (\sigma, \sigma') \in (G_0 \cup G((\mathcal{F}_o^{\mathsf{tok}}(\sigma))^{\perp}))^*)$$

$$\implies (\sigma', \mathcal{F}_o^{\mathsf{eff}}(\sigma)(\sigma')) \in G_0 \cup G(\mathcal{F}_o^{\mathsf{tok}}(\sigma))$$

G

What if no causality?

G*

$$\text{S3. } \forall o, \sigma, \sigma'.\ (\sigma \in I \wedge (\sigma, \sigma') \in (G_0 \cup G((\mathcal{F}_o^{\mathsf{tok}}(\sigma))^{\perp}))^{*})$$
$$\implies (\sigma', \mathcal{F}_o^{\mathsf{eff}}(\sigma)(\sigma')) \in G_0 \cup G(\mathcal{F}_o^{\mathsf{tok}}(\sigma))$$

G

What if no causality?

$\sigma' \in \mathcal{I}$

$$\text{S3. } \forall o, \sigma, \sigma'.\ (\sigma \in I \wedge \sout{(\sigma, \sigma') \in (G_0 \cup G((\mathcal{F}_o^{\mathsf{tok}}(\sigma))^{\perp}))^{*}})$$

$$\implies \sout{(\sigma', \mathcal{F}_o^{\mathsf{eff}}(\sigma)(\sigma')) \in G_0 \cup G(\mathcal{F}_o^{\mathsf{tok}}(\sigma))}$$

$\mathcal{F}_o^{\mathsf{eff}}(\sigma)(\sigma') \in \mathcal{I}$