

# 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



The image is a screenshot of the Amazon.com homepage. At the top, a dark blue banner reads "July 15th is Prime Day" with a "Learn More" button. Below this is the Amazon logo with a "Try Prime" link, a search bar, and navigation links for "Shop by Department", "Your Amazon.com", "Today's Deals", "Gift Cards", "Sell", "Help", "Hello. Sign in Your Account", "Try Prime", and "Wish List". The main content area features a large green play button icon and the text "Try Prime Instant Video Free to stream thousands of movies & TV shows". To the right of this text is a grid of movie and TV show covers, including "VEEP", "NOAH", "TRANSPARENCY", "HANNIBAL", "THE SUNDANCE", "BOSCH", "THE WHITE QUEEN", "HUNG", "CAPTIVE", "DEFIANCE", "UNDER THE DOME", and "CATCHING FIRE". Below the main banner, a section titled "Movies Included with Prime Membership at No Additional Cost" includes a "See more" link and four movie covers: "TRANSFORMERS: AGE OF EXTINCTION", "THE WORDS", "LITTLE ACCIDENTS", and "CATCHING FIRE". On the right side of the page, there is an orange sidebar with the text "Instantly watch movies & TV", the Amazon Prime logo, a "Start Free Trial" button, and the text "Watch as much as you want. Anytime you want."

July 15th is Prime Day [Learn More](#)

amazon [Try Prime](#)

Shop by Department ▾ Your Amazon.com Today's Deals Gift Cards Sell Help Hello. Sign in [Your Account](#) ▾ [Try Prime](#) ▾ [Wish List](#) ▾

Shop Beautiful Things

Try Prime Instant Video Free  
to stream thousands of movies & TV shows

Movies Included with Prime Membership at No Additional Cost  
[See more](#)

TRANSFORMERS: AGE OF EXTINCTION

THE WORDS

LITTLE ACCIDENTS

CATCHING FIRE

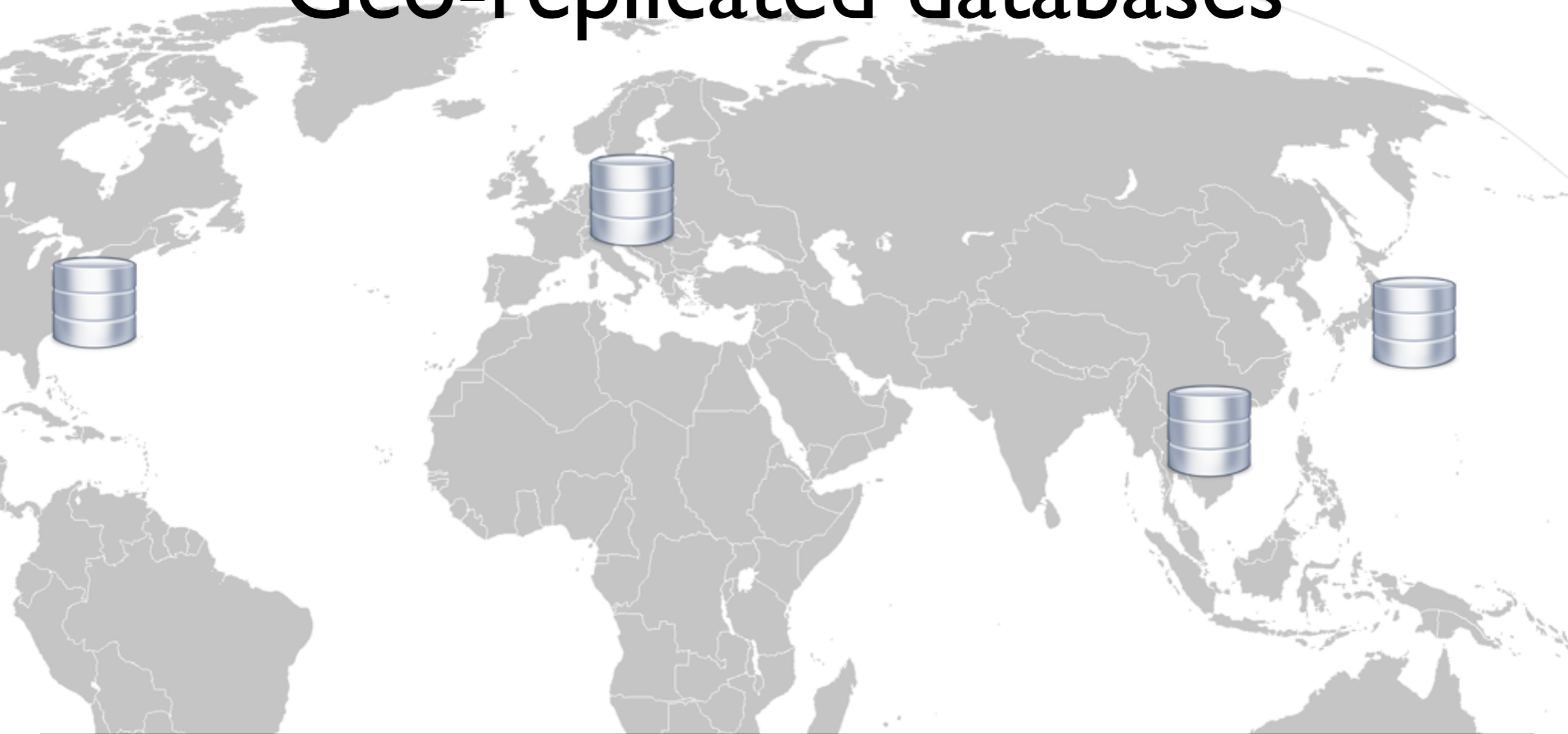
Instantly watch movies & TV

amazon [Prime](#)

[Start Free Trial](#)

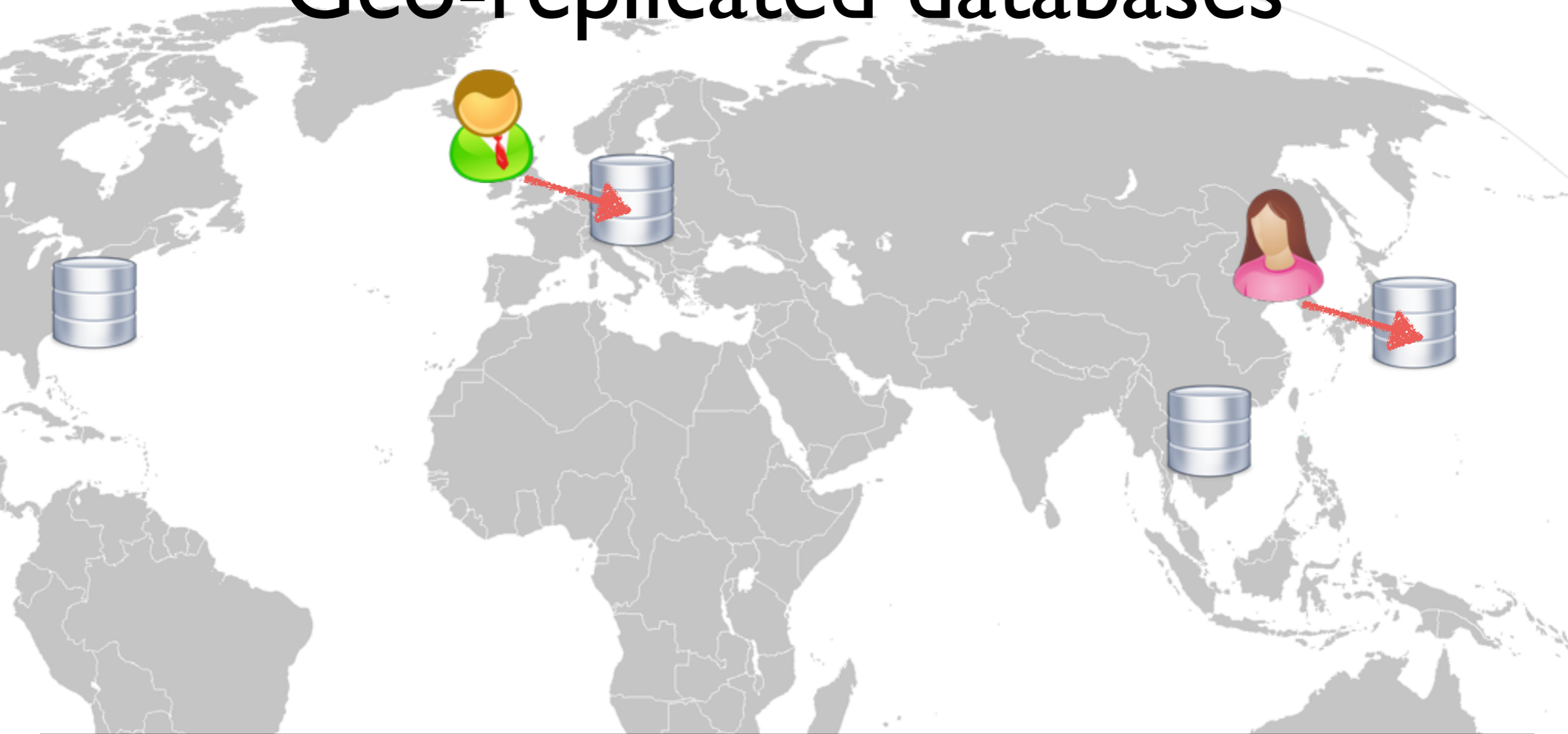
Watch as much as you want.  
Anytime you want.

# Geo-replicated databases



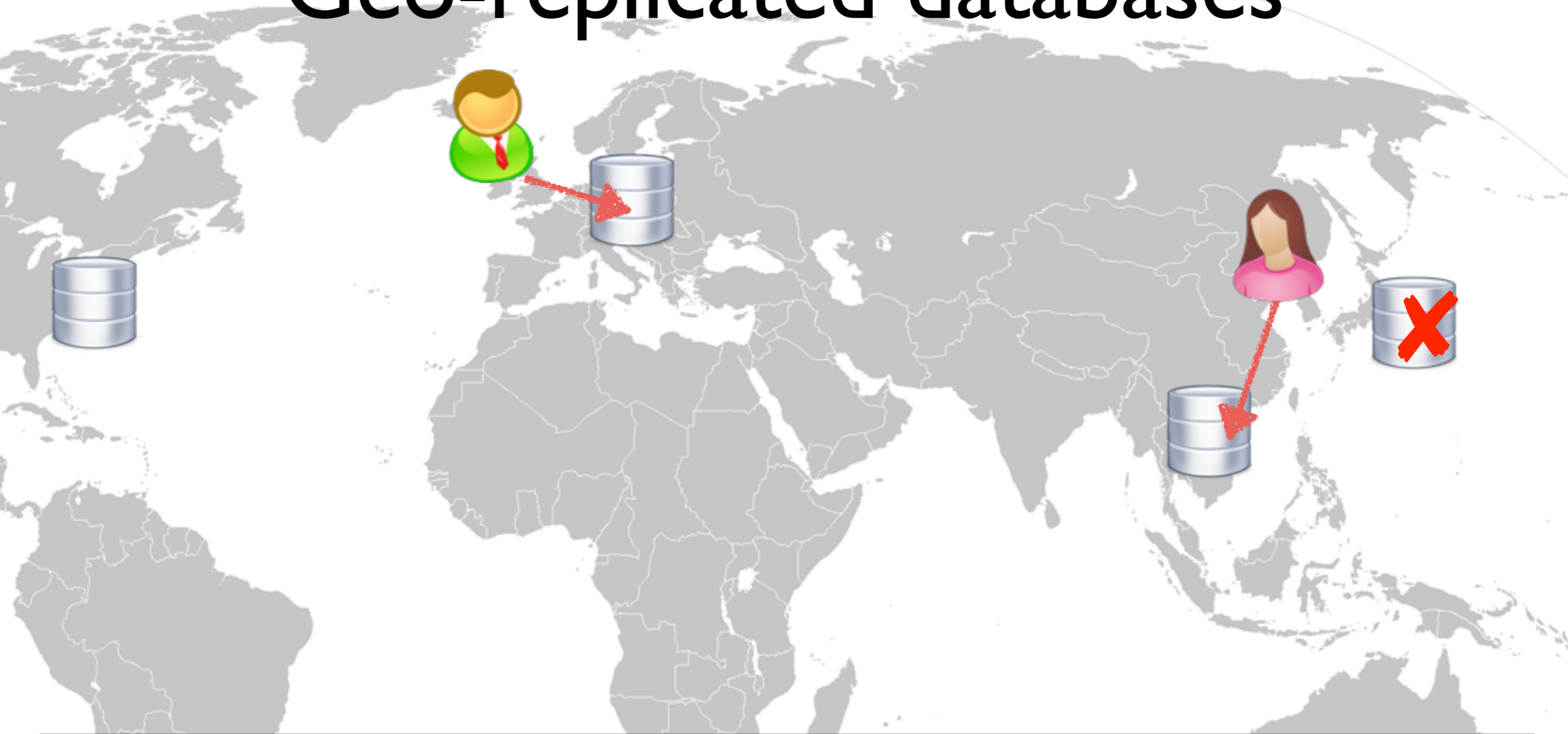
- Every data centre stores a complete replica of data
- Purpose: Minimising latency. Fault tolerance.

# Geo-replicated databases



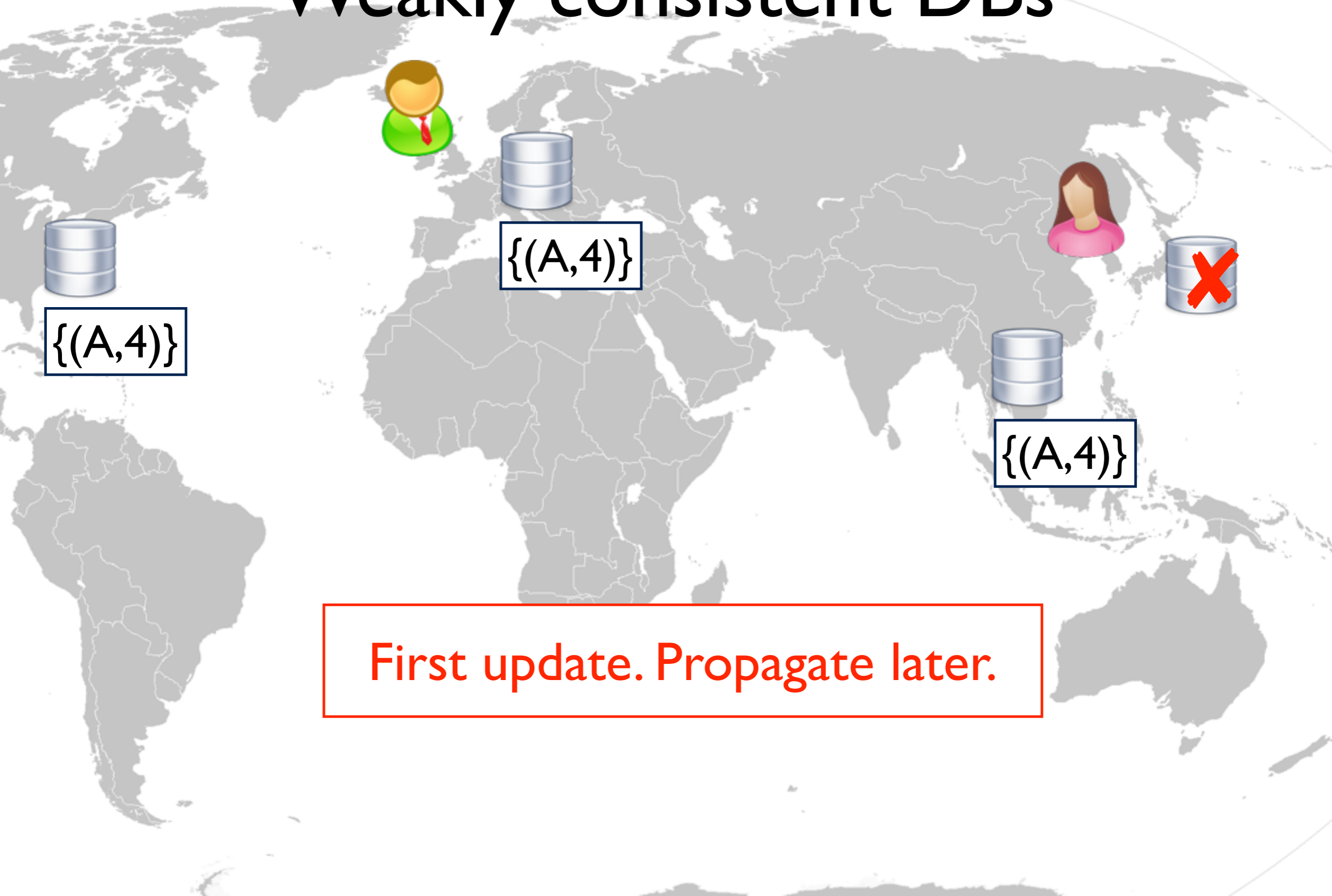
- Every data centre stores a complete replica of data
- Purpose: **Minimising latency.** Fault tolerance.

# Geo-replicated databases

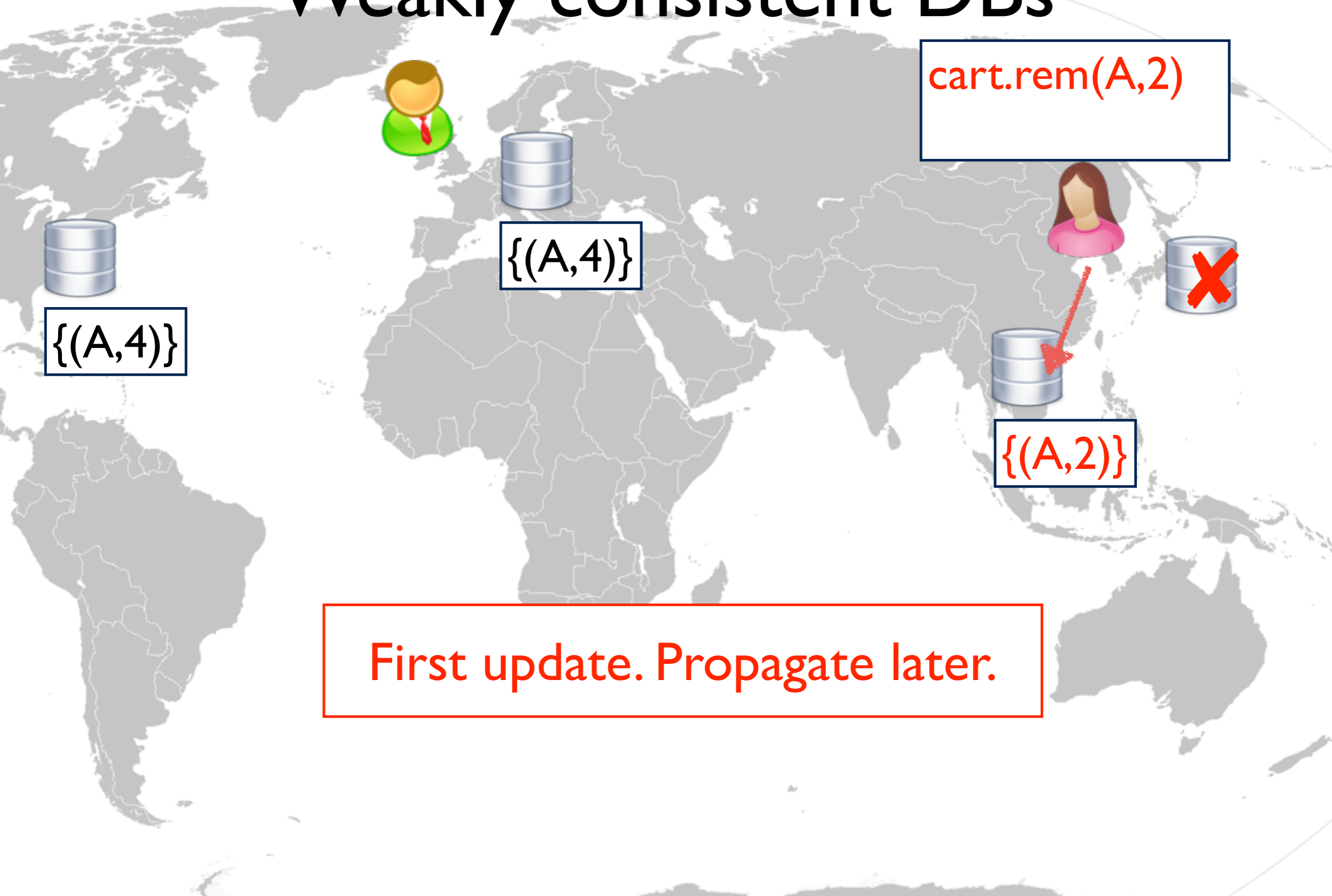


- Every data centre stores a complete replica of data
- Purpose: Minimising latency. **Fault tolerance.**

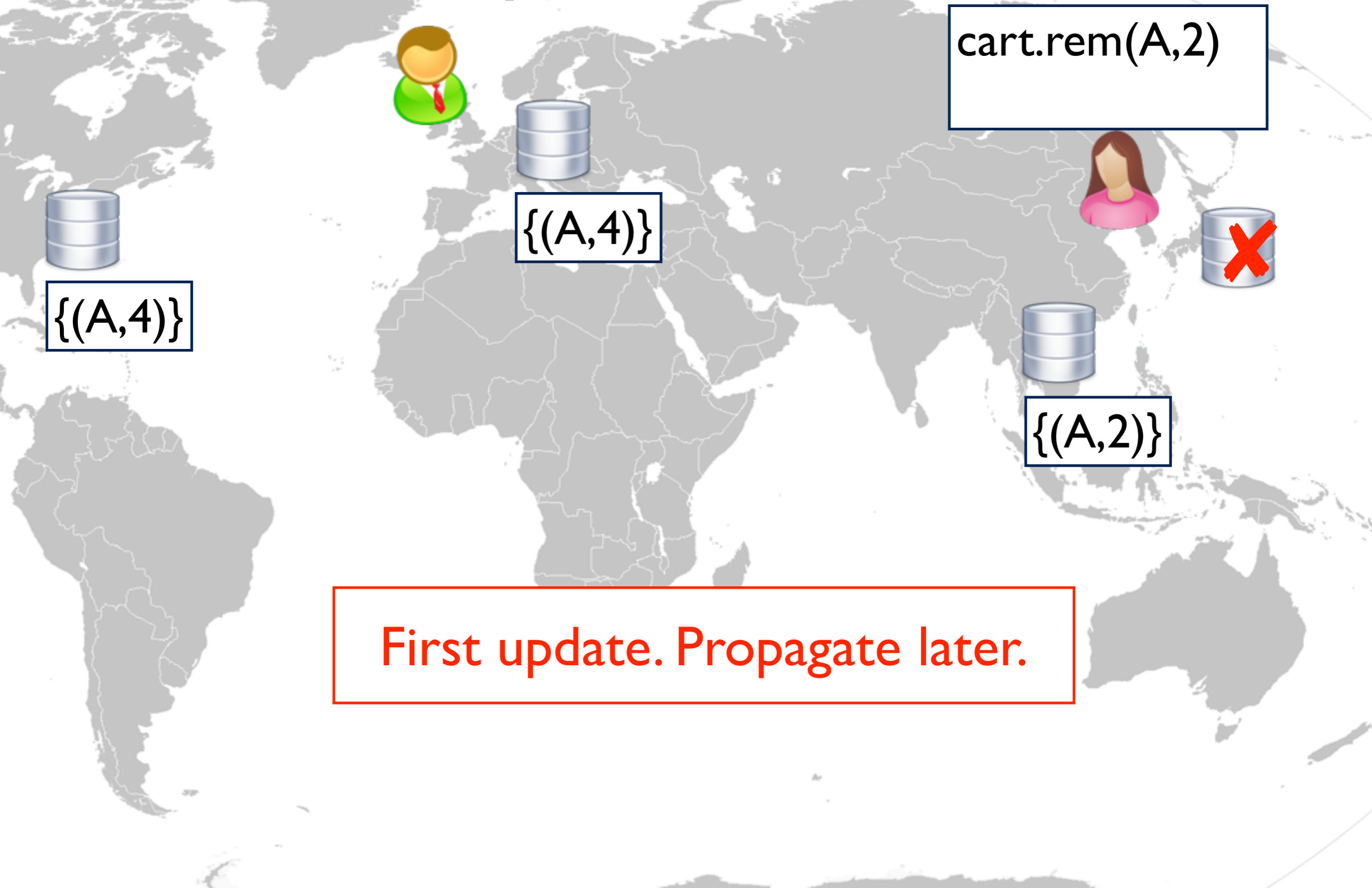
# Weakly consistent DBs



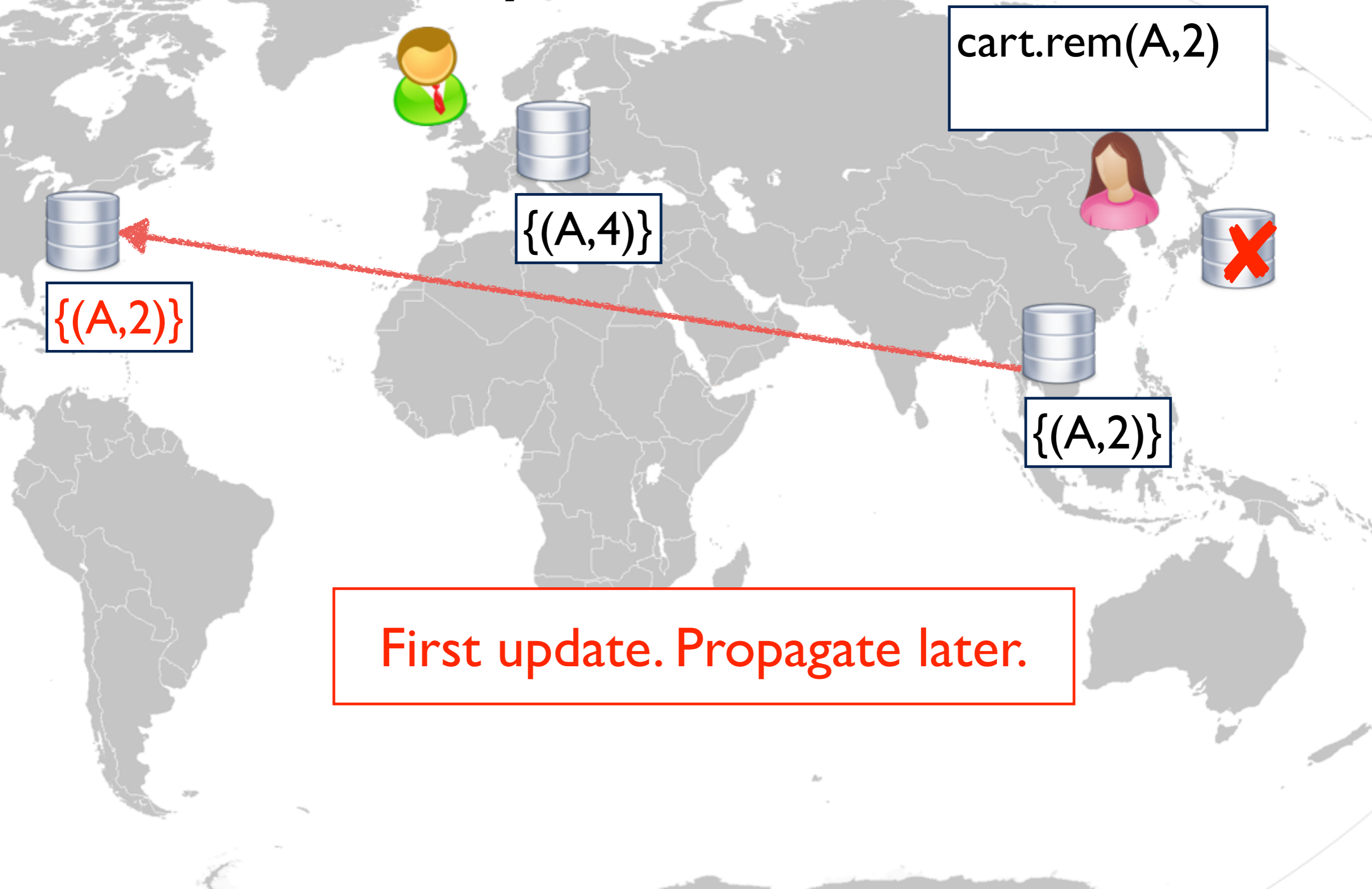
# Weakly consistent DBs



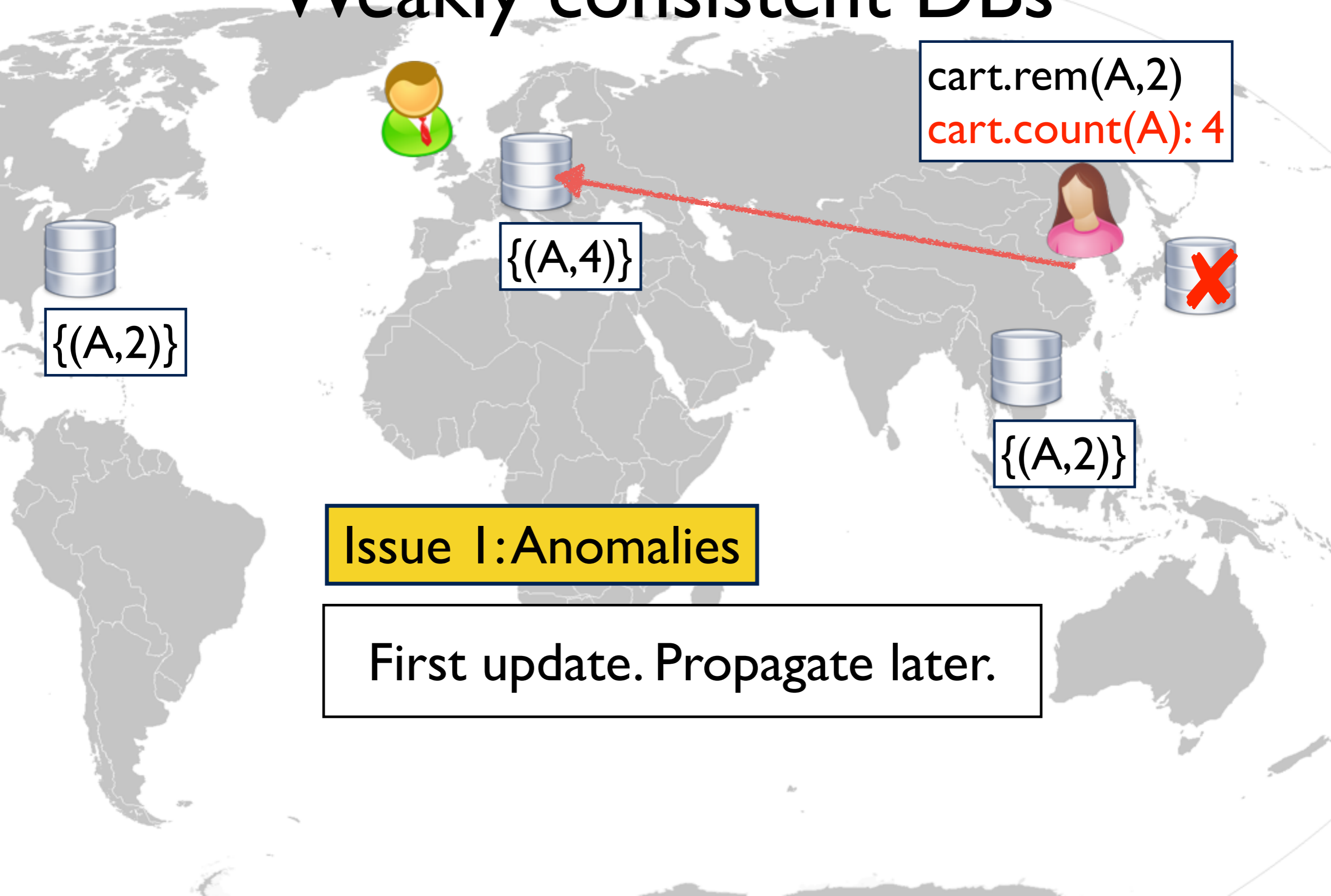
# Weakly consistent DBs



# Weakly consistent DBs



# Weakly consistent DBs



`cart.rem(A,2)`  
`cart.count(A): 4`

`{(A,4)}`

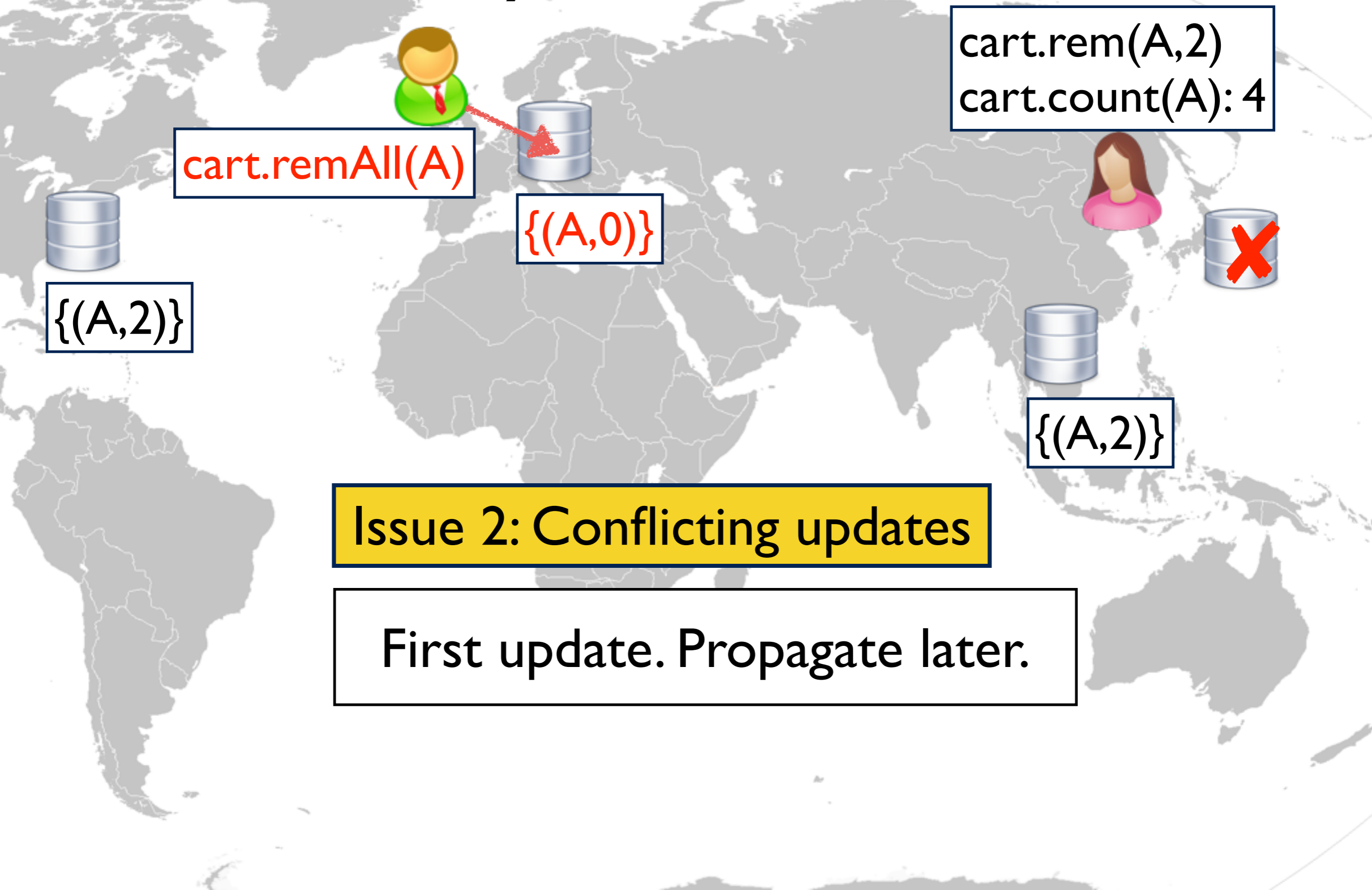
`{(A,2)}`

`{(A,2)}`

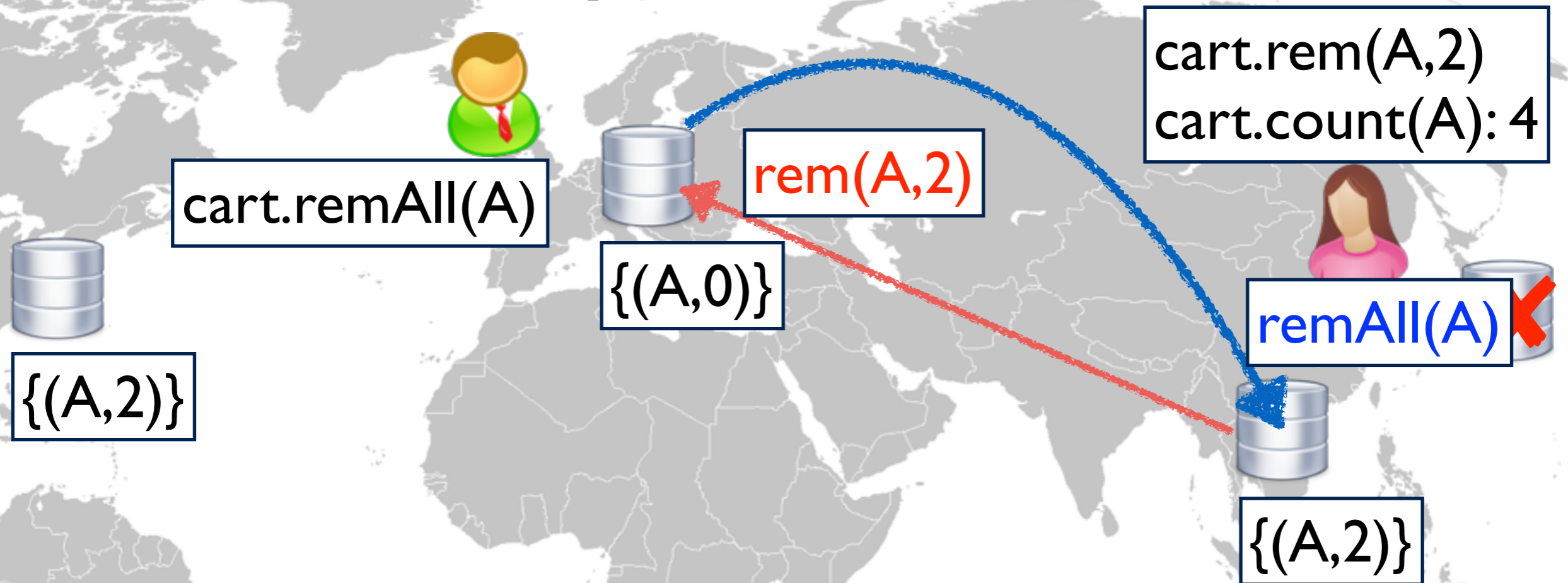
**Issue 1: Anomalies**

First update. Propagate later.

# Weakly consistent DBs



# 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?

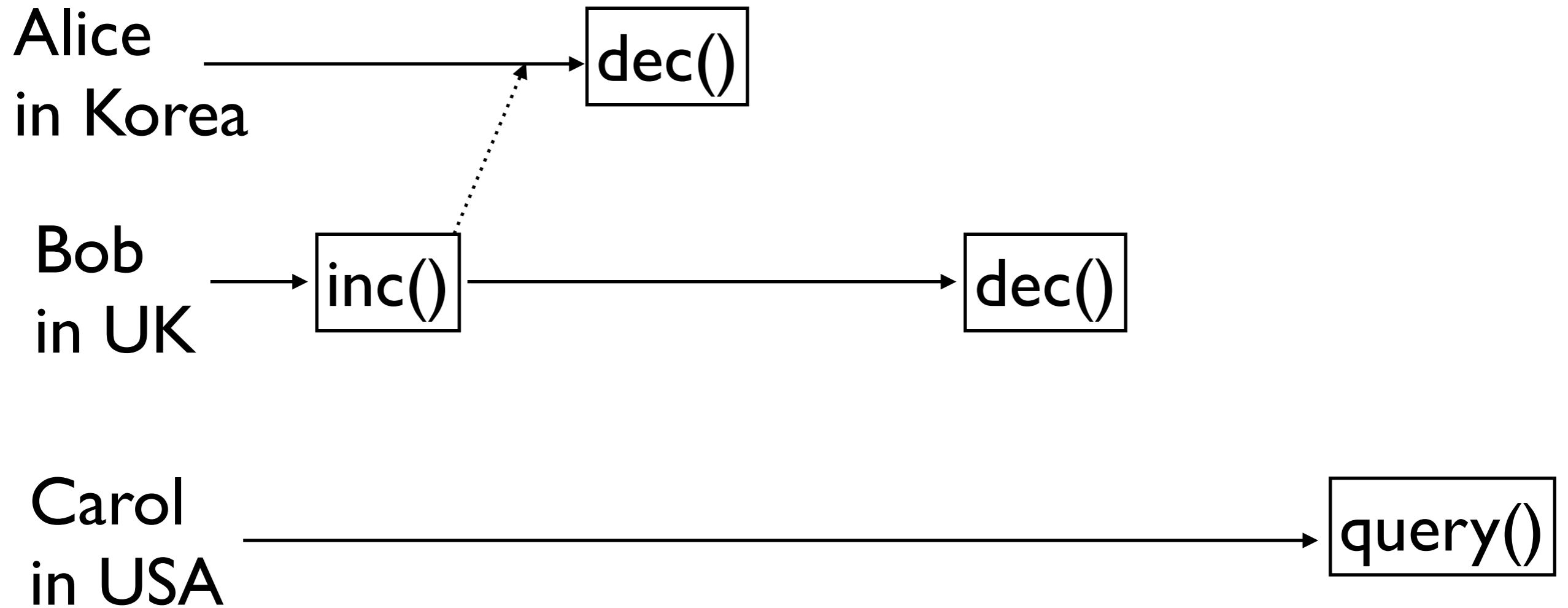
1. Strengthen consistency selectively.
2. **Prove the correctness of a program.**

# Simple bank account

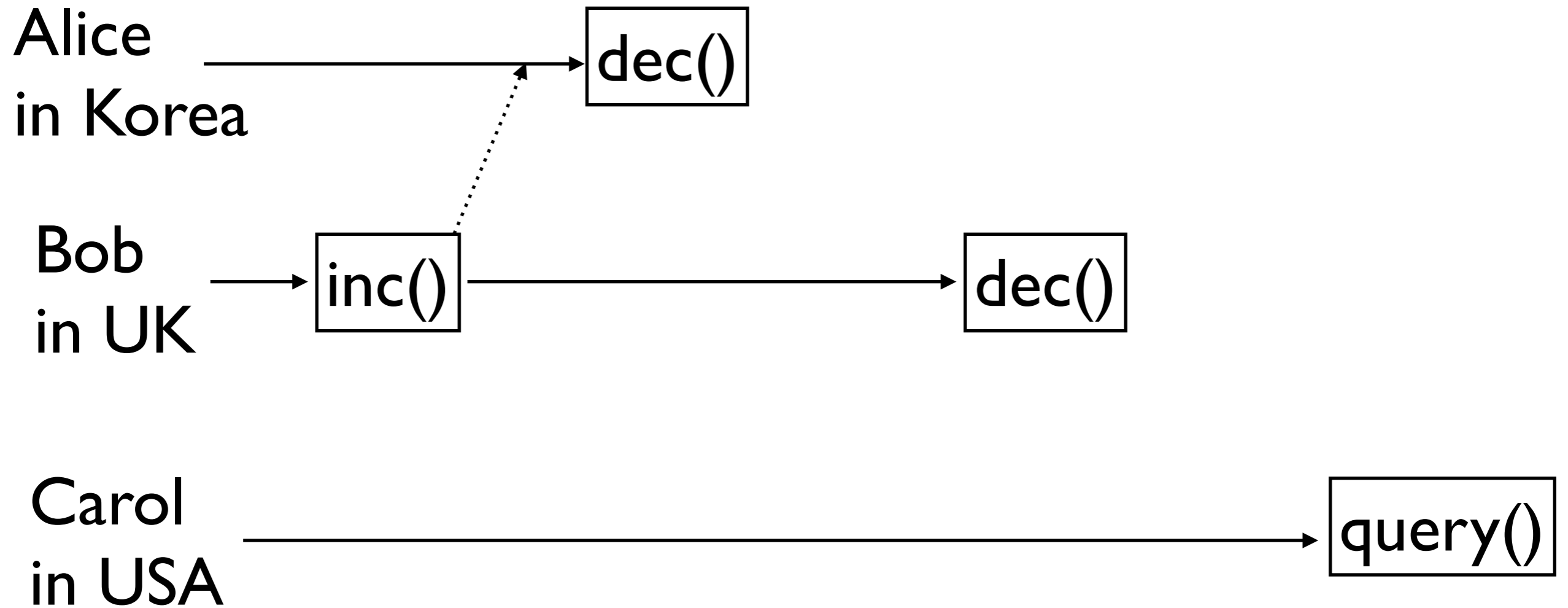
```
class account {  
  // invariant: amount >= 0  
  var amount = 0  
  
  def query() = { return amount }  
  
  def inc() = {  
    amount = amount+1; return true  
  }  
  
  def dec() = {  
    if (amount > 0) {  
      amount = amount-1; return true  
    }  
    else { return false }  
  }  
}
```

# Distributed bank account

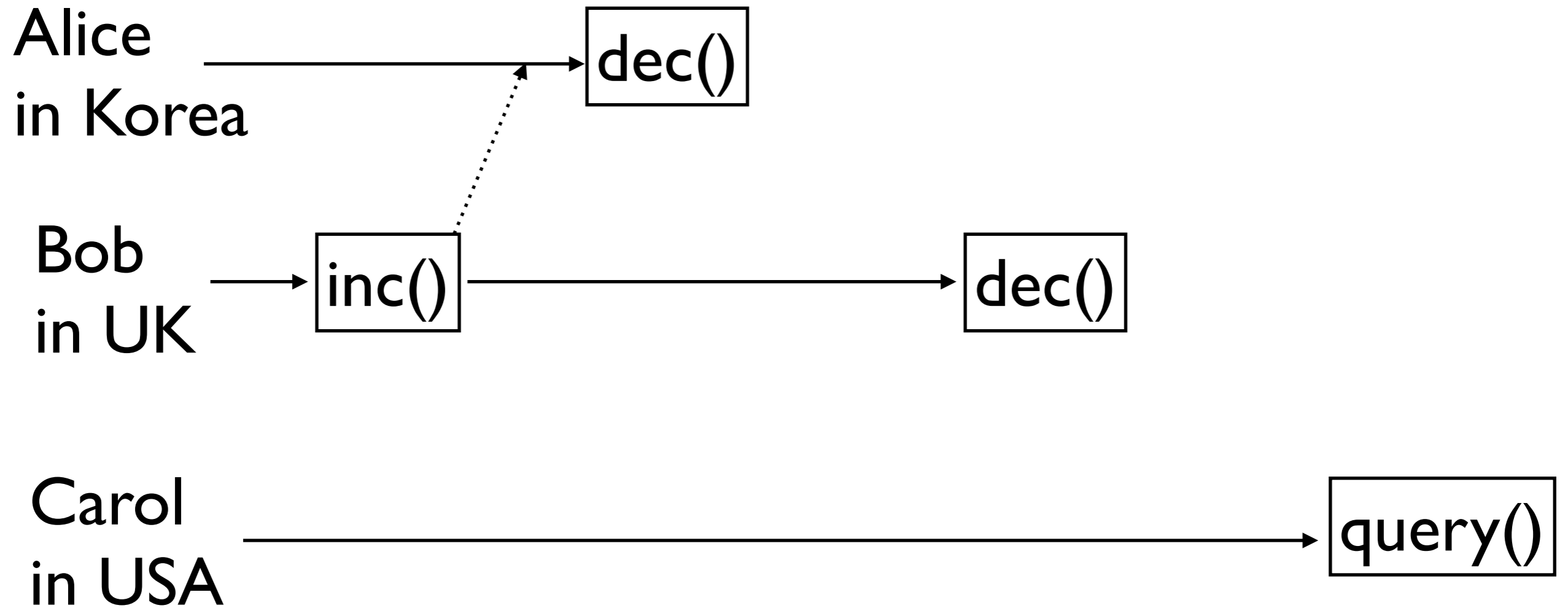
```
class account {  
  // invariant: amount >= 0  
  var[dis] amount = 0  
  
  def query() = { return (amount, (a)=>a) }  
  
  def inc() = {  
    amount = amount+1; return (true, (a)=>a+1)  
  }  
  
  def dec() = {  
    if (amount > 0) {  
      amount = amount-1; return (true, (a)=>a-1)  
    }  
    else { return (false, (a)=>a) }  
  }  
}
```



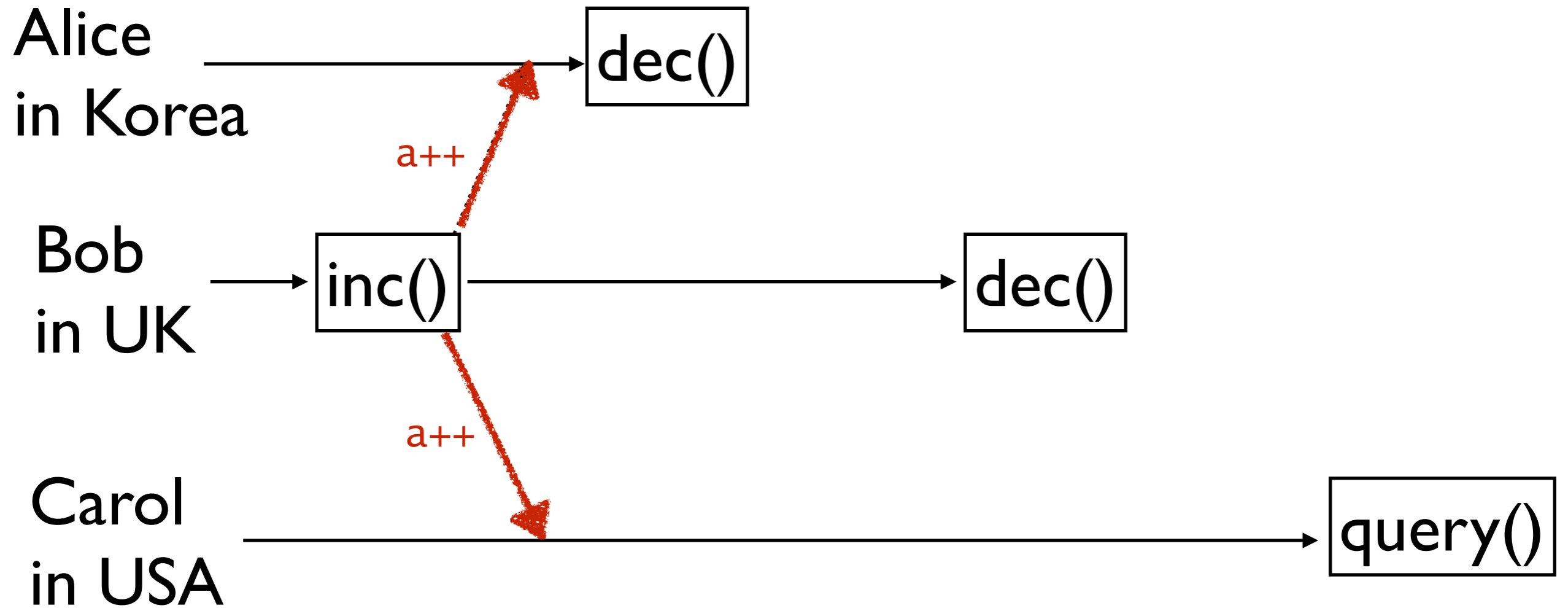
**[Q] What cannot be the result of query()?**  
(a) 0      (b) -1      (c) -2      (d) all possible



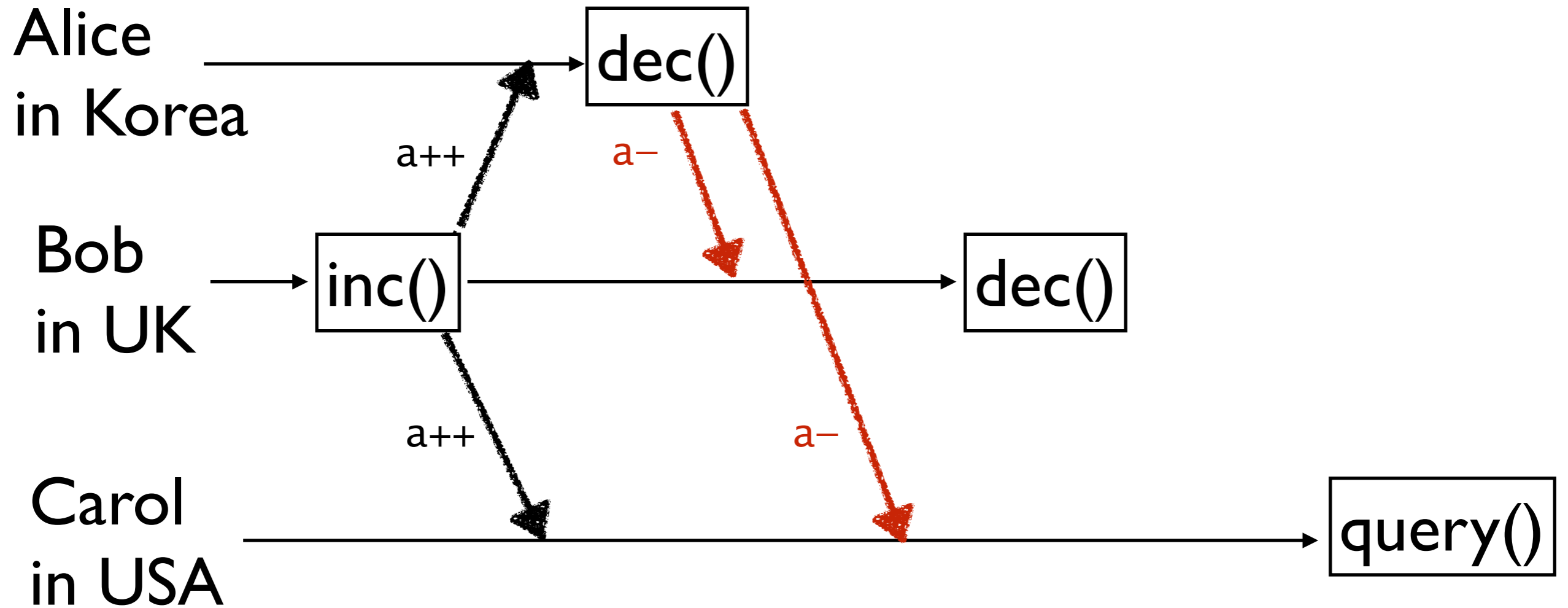
[Q] What cannot be the result of `query()`?  
(a) 0      (b) -1      (c) -2      (d) all possible



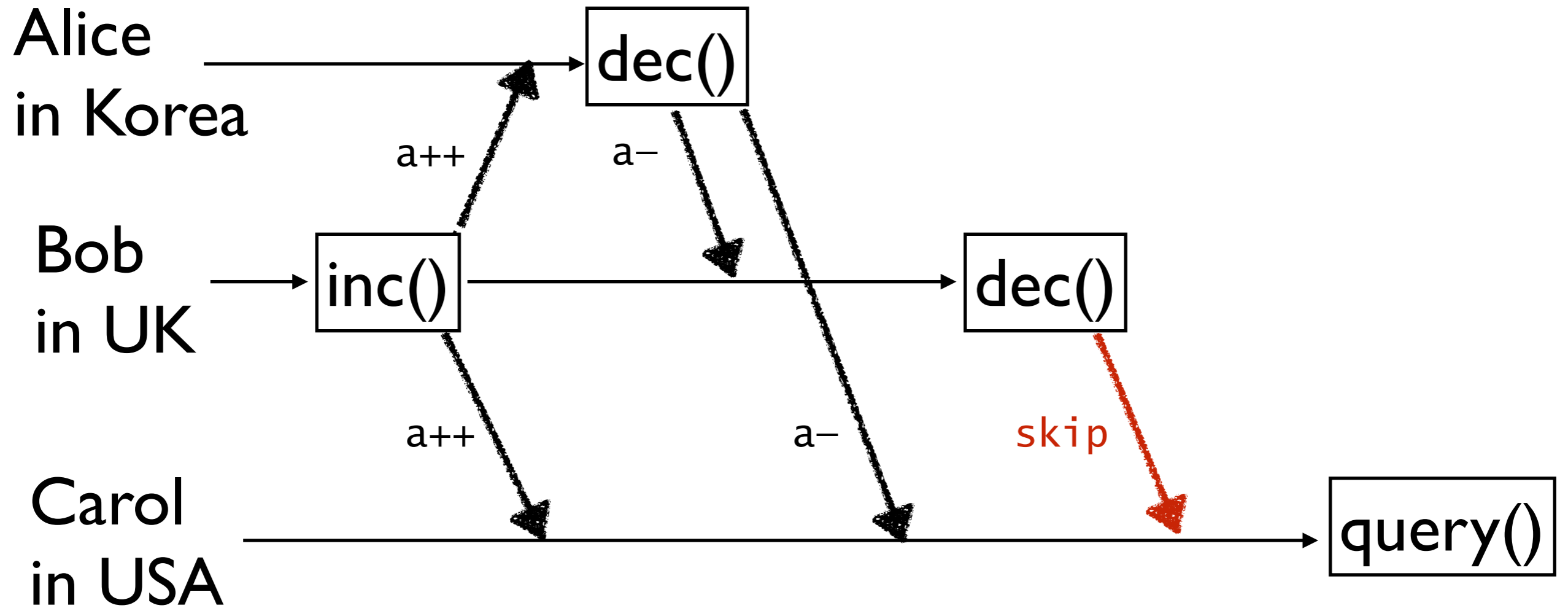
[Q] What cannot be the result of `query()`?  
(a) 0      (b) -1      (c) -2      (d) all possible



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

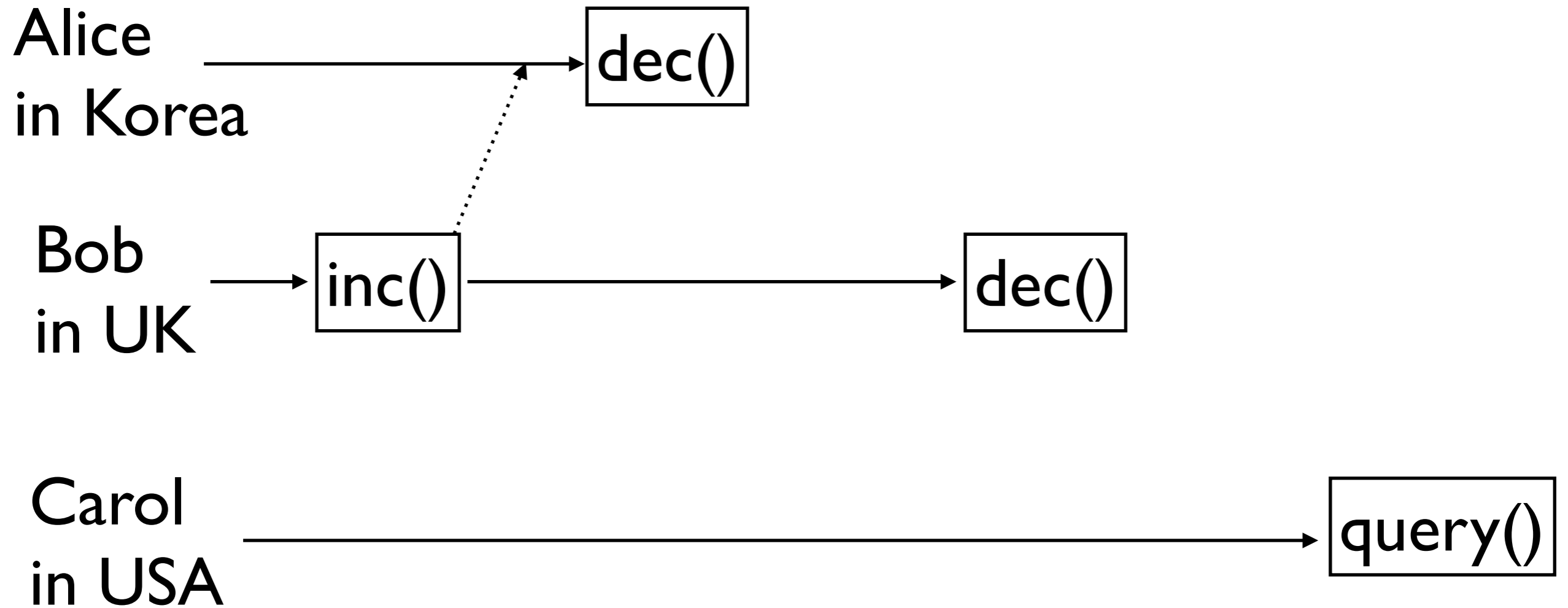


[Q] What cannot be the result of `query()`?  
(a) 0      (b) -1      (c) -2      (d) all possible

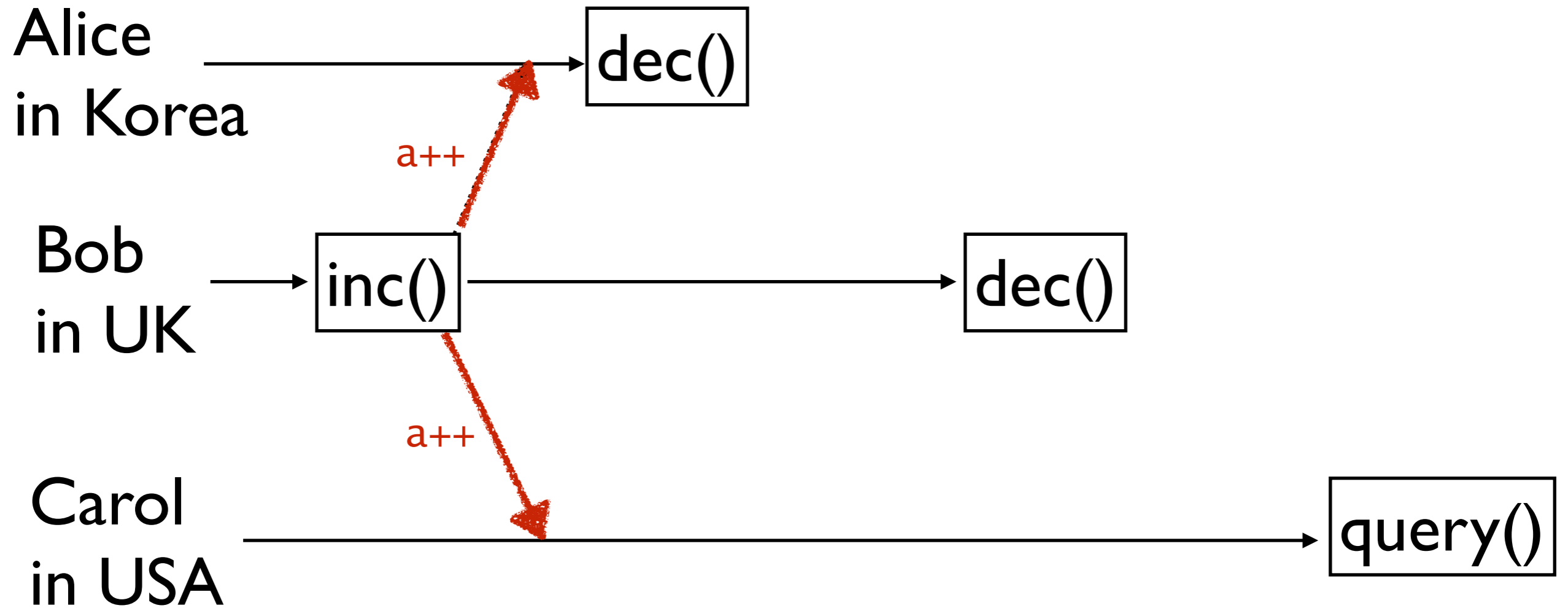


[Q] What cannot be the result of query()?

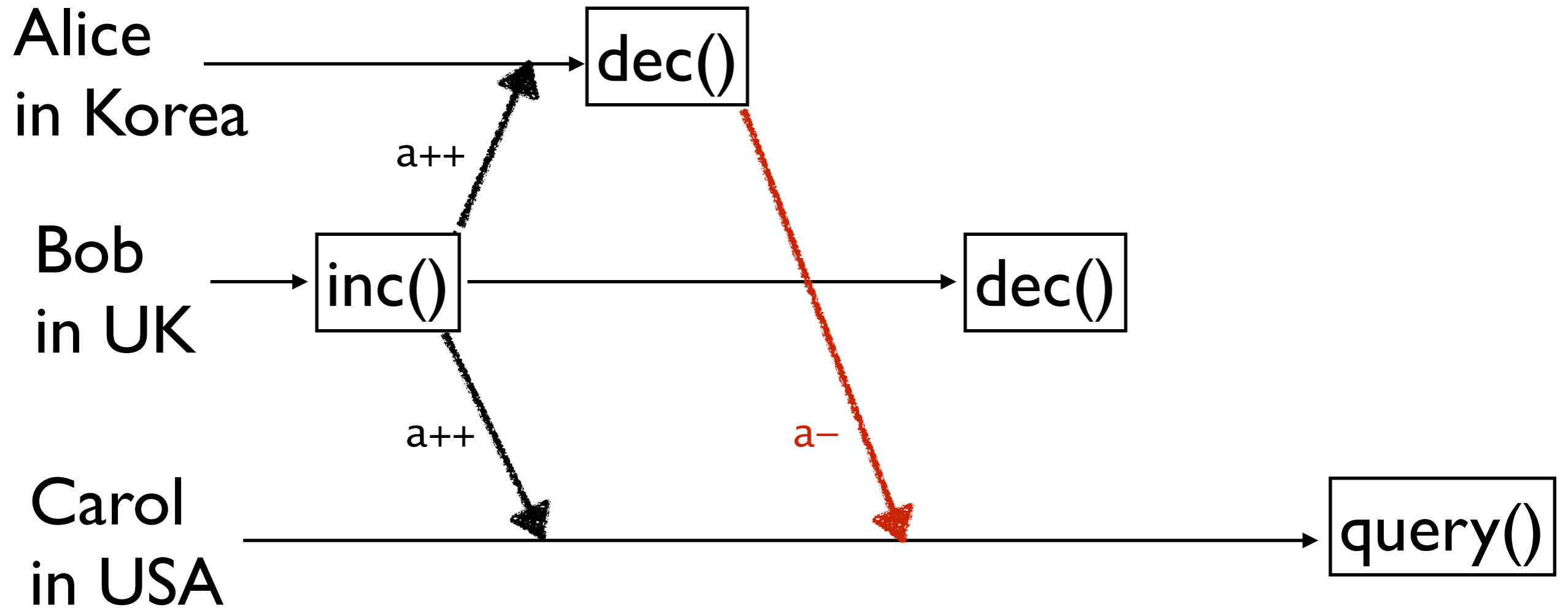
(a) 0      (b) -1      (c) -2      (d) all possible



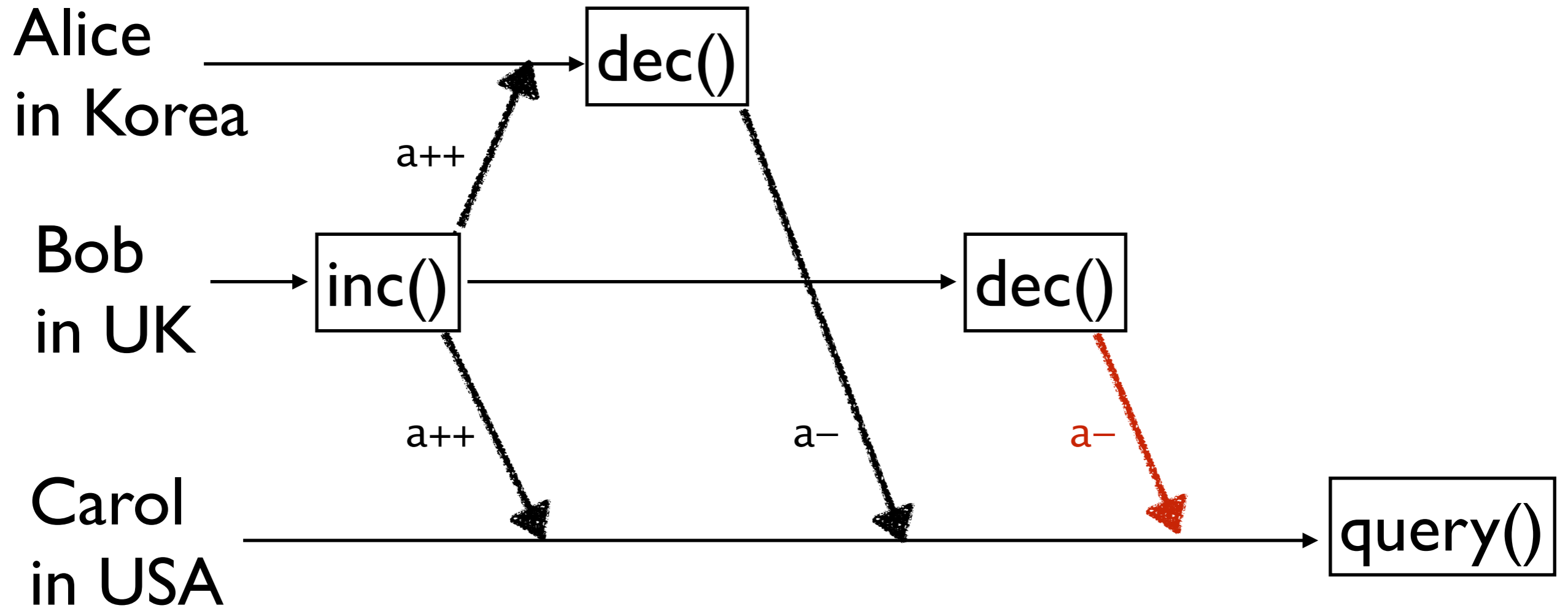
[Q] What cannot be the result of query()?  
(a) 0      (b) -1      (c) -2      (d) all possible



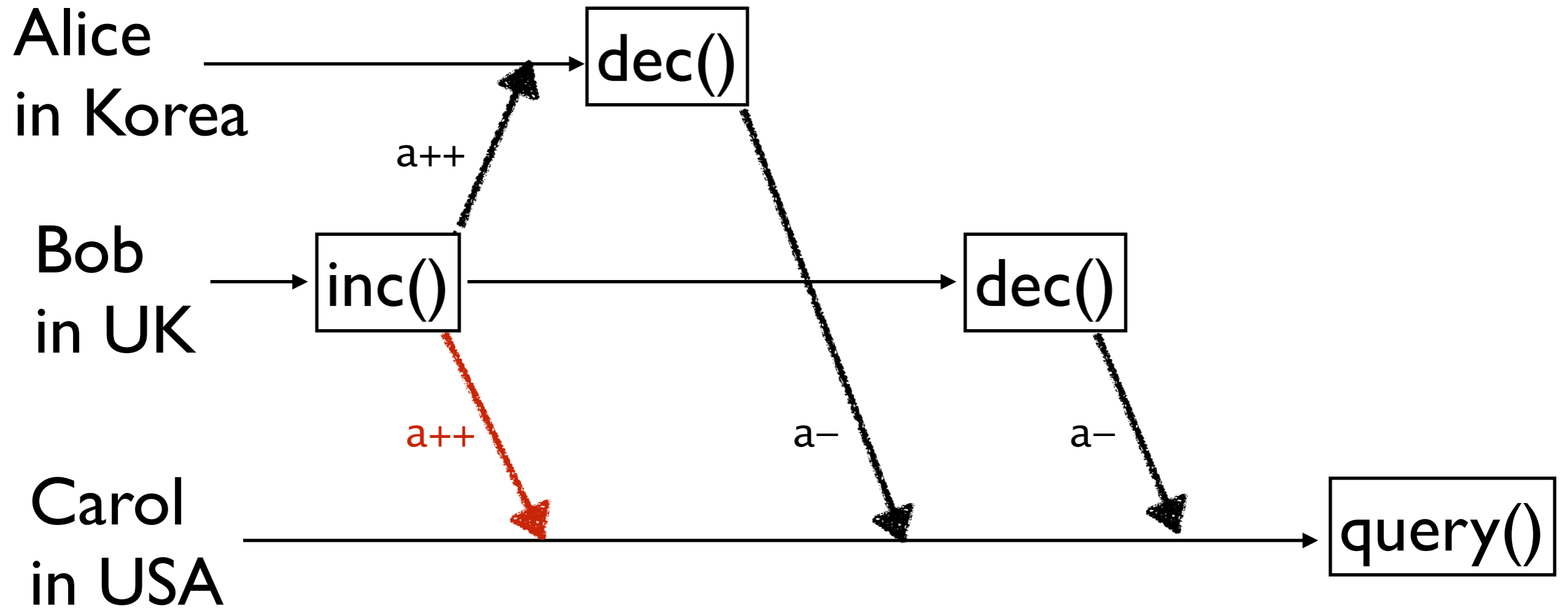
[Q] What cannot be the result of `query()`?  
(a) 0      (b) -1      (c) -2      (d) all possible



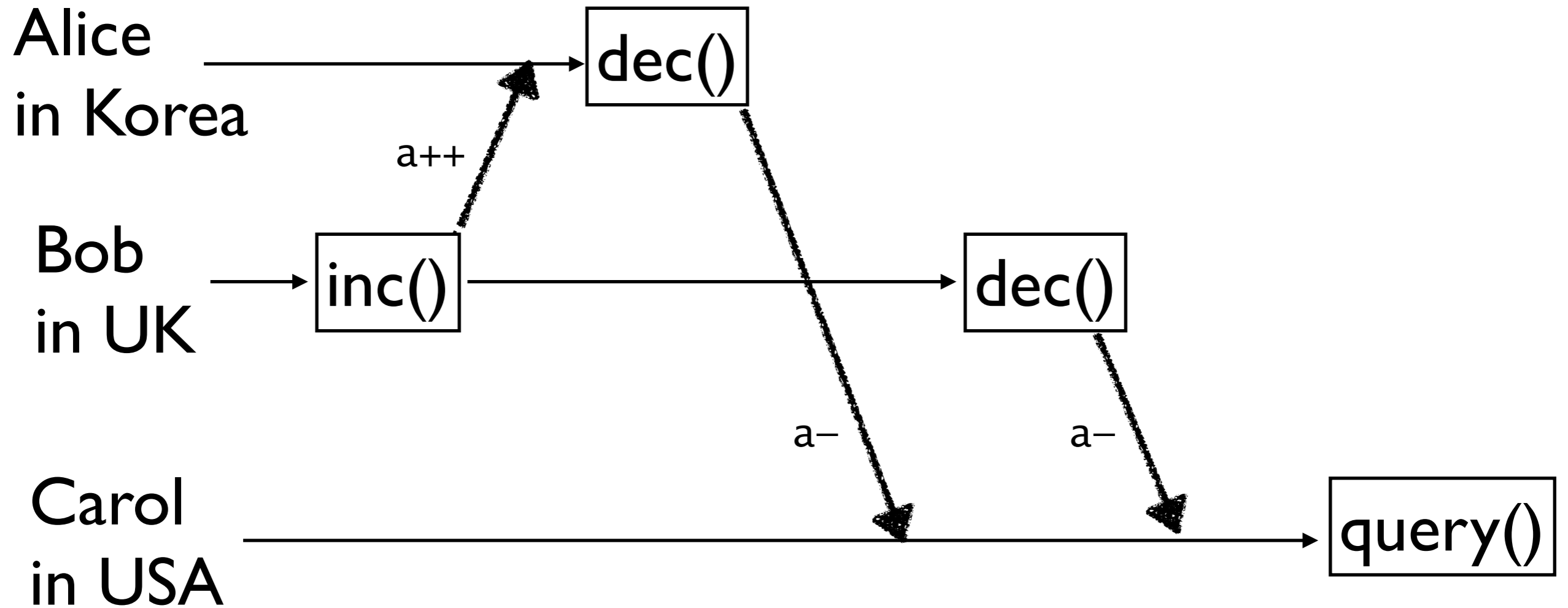
[Q] What cannot be the result of `query()`?  
(a) 0      **(b) -1**      (c) -2      (d) all possible



[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

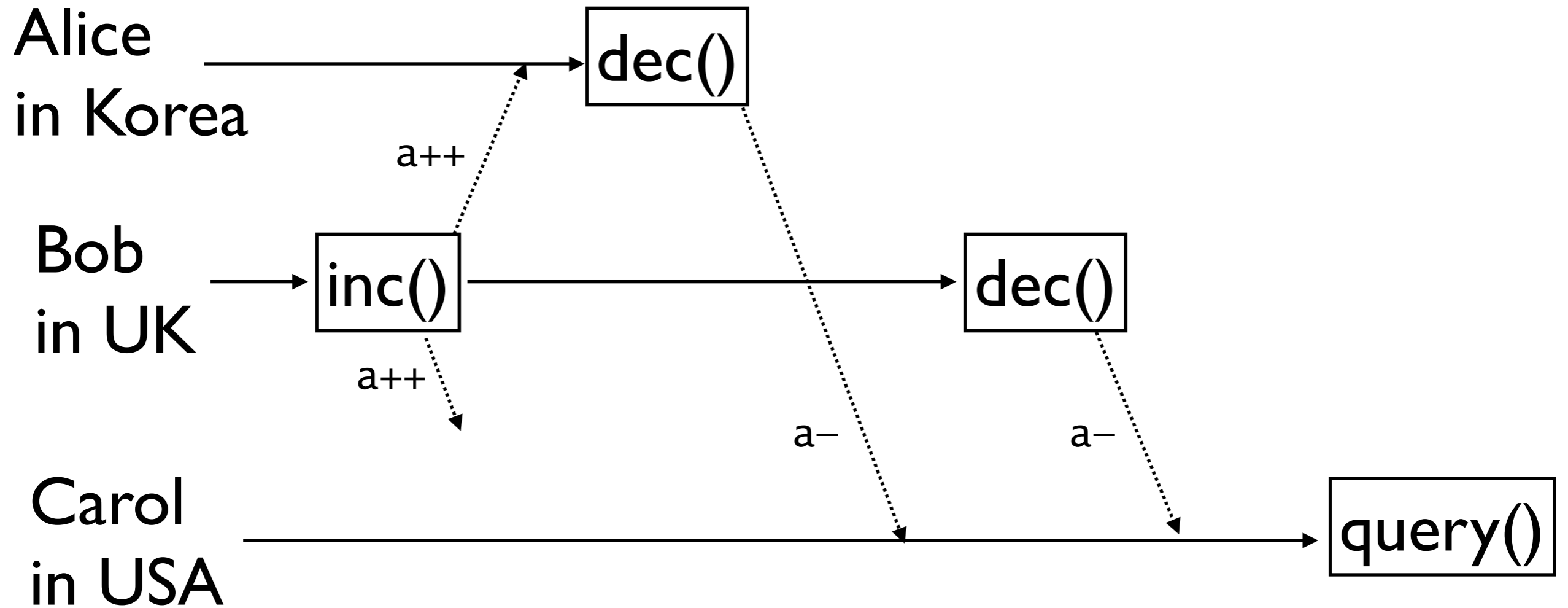
# How to write correct prog.?

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

# Causal consistency

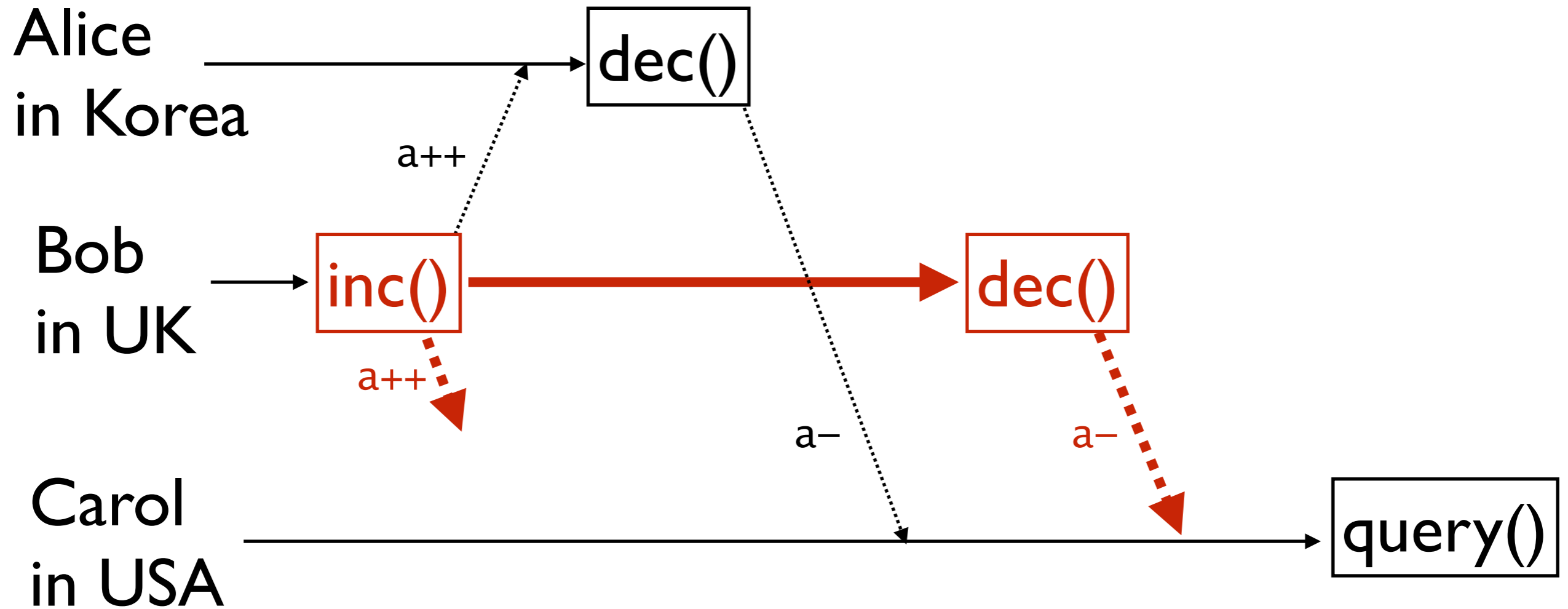
- Message delivery preserves the dependency of events.

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

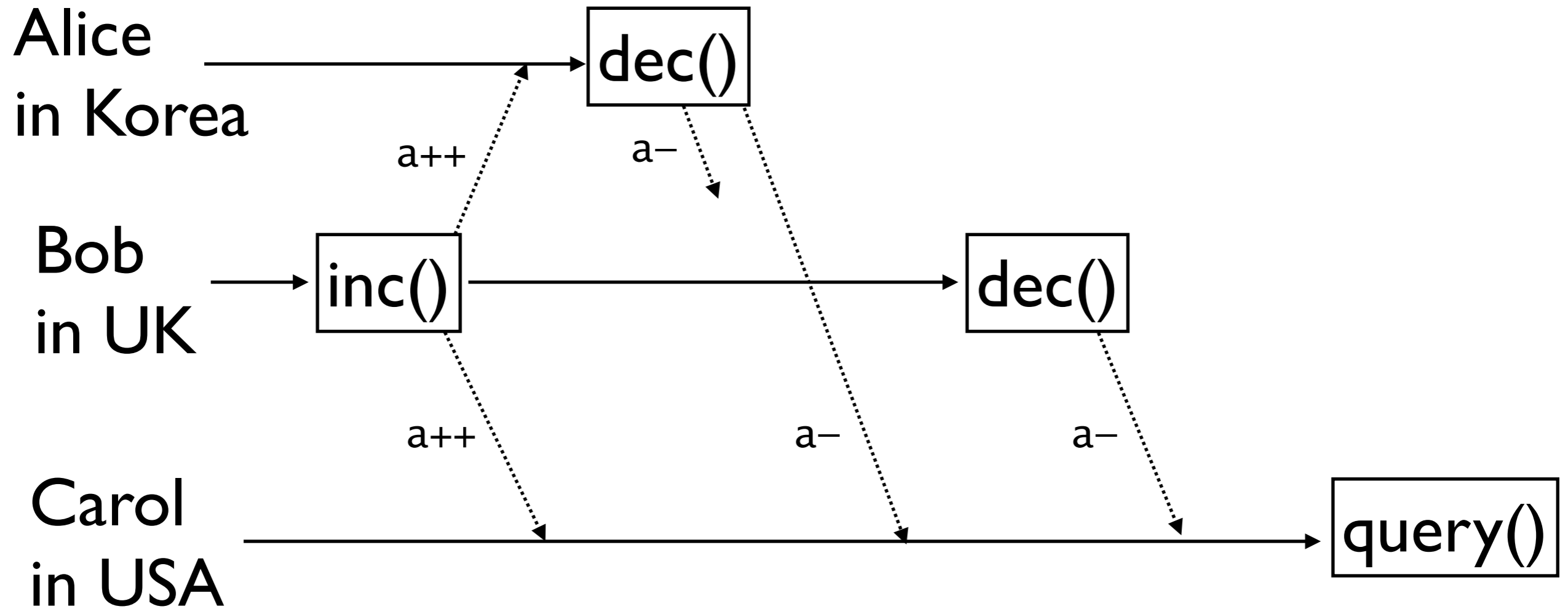
```
class account {  
  // invariant: amount >= 0  
  var[dis] amount = 0  
  
  def query() = { return (amount, (a)=>a) }  
  
  def inc() = {  
    amount = amount+1; return (true, (a)=>a+1)  
  }  
  
  def dec() = {  
    if (amount > 0) {  
      amount = amount-1; return (true, (a)=>a-1)  
    }  
    else { return (false, (a)=>a) }  
  }  
}
```

# Token system

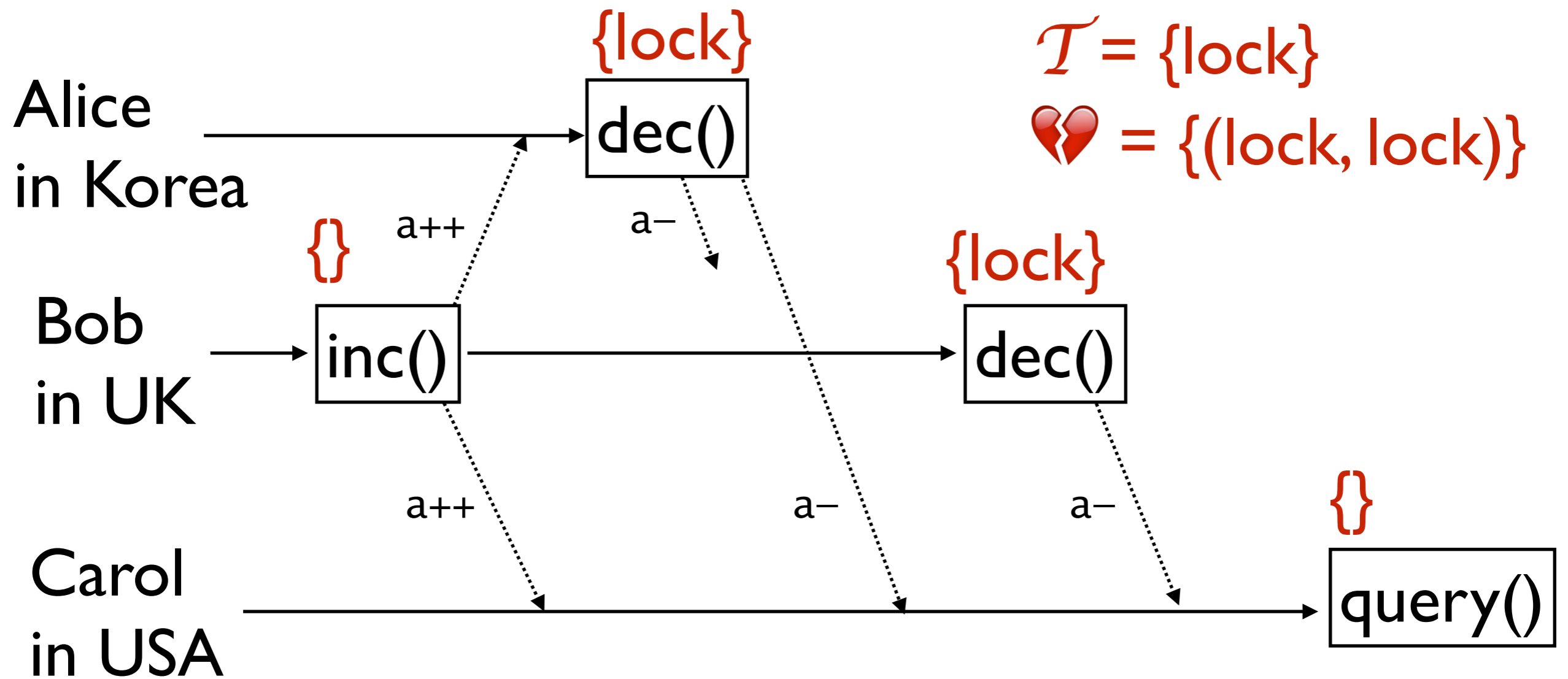
- $(\mathcal{T}, \heartsuit)$  where  $\heartsuit$  is a symmetric rel. on  $\mathcal{T}$ .
- Examples:
  1.  $\mathcal{T} = \{\text{lock}\}$ ,  $\heartsuit = \{(\text{lock}, \text{lock})\}$
  2.  $\mathcal{T} = \{\text{rd}, \text{wr}\}$ ,  $\heartsuit = \{(\text{rd}, \text{wr}), (\text{wr}, \text{wr}), (\text{wr}, \text{rd})\}$

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

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

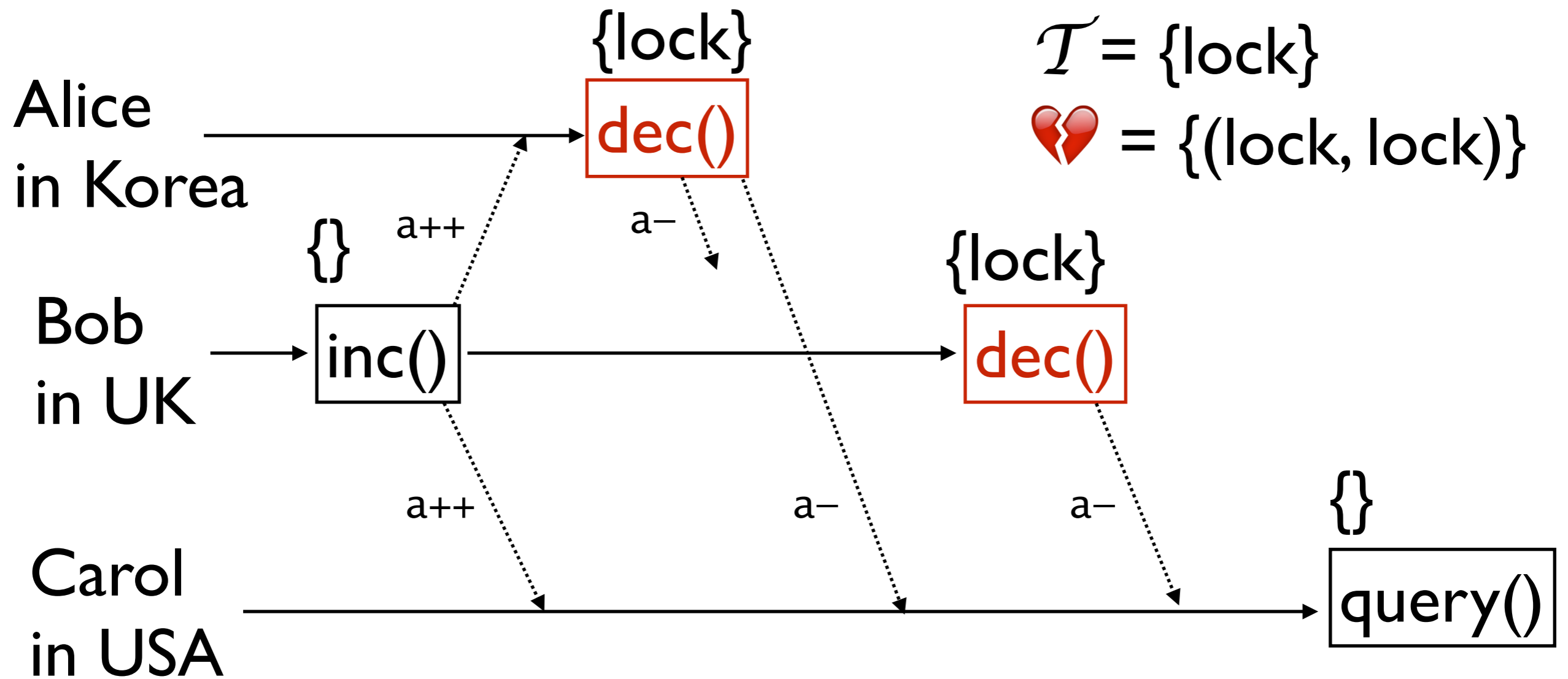


[Q] What cannot be the result of query()?  
(a) 0      **(b) -1**      (c) -2      (d) all possible



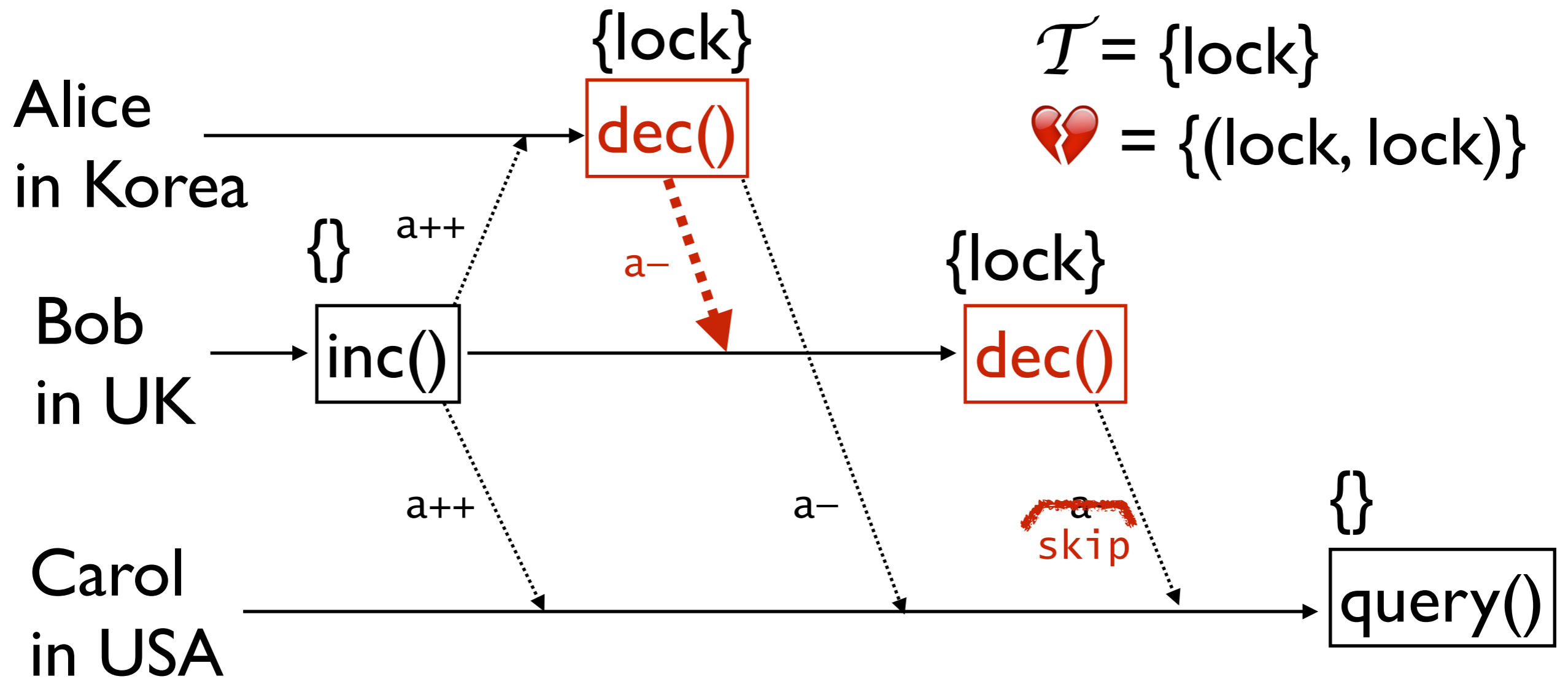
[Q] What cannot be the result of query()?

(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  $\mathcal{I}$  is an invariant

To prove that  $I$  is an invariant

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

To prove that  $I$  is an invariant

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

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

# To prove that $I$ is an invariant

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

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

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

# To prove that $I$ is an invariant

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

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

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

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

To prove that  $I$  is an invariant

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

To prove that  $I$  is an invariant

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

To prove that  $I$  is an invariant

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

To prove that  $I$  is an invariant

$$\begin{aligned} \text{S3. } \forall o, \sigma, \sigma'. (\sigma \in I \wedge (\sigma, \sigma') \in ( \dots )^*) \\ \implies (\sigma', \mathcal{F}_o^{\text{eff}}(\sigma)(\sigma')) \in \dots \end{aligned}$$

To prove that  $I$  is an invariant

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

To prove that  $I$  is an invariant

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

To prove that  $I$  is an invariant

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

$$T^\perp = \{\tau \mid \nexists \tau' \in T. (\tau, \tau') \in \heartsuit\}$$

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

$G_0^*$

$$\begin{array}{l}
 \text{S3. } \forall o, \sigma, \sigma'. (\sigma \in I \wedge (\sigma, \sigma') \in \{ \} (G_0 \cup G((\mathcal{F}_o^{\text{tok}}(\sigma))^\perp))^*) \\
 \text{dec()} \quad \implies (\sigma', \mathcal{F}_o^{\text{eff}}(\sigma)(\sigma')) \in G_0 \cup G(\mathcal{F}_o^{\text{tok}}(\sigma)) \{ \text{lock} \} \\
 \qquad \qquad \qquad \qquad \qquad \qquad \qquad \qquad \qquad \qquad \qquad \qquad \qquad G_0 \cup G_1(\text{lock})
 \end{array}$$

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

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

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

$$\begin{array}{lcl}
 S3. \forall \sigma, \sigma'. (\sigma \in I \wedge (\sigma, \sigma') \in & \boxed{\phantom{G_0^*}} & G_0^* \\
 \text{dec}() & \implies (\sigma', \mathcal{F}_o^{\text{eff}}(\sigma)(\sigma')) \in & \boxed{\phantom{G_0 \cup G_1(\text{lock})}} \\
 & & G_0 \cup G_1(\text{lock})
 \end{array}$$

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

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

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

$$S3. \forall \underline{0}, \sigma, \sigma'. (\sigma \in I \wedge (\sigma, \sigma') \in \boxed{\phantom{G_0^*}} \wedge \text{dec}() \implies (\sigma', \mathcal{F}_o^{\text{eff}}(\sigma)(\sigma')) \in \boxed{\phantom{G_0 \cup G_1(\text{lock})}})$$

$\boxed{\text{if } 0 < \sigma \text{ then } \sigma' - 1 \text{ else } \sigma'}$

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

# What if no on-demand consistency?

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

# What if no on-demand consistency?

$$\begin{aligned}
 S3. \forall o, \sigma, \sigma'. (\sigma \in I \wedge (\sigma, \sigma') \in & \overbrace{(G_0 \cup G((\mathcal{F}_o^{\text{tok}}(\sigma))^\perp))^*}^{G^*}) \\
 \implies (\sigma', \mathcal{F}_o^{\text{eff}}(\sigma)(\sigma')) \in & \underbrace{G_0 \cup G(\mathcal{F}_o^{\text{tok}}(\sigma))}_{G})
 \end{aligned}$$

# What if no causality?

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

# What if no causality?

$$\sigma' \in I$$

$$\begin{aligned}
 S3. \forall o, \sigma, \sigma'. (\sigma \in I \wedge & \cancel{(o, \sigma') \in (G_o \cup G((\mathcal{F}_o^{\text{tok}}(o))^\perp))^*}) \\
 \implies & \cancel{(\sigma', \mathcal{F}_o^{\text{eff}}(\sigma)(\sigma')) \in G_o \cup G(\mathcal{F}_o^{\text{tok}}(\sigma))} \\
 & \mathcal{F}_o^{\text{eff}}(\sigma)(\sigma') \in I
 \end{aligned}$$