
OWASP 2024
GLOBAL
AppSec
CONGRESS CENTRE
LISBON
JUNE 24-28

OWASP 2024 GLOBAL AppSec
CONGRESS CENTRE LISBON JUNE 24-28

A Race to the Bottom

Database Transactions Undermining Your AppSec

VIKTOR CHUCHURSKI

$ whoami

- Viktor Chuchurski

$ whoami

- Viktor Chuchurski
- Sr. Application Security Engineer @ Doyensec

$ whoami

- Viktor Chuchurski
- Sr. Application Security Engineer @ Doyensec
- Software Engineer in a previous life

$ whoami

- Viktor Chuchurski
- Sr. Application Security Engineer @ Doyensec
- Software Engineer in a previous life

- @viktorot on Twitter
- vchuchurski on LinkedIn

```go
func (self *Db) Transfer(source int, destination int, amount int) error {
  ctx := context.Background()
  conn, err := db.Connect(ctx, self.databaseUrl)
  defer conn.Close(ctx)

  tx, err := conn.BeginTx(ctx)

  var user User
  err = conn.QueryRow(ctx, "SELECT id, name, balance FROM users WHERE id = $1", source).
    Scan(&user.Id, &user.Name, &user.Balance)

  if amount < 0 || user.Balance > amount {
    tx.Rollback(ctx)
    return fmt.Errorf("invalid transfer")
  }

  _, err = conn.Exec(ctx, "UPDATE users SET balance = balance - $2 WHERE id = $1", source, amount)
  _, err = conn.Exec(ctx, "UPDATE users SET balance = balance + $2 WHERE id = $1", destination, amount)

  err = tx.Commit(ctx)
  return nil
}
```

```go
func (self *Db) Transfer(source int, destination int, amount int) error {
  ctx := context.Background()
  conn, err := db.Connect(ctx, self.databaseUrl)
  defer conn.Close(ctx)

  tx, err := conn.BeginTx(ctx)

  var user User
  err = conn.QueryRow(ctx, "SELECT id, name, balance FROM users WHERE id = $1", source).
    Scan(&user.Id, &user.Name, &user.Balance)

  if amount < 0 || user.Balance > amount {
    tx.Rollback(ctx)
    return fmt.Errorf("invalid transfer")
  }

  _, err = conn.Exec(ctx, "UPDATE users SET balance = balance - $2 WHERE id = $1", source, amount)
  _, err = conn.Exec(ctx, "UPDATE users SET balance = balance + $2 WHERE id = $1", destination, amount)

  err = tx.Commit(ctx)
  return nil
}
```

```go
func (self *Db) Transfer(source int, destination int, amount int) error {
  ctx := context.Background()
  conn, err := db.Connect(ctx, self.databaseUrl)
  defer conn.Close(ctx)

  tx, err := conn.BeginTx(ctx)

  var user User
  err = conn.QueryRow(ctx, "SELECT id, name, balance FROM users WHERE id = $1", source).
    Scan(&user.Id, &user.Name, &user.Balance)

  if amount < 0 || user.Balance > amount {
    tx.Rollback(ctx)
    return fmt.Errorf("invalid transfer")
  }

  _, err = conn.Exec(ctx, "UPDATE users SET balance = balance - $2 WHERE id = $1", source, amount)
  _, err = conn.Exec(ctx, "UPDATE users SET balance = balance + $2 WHERE id = $1", destination, amount)

  err = tx.Commit(ctx)
  return nil
}
```

```go
func (self *Db) Transfer(source int, destination int, amount int) error {
  ctx := context.Background()
  conn, err := db.Connect(ctx, self.databaseUrl)
  defer conn.Close(ctx)

  tx, err := conn.BeginTx(ctx)

  var user User
  err = conn.QueryRow(ctx, "SELECT id, name, balance FROM users WHERE id = $1", source).
    Scan(&user.Id, &user.Name, &user.Balance)

  if amount < 0 || user.Balance > amount {
    tx.Rollback(ctx)
    return fmt.Errorf("invalid transfer")
  }

  _, err = conn.Exec(ctx, "UPDATE users SET balance = balance - $2 WHERE id = $1", source, amount)
  _, err = conn.Exec(ctx, "UPDATE users SET balance = balance + $2 WHERE id = $1", destination, amount)

  err = tx.Commit(ctx)
  return nil
}
```

```go
func (self *Db) Transfer(source int, destination int, amount int) error {
  ctx := context.Background()
  conn, err := db.Connect(ctx, self.databaseUrl)
  defer conn.Close(ctx)

  tx, err := conn.BeginTx(ctx)

  var user User
  err = conn.Qu
    Scan(&user.

  if amount < 0
    tx.Rollback
    return fmt.Errorf("invalid transfer")
  }

  _, err = conn.Exec(ctx, "UPDATE users SET balance = balance - $2 WHERE id = $1", source, amount)
  _, err = conn.Exec(ctx, "UPDATE users SET balance = balance + $2 WHERE id = $1", destination, amount)

  err = tx.Commit(ctx)
  return nil
}
```

A database transaction represents a **unit of work**, performed within a database management system, that is treated in a coherent and reliable way **independent of other transactions**.

```go
func (self *Db) Transfer(source int, destination int, amount int) error {
  ctx := context.Background()
  conn, err := db.Connect(ctx, self.databaseUrl)
  defer conn.Close(ctx)

  tx, err := conn.BeginTx(ctx)

  var user User
  err = conn.QueryRow(ctx, "SELECT id, name, balance FROM users WHERE id = $1", source).
    Scan(&user.Id, &user.Name, &user.Balance)

  if amount < 0 || user.Balance > amount {
    tx.Rollback(ctx)
    return fmt.Errorf("invalid transfer")
  }

  _, err = conn.Exec(ctx, "UPDATE users SET balance = balance - $2 WHERE id = $1", source, amount)
  _, err = conn.Exec(ctx, "UPDATE users SET balance = balance + $2 WHERE id = $1", destination, amount)

  err = tx.Commit(ctx)
  return nil
}
```

```go
func (self *Db) Transfer(source int, destination int, amount int) error {
  ctx := context.Background()
  conn, err := db.Connect(ctx, self.databaseUrl)
  defer conn.Close(ctx)

  tx, err := conn.BeginTx(ctx)

  var user User
  err = conn.QueryRow(ctx, "SELECT id, name, balance FROM users WHERE id = $1", source).
    Scan(&user.Id, &user.Name, &user.Balance)

  if amount < 0 || user.Balance > amount {
    tx.Rollback(ctx)
    return fmt.Errorf("invalid transfer")
  }

  _, err = conn.Exec(ctx, "UPDATE users SET balance = balance - $2 WHERE id = $1", source, amount)
  _, err = conn.Exec(ctx, "UPDATE users SET balance = balance + $2 WHERE id = $1", destination, amount)

  err = tx.Commit(ctx)
  return nil
}
```

```go
func (self *Db) Transfer(source int, destination int, amount int) error {
  ctx := context.Background()
  conn, err := db.Connect(ctx, self.databaseUrl)
  defer conn.Close(ctx)

  tx, err := conn.BeginTx(ctx)

  var user User
  err = conn.QueryRow(ctx, "SELECT id, name, balance FROM users WHERE id = $1", source).
    Scan(&user.Id, &user.Name, &user.Balance)

  if amount < 0 || user.Balance > amount {
    tx.Rollback(ctx)
    return fmt.Errorf("invalid transfer")
  }

  _, err = conn.Exec(ctx, "UPDATE users SET balance = balance - $2 WHERE id = $1", source, amount)
  _, err = conn.Exec(ctx, "UPDATE users SET balance = balance + $2 WHERE id = $1", destination, amount)

  err = tx.Commit(ctx)
  return nil
}
```

```go
func (self *Db) Transfer(source int, destination int, amount int) error {
  ctx := context.Background()
  conn, err := db.Connect(ctx, self.databaseUrl)
  defer conn.Close(ctx)

  tx, err := conn.BeginTx(ctx)

  var user User
  err = conn.QueryRow(ctx, "SELECT id, name, balance FROM users WHERE id = $1", source).
    Scan(&user.Id, &user.Name, &user.Balance)

  if amount < 0 || user.Balance > amount {
    tx.Rollback(ctx)
    return fmt.Errorf("invalid transfer")
  }

  _, err = conn.Exec(ctx, "UPDATE users SET balance = balance - $2 WHERE id = $1", source, amount)
  _, err = conn.Exec(ctx, "UPDATE users SET balance = balance + $2 WHERE id = $1", destination, amount)

  err = tx.Commit(ctx)
  return nil
}
```

```go
func (self *Db) Transfer(source int, destination int, amount int) error {
  ctx := context.Background()
  conn, err := db.Connect(ctx, self.databaseUrl)
  defer conn.Close(ctx)

  tx, err := conn.BeginTx(ctx)

  var user User
  err = conn.QueryRow(ctx, "SELECT id, name, balance FROM users WHERE id = $1", source).
    Scan(&user.Id, &user.Name, &user.Balance)

  if amount < 0 || user.Balance > amount {
    tx.Rollback(ctx)
    return fmt.Errorf("invalid transfer")
  }

  _, err = conn.Exec(ctx, "UPDATE users SET balance = balance - $2 WHERE id = $1", source, amount)
  _, err = conn.Exec(ctx, "UPDATE users SET balance = balance + $2 WHERE id = $1", destination, amount)

  err = tx.Commit(ctx)
  return nil
}
```

```go
func (self *Db) Transfer(source int, destination int, amount int) error {
  ctx := context.Background()
  conn, err := db.Connect(ctx, self.databaseUrl)
  defer conn.Close(ctx)

  tx, err := conn.BeginTx(ctx)

  var user User
  err = conn.QueryRow(ctx, "SELECT id, name, balance FROM users WHERE id = $1", source).
    Scan(&user.Id, &user.Name, &user.Balance)

  if amount < 0 || user.Balance > amount {
    tx.Rollback(ctx)
    return fmt.Errorf("invalid transfer")
  }

  _, err = conn.Exec(ctx, "UPDATE users SET balance = balance - $2 WHERE id = $1", source, amount)
  _, err = conn.Exec(ctx, "UPDATE users SET balance = balance + $2 WHERE id = $1", destination, amount)

  err = tx.Commit(ctx)
  return nil
}
```

```go
func (self *Db) Transfer(source int, destination int, amount int) error {
  ctx := context.Background()
  conn, err := db.Connect(ctx, self.databaseUrl)
  defer conn.Close(ctx)

  tx, err := conn.BeginTx(ctx)

  var user User
  err = conn.QueryRow(ctx, "SELECT id, name, balance FROM users WHERE id = $1", source).
    Scan(&user.Id, &user.Name, &user.Balance)

  if amount < 0 || user.Balance > amount {
    tx.Rollback(ctx)
    return fmt.Errorf("invalid transfer")
  }

  _, err = conn.Exec(ctx, "UPDATE users SET balance = balance - $2 WHERE id = $1", source, amount)
  _, err = conn.Exec(ctx, "UPDATE users SET balance = balance + $2 WHERE id = $1", destination, amount)

  err = tx.Commit(ctx)
  return nil
}
```

Transaction Execution Logs

```
[41] BEGIN
[72] BEGIN
[41] SELECT id, balance FROM users WHERE id = 2
[72] SELECT id, balance FROM users WHERE id = 2
[41] UPDATE users SET balance = balance - 20 WHERE id = 2
[72] UPDATE users SET balance = balance - 20 WHERE id = 2
[41] UPDATE users SET balance = balance + 20 WHERE id = 1
[41] COMMIT
[72] UPDATE users SET balance = balance + 20 WHERE id = 1
[72] COMMIT
```

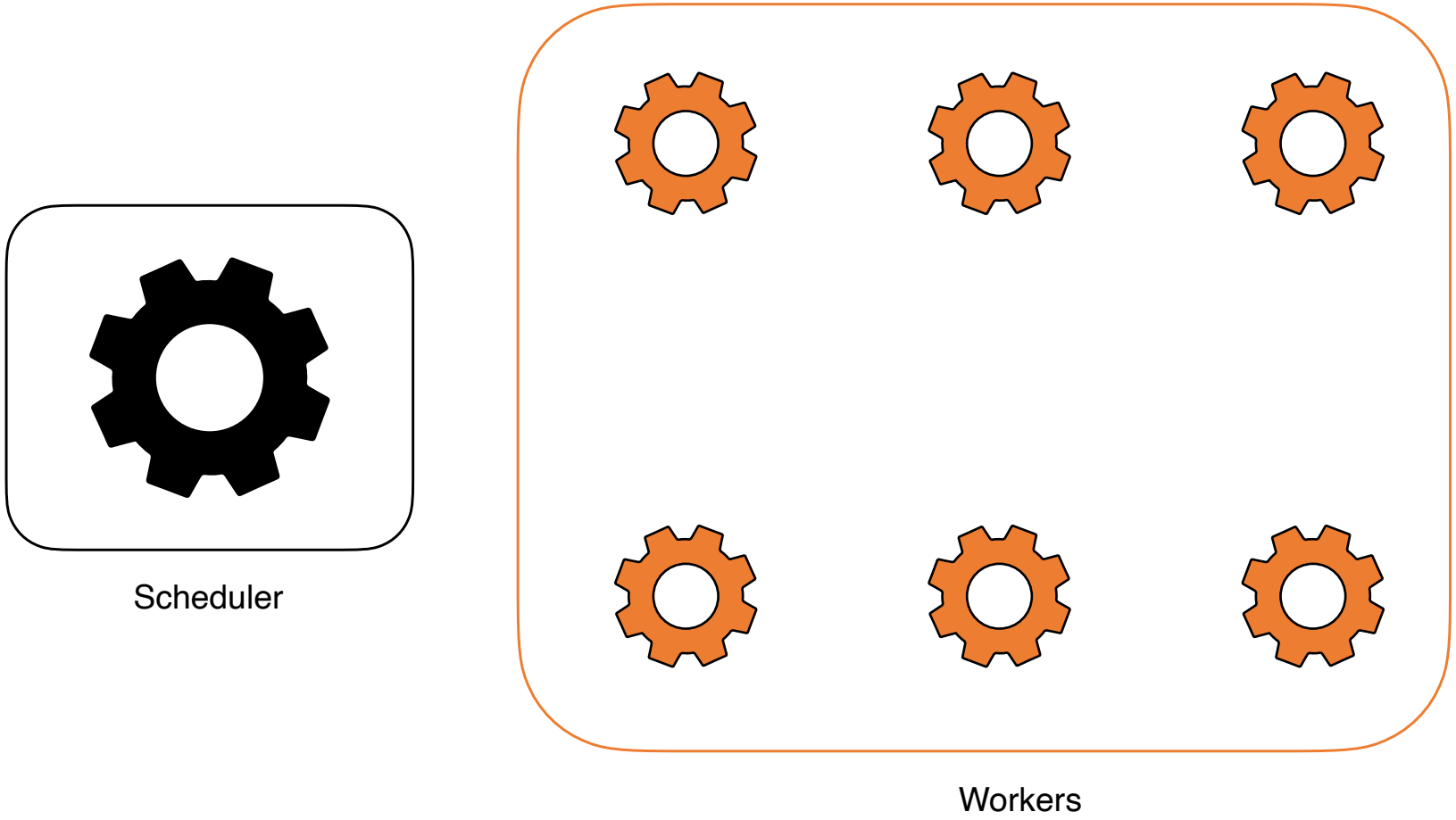
Transaction Execution Logs

```
[Tx1] BEGIN
[Tx2] BEGIN
[Tx1] SELECT id, balance FROM users WHERE id = 2
[Tx2] SELECT id, balance FROM users WHERE id = 2
[Tx1] UPDATE users SET balance = balance - 20 WHERE id = 2
[Tx2] UPDATE users SET balance = balance - 20 WHERE id = 2
[Tx1] UPDATE users SET balance = balance + 20 WHERE id = 1
[Tx1] COMMIT
[Tx2] UPDATE users SET balance = balance + 20 WHERE id = 1
[Tx2] COMMIT
```

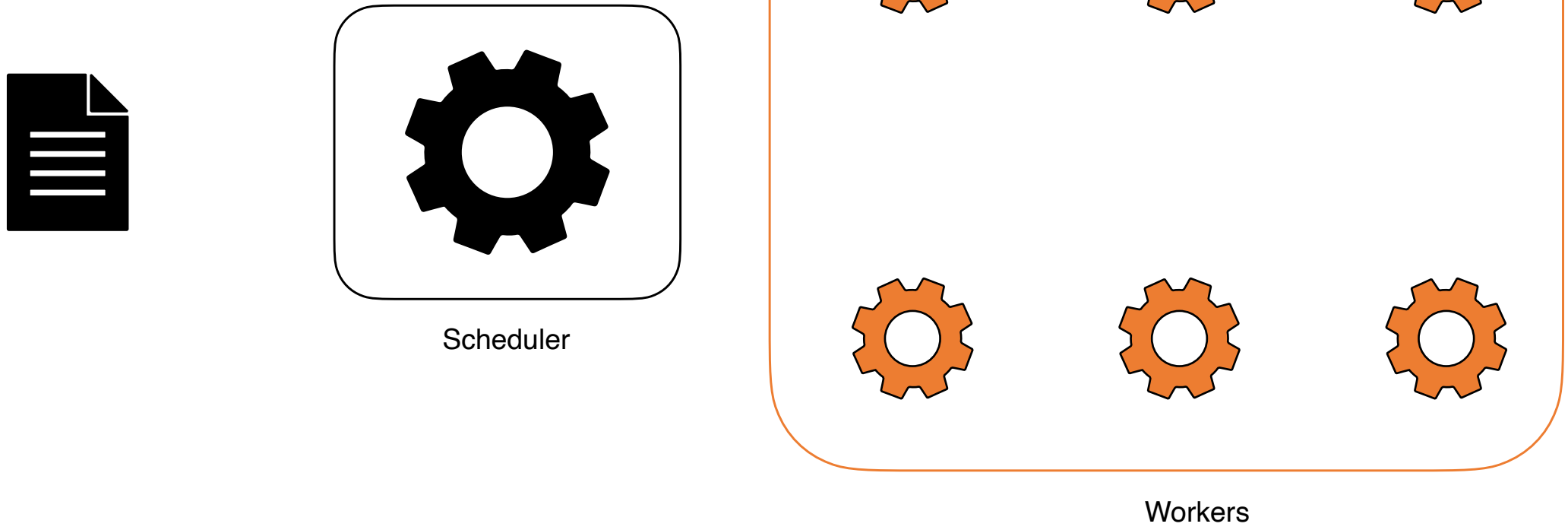
Transaction Execution Logs

```
[Tx1] BEGIN
[Tx2] BEGIN
[Tx1] SELECT id, balance FROM users WHERE id = 2
[Tx2] SELECT id, balance FROM users WHERE id = 2
[Tx1] UPDATE users SET balance = balance - 20 WHERE id = 2
[Tx2] UPDATE users SET balance = balance - 20 WHERE id = 2
[Tx1] UPDATE users SET balance = balance + 20 WHERE id = 1
[Tx1] COMMIT
[Tx2] UPDATE users SET balance = balance + 20 WHERE id = 1
[Tx2] COMMIT
```

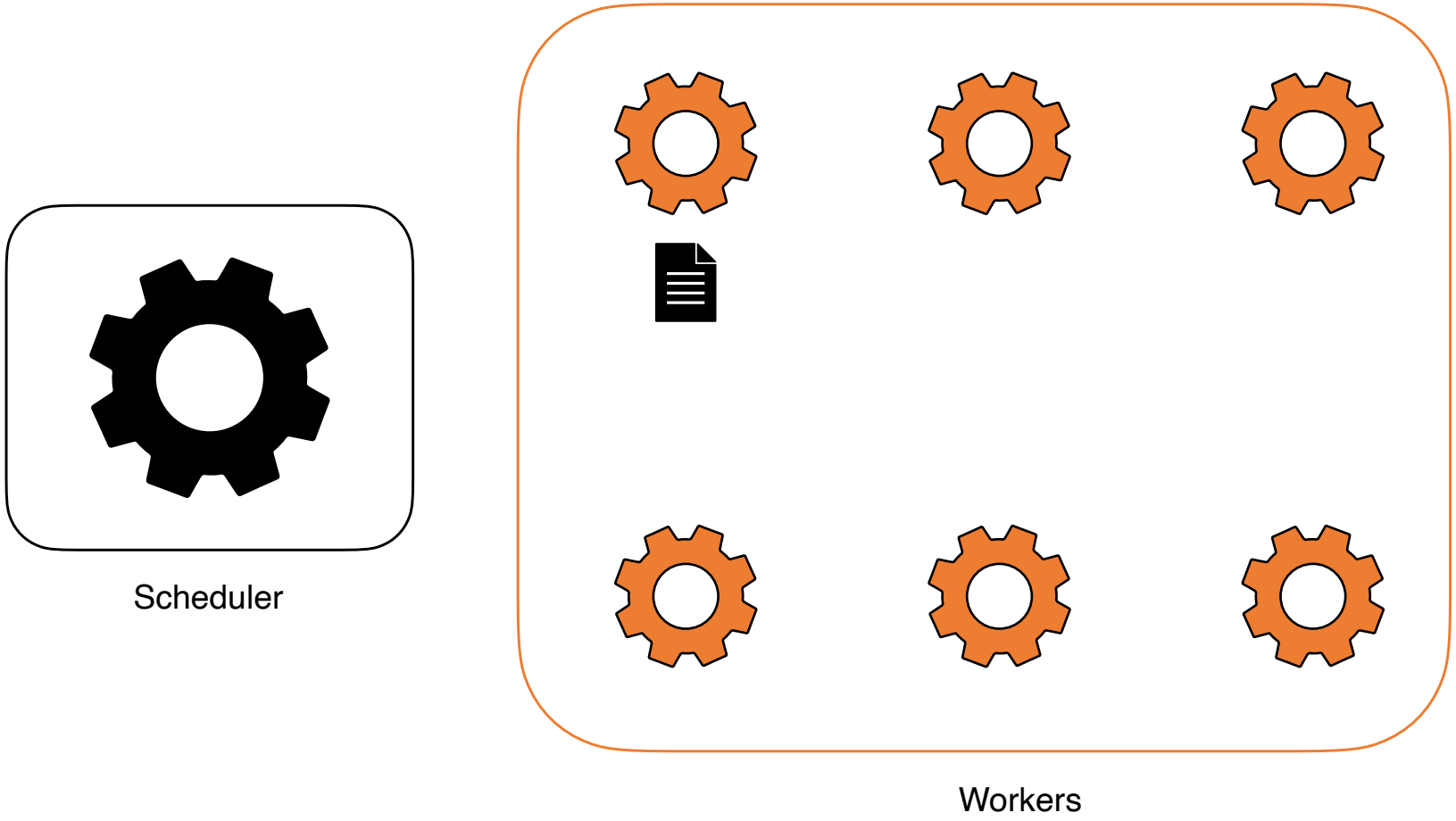
Query Scheduling



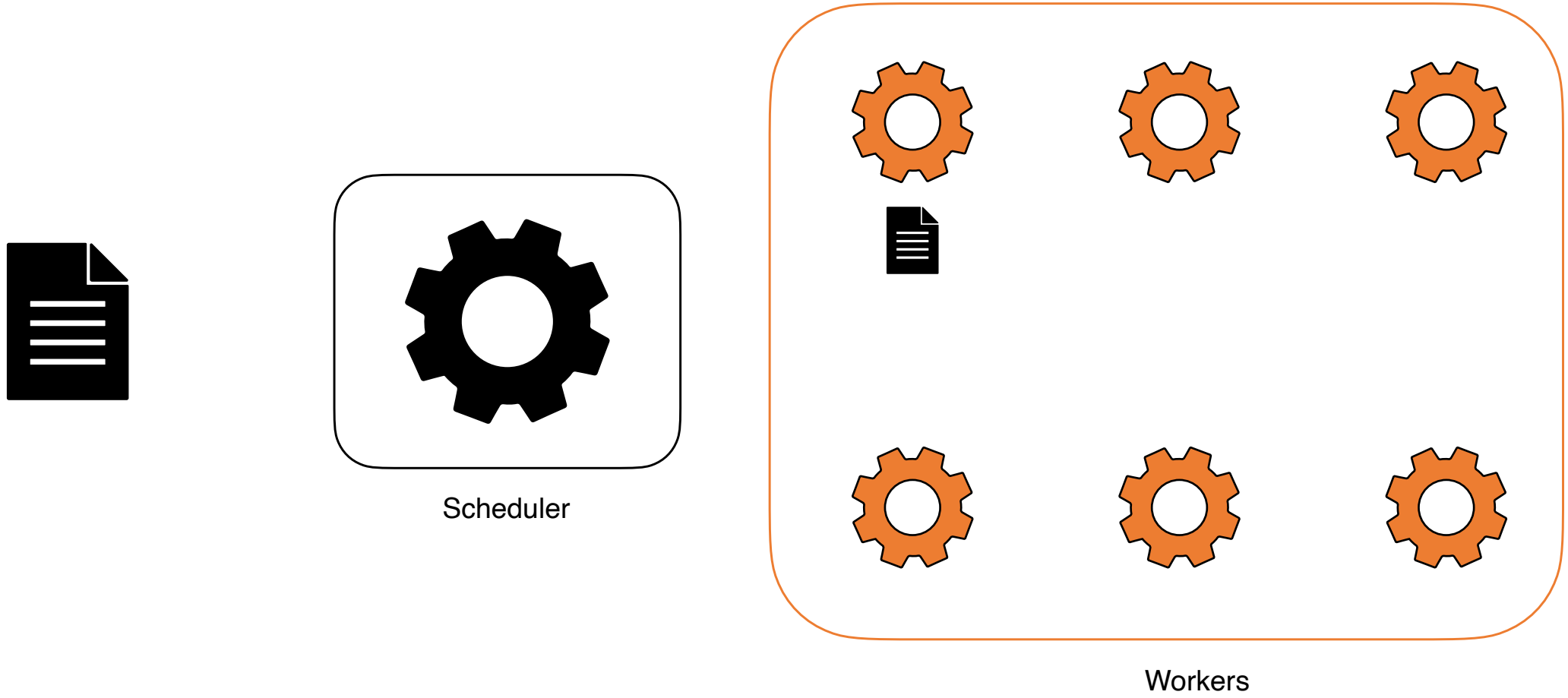
Query Scheduling



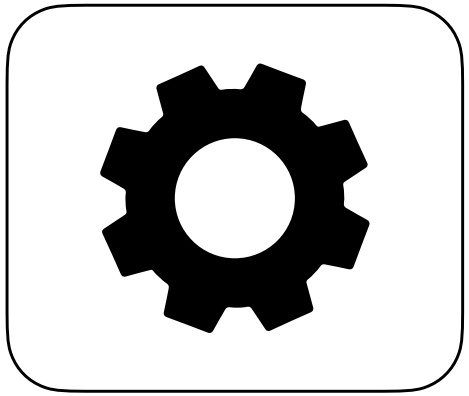
Query Scheduling



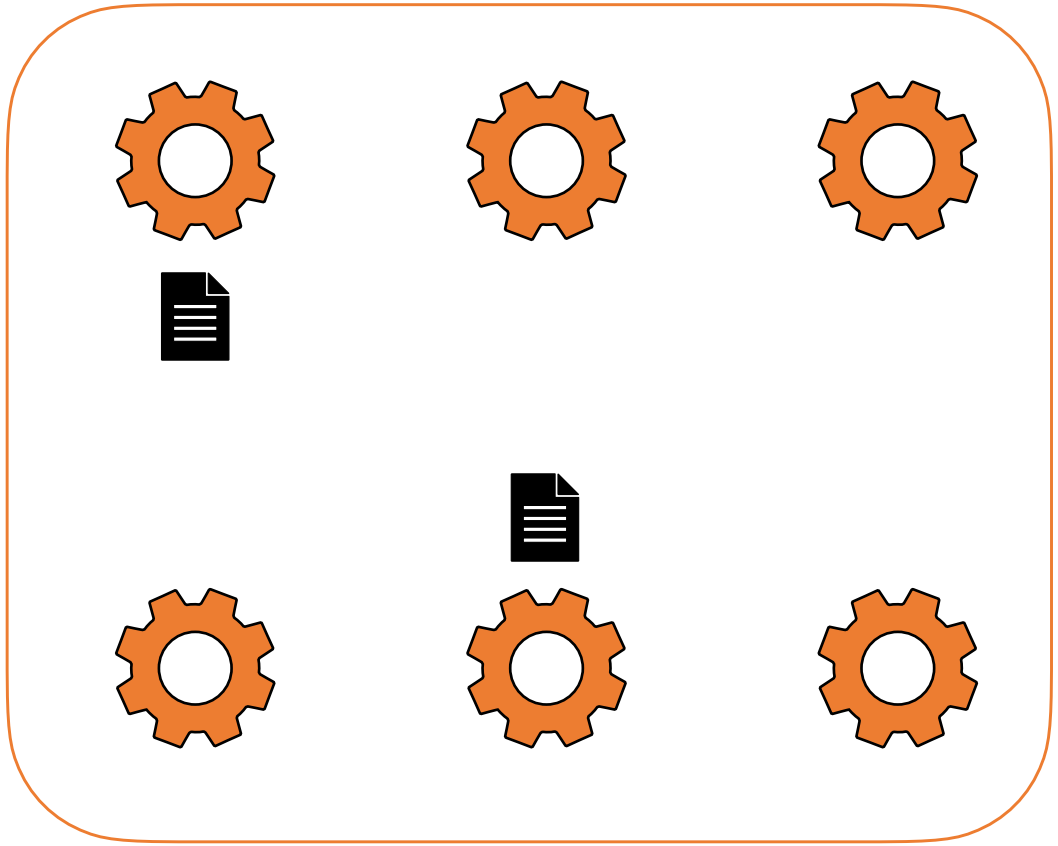
Query Scheduling



Query Scheduling

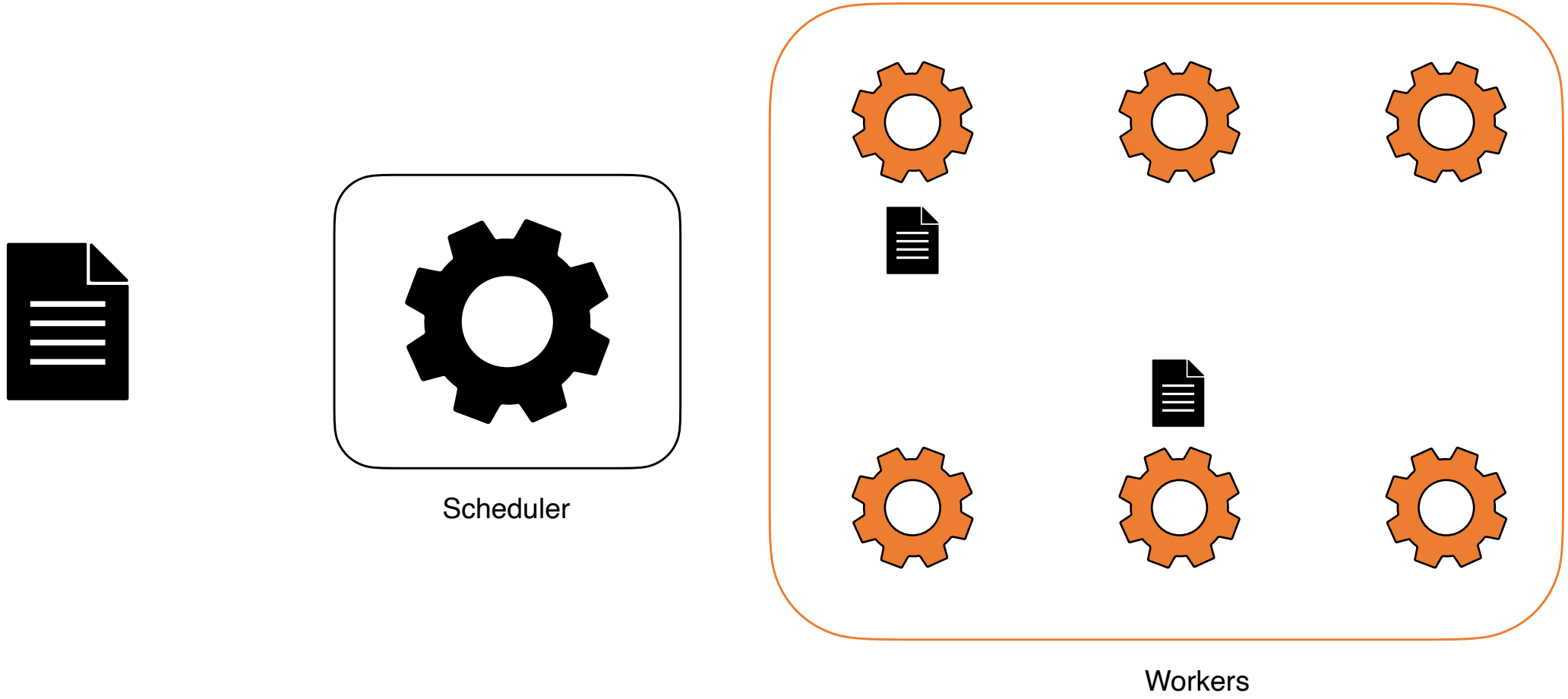


Scheduler

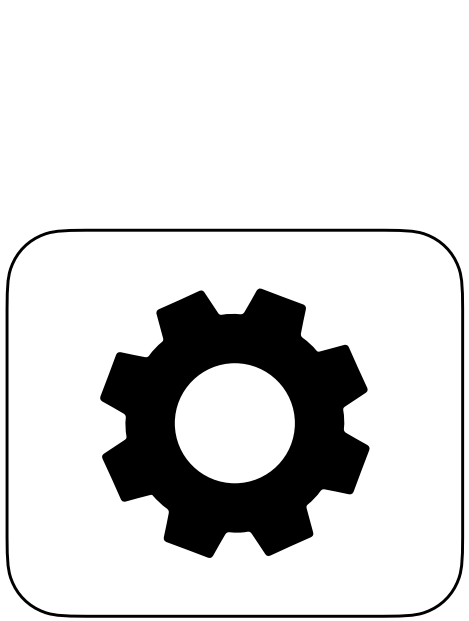


Workers

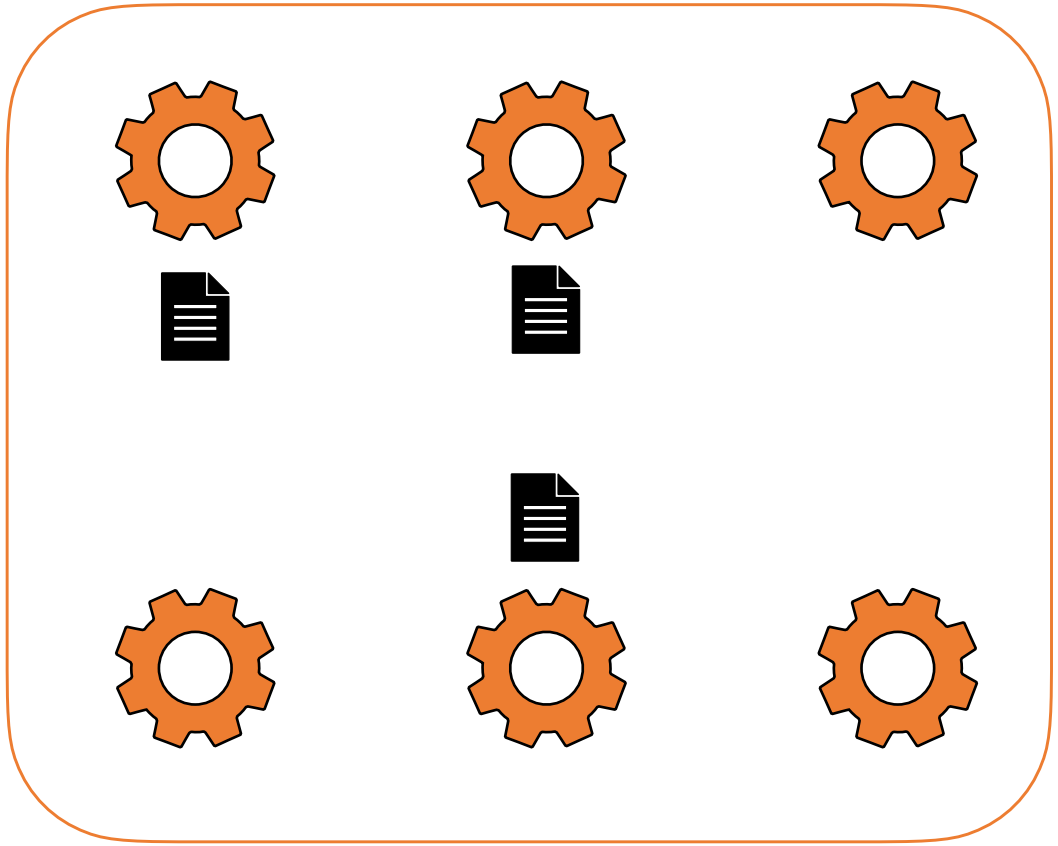
Query Scheduling



Query Scheduling

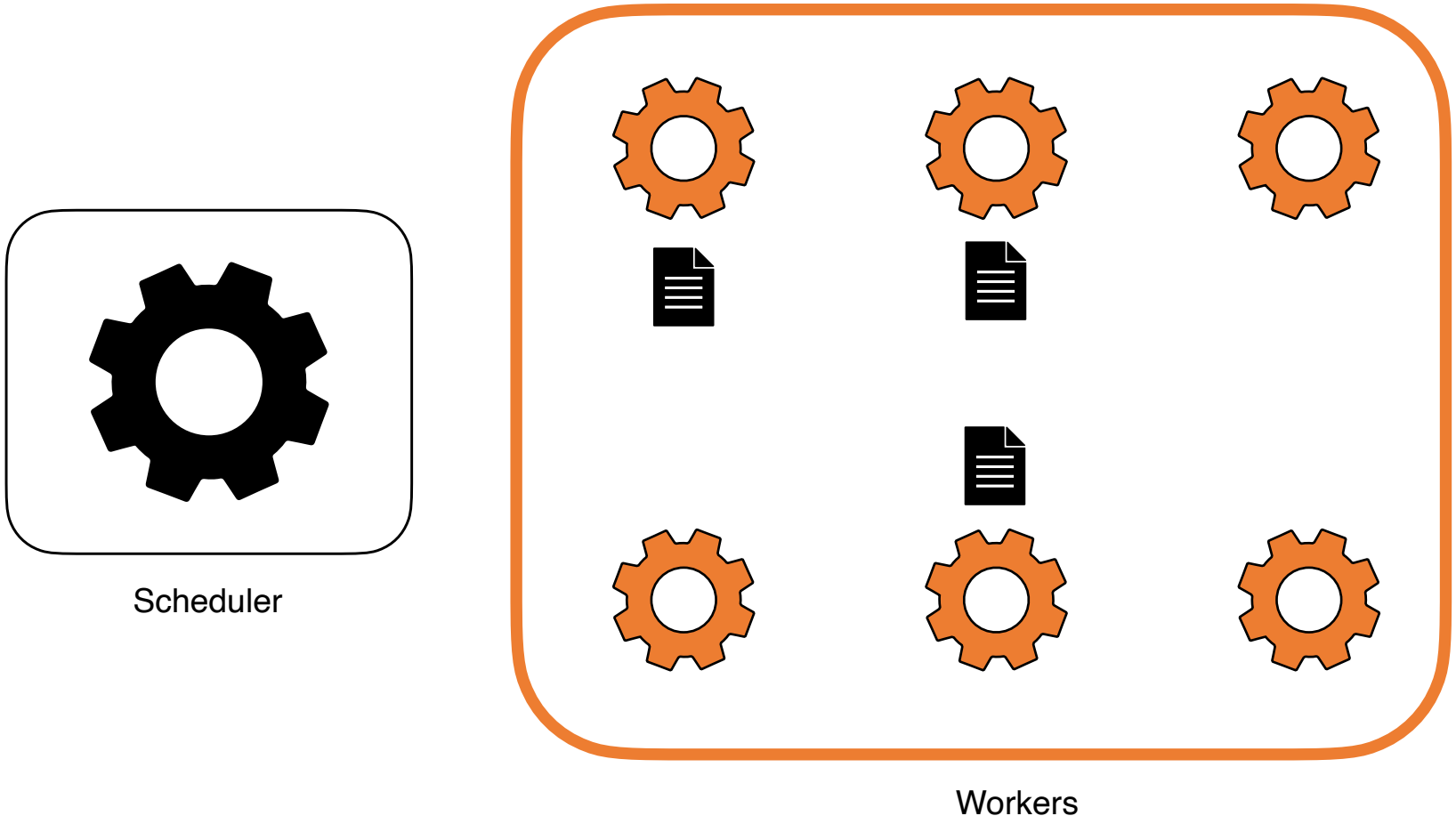


Scheduler



Workers

Query Scheduling



Locks

Locks

- Read/Shared (S)

Locks

- Read/Shared (S)
- Write/Exclusive (X)

Locks

- Read/Shared (S)
- Write/Exclusive (X)

		Requested	
		Shared	Exclusive
Active	None	✓	✓
	Shared	✓	✗
	Exclusive	✗	✗

Locks

- Read/Shared (S)
- Write/Exclusive (X)

		Requested	
		Shared	Exclusive
Active	None	✓	✓
	Shared	✓	✗
	Exclusive	✗	✗

- Under normal circumstances, `SELECT` statements do not acquire any locks!

Read Phenomena

Read Phenomena

- Dirty Reads

Read Phenomena

- Dirty Reads
- Non-Repeatable Reads

Read Phenomena

- Dirty Reads
- Non-Repeatable Reads
- Phantom Reads

Dirty Read

Dirty Read

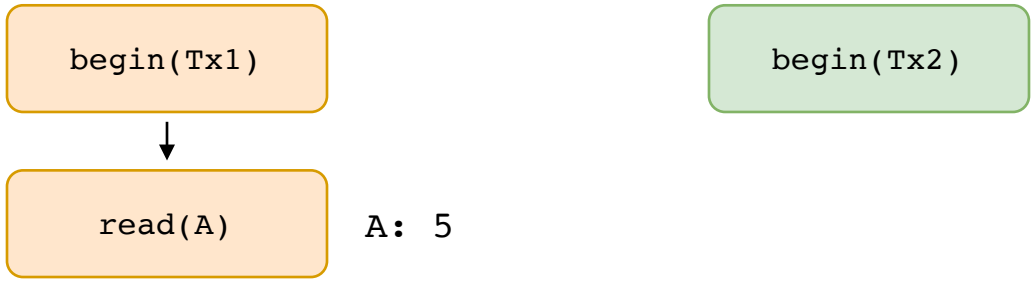
```
begin(Tx1)
```

Dirty Read

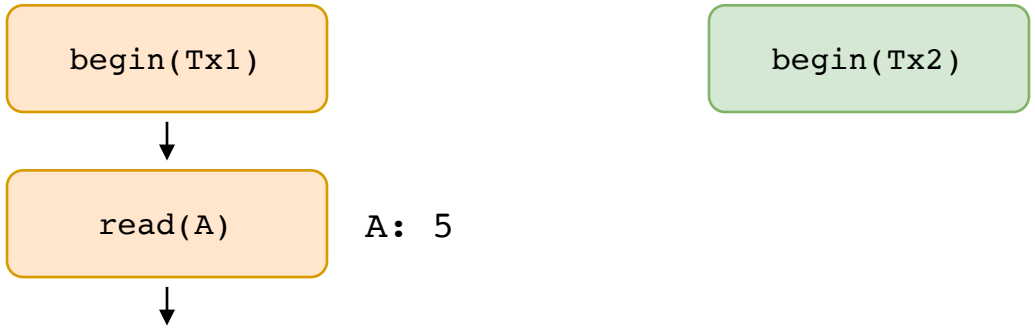
Dirty Read

Dirty Read

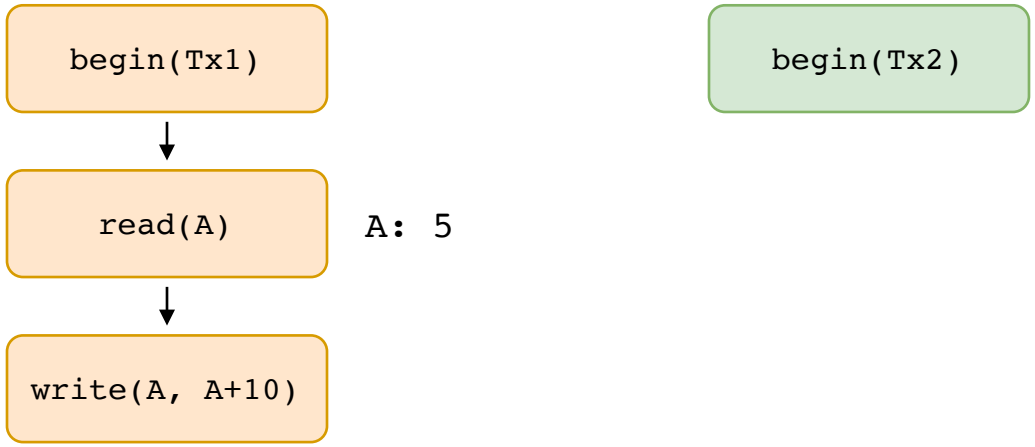
Dirty Read



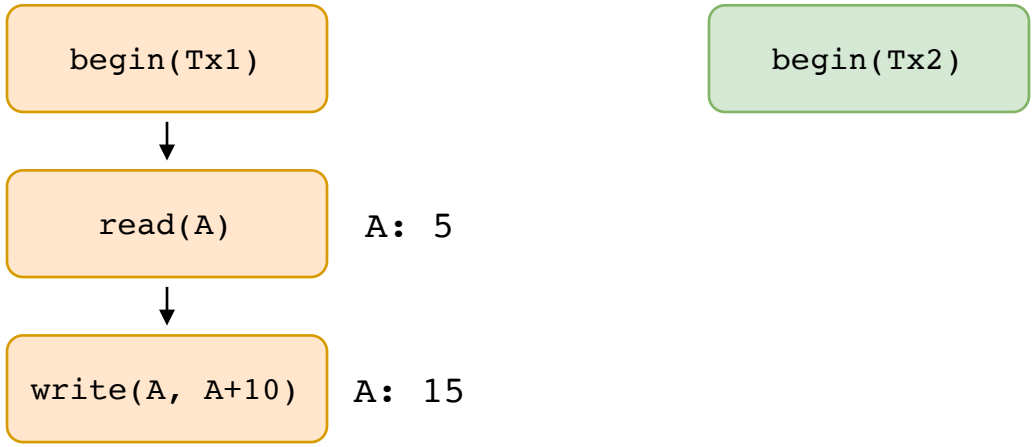
Dirty Read



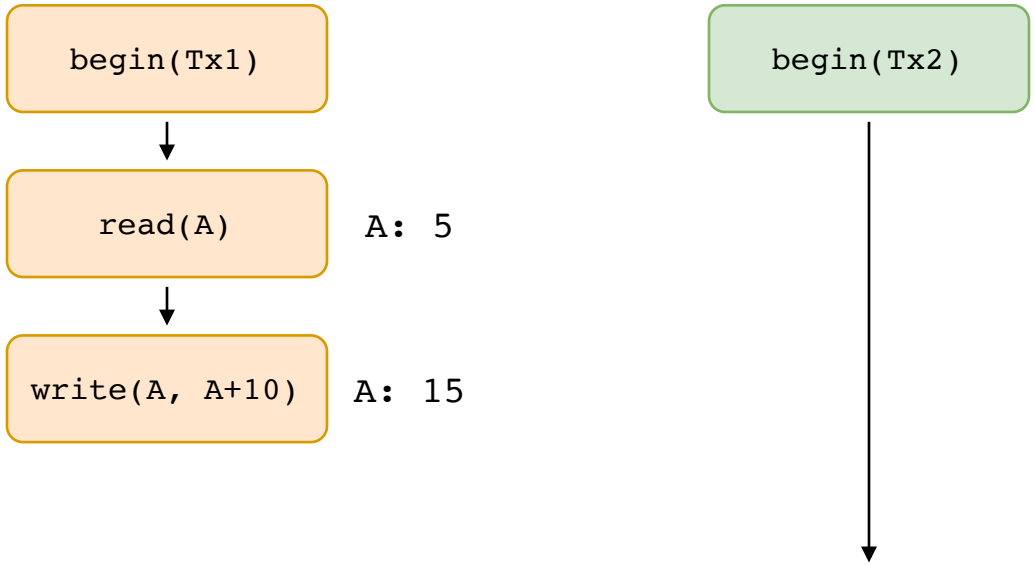
Dirty Read



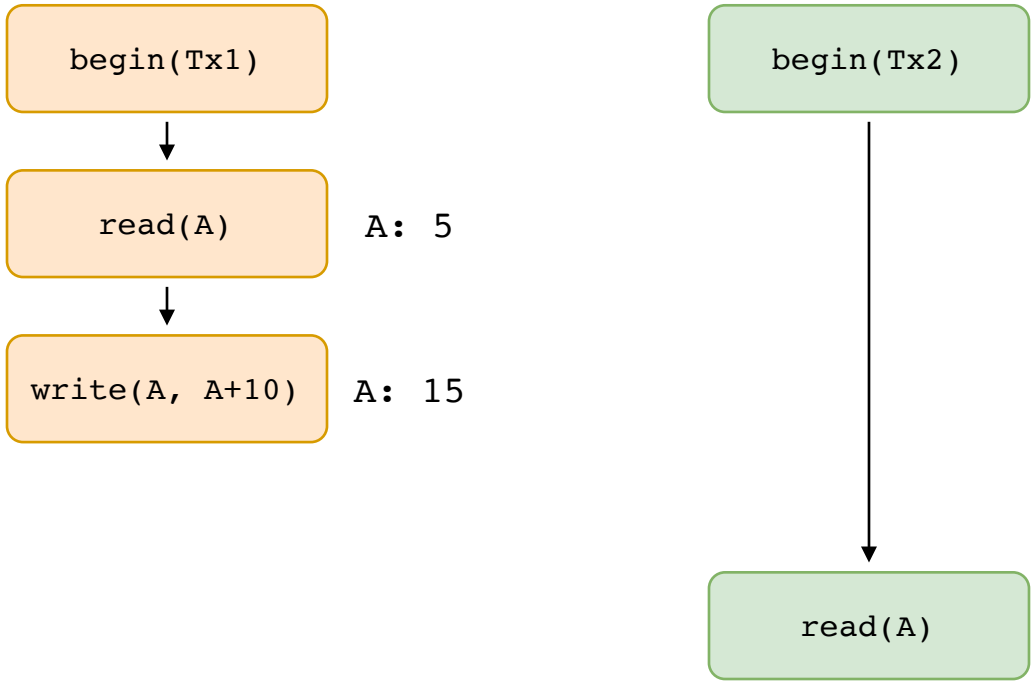
Dirty Read



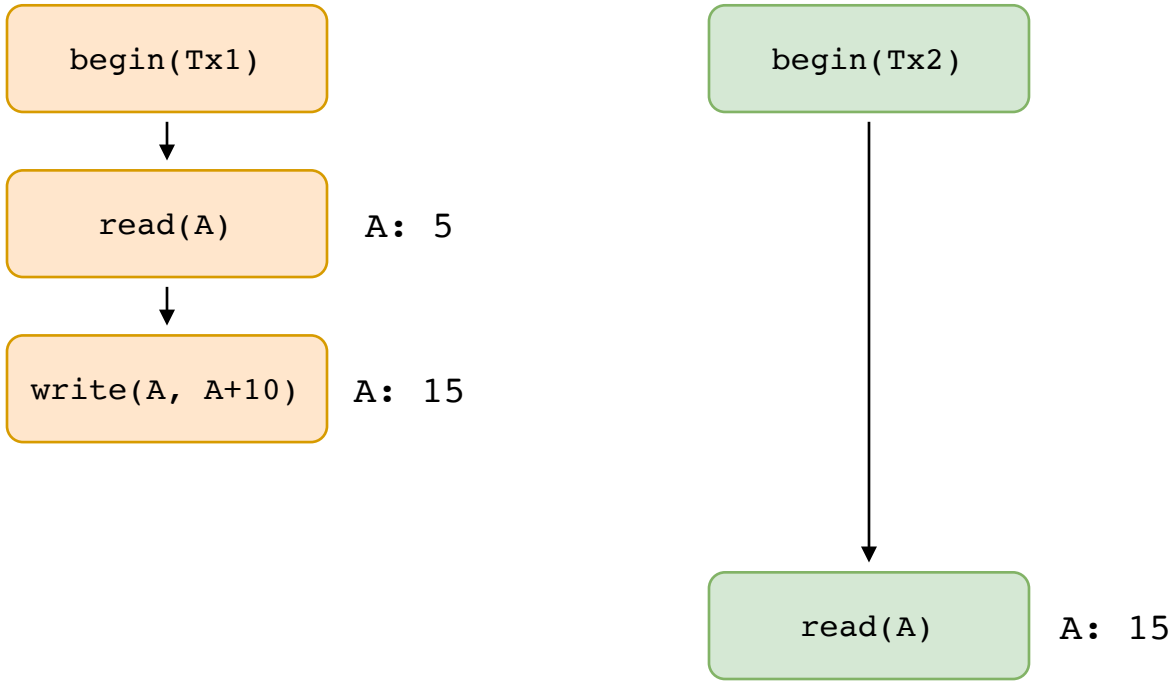
Dirty Read



Dirty Read



Dirty Read



Non-Repeatable Reads

Non-Repeatable Reads

```
begin(Tx1)
```

Non-Repeatable Reads

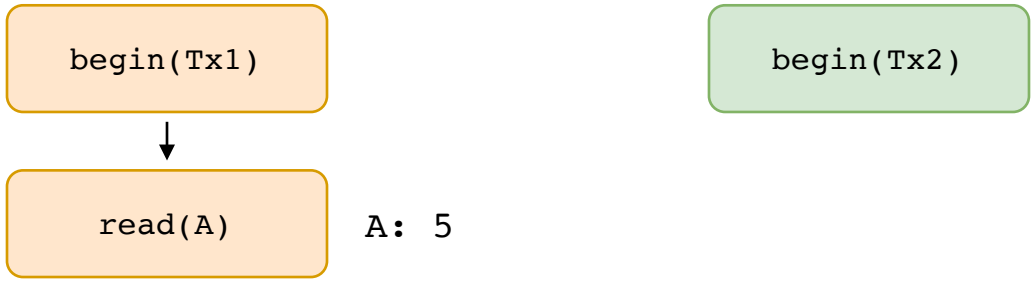
```
begin(Tx1)
```

```
begin(Tx2)
```

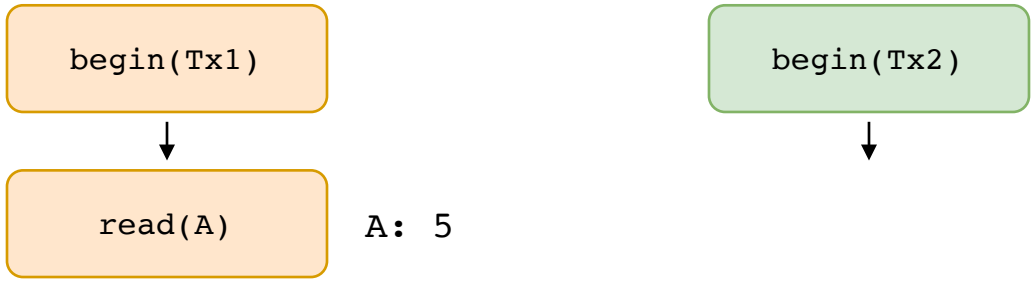
Non-Repeatable Reads

Non-Repeatable Reads

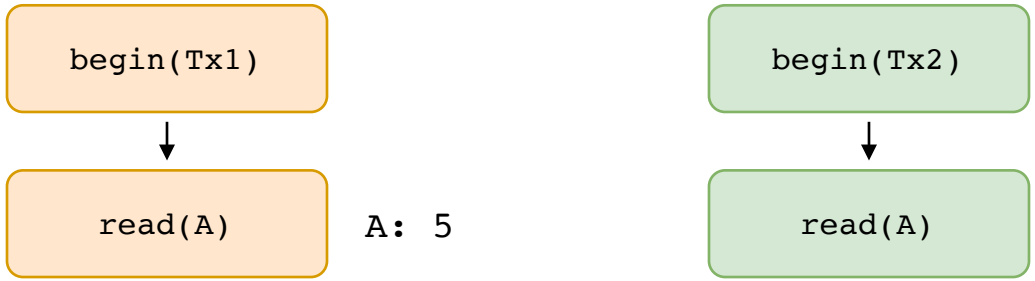
Non-Repeatable Reads



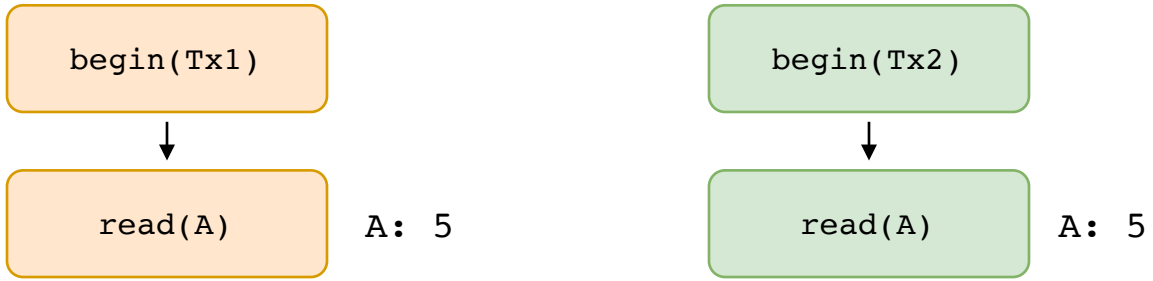
Non-Repeatable Reads



Non-Repeatable Reads

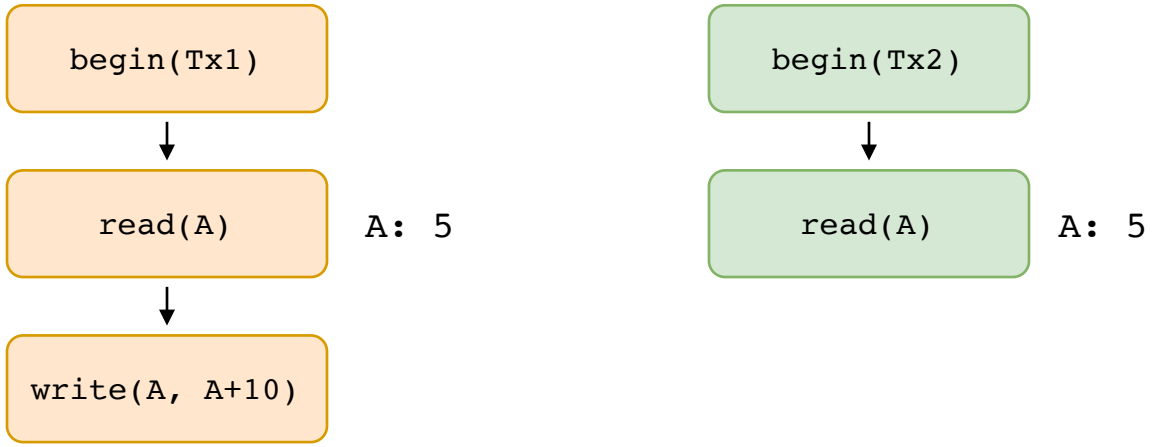


Non-Repeatable Reads

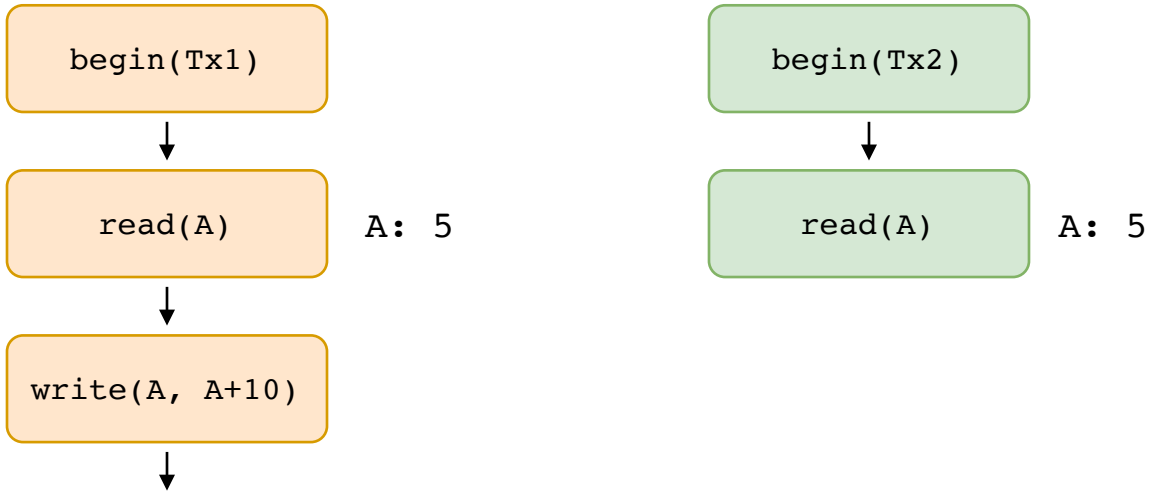


Non-Repeatable Reads

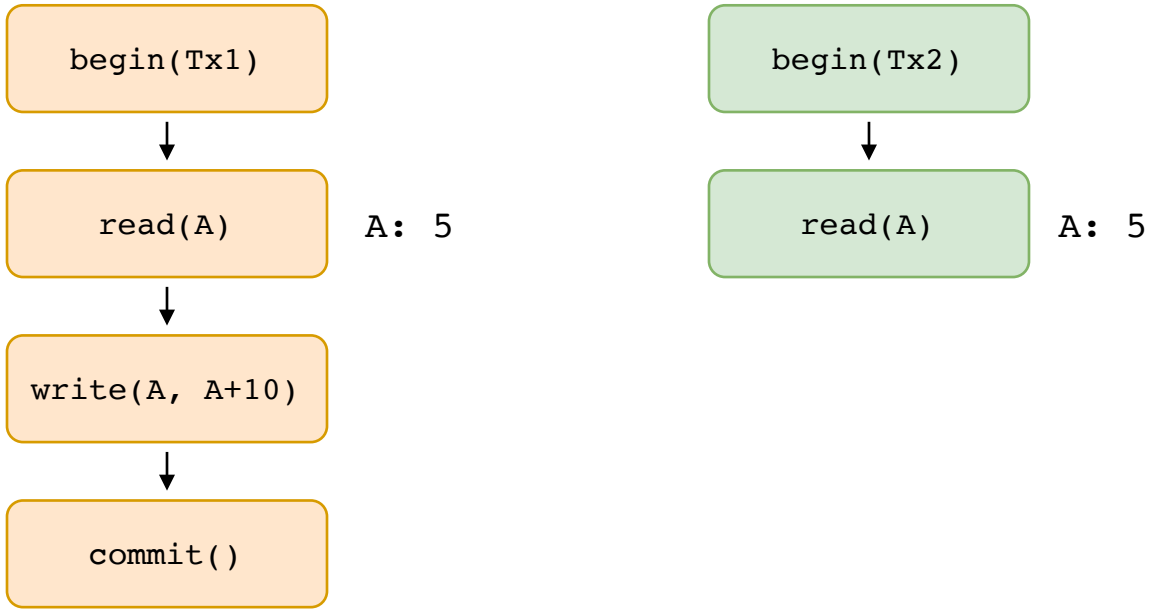
Non-Repeatable Reads



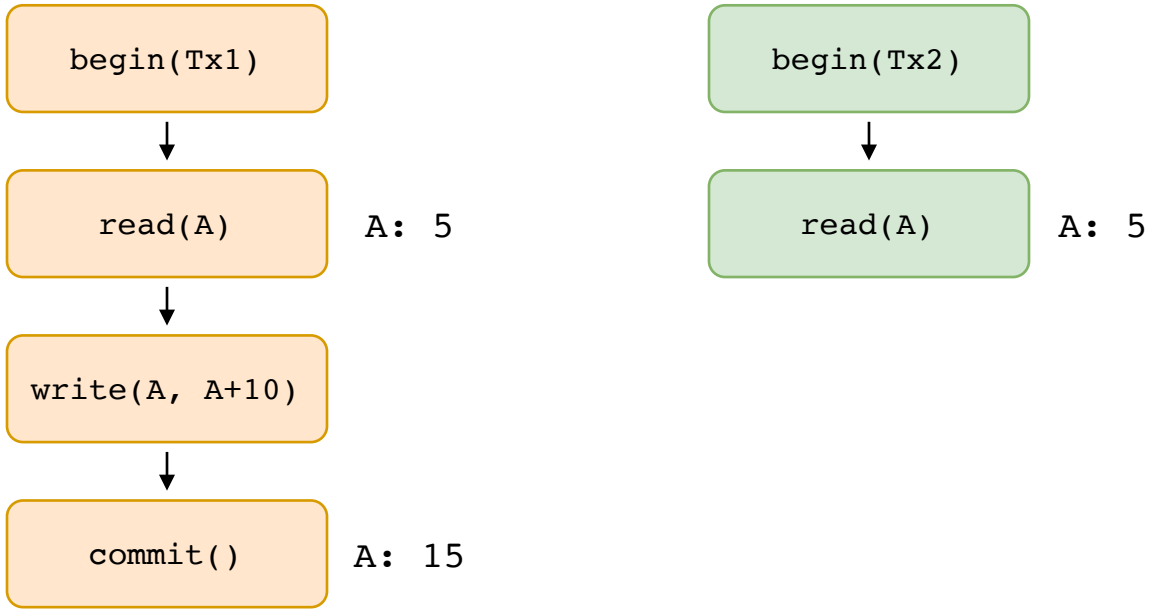
Non-Repeatable Reads



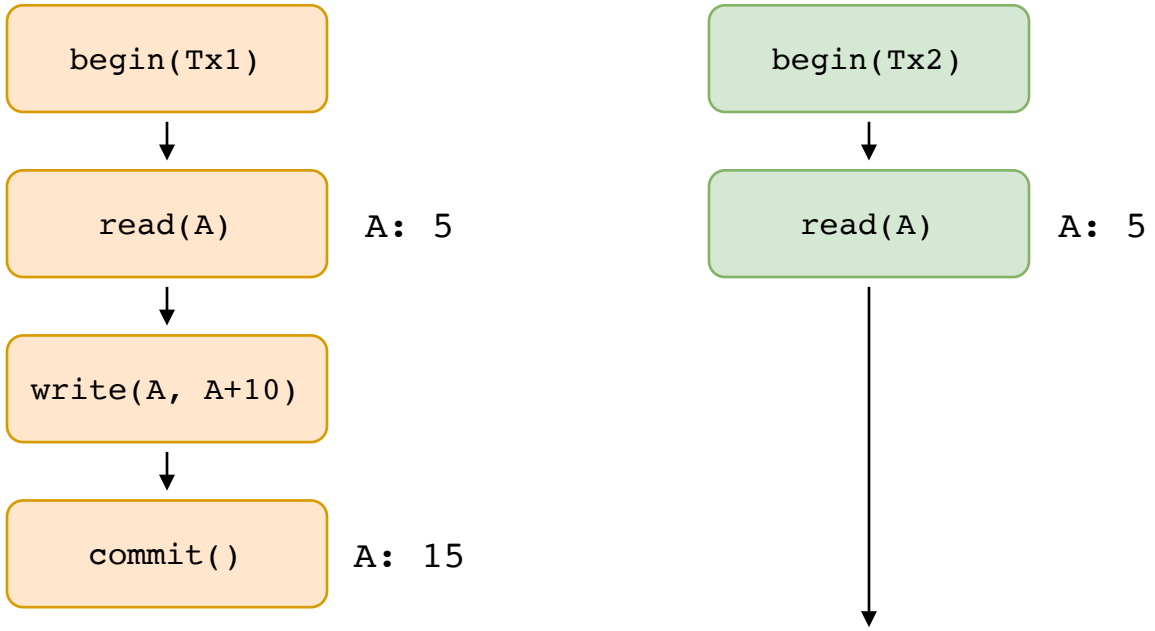
Non-Repeatable Reads



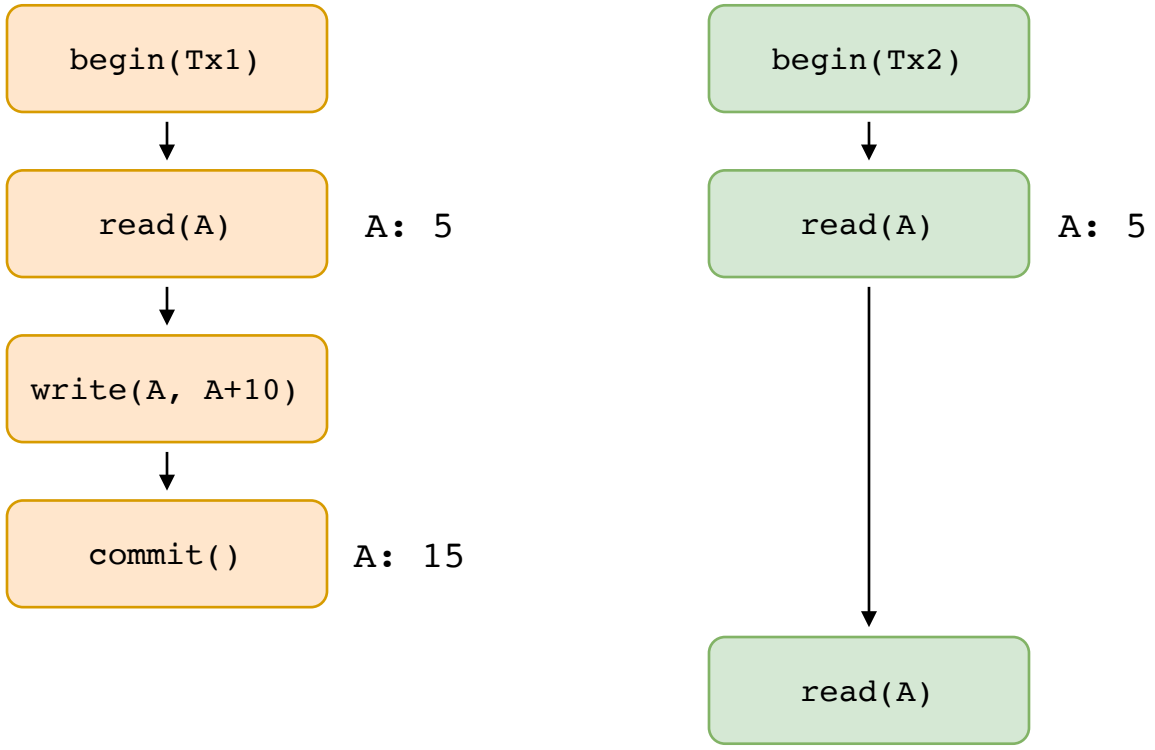
Non-Repeatable Reads



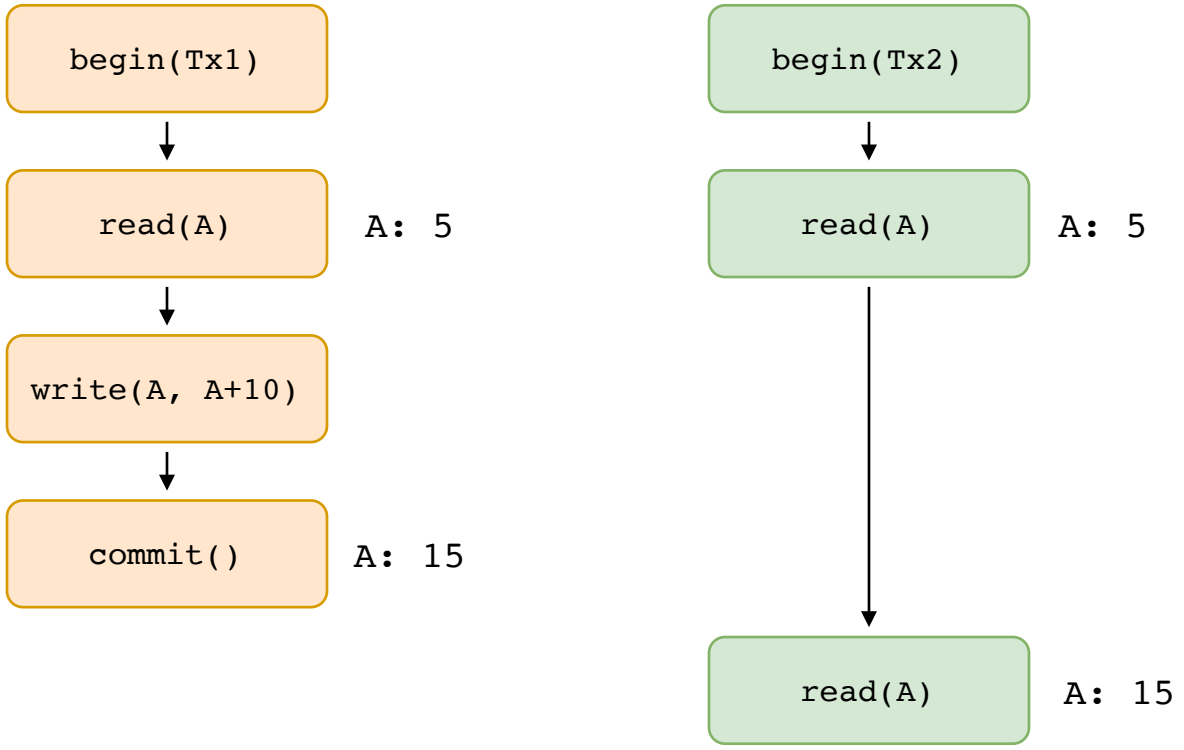
Non-Repeatable Reads



Non-Repeatable Reads



Non-Repeatable Reads



Phantom Reads

Phantom Reads

```
begin(Tx1)
```

Phantom Reads

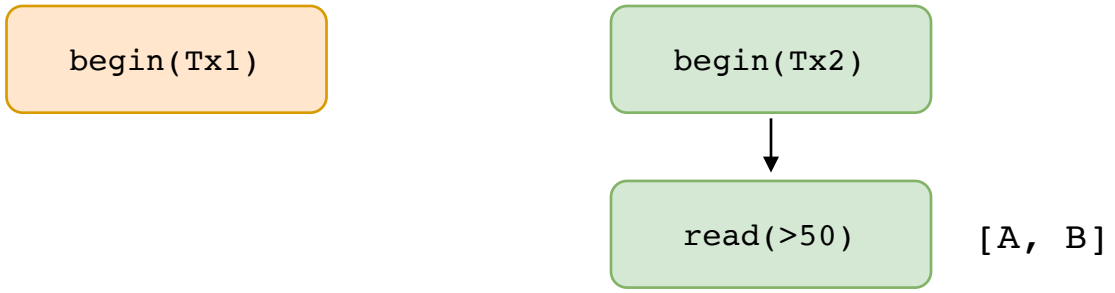
```
begin(Tx1)
```

```
begin(Tx2)
```

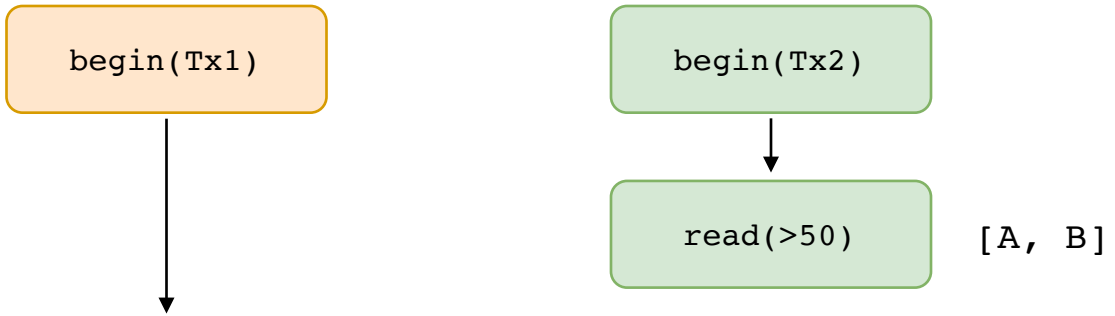
Phantom Reads

Phantom Reads

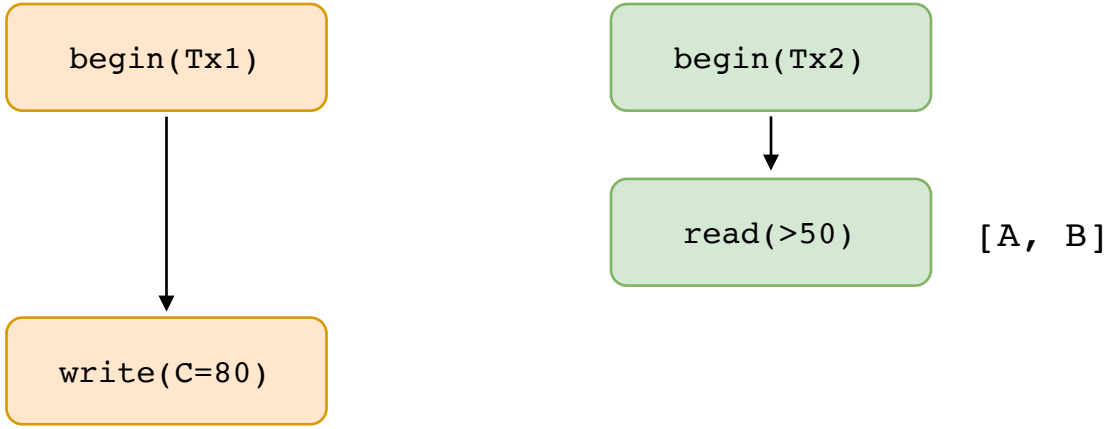
Phantom Reads



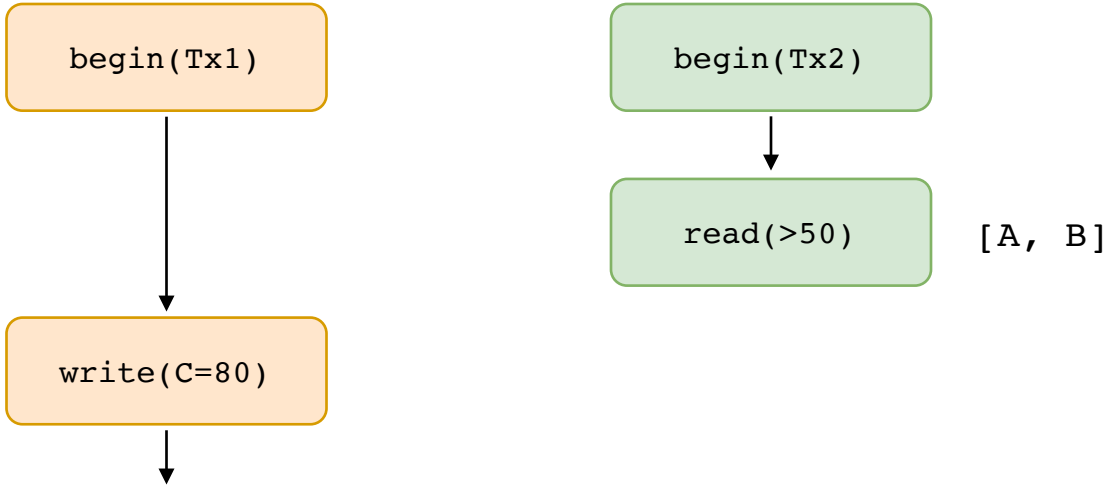
Phantom Reads



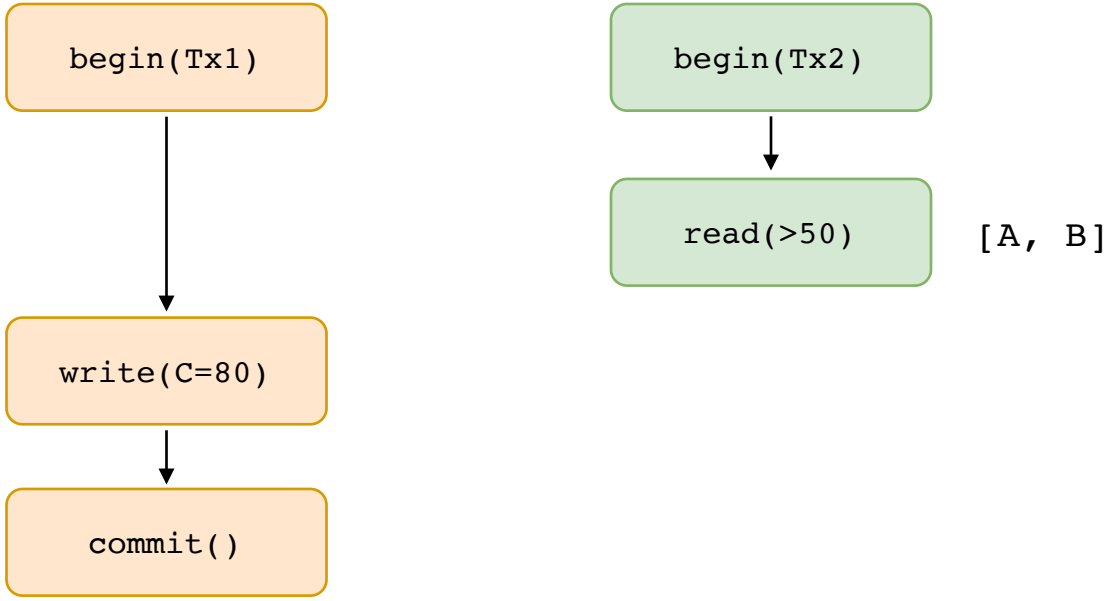
Phantom Reads



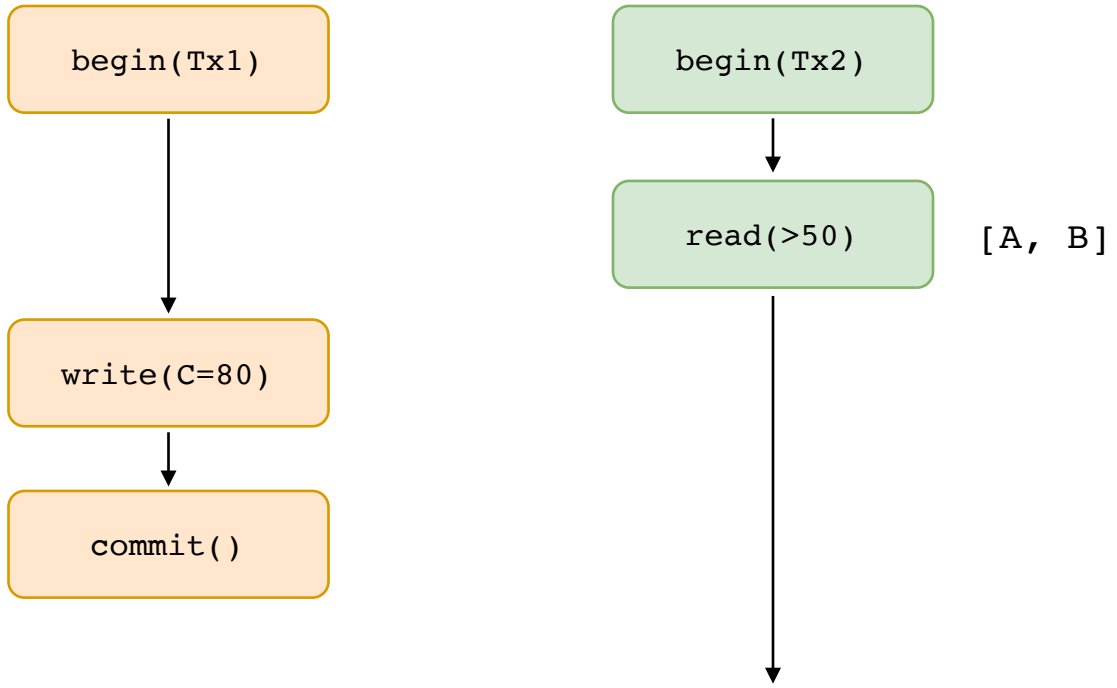
Phantom Reads



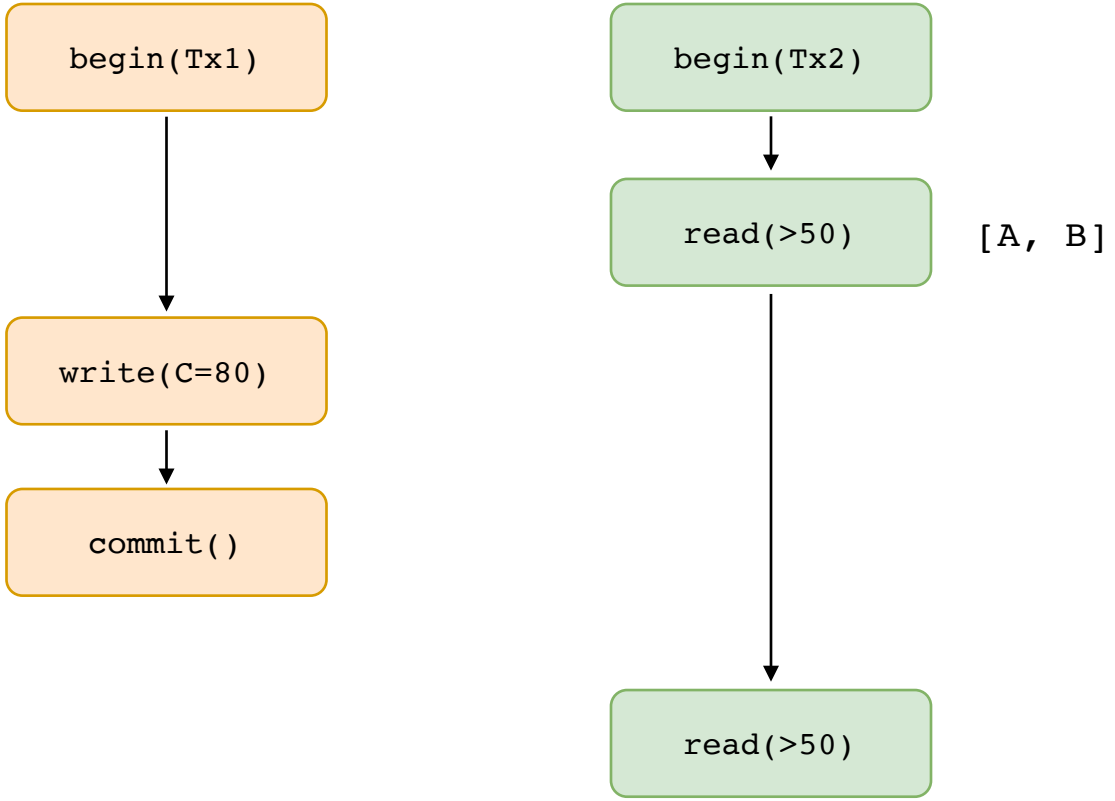
Phantom Reads



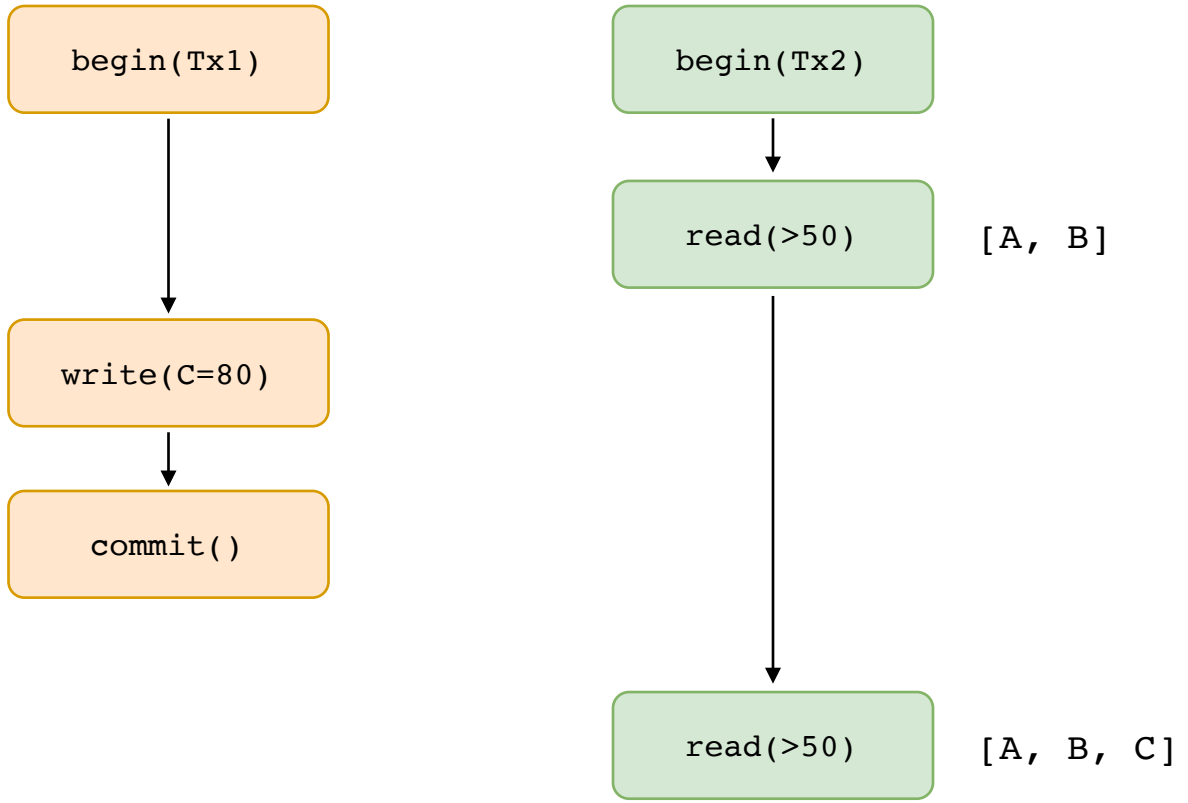
Phantom Reads



Phantom Reads



Phantom Reads



Read Phenomena - Bonus

- Dirty Reads
- Non-Repeatable Reads
- Phantom Reads

Read Phenomena - Bonus

- Dirty Reads
- Non-Repeatable Reads
- Phantom Reads
- Read Skews

Read Phenomena - Bonus

- Dirty Reads
- Non-Repeatable Reads
- Phantom Reads
- Read Skews
- Write Skews

Read Phenomena - Bonus

- Dirty Reads
- Non-Repeatable Reads
- Phantom Reads
- Read Skews
- Write Skews
- Lost Updates

Read Phenomena - Bonus

- Dirty Reads
- Non-Repeatable Reads
- Phantom Reads
- Read Skews
- Write Skews
- Lost Updates

- https://www.microsoft.com/en-us/research/wp-content/uploads/2016/02/tr-95-51.pdf

Transaction's Critical Section

```
BEGIN
SELECT id, balance FROM users WHERE id = 2
UPDATE users SET balance = balance - 20 WHERE id = 2
UPDATE users SET balance = balance + 20 WHERE id = 1
COMMIT
```

Transaction's Critical Section

```
BEGIN
SELECT id, balance FROM users WHERE id = 2
UPDATE users SET balance = balance - 20 WHERE id = 2
UPDATE users SET balance = balance + 20 WHERE id = 1
COMMIT
```

Vulnerable Pattern #1

```go
func (db *Db) Transfer(source int, destination int, amount int) error {
  ctx := context.Background()
  conn, err := pgx.Connect(ctx, db.databaseUrl)
  defer conn.Close(ctx)

  tx, err := conn.BeginTx(ctx)

  var user User
  err = conn.QueryRow(ctx, "SELECT id, name, balance FROM users WHERE id = $1", from).
    Scan(&user.Id, &user.Name, &user.Balance)

  if amount < 0 || user.Balance > amount {
    tx.Rollback(ctx)
    return fmt.Errorf("invalid transfer")
  }

  _, err = conn.Exec(ctx, "UPDATE users SET balance = balance - $2 WHERE id = $1", source, amount)
  _, err = conn.Exec(ctx, "UPDATE users SET balance = balance + $2 WHERE id = $1", destination, amount)

  err = tx.Commit(ctx)
  return nil
}
```

begin(Tx1)

begin(Tx2)

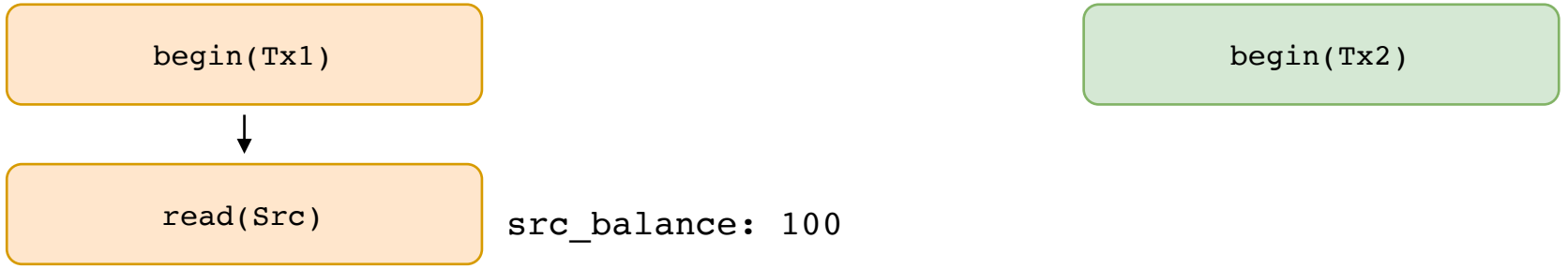
Scenario	
Source Balance :	100
Destination Balance:	20
Transfer Amount:	100

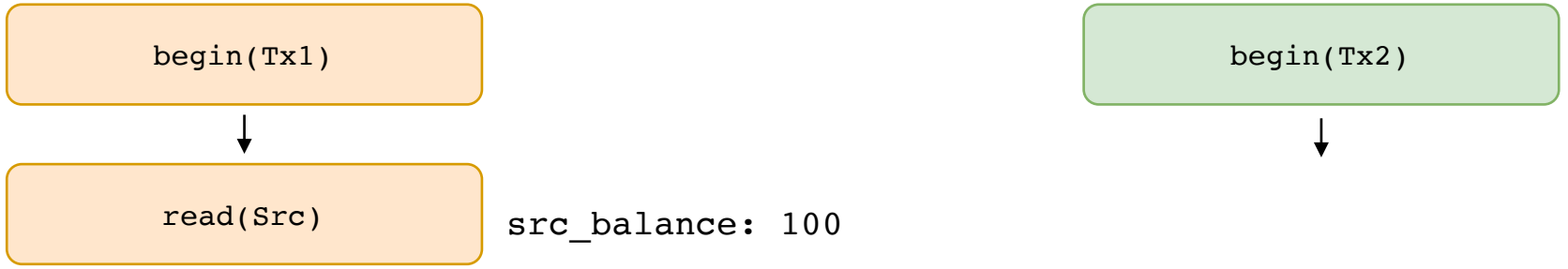
```
begin(Tx1)
```

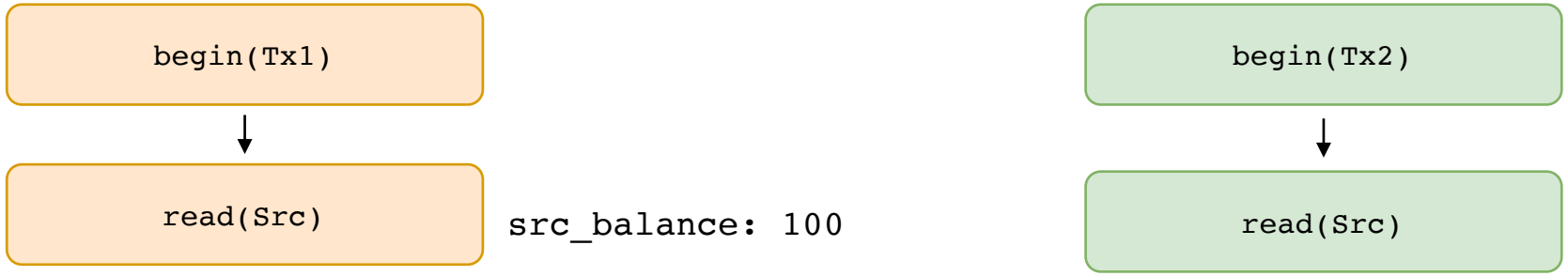
```
begin(Tx2)
```

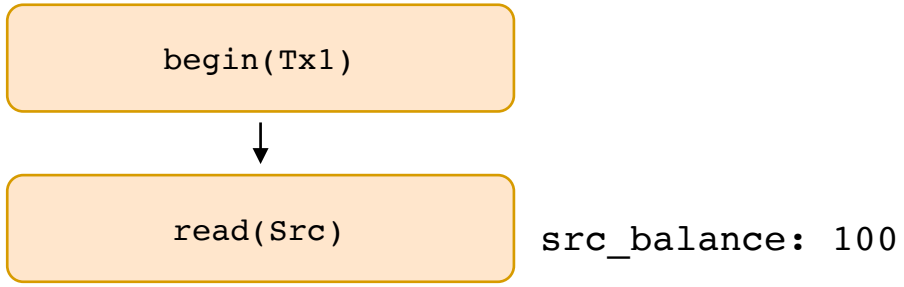

begin(Tx1)
begin(Tx2)

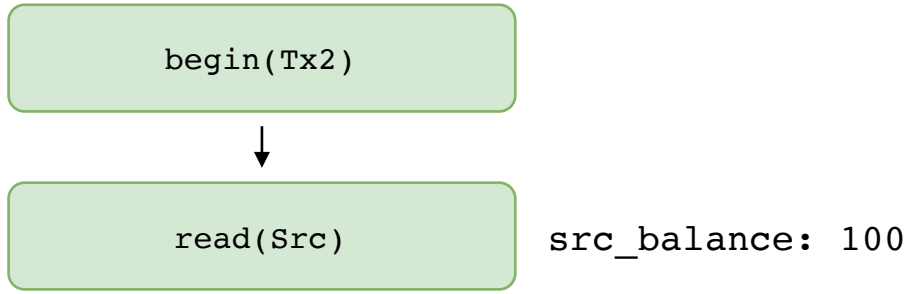

begin(Tx1)
read(Src)
begin(Tx2)

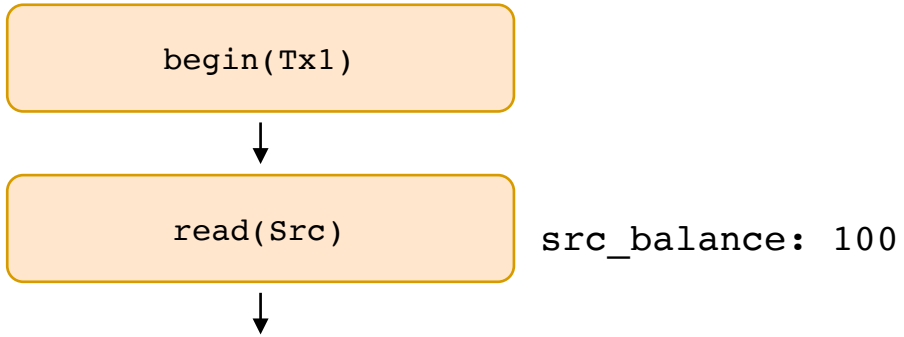

begin(Tx1)
read(Src)
src_balance: 100
begin(Tx2)

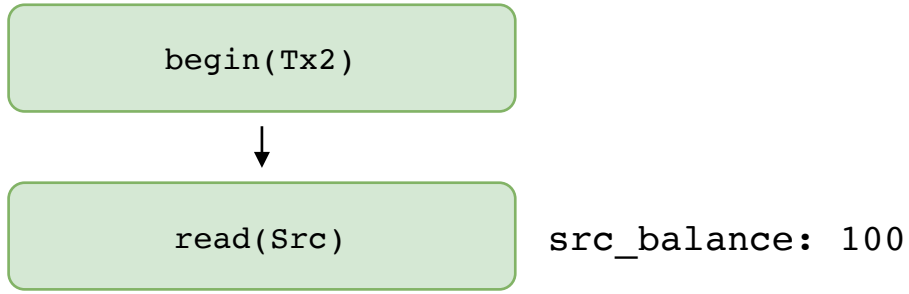

begin(Tx1)
read(Src)
src_balance: 100
begin(Tx2)

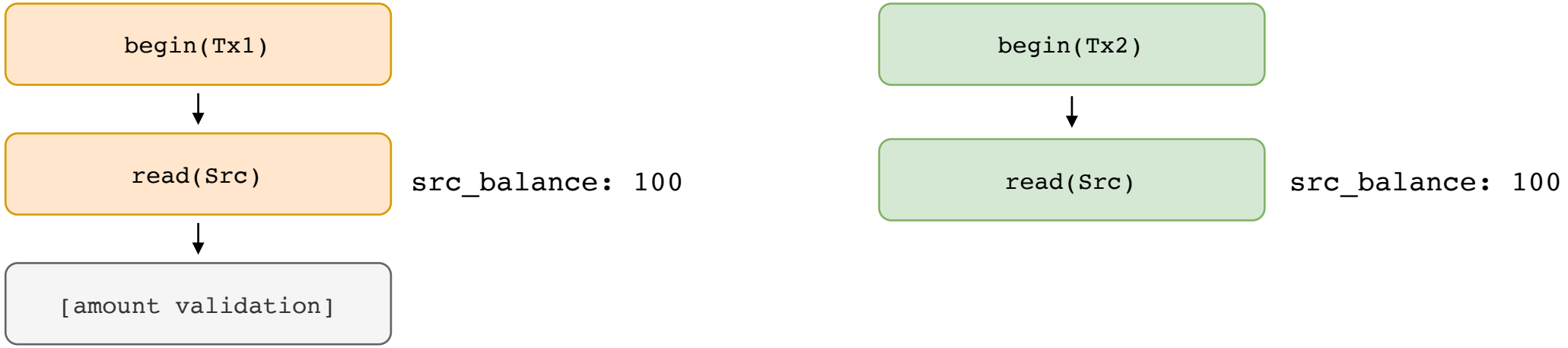

begin(Tx1)
read(Src)
src_balance: 100
begin(Tx2)
read(Src)

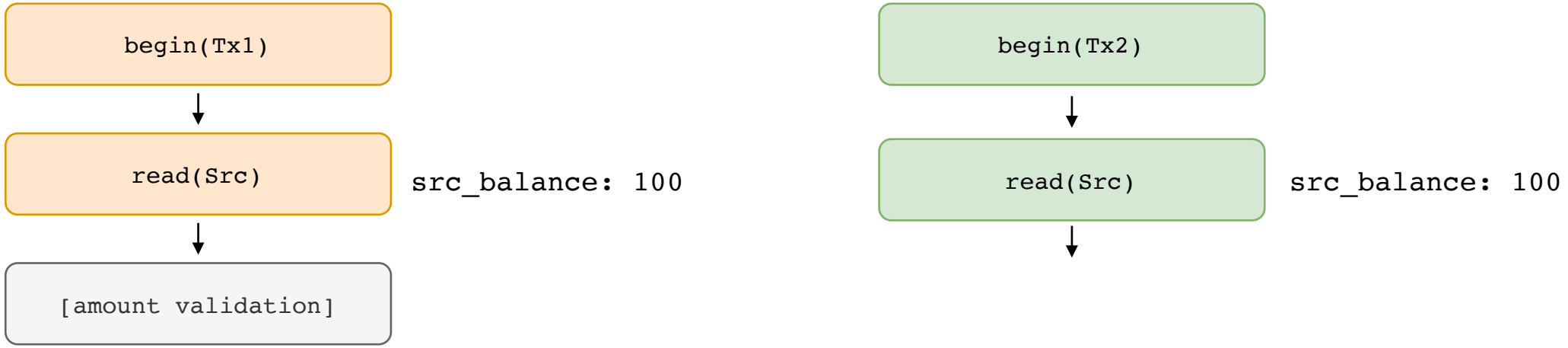

begin(Tx1)
read(Src)
src_balance: 100

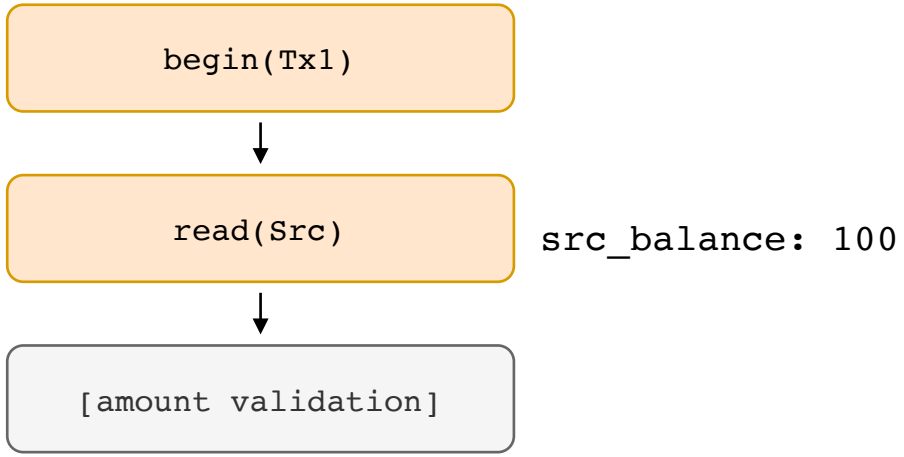

begin(Tx2)
read(Src)
src_balance: 100

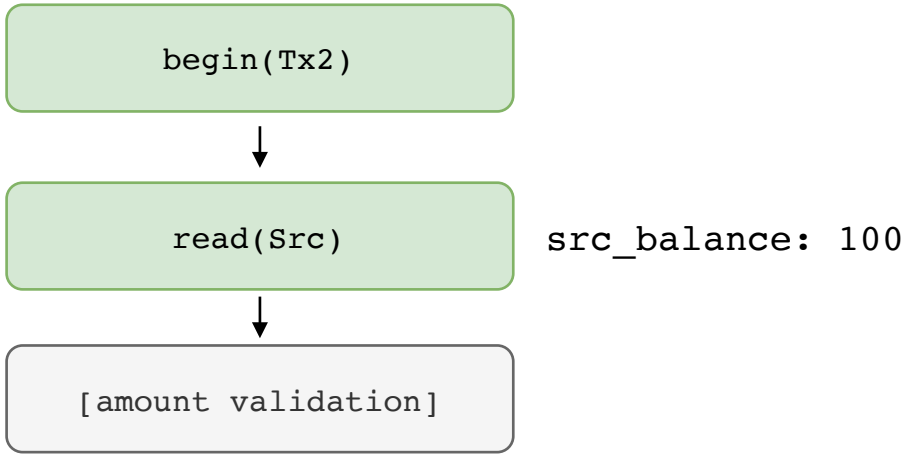

begin(Tx1)
read(Src)
src_balance: 100

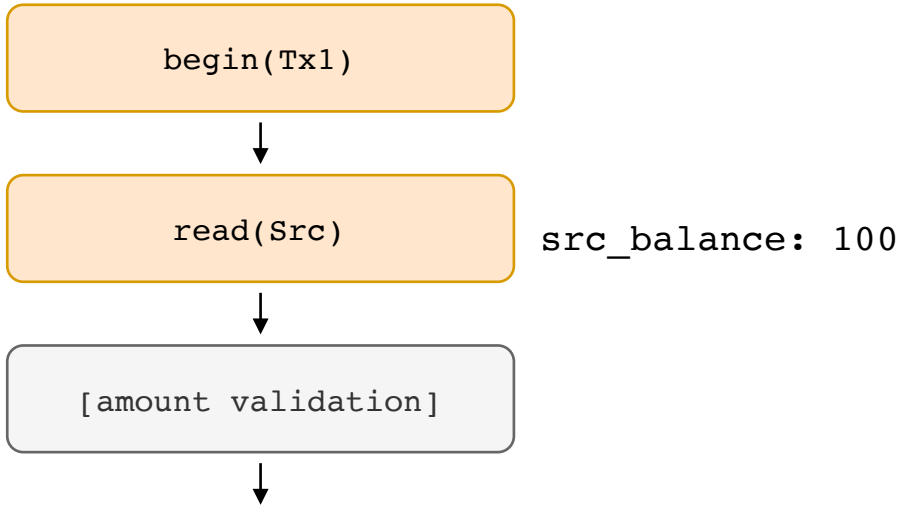

begin(Tx2)
read(Src)
src_balance: 100


begin(Tx1)
read(Src)
src_balance: 100
[amount validation]
begin(Tx2)
read(Src)
src_balance: 100

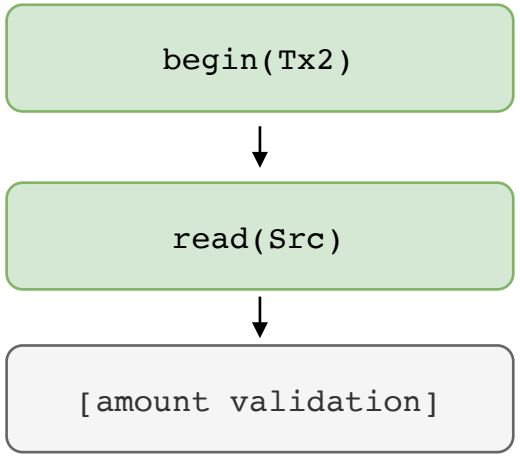

begin(Tx1)
read(Src)
src_balance: 100
[amount validation]
begin(Tx2)
read(Src)
src_balance: 100


begin(Tx1)
read(Src)
[amount validation]
src_balance: 100

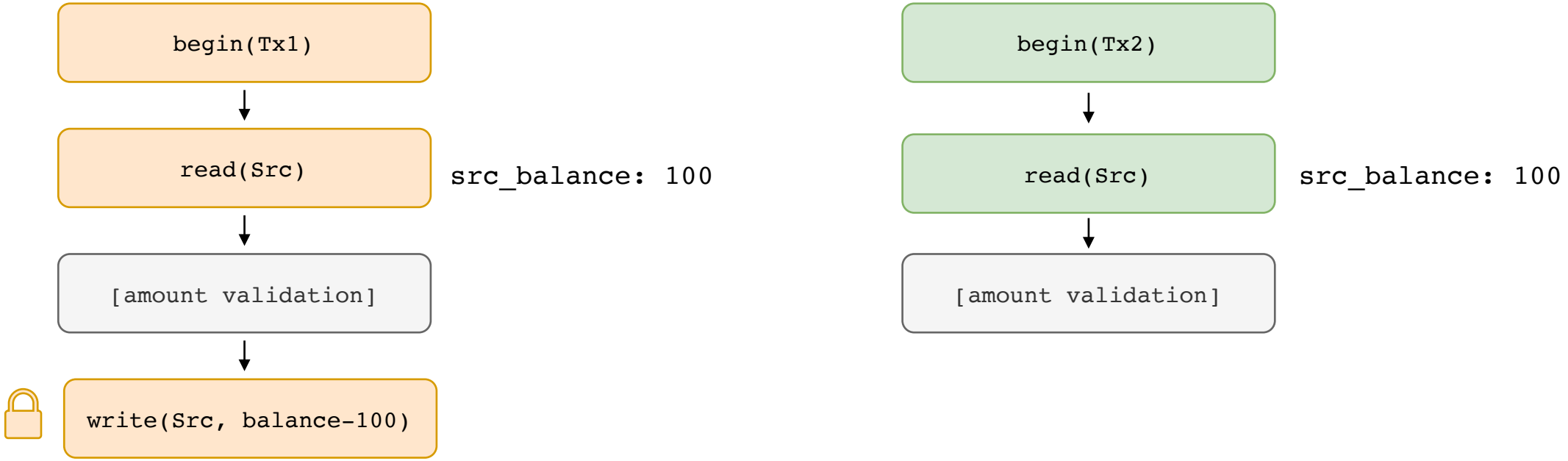

begin(Tx2)
read(Src)
[amount validation]
src_balance: 100

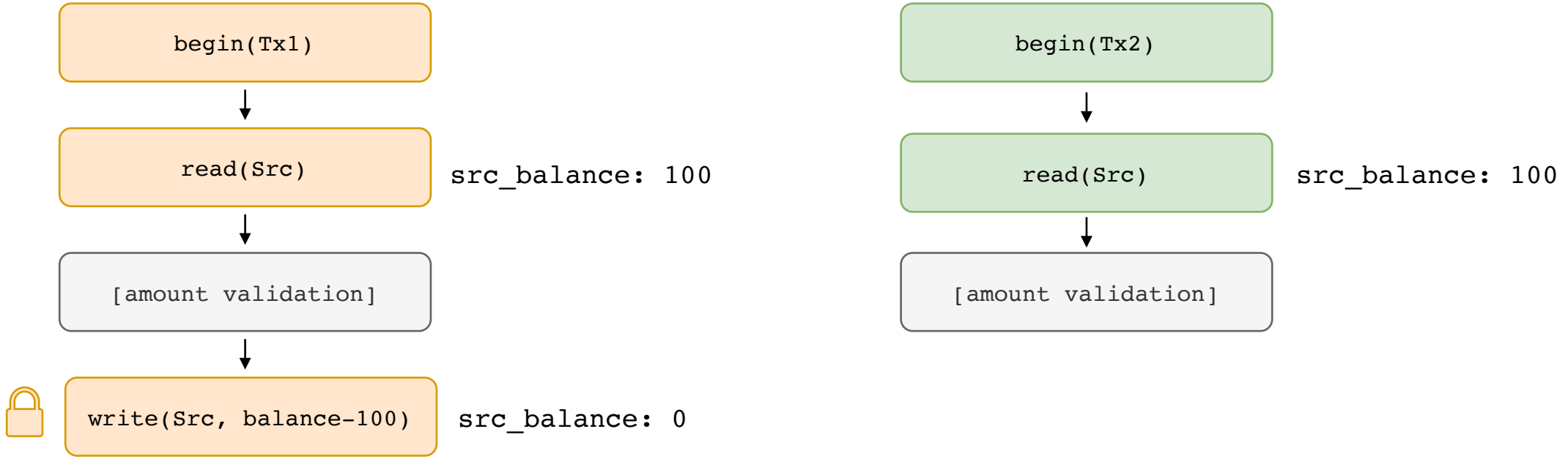


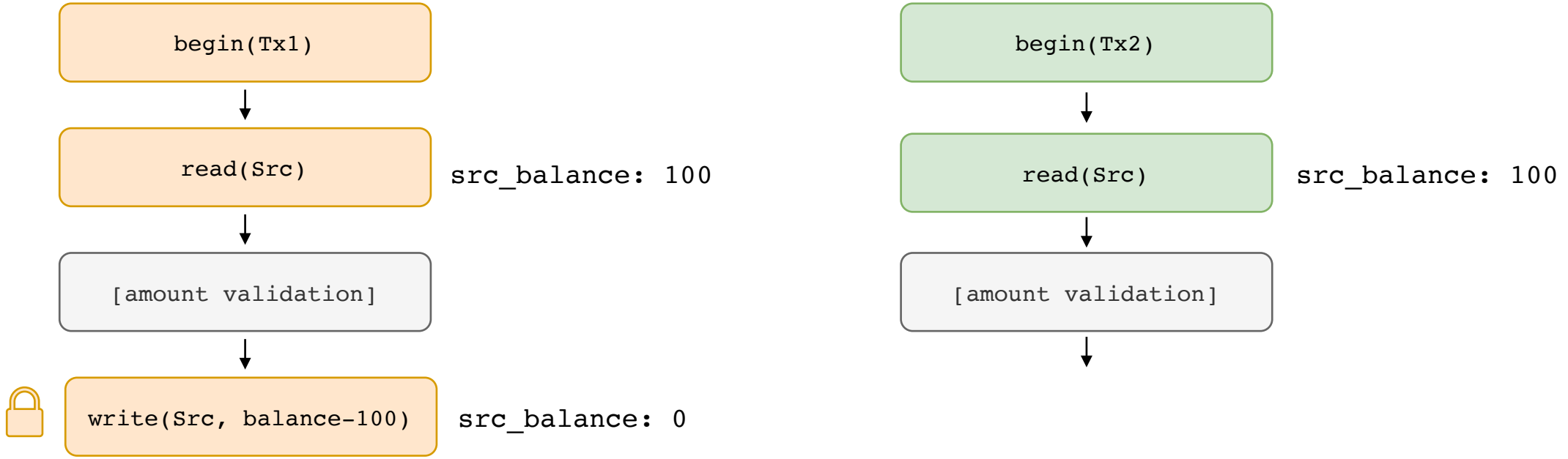
src_balance: 100

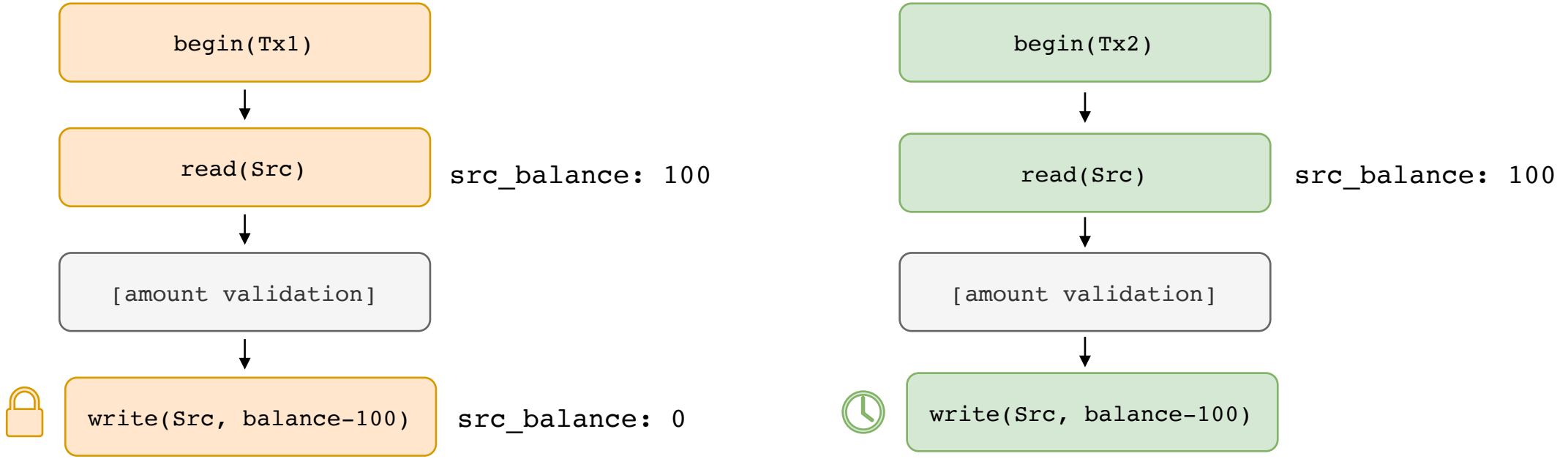


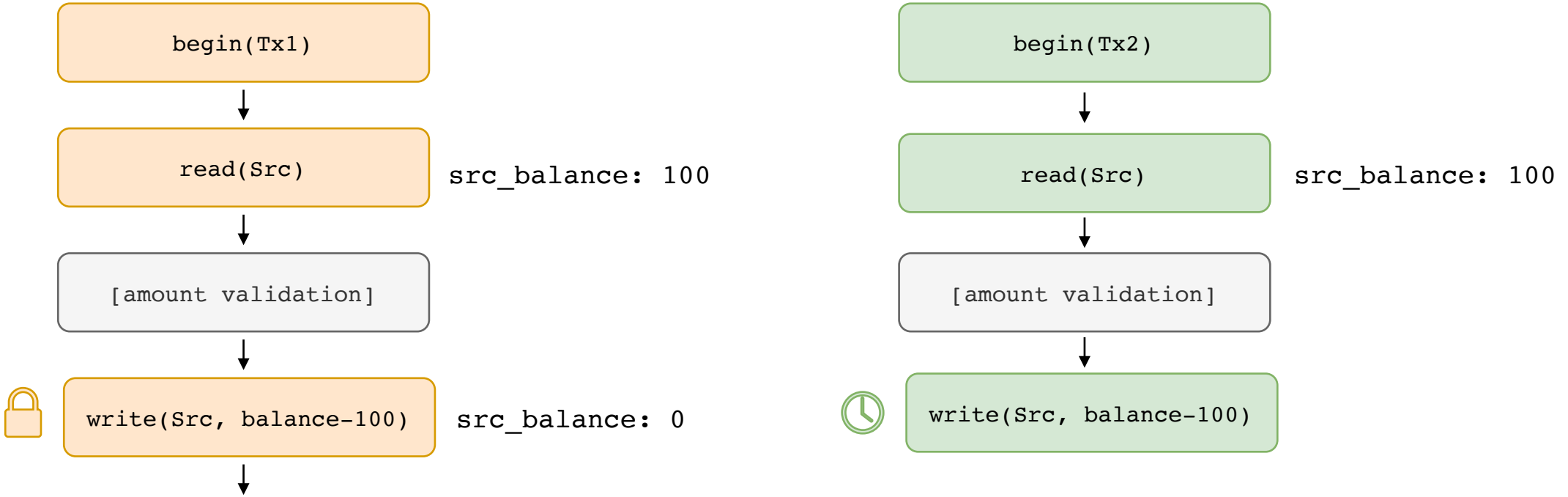
src_balance: 100


begin(Tx1)
read(Src)
src_balance: 100
[amount validation]
write(Src, balance-100)
begin(Tx2)
read(Src)
src_balance: 100
[amount validation]


begin(Tx1)
read(Src)
src_balance: 100
[amount validation]
write(Src, balance-100)
src_balance: 0
begin(Tx2)
read(Src)
src_balance: 100
[amount validation]


begin(Tx1)
read(Src)
src_balance: 100
[amount validation]
write(Src, balance-100)
src_balance: 0
begin(Tx2)
read(Src)
src_balance: 100
[amount validation]

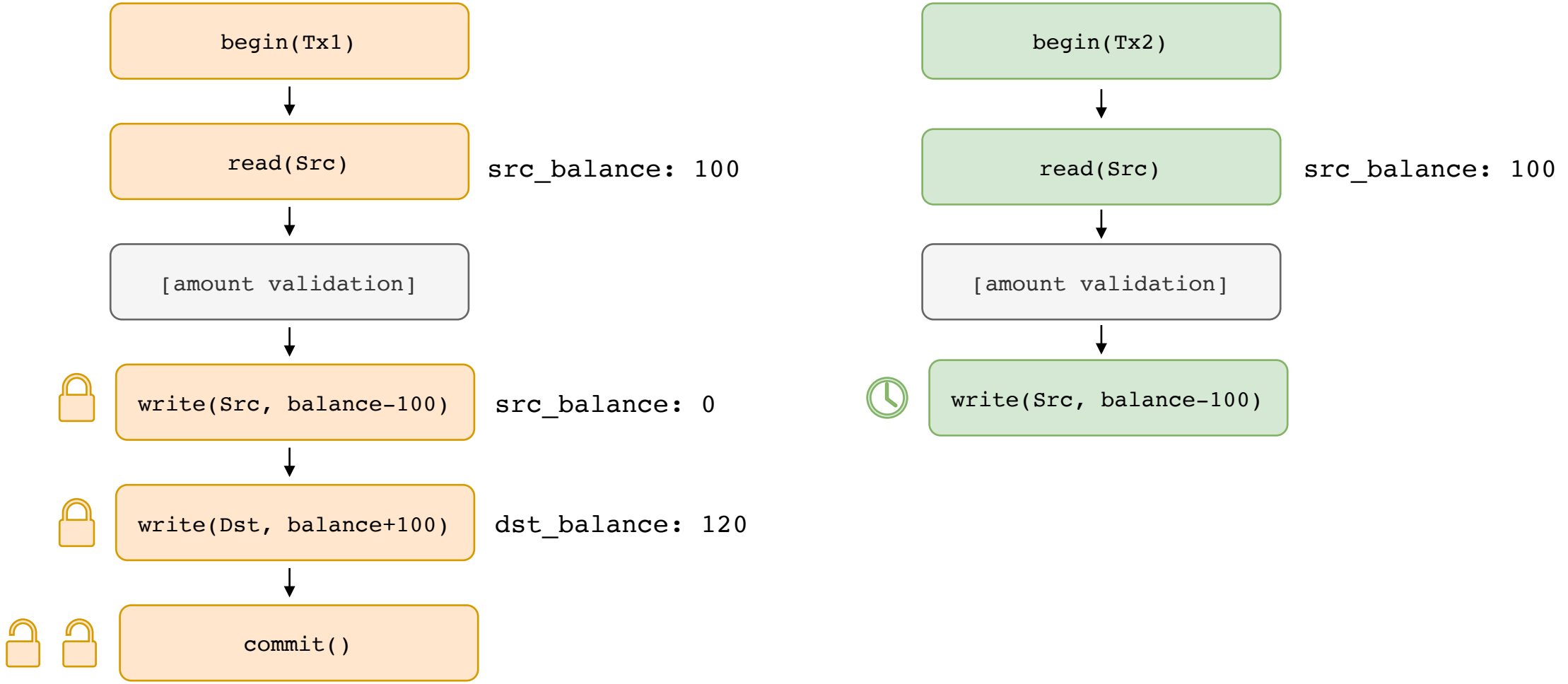

begin(Tx1)
read(Src)
src_balance: 100
[amount validation]
write(Src, balance-100)
src_balance: 0
begin(Tx2)
read(Src)
src_balance: 100
[amount validation]
write(Src, balance-100)

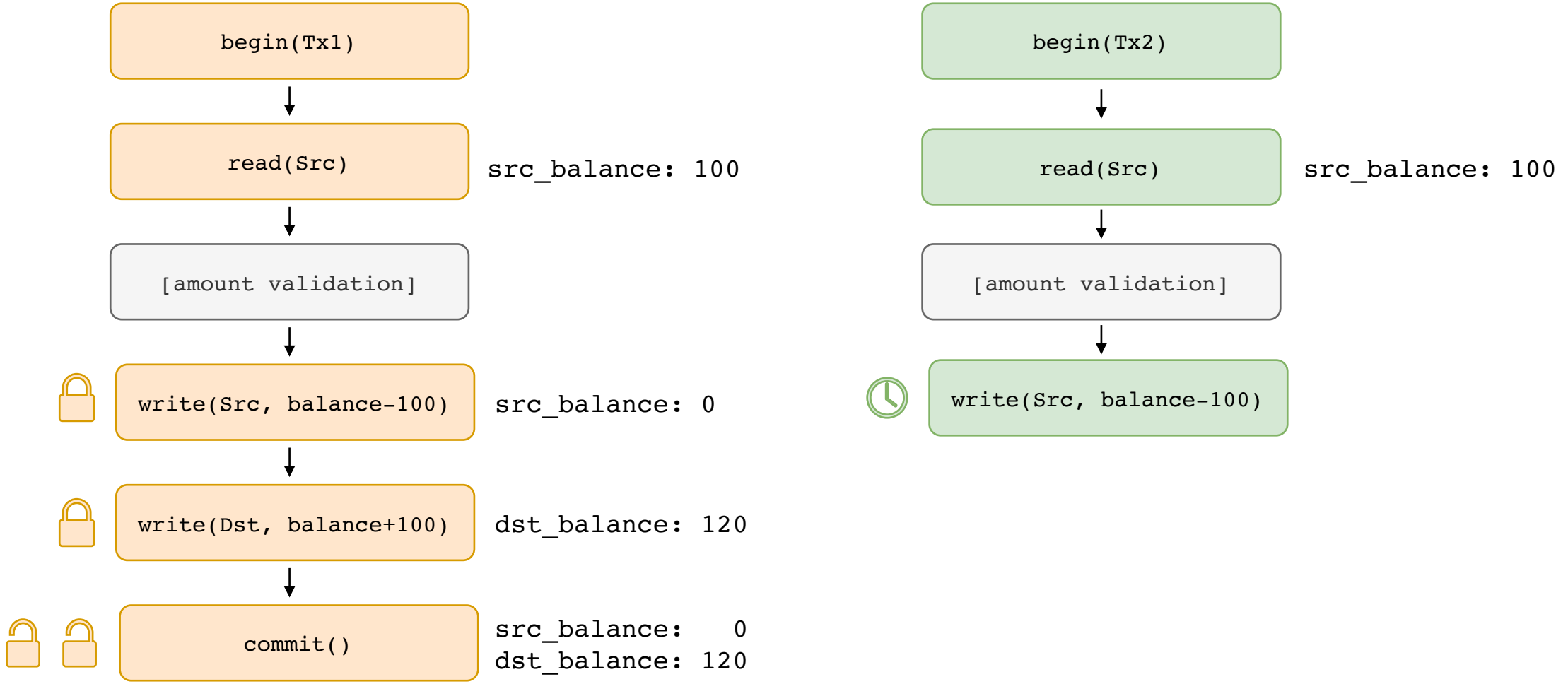

begin(Tx1)
read(Src)
src_balance: 100
[amount validation]
write(Src, balance-100)
src_balance: 0
begin(Tx2)
read(Src)
src_balance: 100
[amount validation]
write(Src, balance-100)

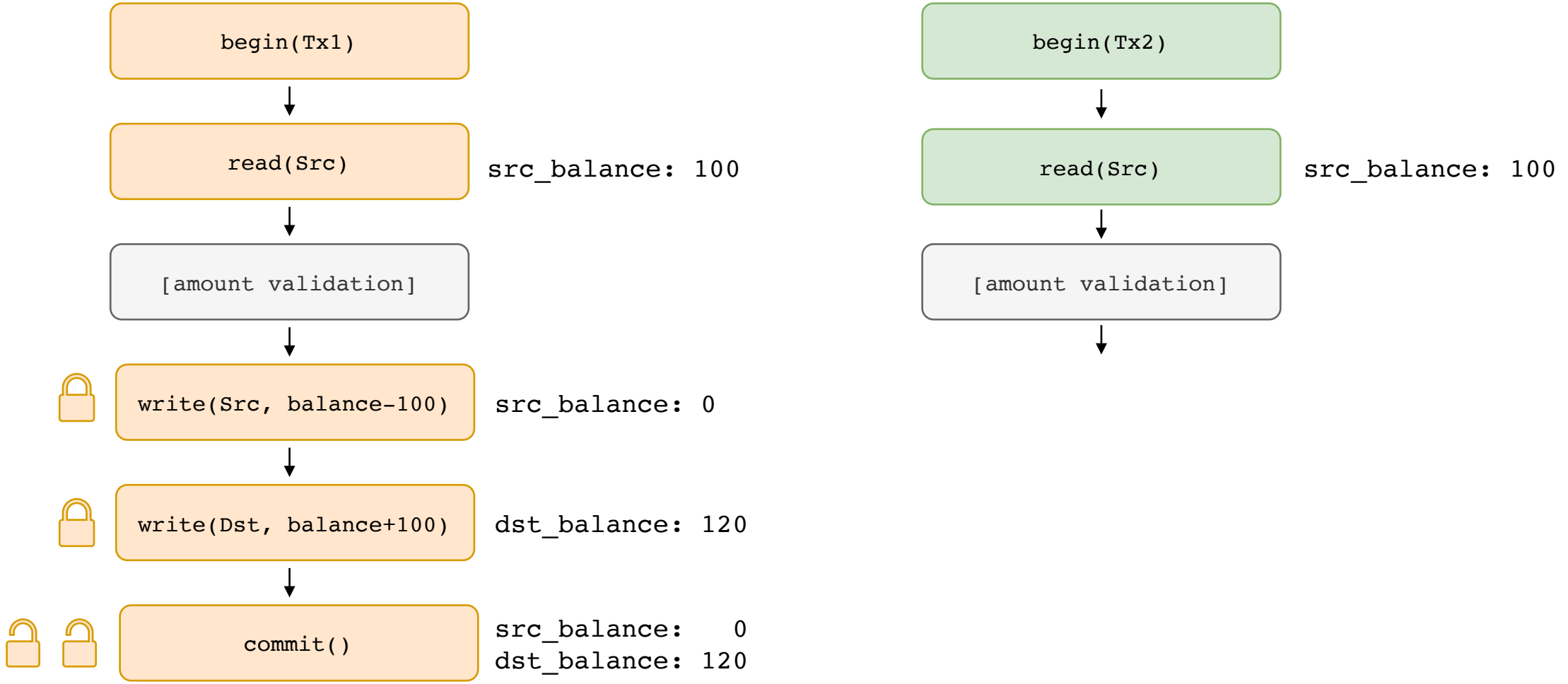

begin(Tx1)
read(Src)
src_balance: 100
[amount validation]
write(Src, balance-100)
src_balance: 0
write(Dst, balance+100)
begin(Tx2)
read(Src)
src_balance: 100
[amount validation]
write(Src, balance-100)

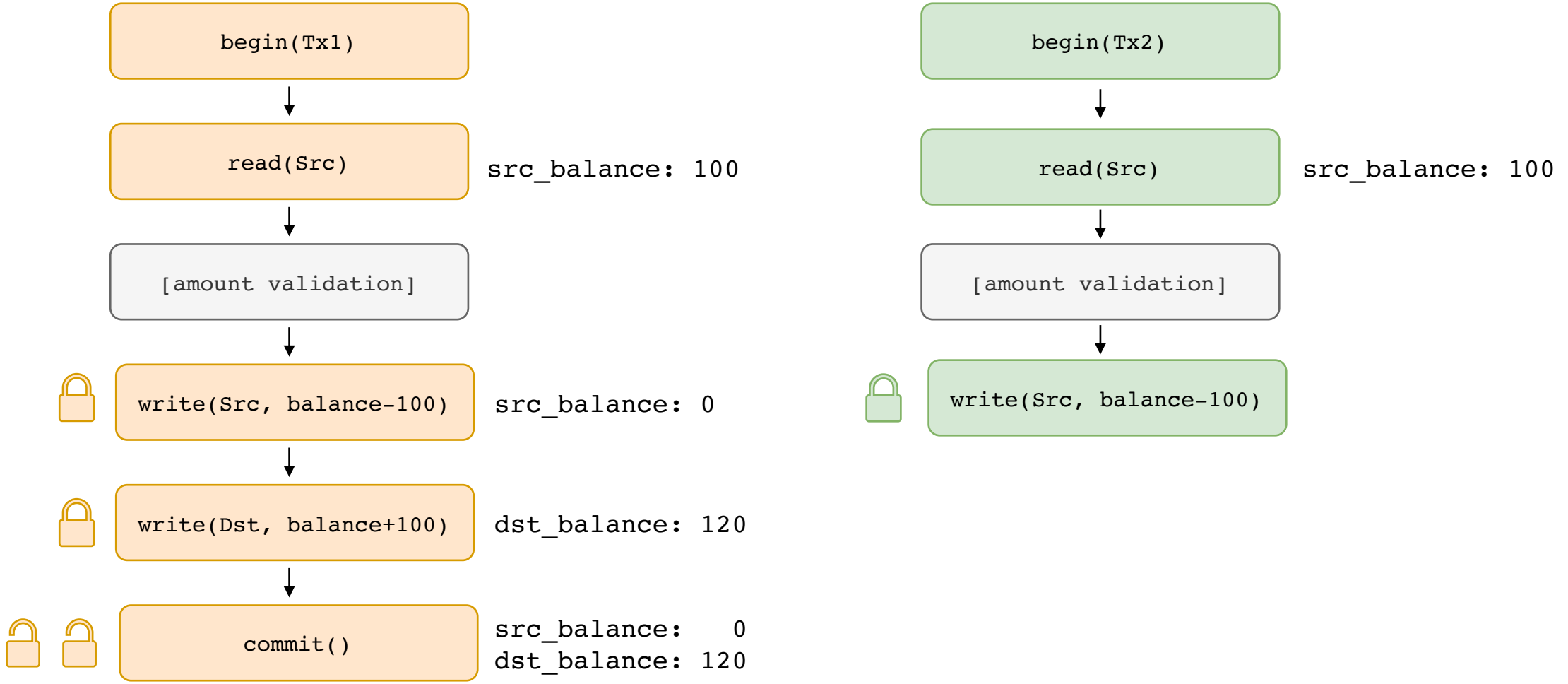

begin(Tx1)
read(Src)
src_balance: 100
[amount validation]
write(Src, balance-100)
src_balance: 0
write(Dst, balance+100)
dst_balance: 120
begin(Tx2)
read(Src)
src_balance: 100
[amount validation]
write(Src, balance-100)

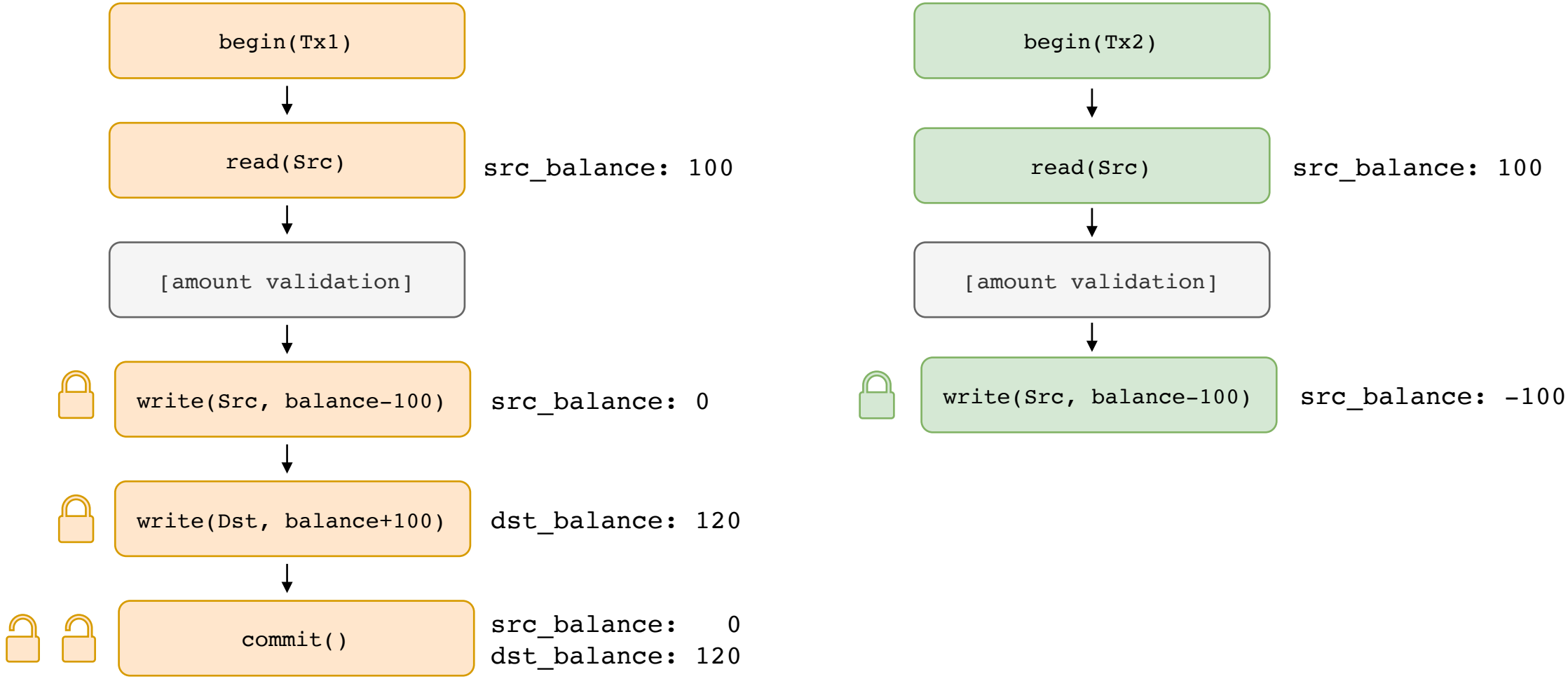

begin(Tx1)
read(Src)
src_balance: 100
[amount validation]
write(Src, balance-100)
src_balance: 0
write(Dst, balance+100)
dst_balance: 120
begin(Tx2)
read(Src)
src_balance: 100
[amount validation]
write(Src, balance-100)

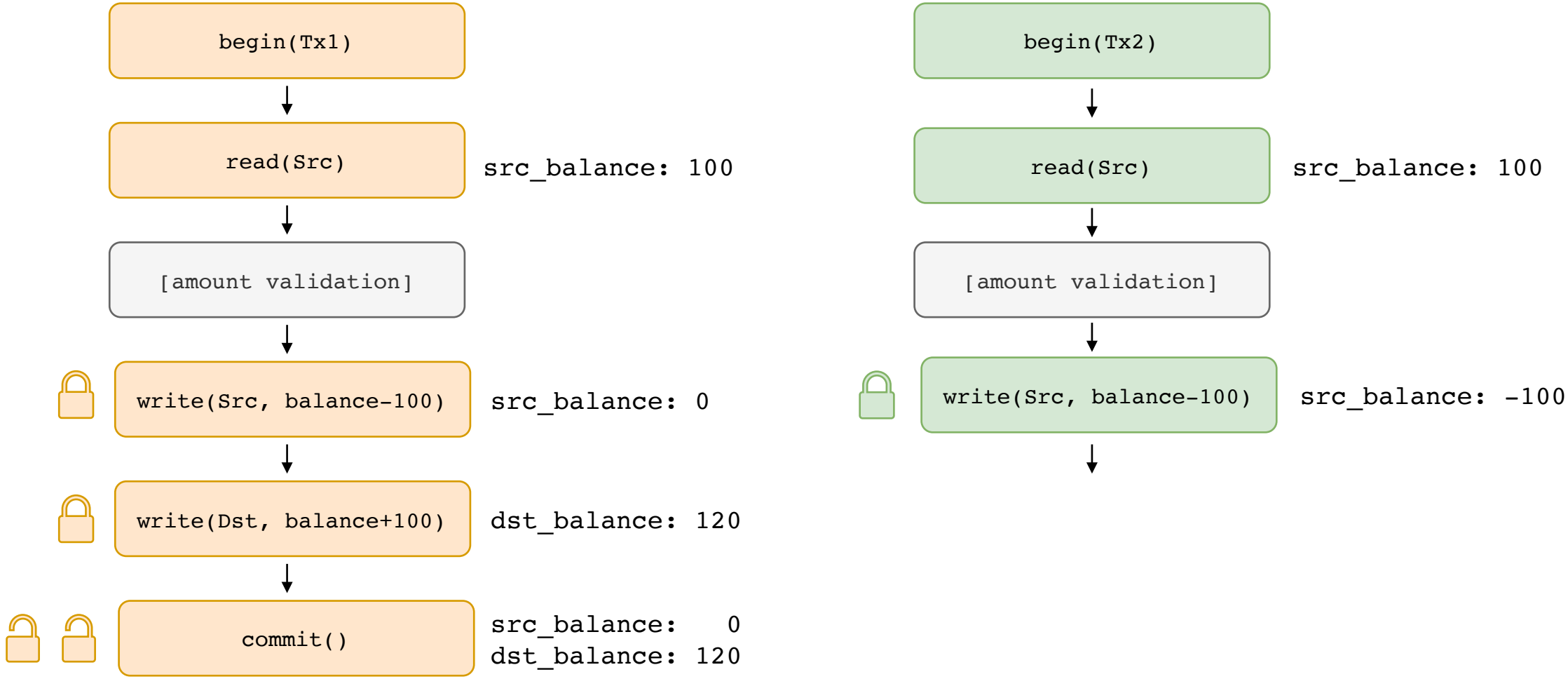

begin(Tx1)
read(Src)
src_balance: 100
[amount validation]
write(Src, balance-100)
src_balance: 0
write(Dst, balance+100)
dst_balance: 120
commit()
begin(Tx2)
read(Src)
src_balance: 100
[amount validation]
write(Src, balance-100)

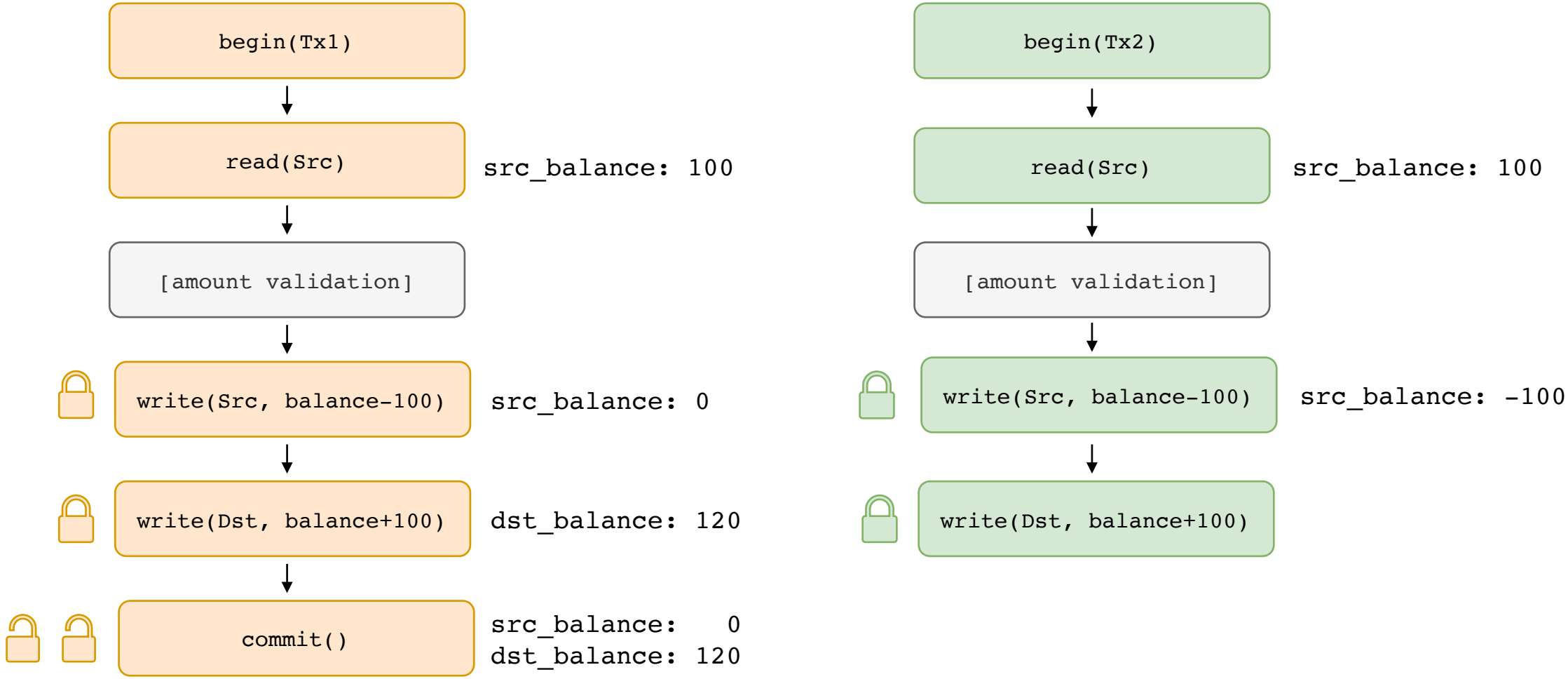

begin(Tx1)
read(Src)
src_balance: 100
[amount validation]
write(Src, balance-100)
src_balance: 0
write(Dst, balance+100)
dst_balance: 120
commit()
src_balance: 0
dst_balance: 120
begin(Tx2)
read(Src)
src_balance: 100
[amount validation]
write(Src, balance-100)

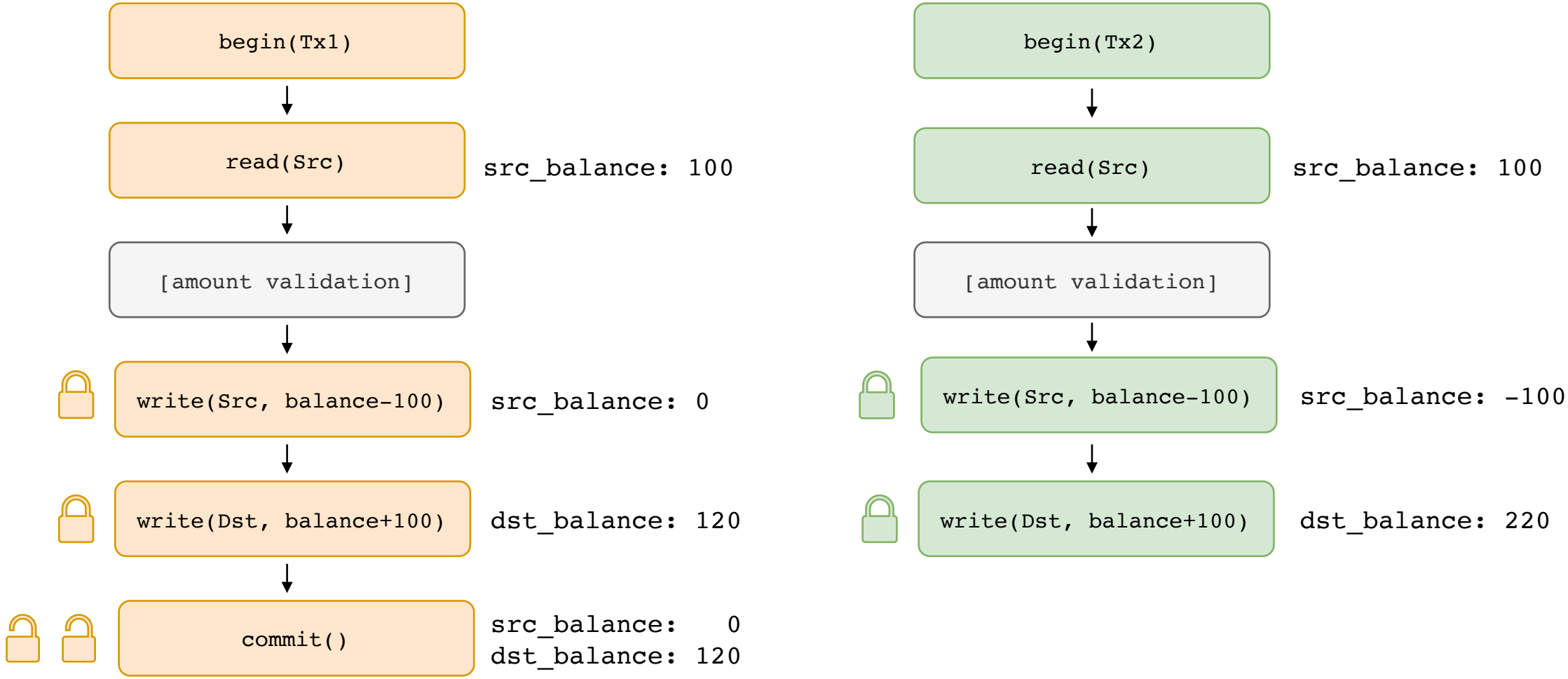

begin(Tx1)
read(Src)
src_balance: 100
[amount validation]
write(Src, balance-100)
src_balance: 0
write(Dst, balance+100)
dst_balance: 120
commit()
src_balance: 0
dst_balance: 120
begin(Tx2)
read(Src)
src_balance: 100
[amount validation]

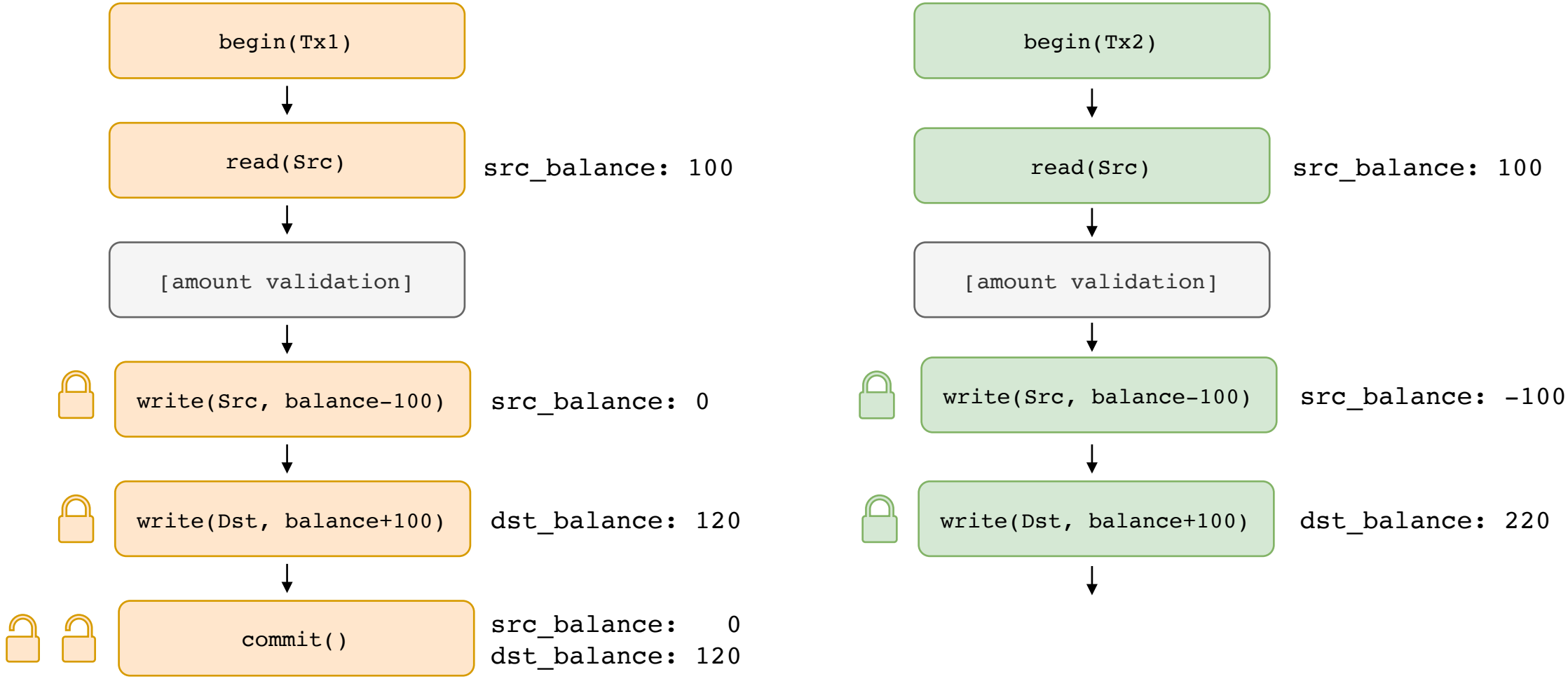

begin(Tx1)
read(Src)
src_balance: 100
[amount validation]
write(Src, balance-100)
src_balance: 0
write(Dst, balance+100)
dst_balance: 120
commit()
src_balance: 0
dst_balance: 120
begin(Tx2)
read(Src)
src_balance: 100
[amount validation]
write(Src, balance-100)

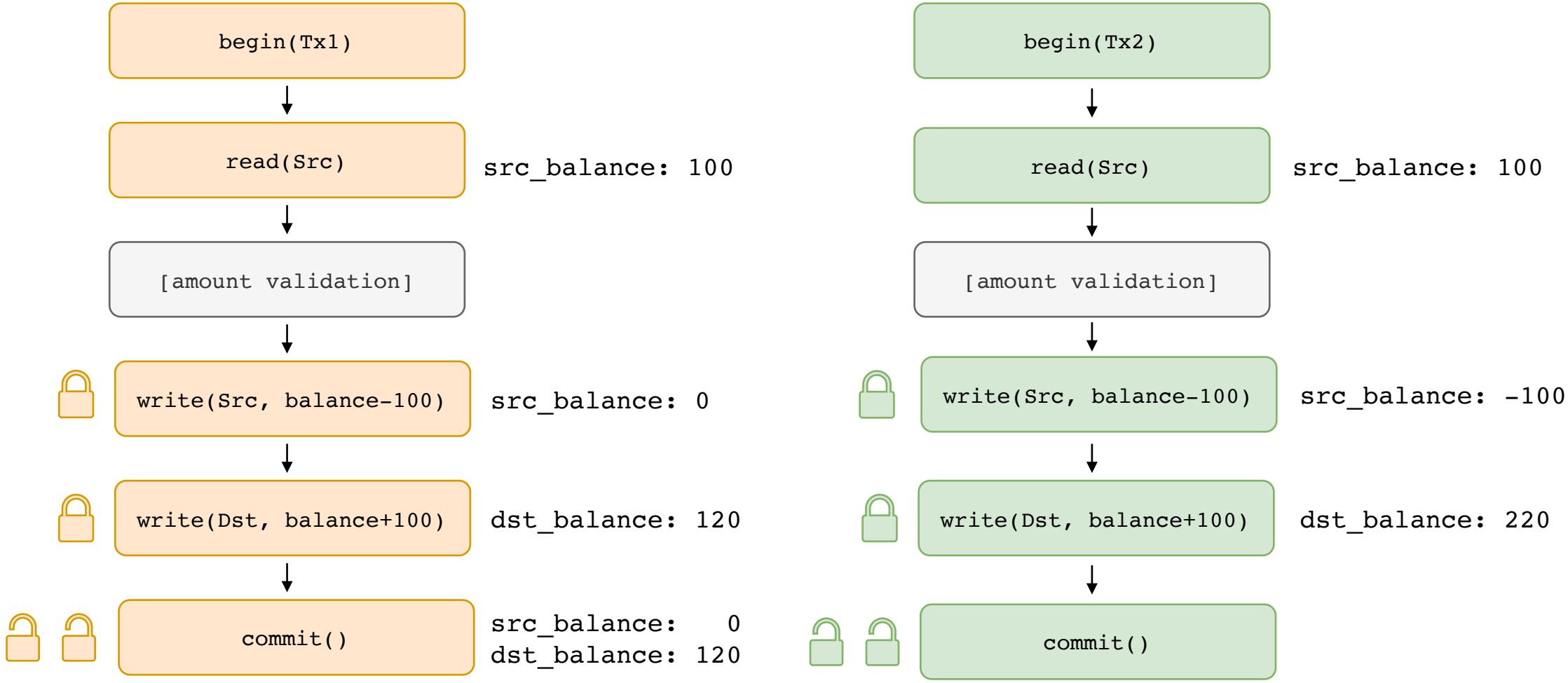

begin(Tx1)
read(Src)
src_balance: 100
[amount validation]
write(Src, balance-100)
src_balance: 0
write(Dst, balance+100)
dst_balance: 120
commit()
src_balance: 0
dst_balance: 120
begin(Tx2)
read(Src)
src_balance: 100
[amount validation]
write(Src, balance-100)
src_balance: -100

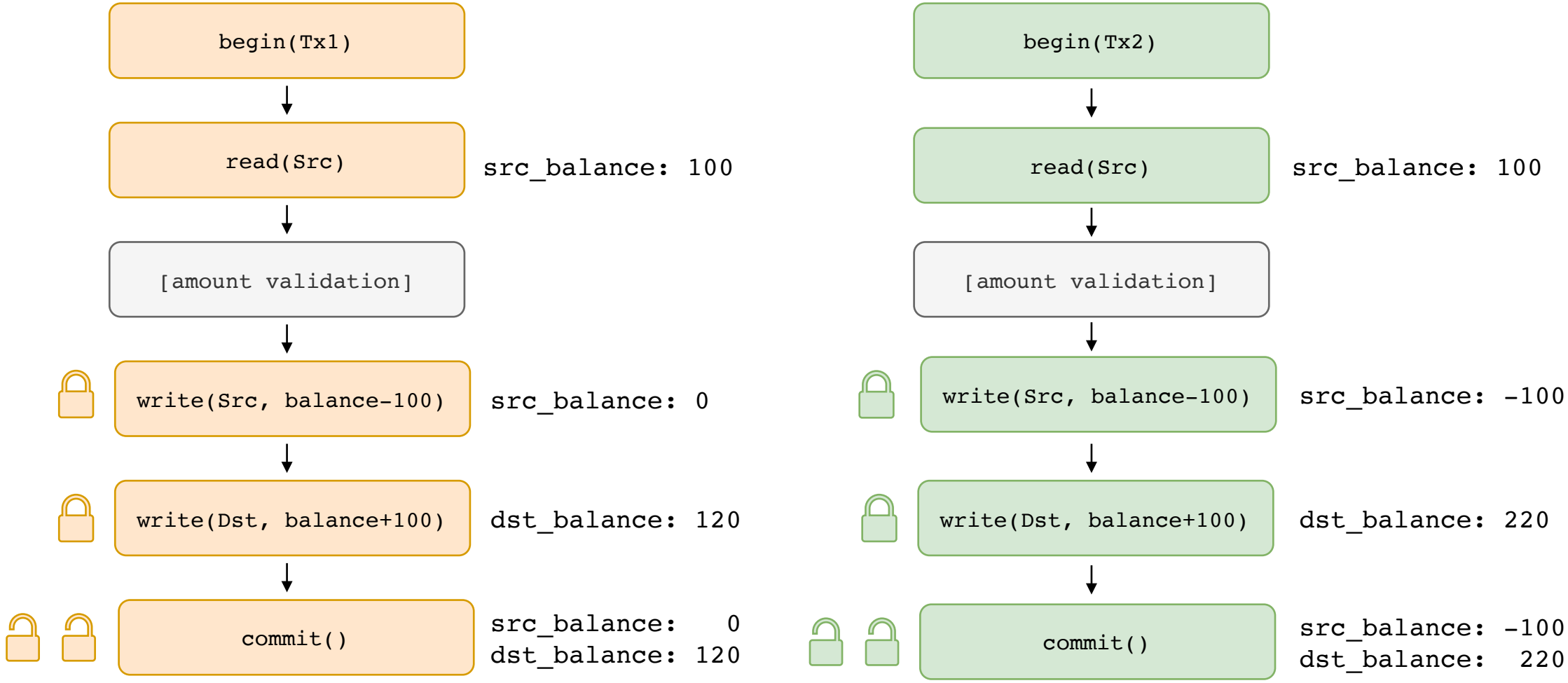

begin(Tx1)
read(Src)
src_balance: 100
[amount validation]
write(Src, balance-100)
src_balance: 0
write(Dst, balance+100)
dst_balance: 120
commit()
src_balance: 0
dst_balance: 120
begin(Tx2)
read(Src)
src_balance: 100
[amount validation]
write(Src, balance-100)
src_balance: -100


begin(Tx1)
read(Src)
src_balance: 100
[amount validation]
write(Src, balance-100)
src_balance: 0
write(Dst, balance+100)
dst_balance: 120
commit()
src_balance: 0
dst_balance: 120
begin(Tx2)
read(Src)
src_balance: 100
[amount validation]
write(Src, balance-100)
src_balance: -100
write(Dst, balance+100)


begin(Tx1)
read(Src)
src_balance: 100
[amount validation]
write(Src, balance-100)
src_balance: 0
write(Dst, balance+100)
dst_balance: 120
commit()
src_balance: 0
dst_balance: 120
begin(Tx2)
read(Src)
src_balance: 100
[amount validation]
write(Src, balance-100)
src_balance: -100
write(Dst, balance+100)
dst_balance: 220


begin(Tx1)
read(Src)
src_balance: 100
[amount validation]
write(Src, balance-100)
src_balance: 0
write(Dst, balance+100)
dst_balance: 120
commit()
src_balance: 0
dst_balance: 120
begin(Tx2)
read(Src)
src_balance: 100
[amount validation]
write(Src, balance-100)
src_balance: -100
write(Dst, balance+100)
dst_balance: 220


begin(Tx1)
read(Src)
src_balance: 100
[amount validation]
write(Src, balance-100)
src_balance: 0
write(Dst, balance+100)
dst_balance: 120
commit()
src_balance: 0
dst_balance: 120
begin(Tx2)
read(Src)
src_balance: 100
[amount validation]
write(Src, balance-100)
src_balance: -100
write(Dst, balance+100)
dst_balance: 220
commit()


begin(Tx1)
read(Src)
src_balance: 100
[amount validation]
write(Src, balance-100)
src_balance: 0
write(Dst, balance+100)
dst_balance: 120
commit()
src_balance: 0
dst_balance: 120
begin(Tx2)
read(Src)
src_balance: 100
[amount validation]
write(Src, balance-100)
src_balance: -100
write(Dst, balance+100)
dst_balance: 220
commit()
src_balance: -100
dst_balance: 220

Vulnerable Pattern #2

```go
func (self *Db) Transfer(source int, destination int, amount int) error {
  // ---snip---
  defer conn.Close(ctx)

  tx, err := conn.BeginTx(ctx)

  var srcUser, destUser User
  err = conn.QueryRow(ctx, "SELECT id, balance FROM users WHERE id = $1", source).
    Scan(&srcUser.Id, &srcUser.Balance)
  err = conn.QueryRow(ctx, "SELECT id, balance FROM users WHERE id = $1", destination).
    Scan(&destUser.Id, &destUser.Balance)

  if amount < 0 || user.Balance > amount {
    tx.Rollback(ctx)
    return fmt.Errorf("invalid transfer")
  }

  newSrcBalance := srcUser.Balance - amount
  newDestBalance := destUser.Balance + amount

  _, err = conn.Exec(ctx, "UPDATE users SET balance = $2 WHERE id = $1", source, newSrcBalance)
  _, err = conn.Exec(ctx, "UPDATE users SET balance = $2 WHERE id = $1", destination, newDestBalance)

  err = tx.Commit(ctx)
  return nil
}
```

begin(Tx1)

begin(Tx2)

Scenario

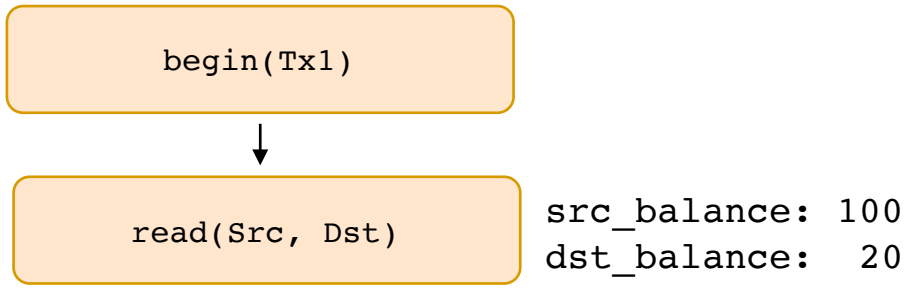
Source Balance :	100
Destination Balance:	20
Transfer Amount:	100

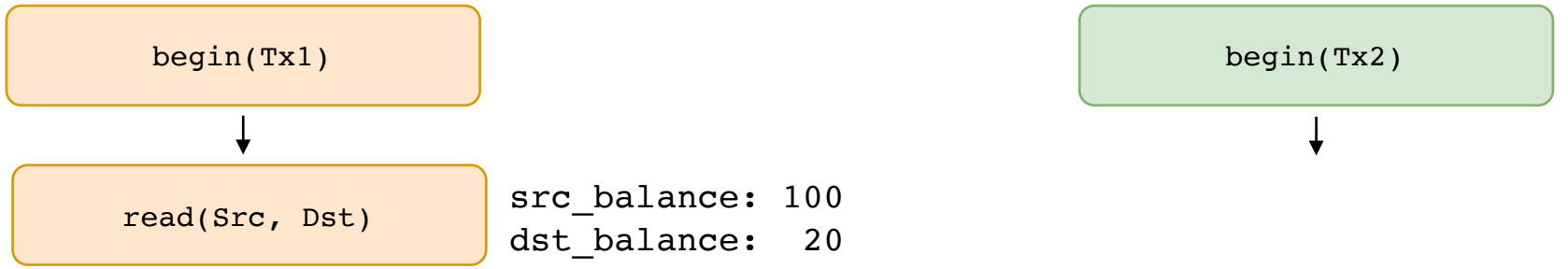
```
begin(Tx1)
```

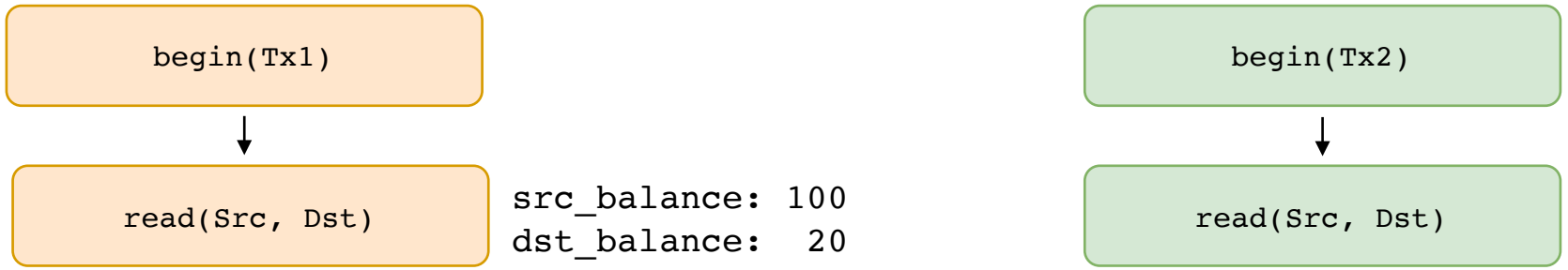
```
begin(Tx2)
```

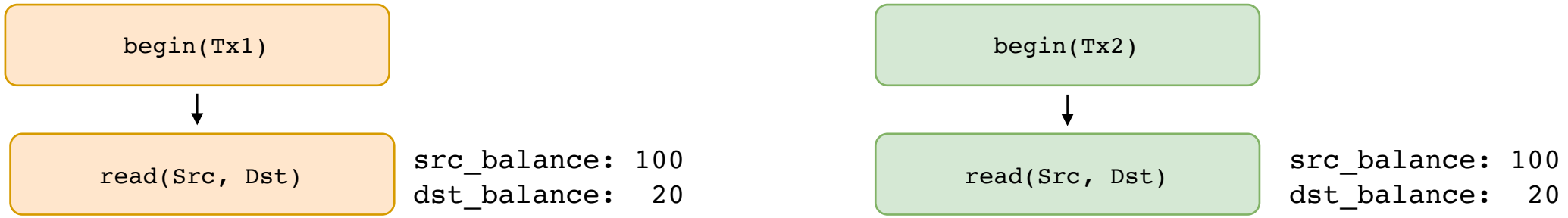

begin(Tx1)
begin(Tx2)


begin(Tx1)
read(Src, Dst)
begin(Tx2)

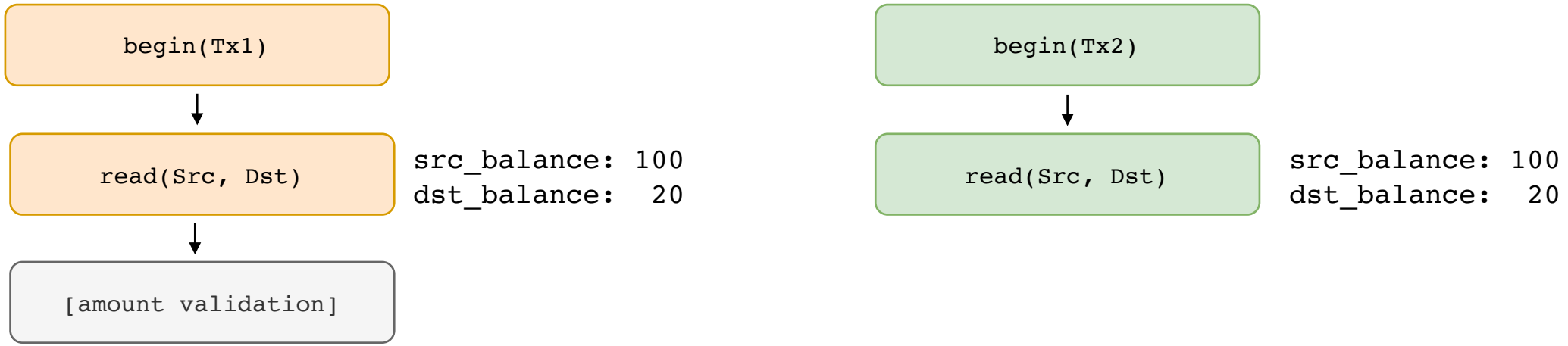

begin(Tx1)
read(Src, Dst)
src_balance: 100
dst_balance: 20
begin(Tx2)

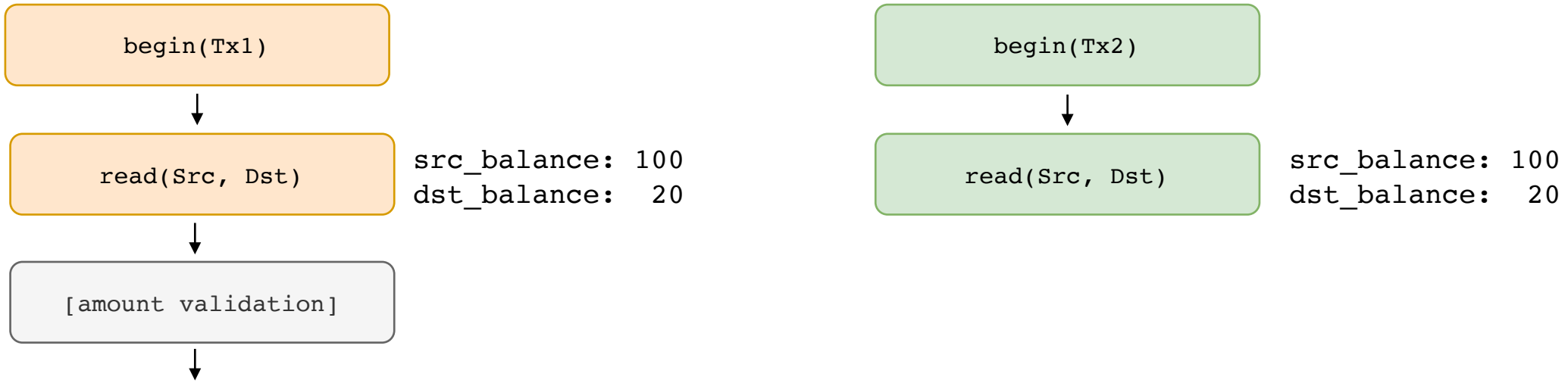

begin(Tx1)
read(Src, Dst)
src_balance: 100
dst_balance: 20
begin(Tx2)

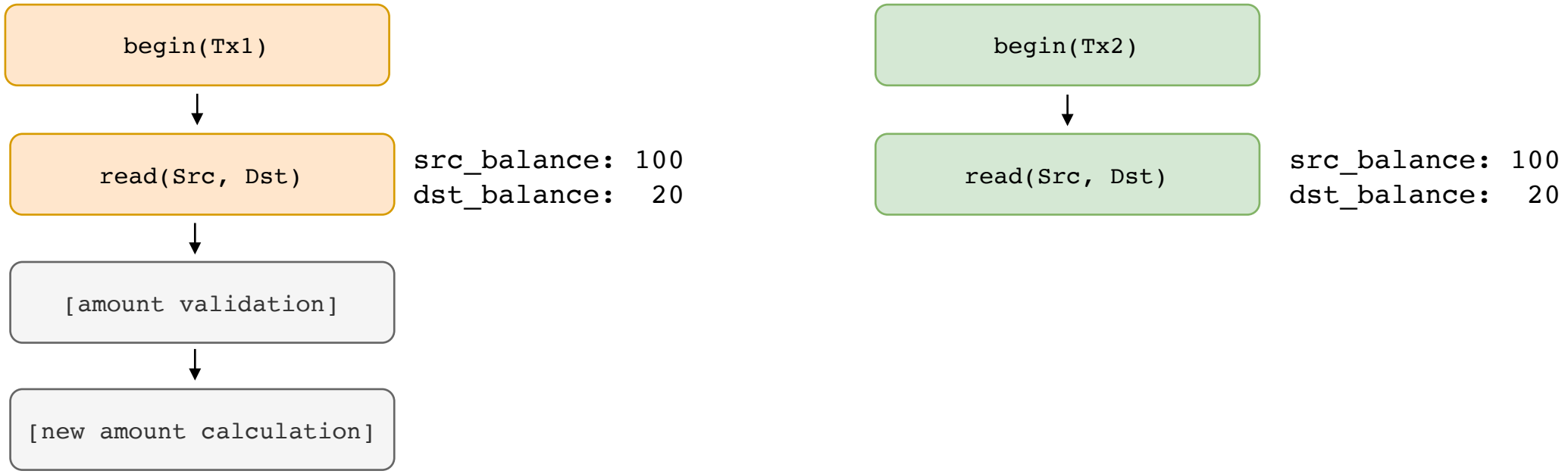

begin(Tx1)
read(Src, Dst)
src_balance: 100
dst_balance: 20
begin(Tx2)
read(Src, Dst)

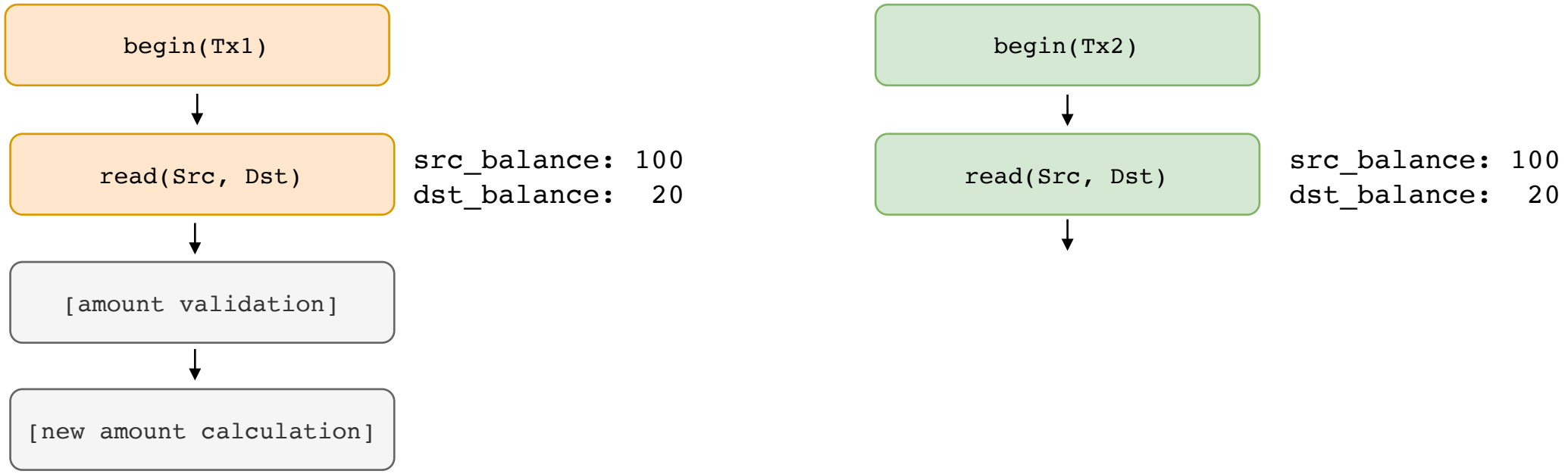

begin(Tx1)
read(Src, Dst)
src_balance: 100
dst_balance: 20
begin(Tx2)
read(Src, Dst)
src_balance: 100
dst_balance: 20


begin(Tx1)
read(Src, Dst)
src_balance: 100
dst_balance: 20
begin(Tx2)
read(Src, Dst)
src_balance: 100
dst_balance: 20

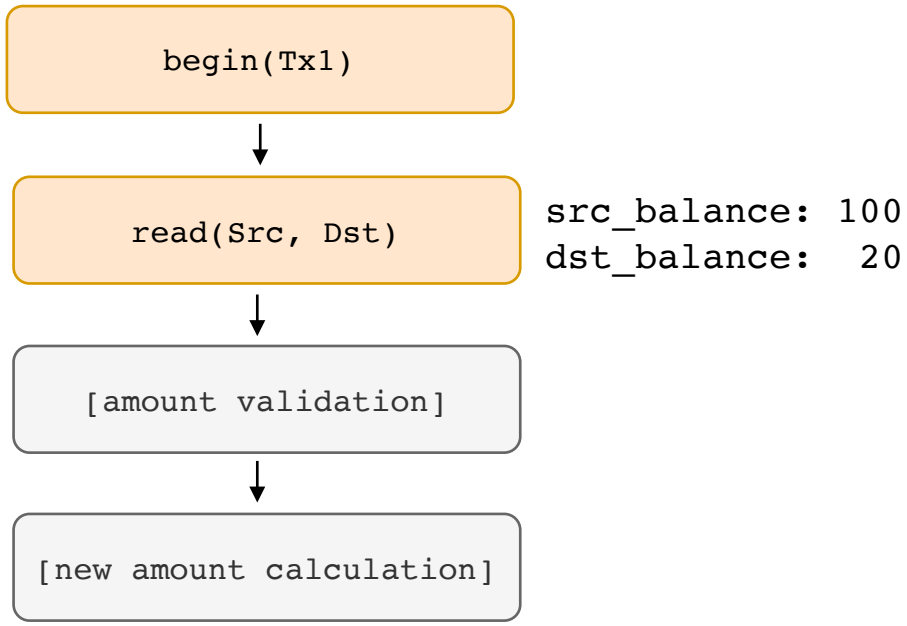

begin(Tx1)
read(Src, Dst)
[amount validation]
src_balance: 100
dst_balance: 20
begin(Tx2)
read(Src, Dst)
src_balance: 100
dst_balance: 20

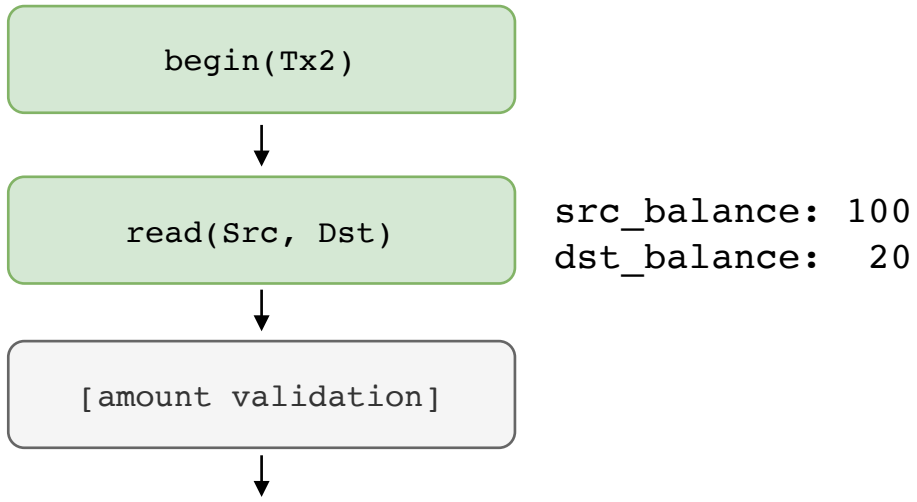

begin(Tx1)
read(Src, Dst)
src_balance: 100
dst_balance: 20
[amount validation]
begin(Tx2)
read(Src, Dst)
src_balance: 100
dst_balance: 20


begin(Tx1)
read(Src, Dst)
src_balance: 100
dst_balance: 20
[amount validation]
[new amount calculation]
begin(Tx2)
read(Src, Dst)
src_balance: 100
dst_balance: 20

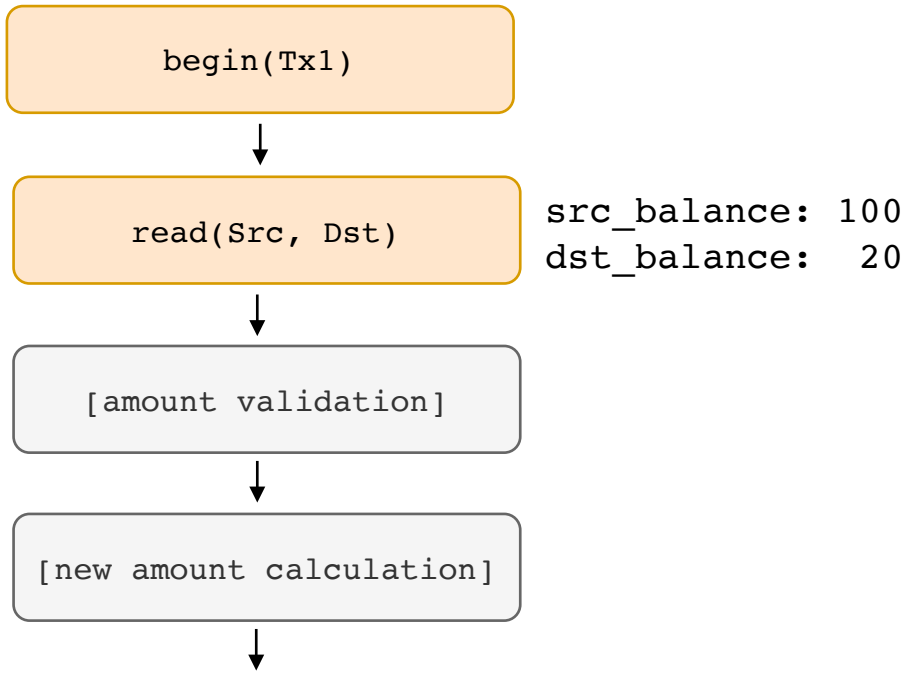

begin(Tx1)
read(Src, Dst)
src_balance: 100
dst_balance: 20
[amount validation]
[new amount calculation]
begin(Tx2)
read(Src, Dst)
src_balance: 100
dst_balance: 20

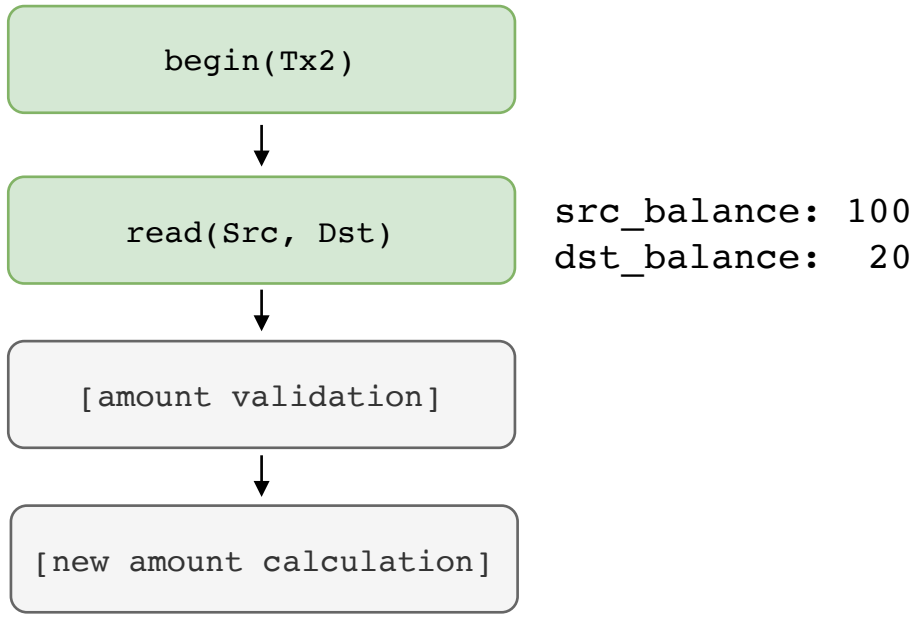

begin(Tx1)
read(Src, Dst)
src_balance: 100
dst_balance: 20
[amount validation]
[new amount calculation]
begin(Tx2)
read(Src, Dst)
src_balance: 100
dst_balance: 20
[amount validation]

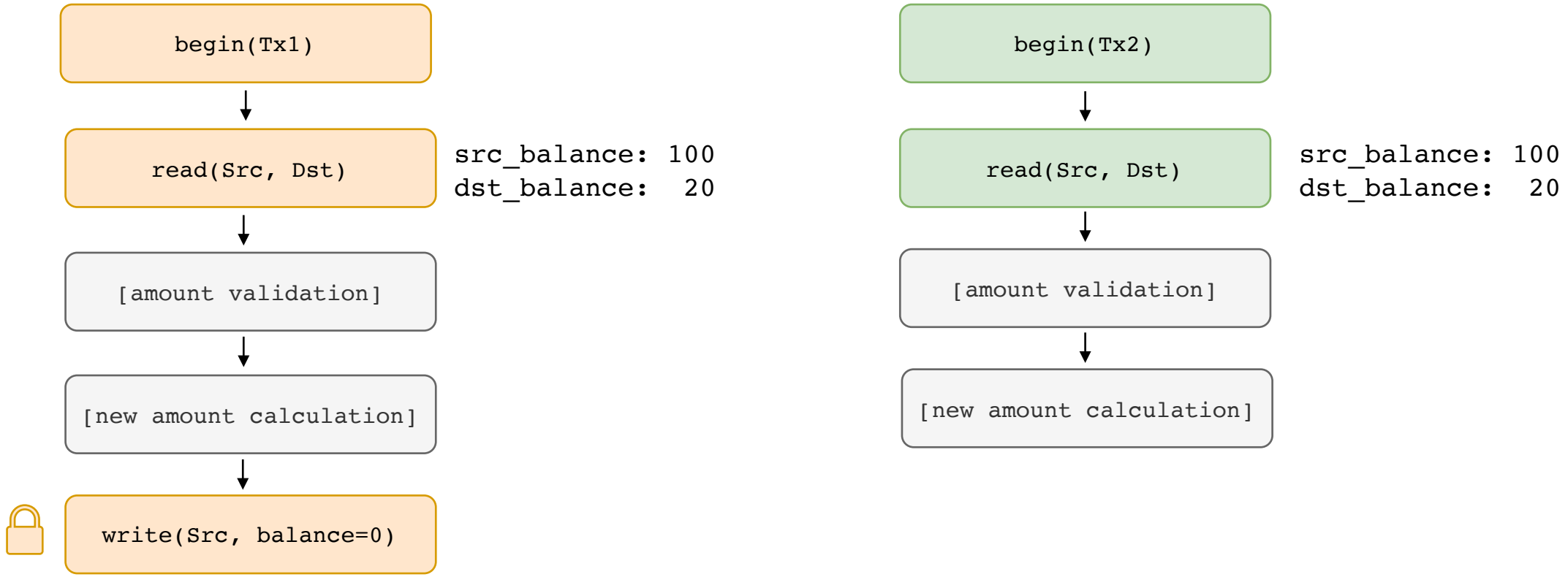

begin(Tx1)
read(Src, Dst)
src_balance: 100
dst_balance: 20
[amount validation]
[new amount calculation]

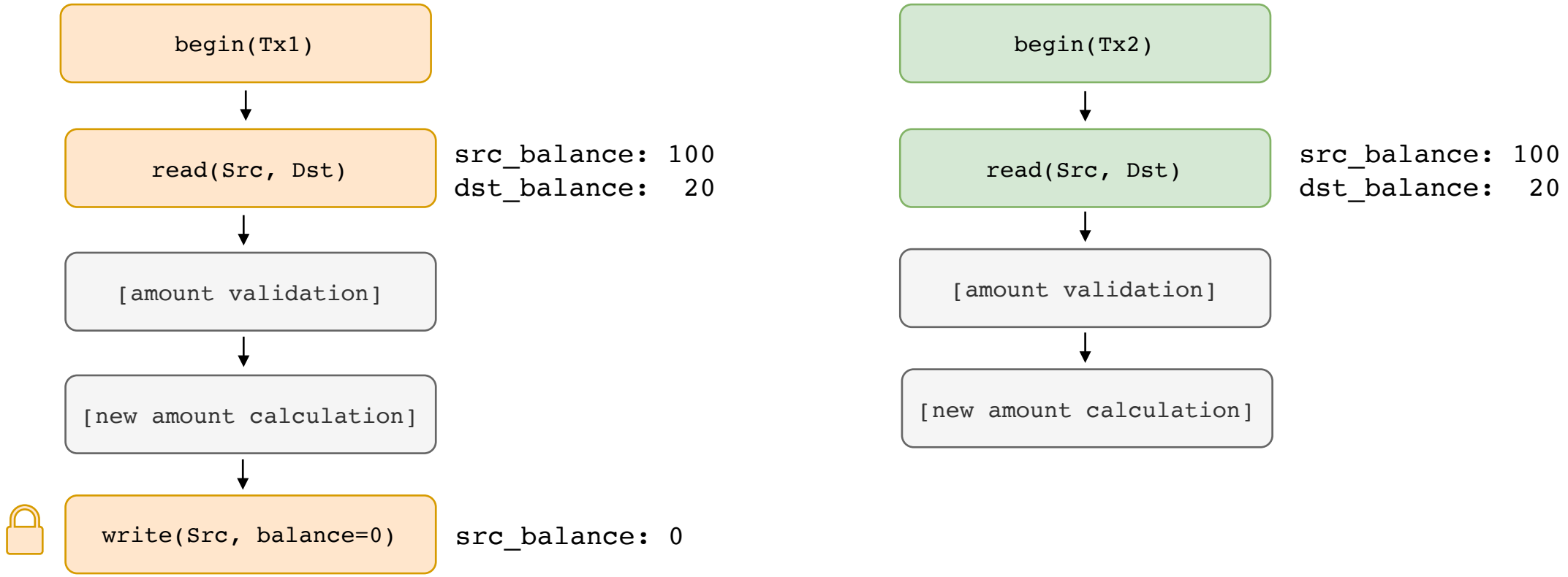

begin(Tx2)
read(Src, Dst)
src_balance: 100
dst_balance: 20
[amount validation]

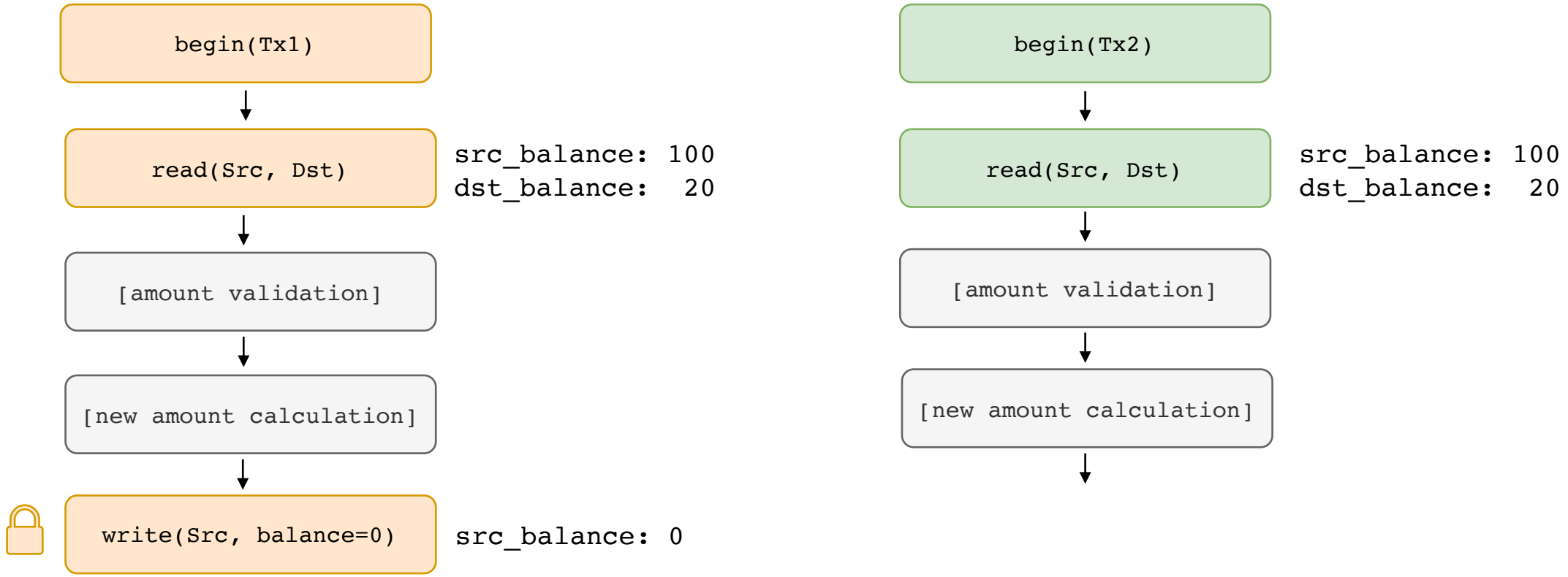

begin(Tx1)
read(Src, Dst)
src_balance: 100
dst_balance: 20
[amount validation]
[new amount calculation]
begin(Tx2)
read(Src, Dst)
src_balance: 100
dst_balance: 20
[amount validation]
[new amount calculation]

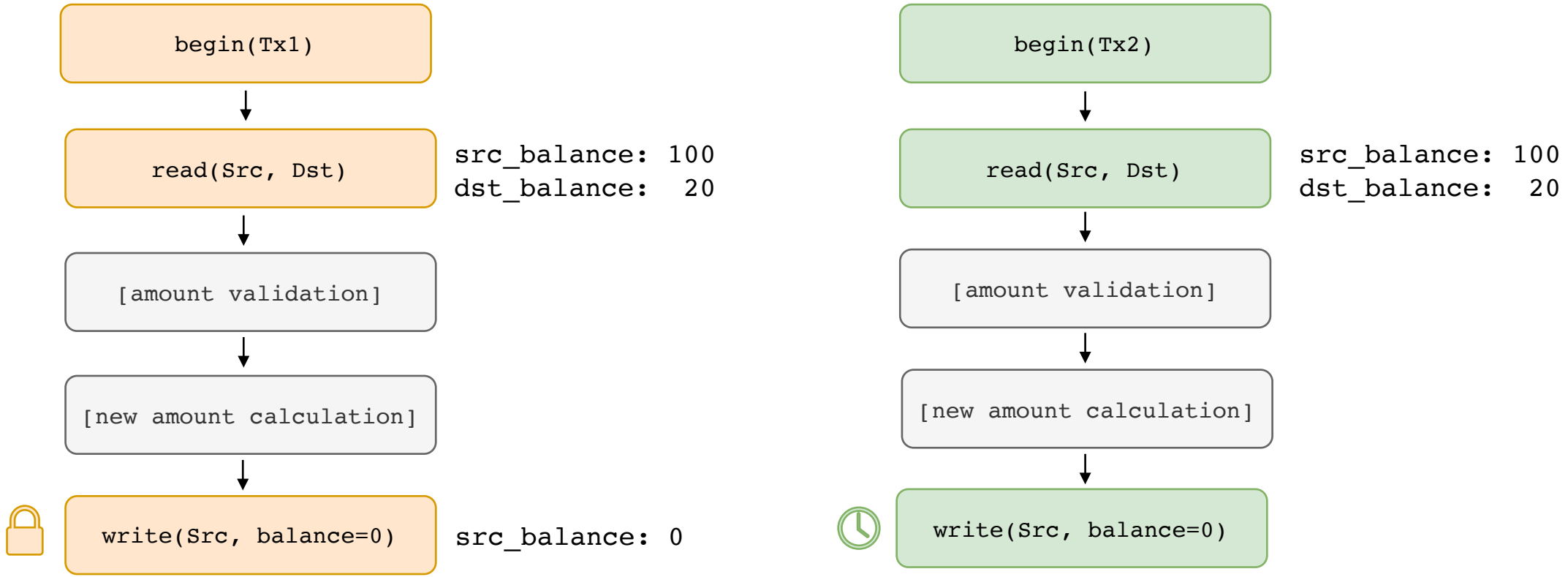

begin(Tx1)
read(Src, Dst)
src_balance: 100
dst_balance: 20
[amount validation]
[new amount calculation]

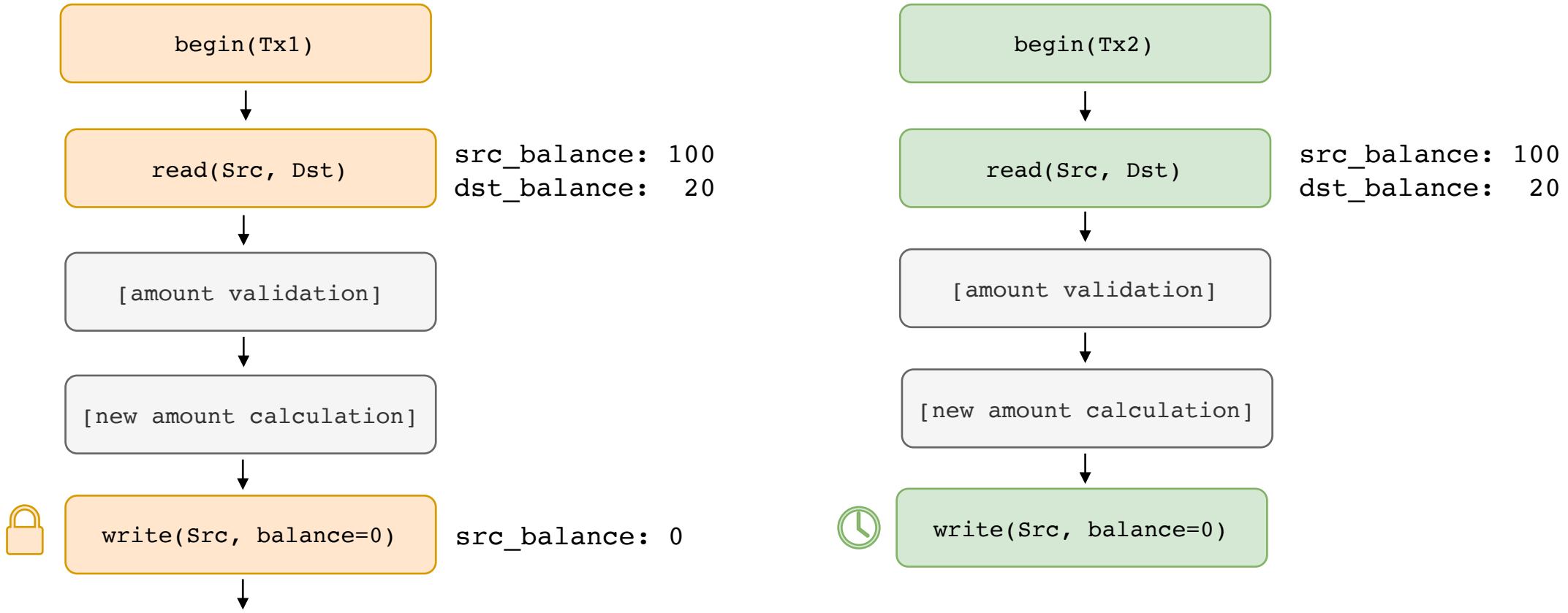

begin(Tx2)
read(Src, Dst)
src_balance: 100
dst_balance: 20
[amount validation]
[new amount calculation]

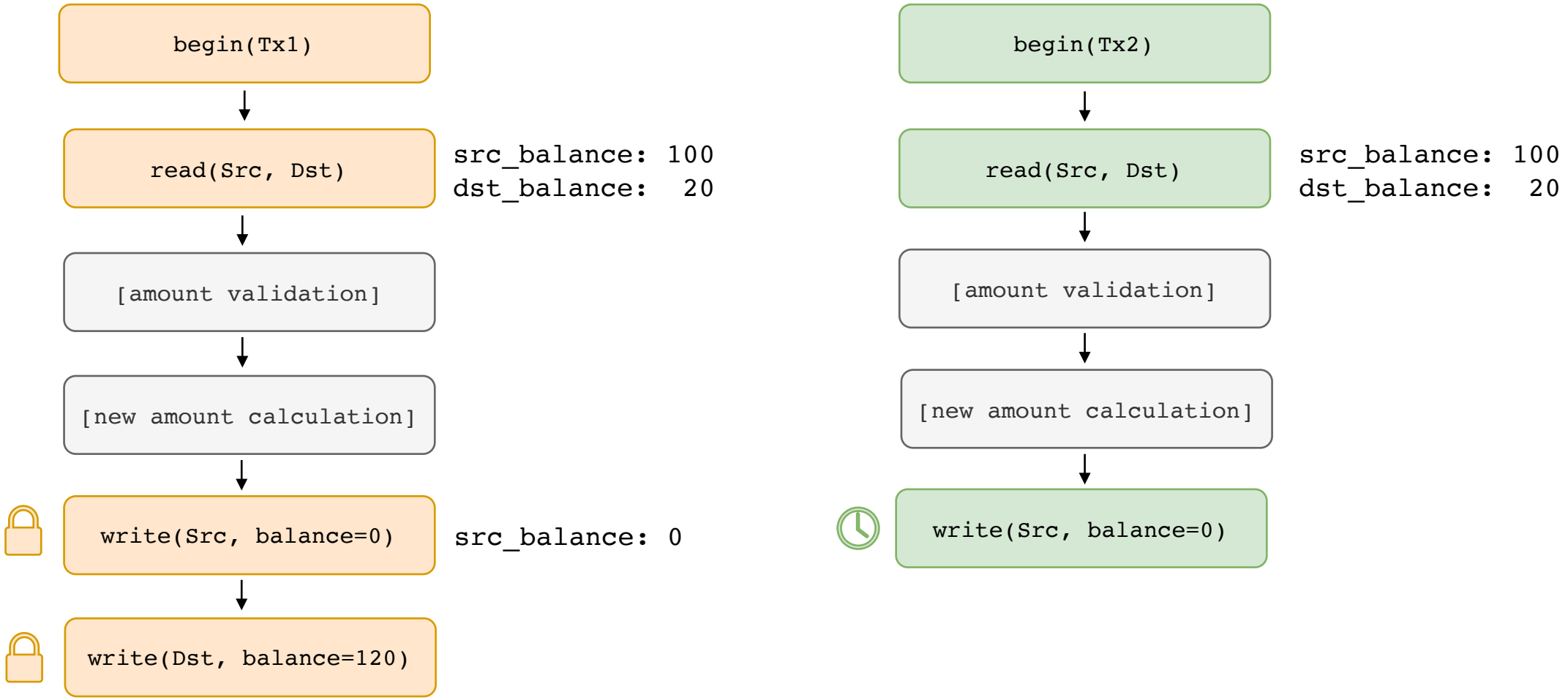

begin(Tx1)
read(Src, Dst)
src_balance: 100
dst_balance: 20
[amount validation]
[new amount calculation]
write(Src, balance=0)
begin(Tx2)
read(Src, Dst)
src_balance: 100
dst_balance: 20
[amount validation]
[new amount calculation]

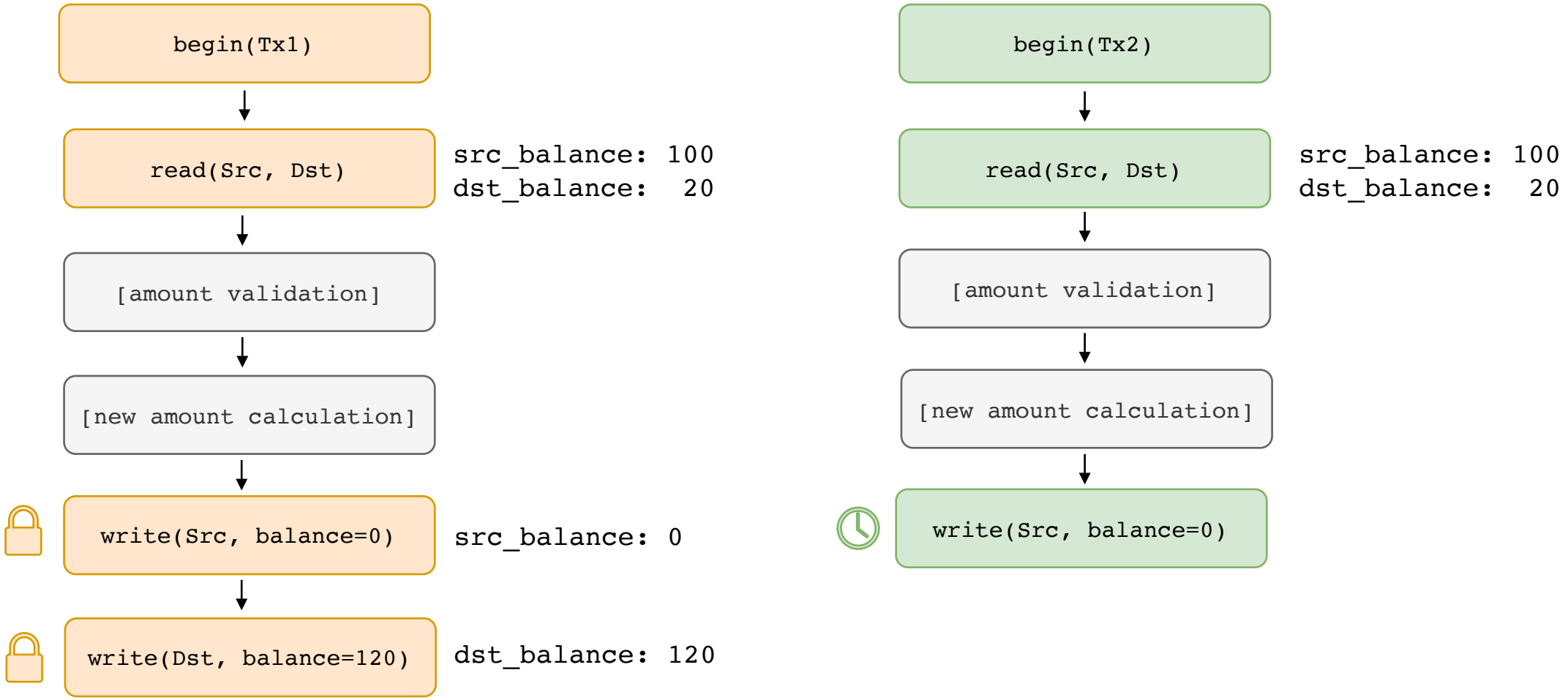

begin(Tx1)
read(Src, Dst)
src_balance: 100
dst_balance: 20
[amount validation]
[new amount calculation]
write(Src, balance=0)
src_balance: 0
begin(Tx2)
read(Src, Dst)
src_balance: 100
dst_balance: 20
[amount validation]
[new amount calculation]

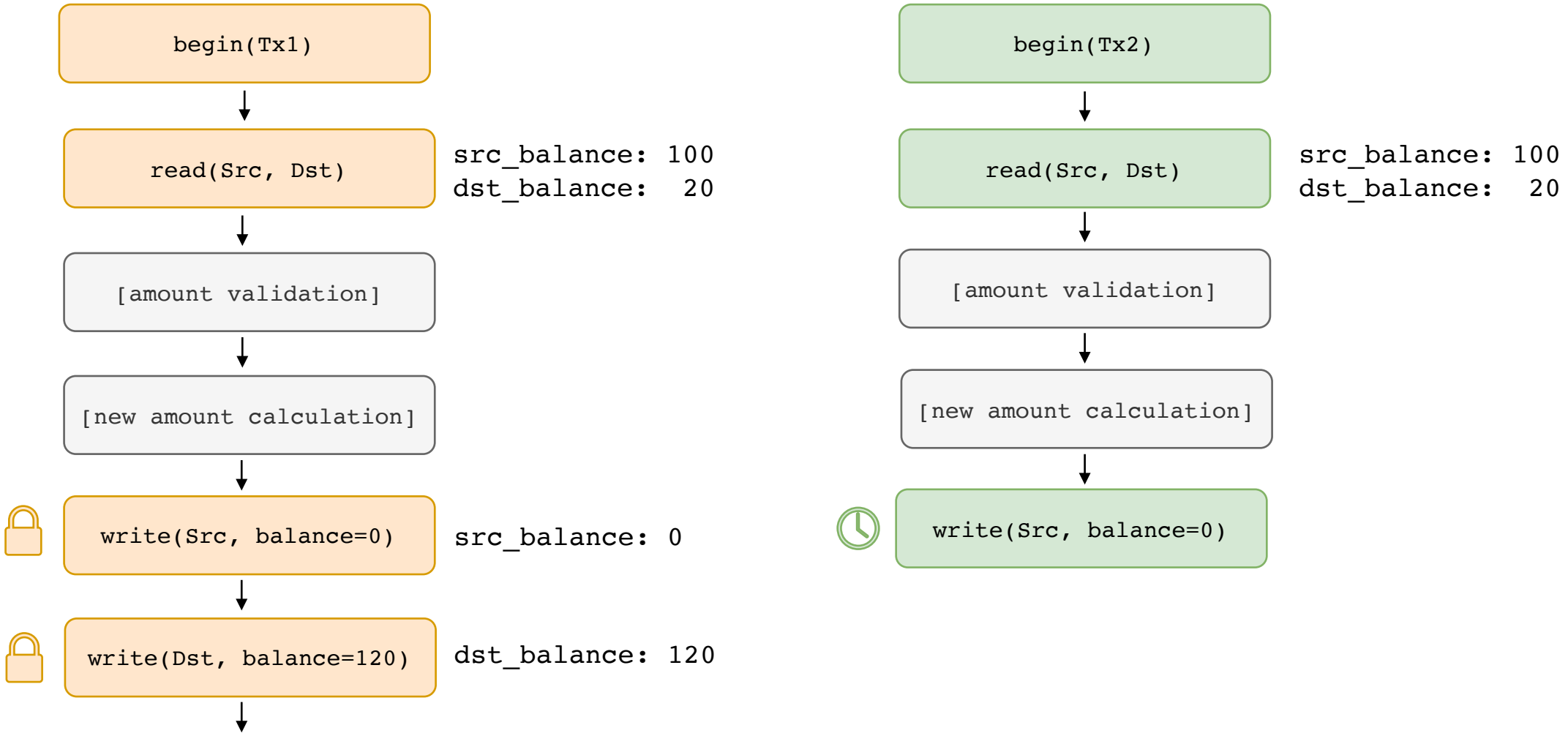

begin(Tx1)
read(Src, Dst)
src_balance: 100
dst_balance: 20
[amount validation]
[new amount calculation]
write(Src, balance=0)
src_balance: 0
begin(Tx2)
read(Src, Dst)
src_balance: 100
dst_balance: 20
[amount validation]
[new amount calculation]

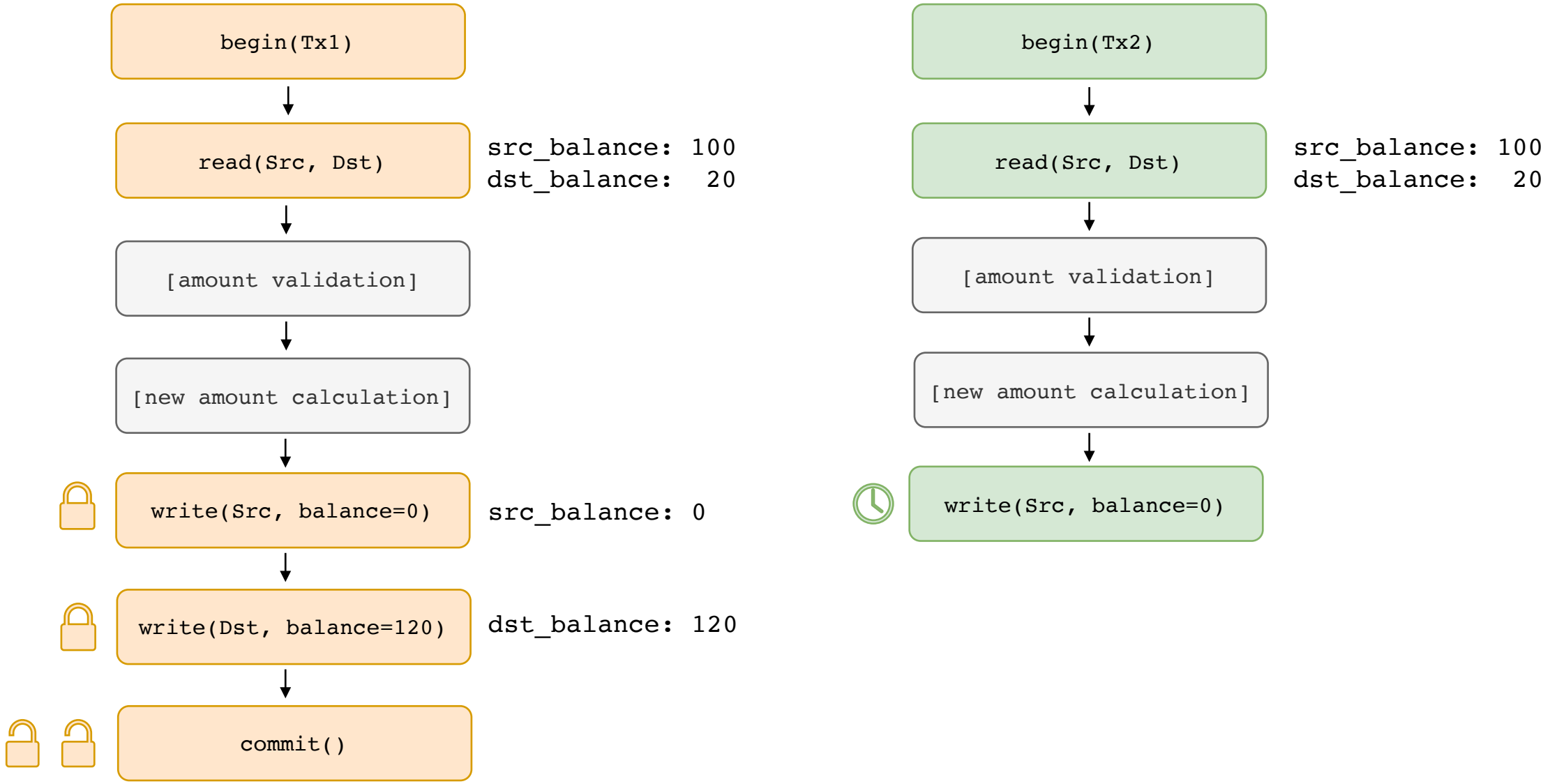

begin(Tx1)
read(Src, Dst)
src_balance: 100
dst_balance: 20
[amount validation]
[new amount calculation]
write(Src, balance=0)
src_balance: 0
begin(Tx2)
read(Src, Dst)
src_balance: 100
dst_balance: 20
[amount validation]
[new amount calculation]
write(Src, balance=0)

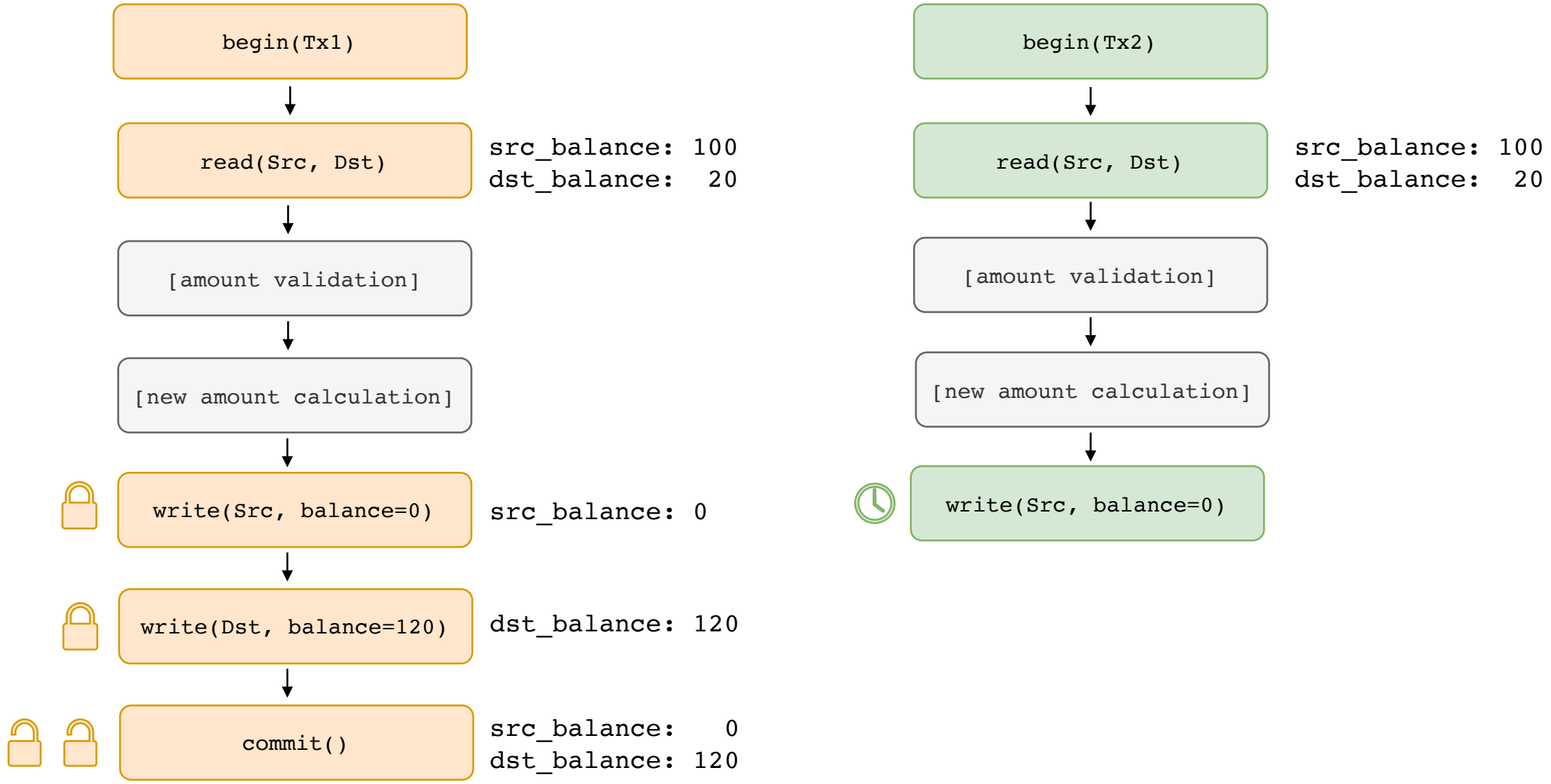

begin(Tx1)
read(Src, Dst)
src_balance: 100
dst_balance: 20
[amount validation]
[new amount calculation]
write(Src, balance=0)
src_balance: 0
begin(Tx2)
read(Src, Dst)
src_balance: 100
dst_balance: 20
[amount validation]
[new amount calculation]
write(Src, balance=0)

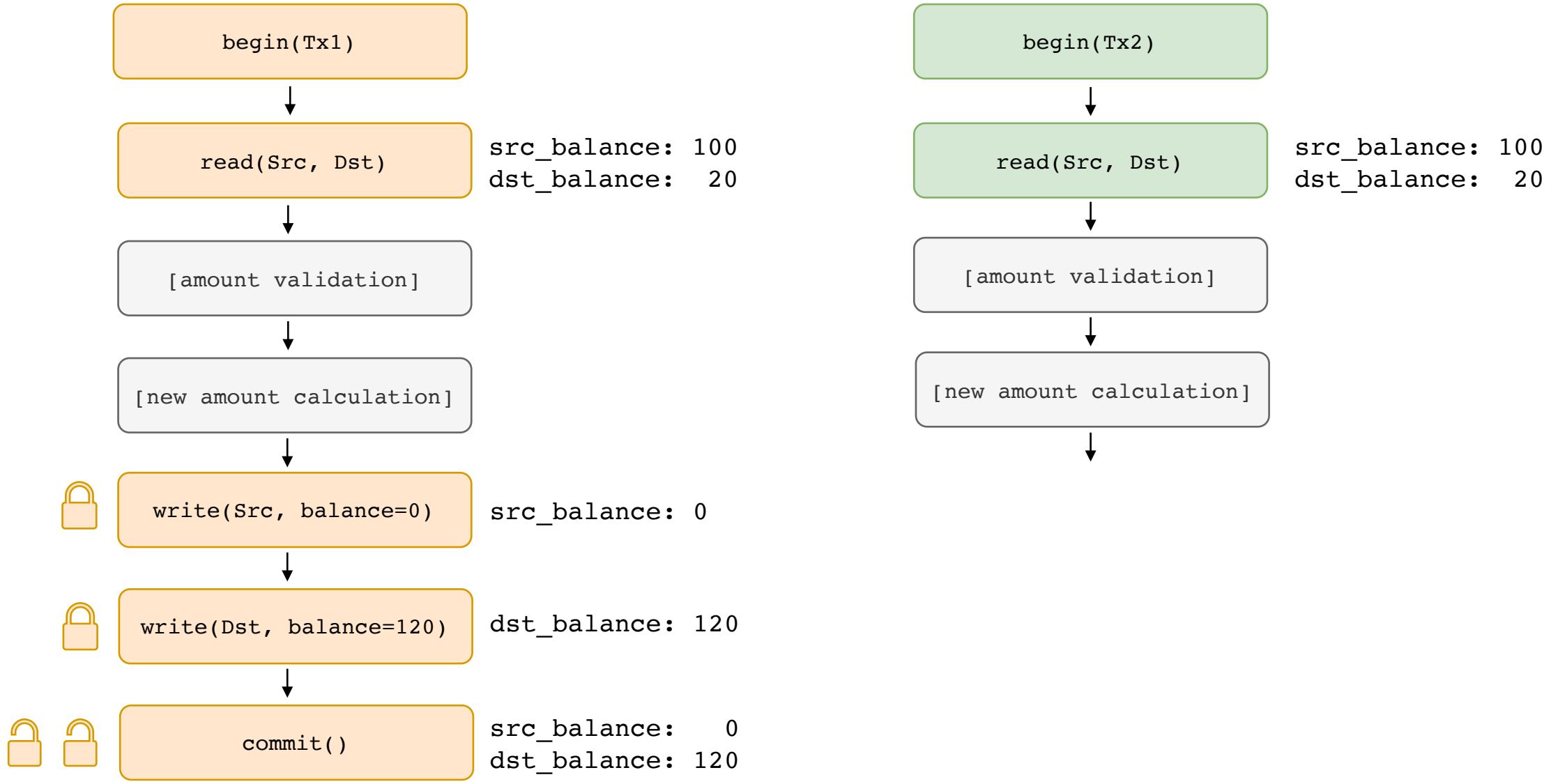

begin(Tx1)
read(Src, Dst)
src_balance: 100
dst_balance: 20
[amount validation]
[new amount calculation]
write(Src, balance=0)
src_balance: 0
write(Dst, balance=120)
begin(Tx2)
read(Src, Dst)
src_balance: 100
dst_balance: 20
[amount validation]
[new amount calculation]
write(Src, balance=0)

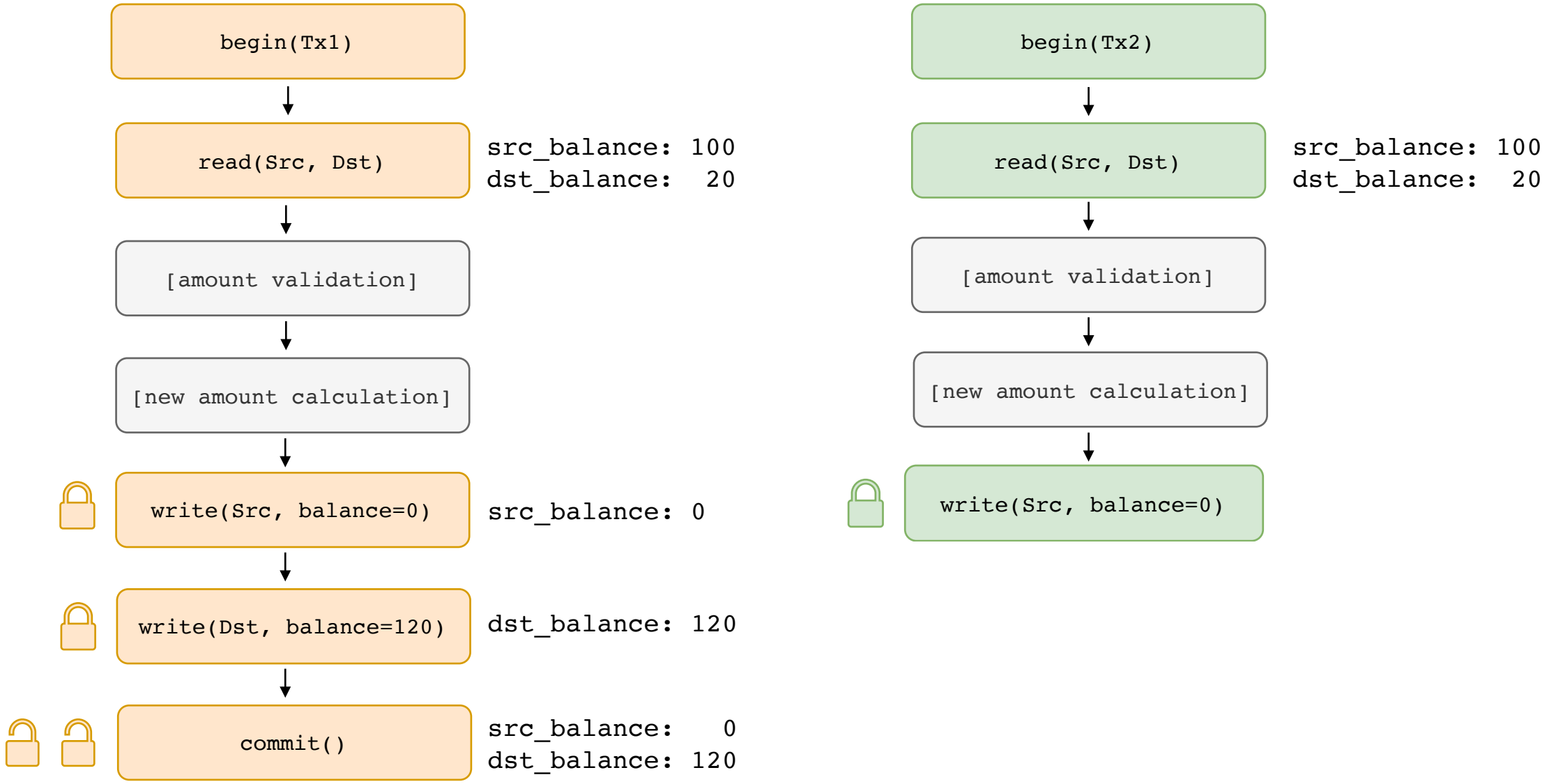

begin(Tx1)
read(Src, Dst)
src_balance: 100
dst_balance: 20
[amount validation]
[new amount calculation]
write(Src, balance=0)
src_balance: 0
write(Dst, balance=120)
dst_balance: 120
begin(Tx2)
read(Src, Dst)
src_balance: 100
dst_balance: 20
[amount validation]
[new amount calculation]
write(Src, balance=0)

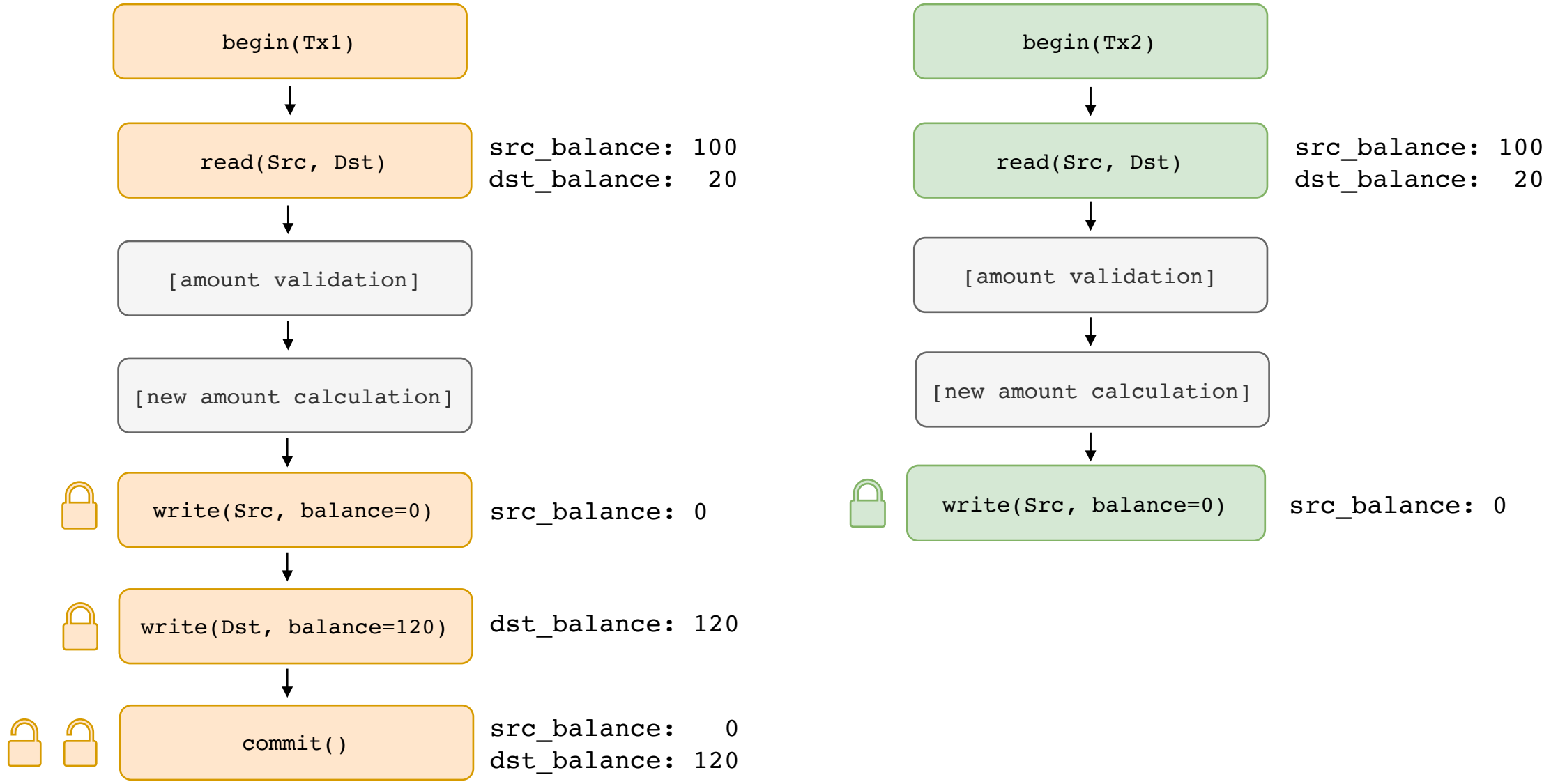

begin(Tx1)
read(Src, Dst)
src_balance: 100
dst_balance: 20
[amount validation]
[new amount calculation]
write(Src, balance=0)
src_balance: 0
write(Dst, balance=120)
dst_balance: 120
begin(Tx2)
read(Src, Dst)
src_balance: 100
dst_balance: 20
[amount validation]
[new amount calculation]
write(Src, balance=0)

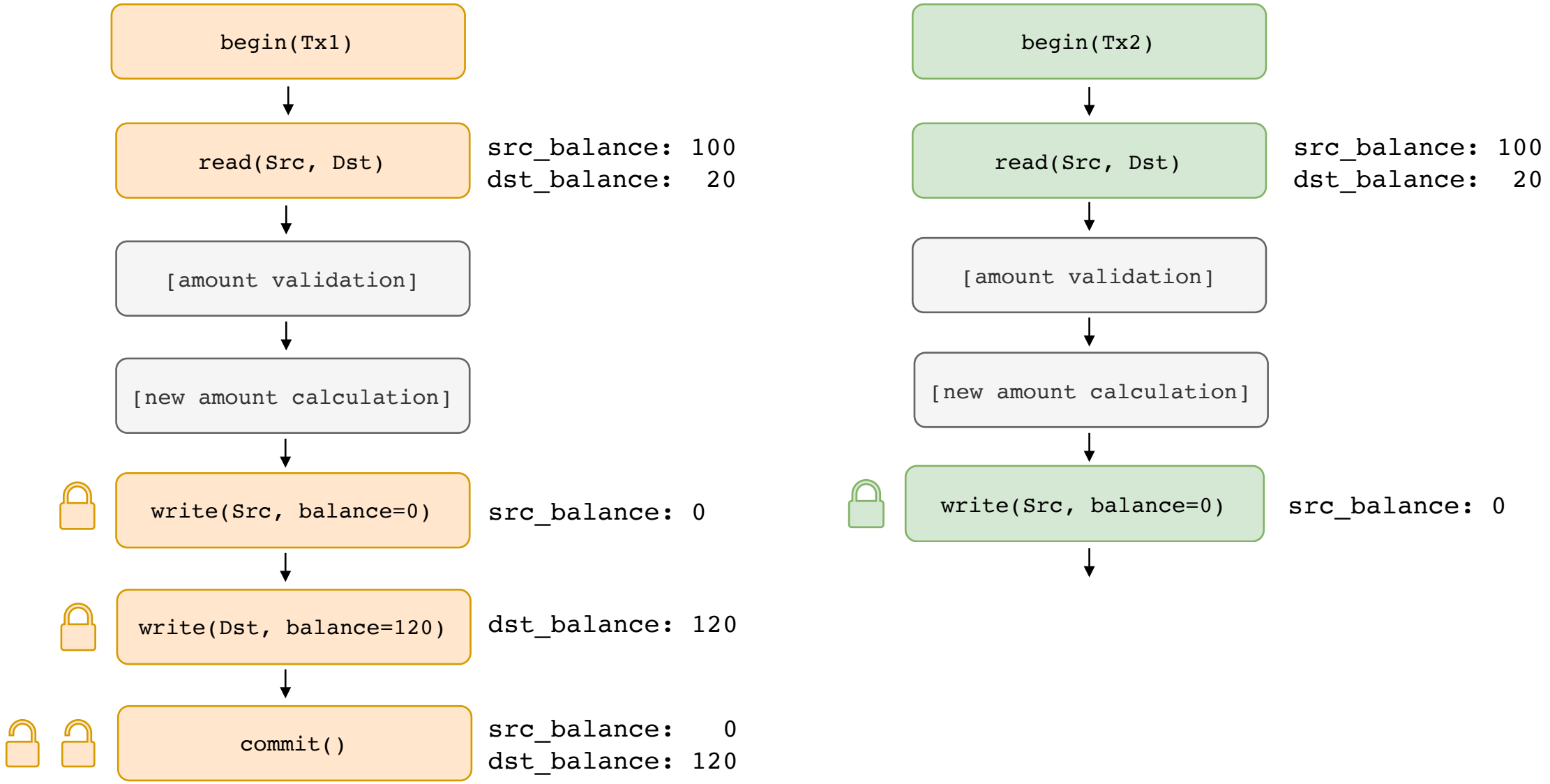

begin(Tx1)
read(Src, Dst)
src_balance: 100
dst_balance: 20
[amount validation]
[new amount calculation]
write(Src, balance=0)
src_balance: 0
write(Dst, balance=120)
dst_balance: 120
commit()
begin(Tx2)
read(Src, Dst)
src_balance: 100
dst_balance: 20
[amount validation]
[new amount calculation]
write(Src, balance=0)

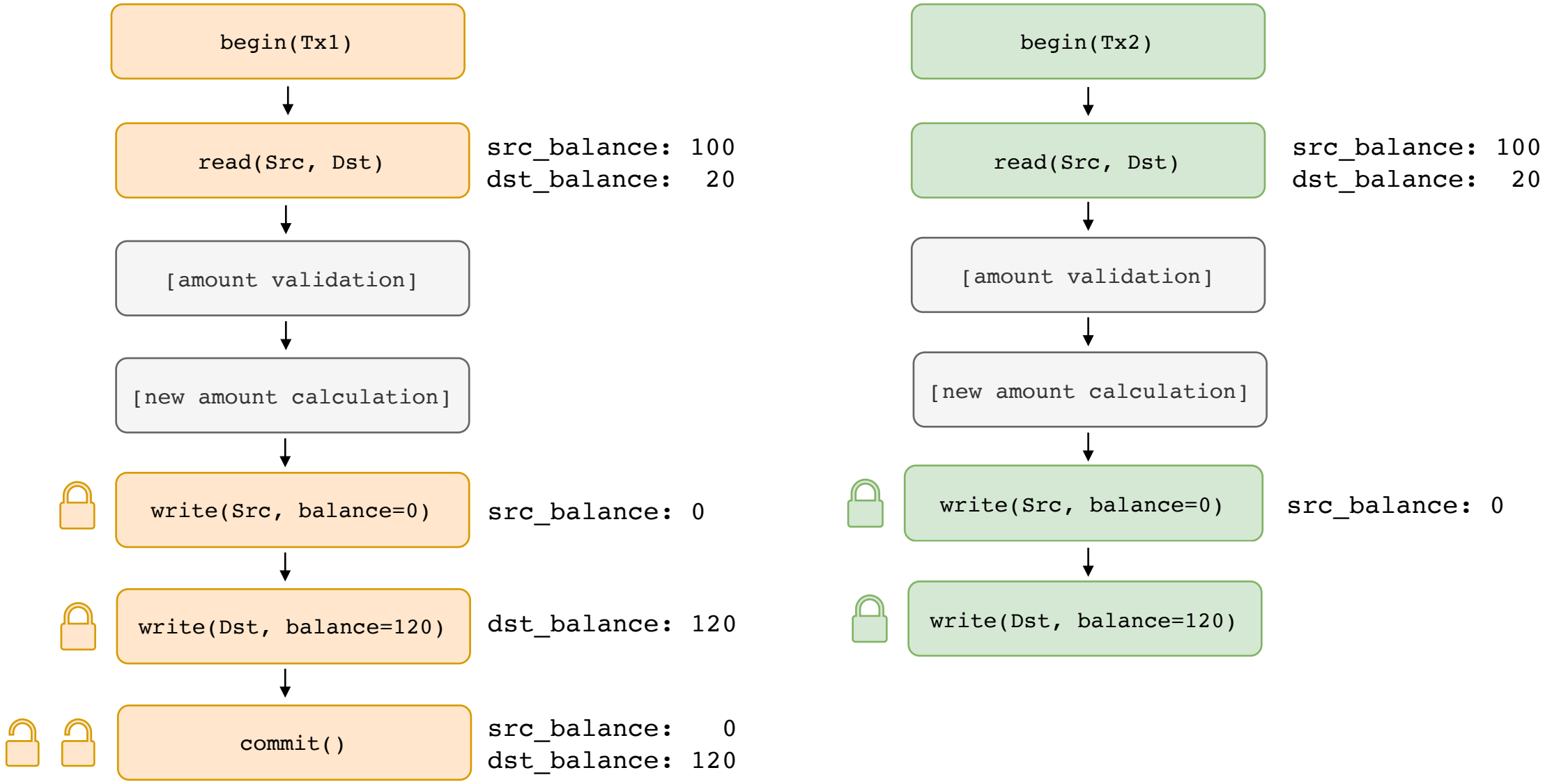

begin(Tx1)
read(Src, Dst)
src_balance: 100
dst_balance: 20
[amount validation]
[new amount calculation]
write(Src, balance=0)
src_balance: 0
write(Dst, balance=120)
dst_balance: 120
commit()
src_balance: 0
dst_balance: 120
begin(Tx2)
read(Src, Dst)
src_balance: 100
dst_balance: 20
[amount validation]
[new amount calculation]
write(Src, balance=0)

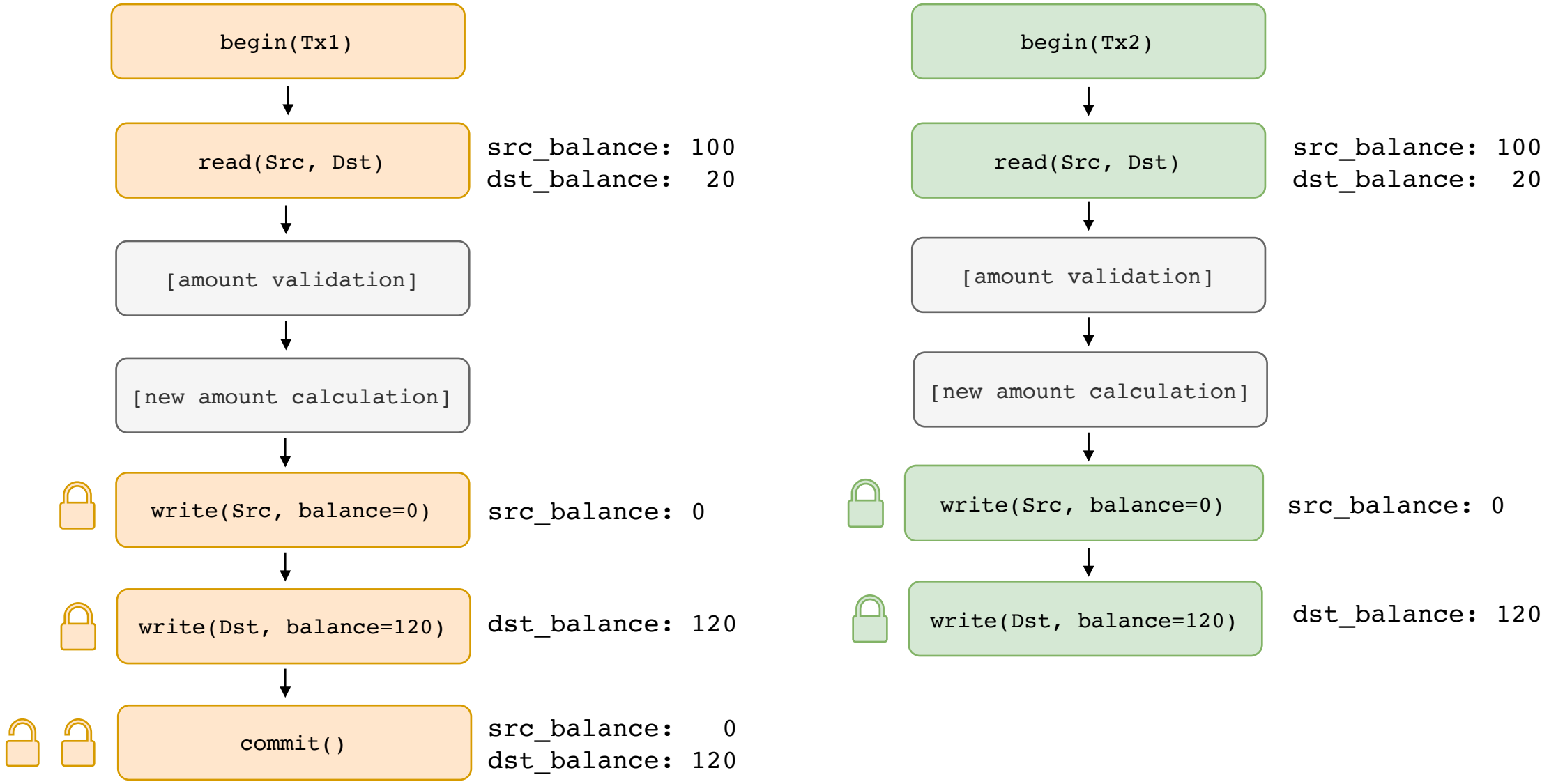

begin(Tx1)
read(Src, Dst)
src_balance: 100
dst_balance: 20
[amount validation]
[new amount calculation]
write(Src, balance=0)
src_balance: 0
write(Dst, balance=120)
dst_balance: 120
commit()
src_balance: 0
dst_balance: 120
begin(Tx2)
read(Src, Dst)
src_balance: 100
dst_balance: 20
[amount validation]
[new amount calculation]

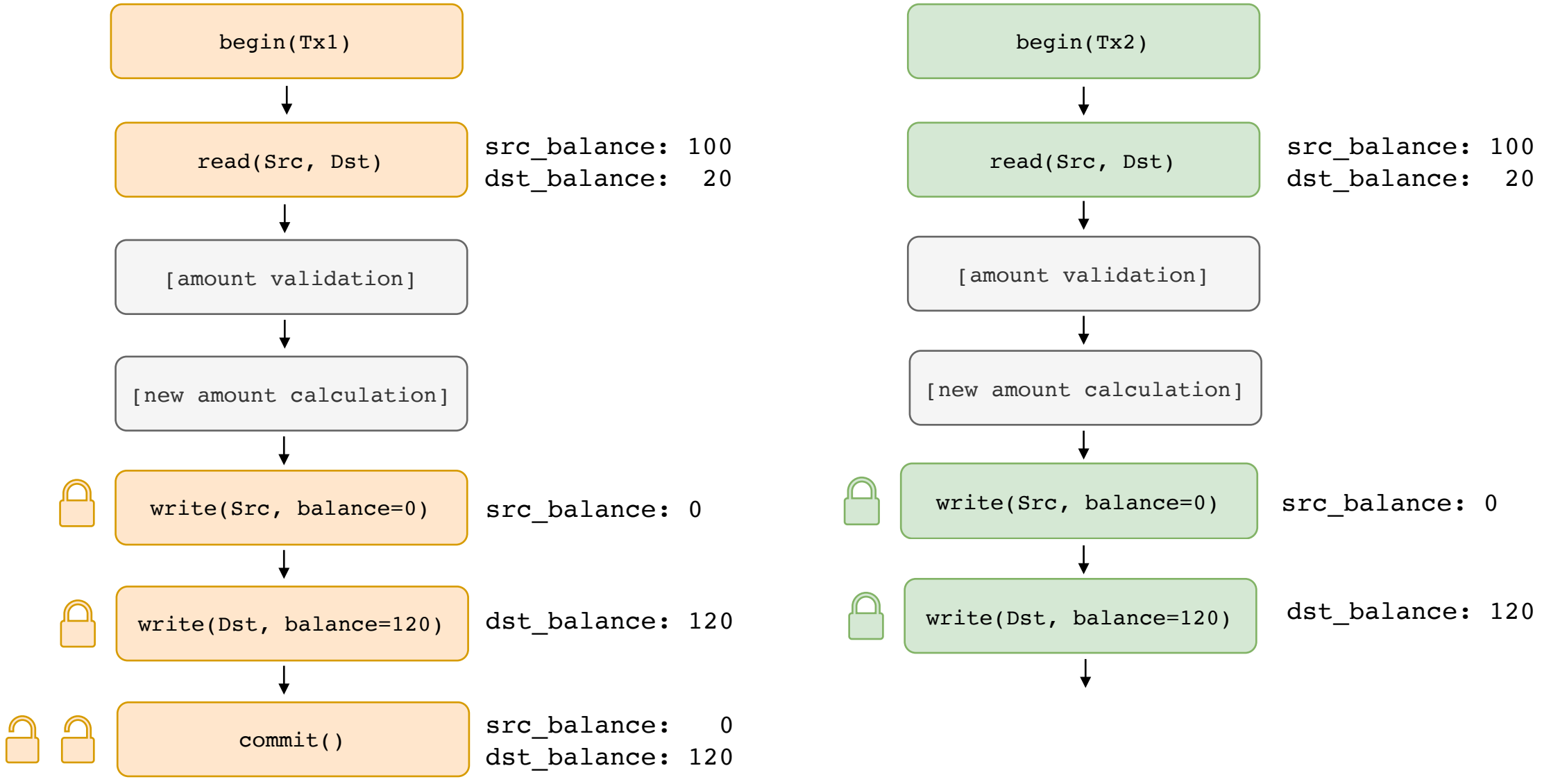

begin(Tx1)
read(Src, Dst)
src_balance: 100
dst_balance: 20
[amount validation]
[new amount calculation]
write(Src, balance=0)
src_balance: 0
write(Dst, balance=120)
dst_balance: 120
commit()
src_balance: 0
dst_balance: 120
begin(Tx2)
read(Src, Dst)
src_balance: 100
dst_balance: 20
[amount validation]
[new amount calculation]
write(Src, balance=0)

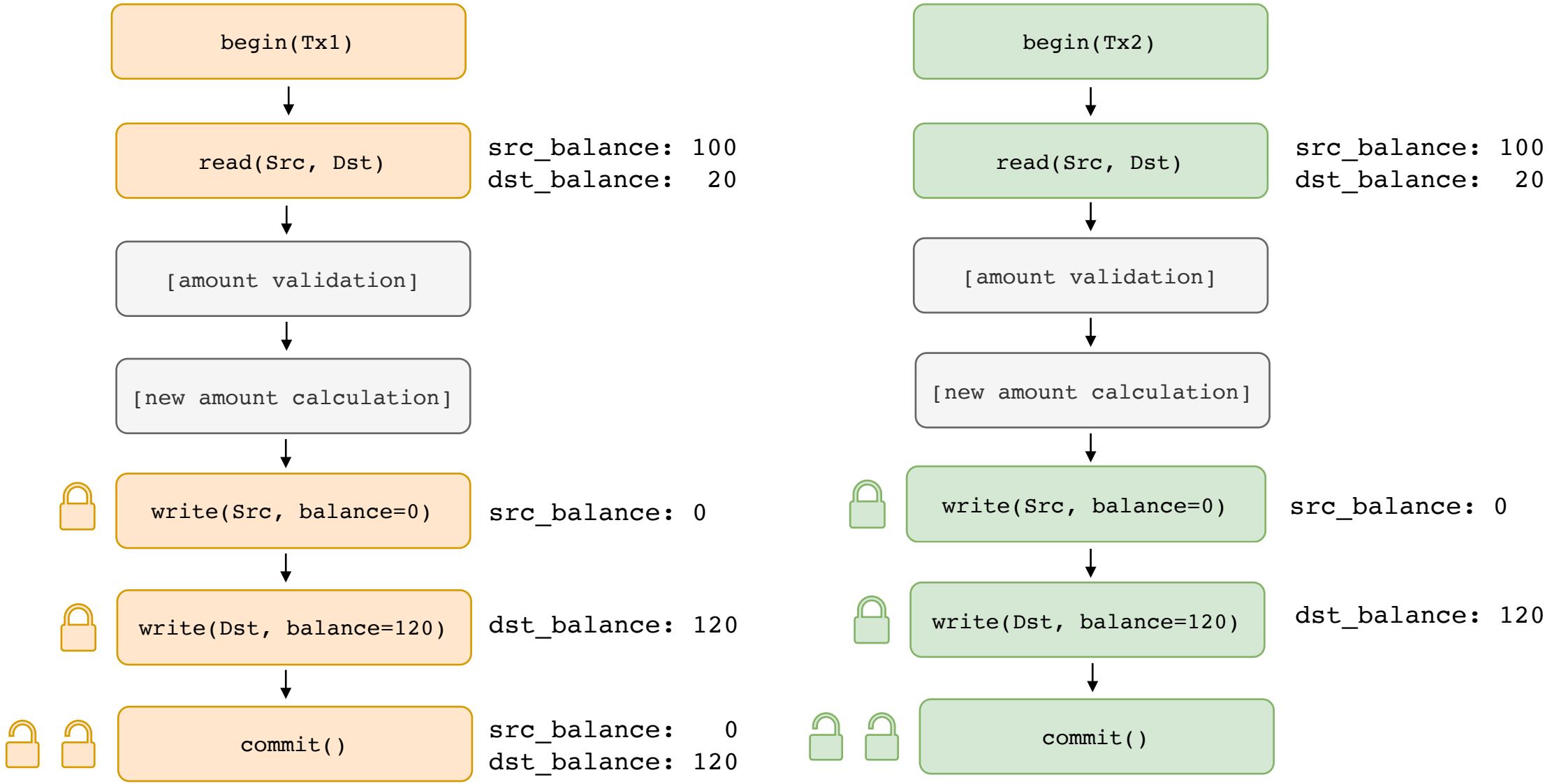

begin(Tx1)
read(Src, Dst)
src_balance: 100
dst_balance: 20
[amount validation]
[new amount calculation]
write(Src, balance=0)
src_balance: 0
write(Dst, balance=120)
dst_balance: 120
commit()
src_balance: 0
dst_balance: 120
begin(Tx2)
read(Src, Dst)
src_balance: 100
dst_balance: 20
[amount validation]
[new amount calculation]
write(Src, balance=0)
src_balance: 0

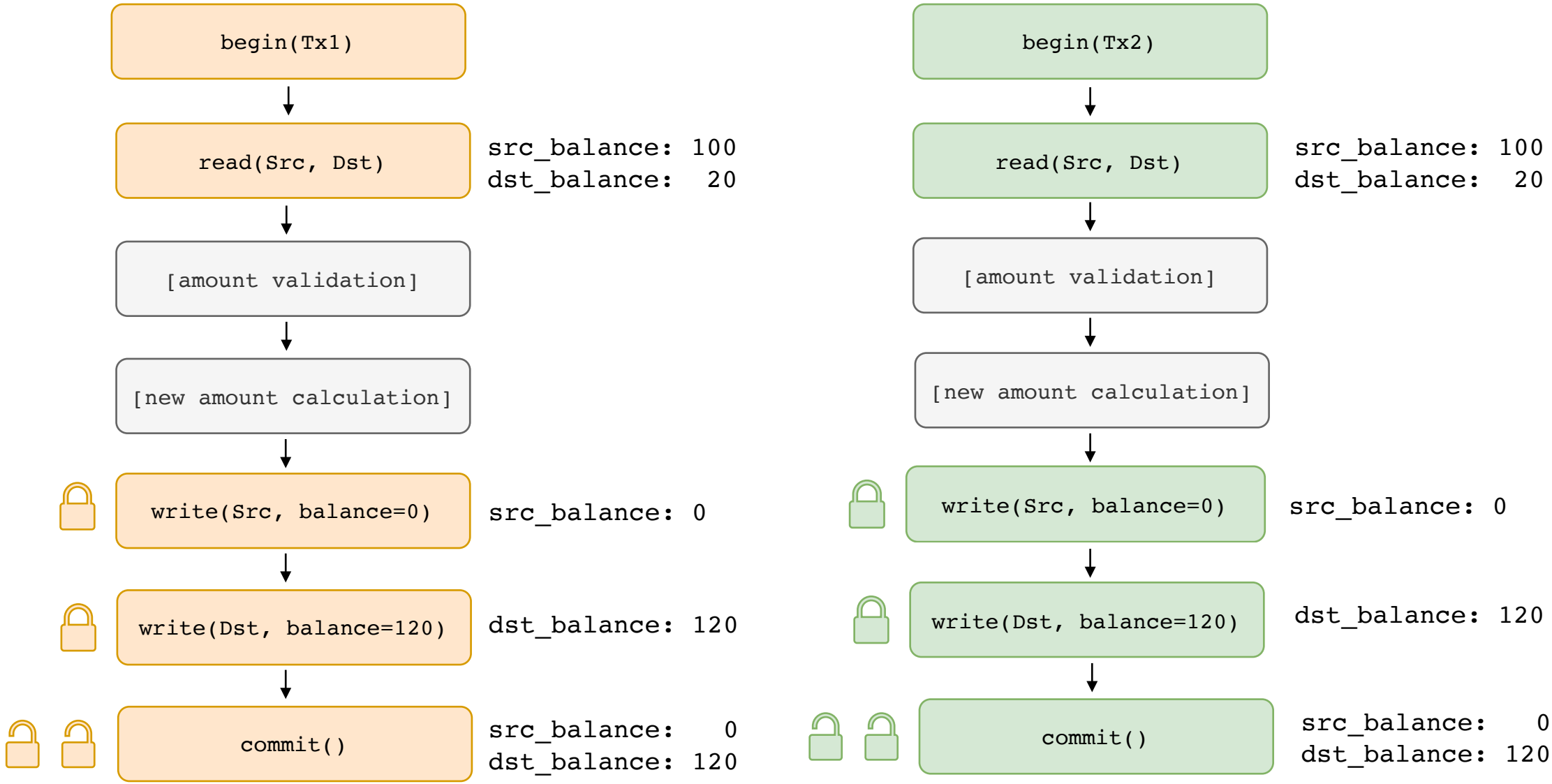

begin(Tx1)
read(Src, Dst)
src_balance: 100
dst_balance: 20
[amount validation]
[new amount calculation]
write(Src, balance=0)
src_balance: 0
write(Dst, balance=120)
dst_balance: 120
commit()
src_balance: 0
dst_balance: 120
begin(Tx2)
read(Src, Dst)
src_balance: 100
dst_balance: 20
[amount validation]
[new amount calculation]
write(Src, balance=0)
src_balance: 0


begin(Tx1)
read(Src, Dst)
src_balance: 100
dst_balance: 20
[amount validation]
[new amount calculation]
write(Src, balance=0)
src_balance: 0
write(Dst, balance=120)
dst_balance: 120
commit()
src_balance: 0
dst_balance: 120
begin(Tx2)
read(Src, Dst)
src_balance: 100
dst_balance: 20
[amount validation]
[new amount calculation]
write(Src, balance=0)
src_balance: 0
write(Dst, balance=120)

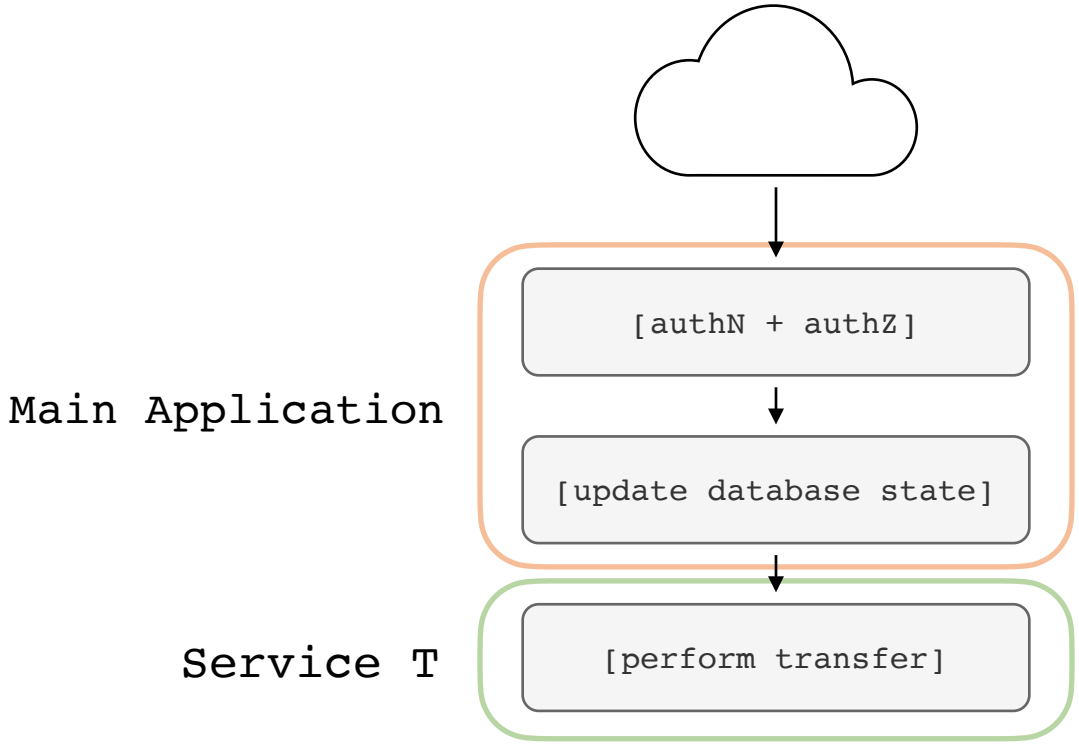

begin(Tx1)
read(Src, Dst)
src_balance: 100
dst_balance: 20
[amount validation]
[new amount calculation]
write(Src, balance=0)
src_balance: 0
write(Dst, balance=120)
dst_balance: 120
commit()
src_balance: 0
dst_balance: 120
begin(Tx2)
read(Src, Dst)
src_balance: 100
dst_balance: 20
[amount validation]
[new amount calculation]
write(Src, balance=0)
src_balance: 0
write(Dst, balance=120)
dst_balance: 120


begin(Tx1)
read(Src, Dst)
src_balance: 100
dst_balance: 20
[amount validation]
[new amount calculation]
write(Src, balance=0)
src_balance: 0
write(Dst, balance=120)
dst_balance: 120
commit()
src_balance: 0
dst_balance: 120
begin(Tx2)
read(Src, Dst)
src_balance: 100
dst_balance: 20
[amount validation]
[new amount calculation]
write(Src, balance=0)
src_balance: 0
write(Dst, balance=120)
dst_balance: 120

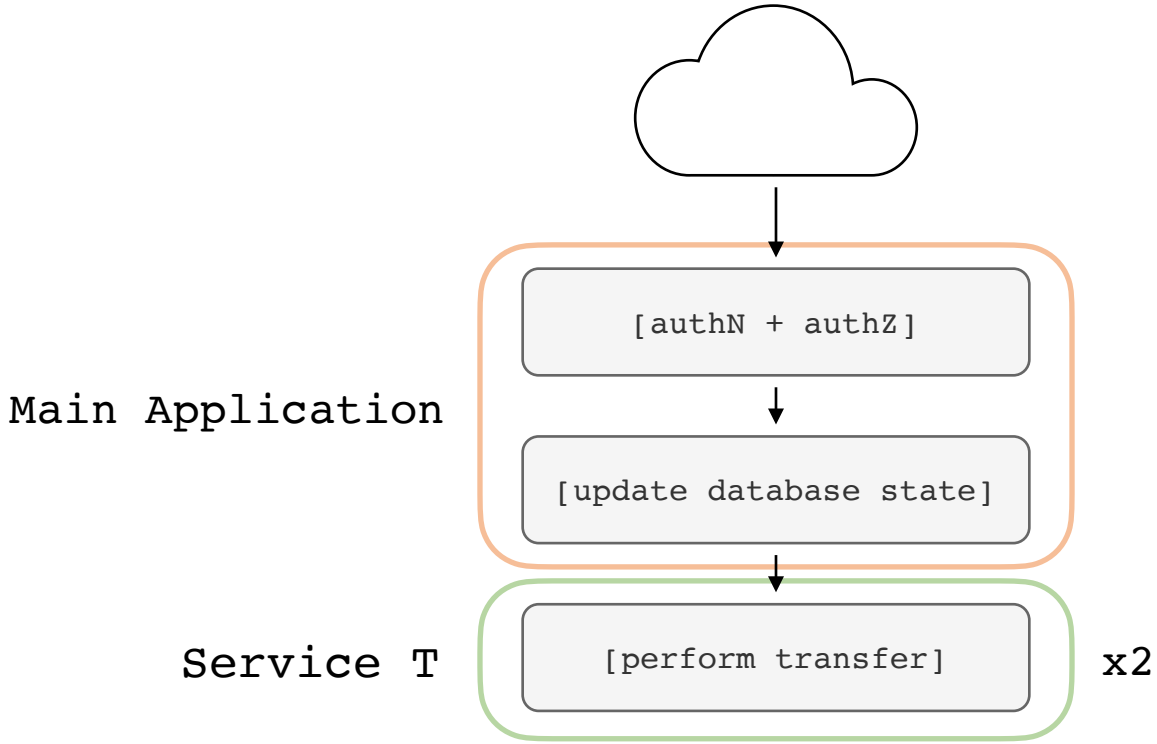

begin(Tx1)
read(Src, Dst)
src_balance: 100
dst_balance: 20
[amount validation]
[new amount calculation]
write(Src, balance=0)
src_balance: 0
write(Dst, balance=120)
dst_balance: 120
commit()
src_balance: 0
dst_balance: 120
begin(Tx2)
read(Src, Dst)
src_balance: 100
dst_balance: 20
[amount validation]
[new amount calculation]
write(Src, balance=0)
src_balance: 0
write(Dst, balance=120)
dst_balance: 120
commit()


begin(Tx1)
read(Src, Dst)
src_balance: 100
dst_balance: 20
[amount validation]
[new amount calculation]
write(Src, balance=0)
src_balance: 0
write(Dst, balance=120)
dst_balance: 120
commit()
src_balance: 0
dst_balance: 120
begin(Tx2)
read(Src, Dst)
src_balance: 100
dst_balance: 20
[amount validation]
[new amount calculation]
write(Src, balance=0)
src_balance: 0
write(Dst, balance=120)
dst_balance: 120
commit()
src_balance: 0
dst_balance: 120

Microservice Architecture



Microservice Architecture



begin(Tx1)

begin(Tx2)

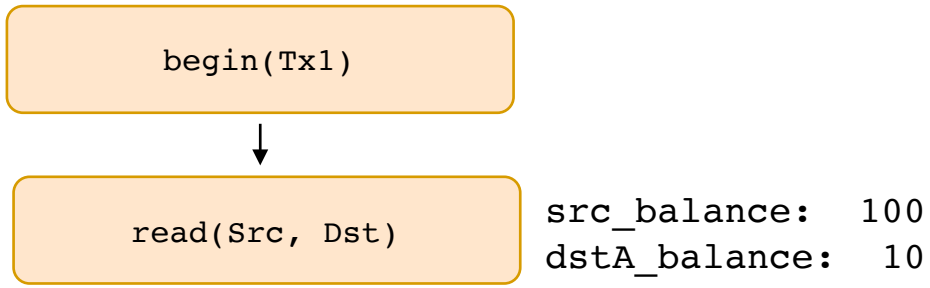
Scenario	
Source Balance :	100
DestinationA Balance:	10
DestinationB Balance:	20
Total Balance:	130
Transfer Amount:	100

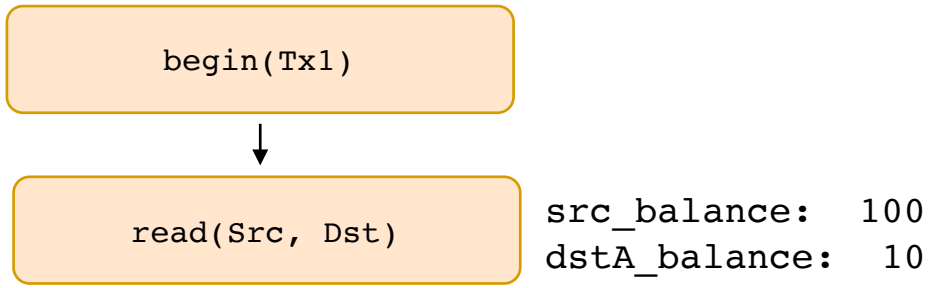
```
begin(Tx1)
```

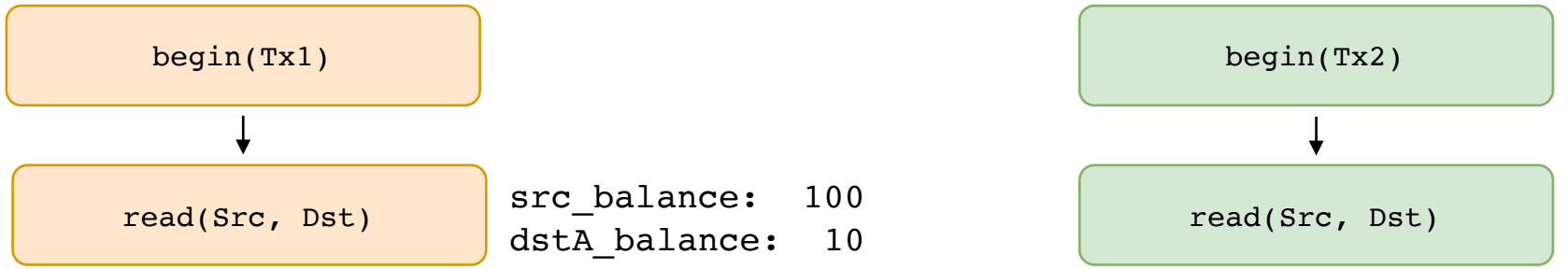
```
begin(Tx2)
```

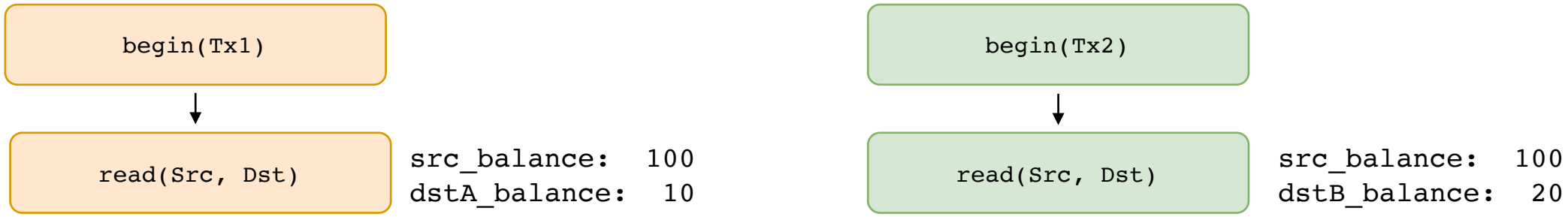

begin(Tx1)
begin(Tx2)

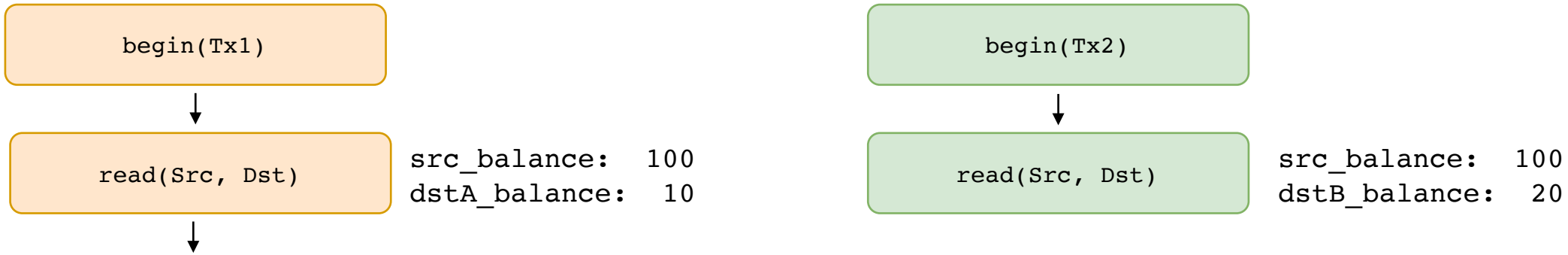

begin(Tx1)
read(Src, Dst)
begin(Tx2)

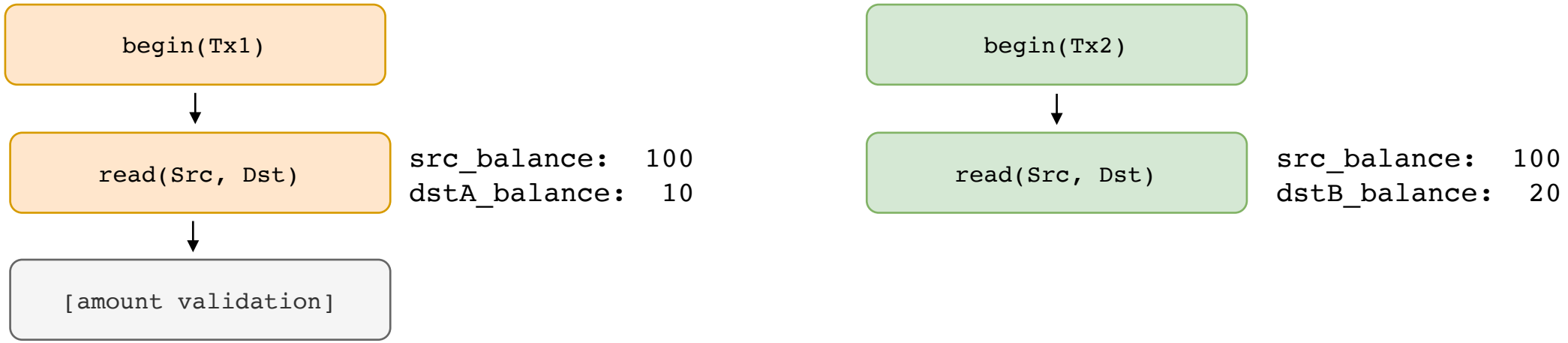

begin(Tx1)
read(Src, Dst)
src_balance: 100
dstA_balance: 10
begin(Tx2)

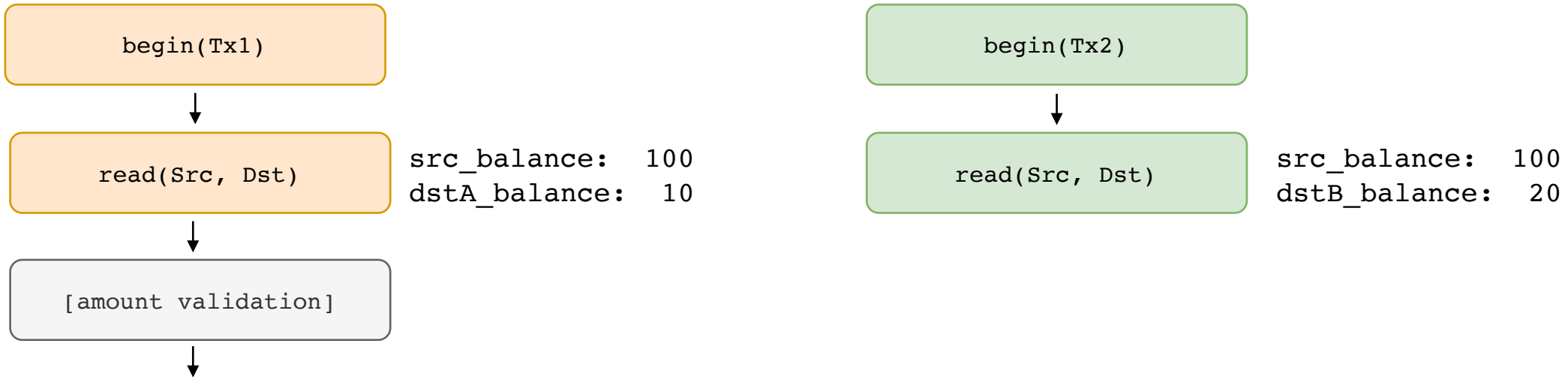

begin(Tx1)
read(Src, Dst)
src_balance: 100
dstA_balance: 10
begin(Tx2)

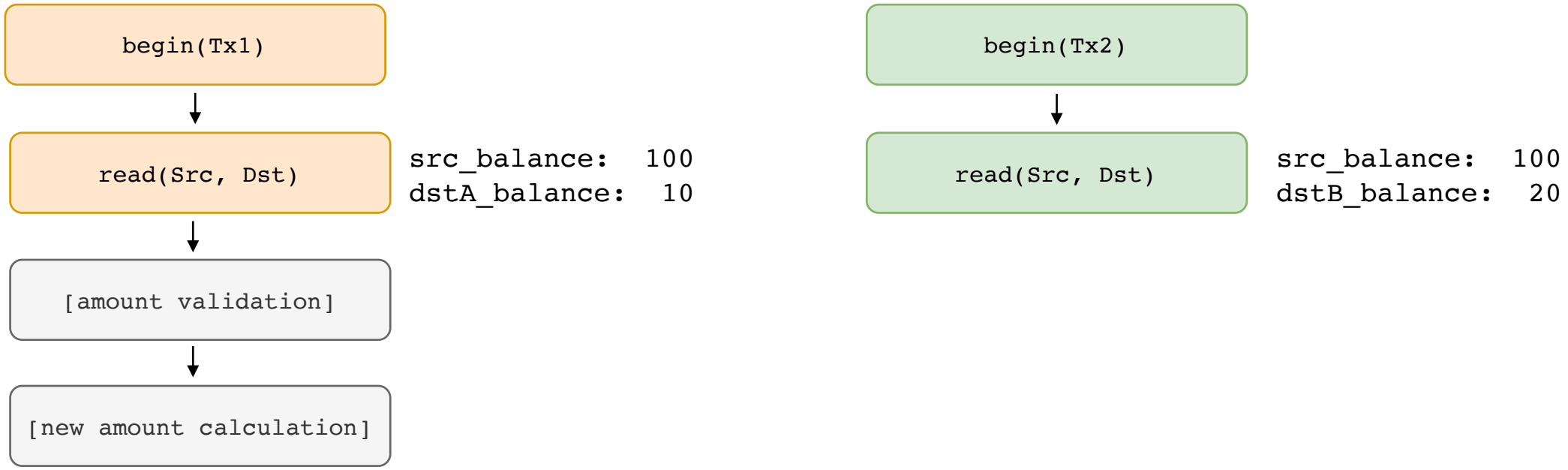

begin(Tx1)
read(Src, Dst)
src_balance: 100
dstA_balance: 10
begin(Tx2)
read(Src, Dst)

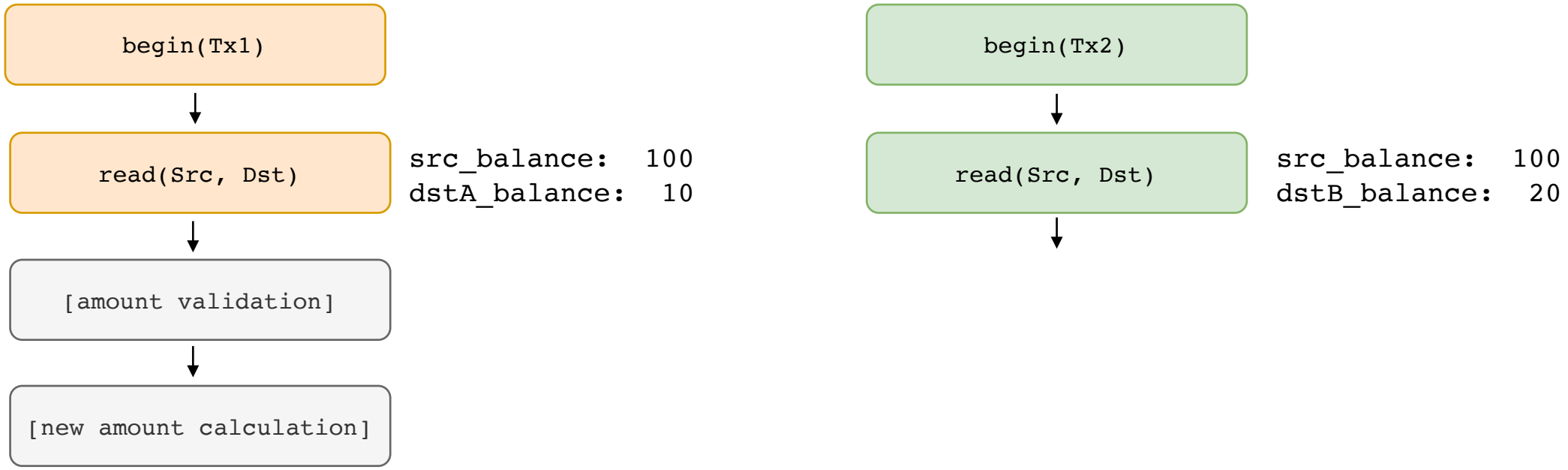

begin(Tx1)
read(Src, Dst)
src_balance: 100
dstA_balance: 10
begin(Tx2)
read(Src, Dst)
src_balance: 100
dstB_balance: 20

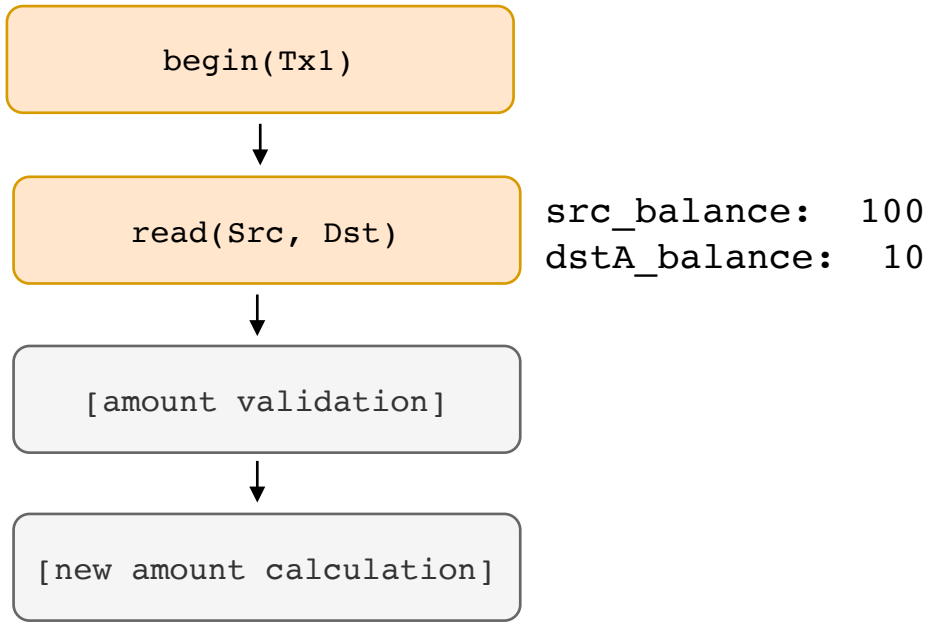

begin(Tx1)
read(Src, Dst)
src_balance: 100
dstA_balance: 10
begin(Tx2)
read(Src, Dst)
src_balance: 100
dstB_balance: 20

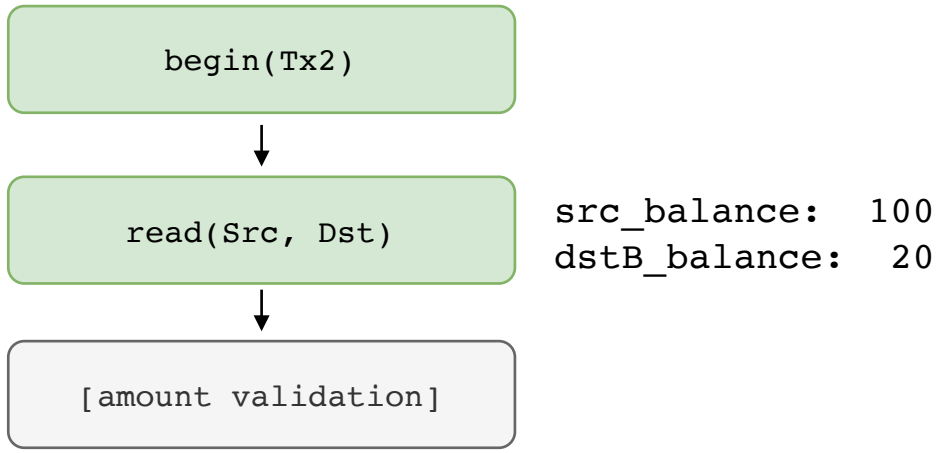

begin(Tx1)
read(Src, Dst)
[amount validation]
src_balance: 100
dstA_balance: 10
begin(Tx2)
read(Src, Dst)
src_balance: 100
dstB_balance: 20

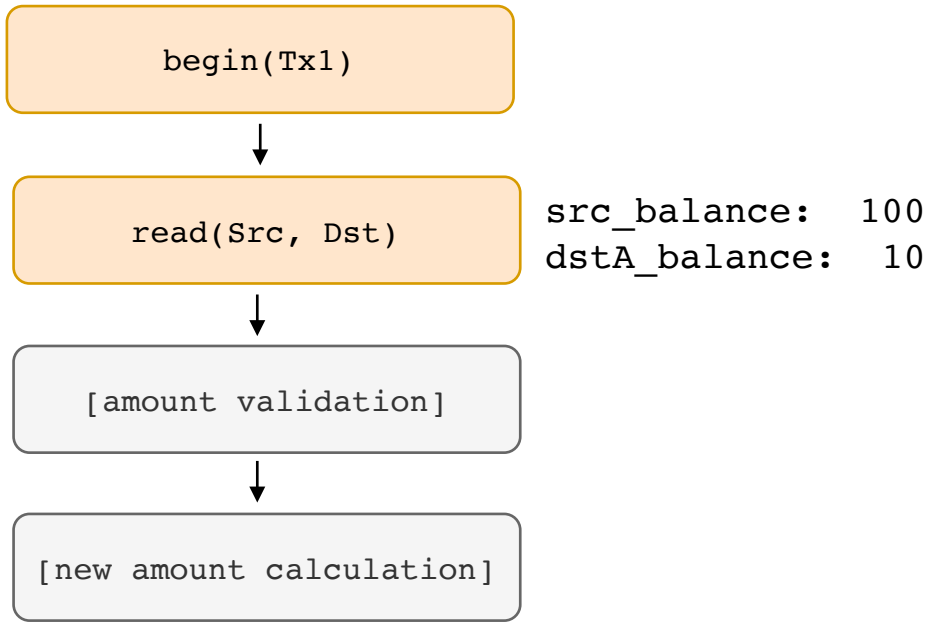

begin(Tx1)
read(Src, Dst)
src_balance: 100
dstA_balance: 10
[amount validation]
begin(Tx2)
read(Src, Dst)
src_balance: 100
dstB_balance: 20

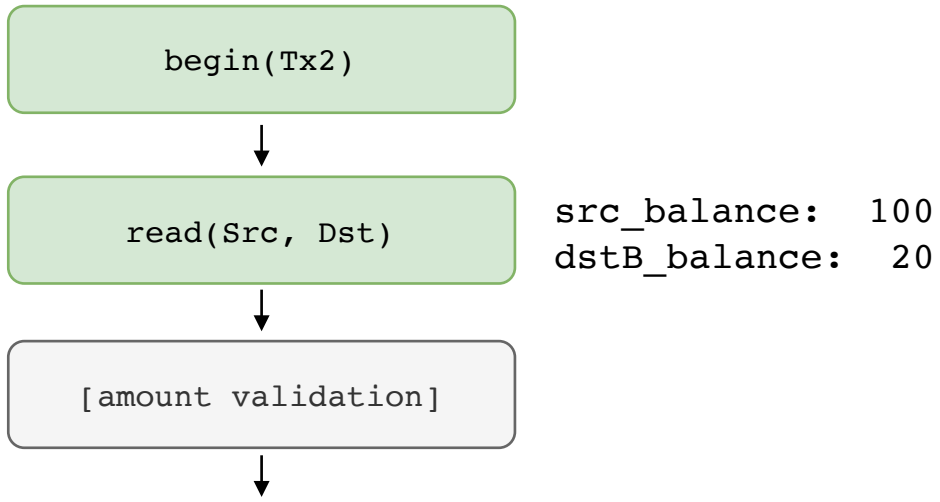

begin(Tx1)
read(Src, Dst)
src_balance: 100
dstA_balance: 10
[amount validation]
[new amount calculation]
begin(Tx2)
read(Src, Dst)
src_balance: 100
dstB_balance: 20

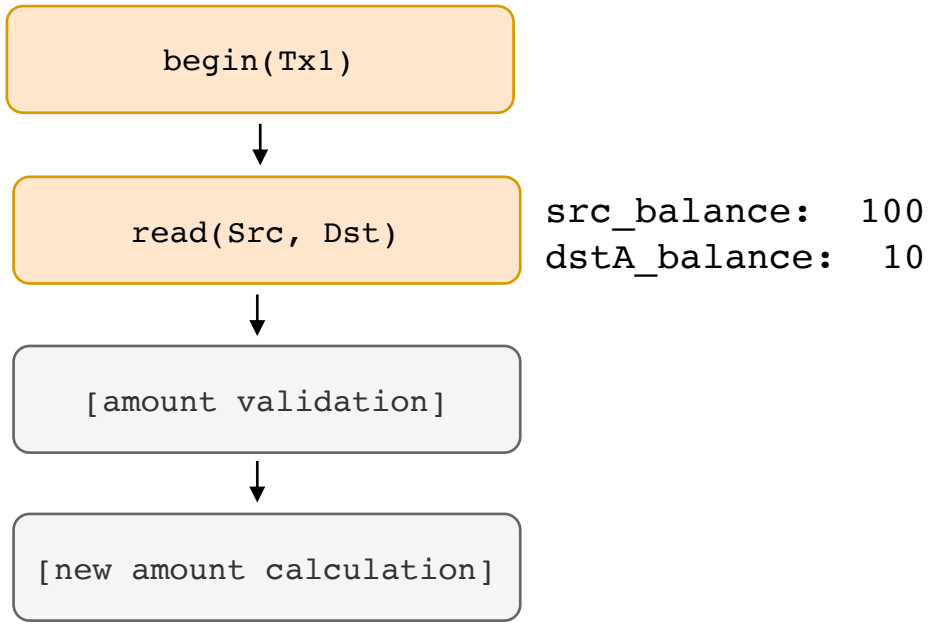

begin(Tx1)
read(Src, Dst)
src_balance: 100
dstA_balance: 10
[amount validation]
[new amount calculation]
begin(Tx2)
read(Src, Dst)
src_balance: 100
dstB_balance: 20

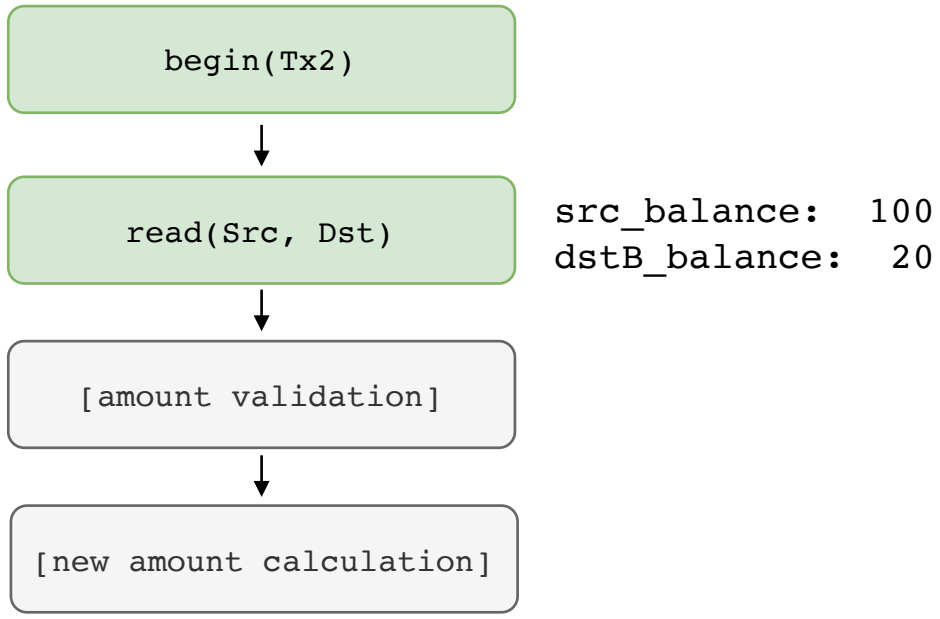

begin(Tx1)
read(Src, Dst)
[amount validation]
[new amount calculation]
src_balance: 100
dstA_balance: 10

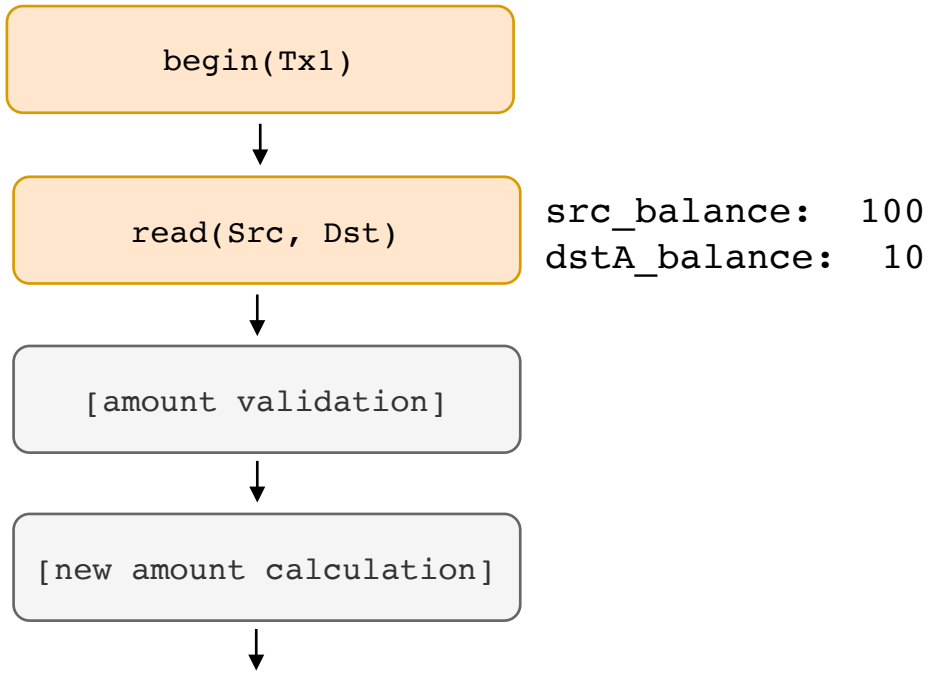

begin(Tx2)
read(Src, Dst)
[amount validation]
src_balance: 100
dstB_balance: 20

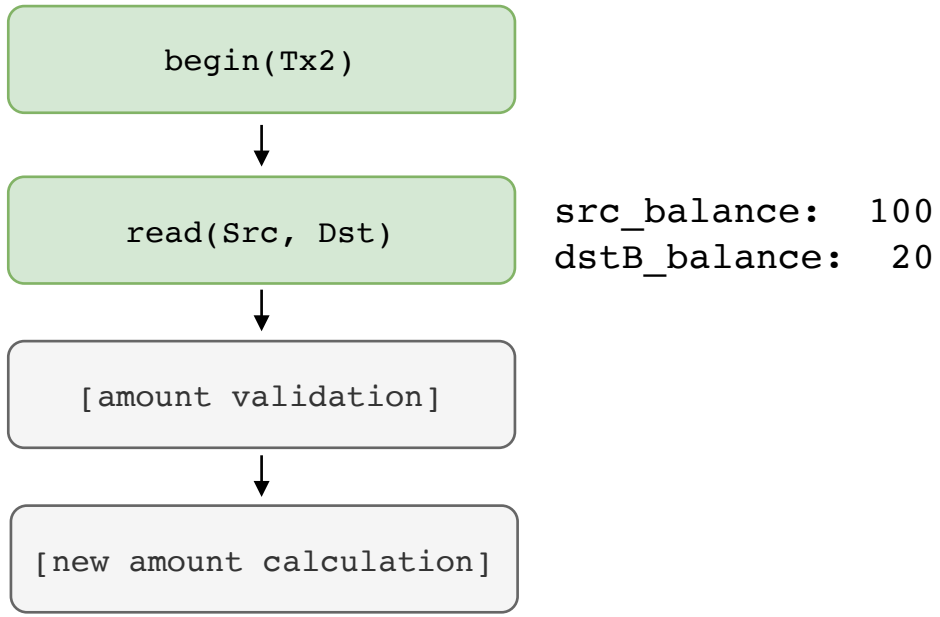

begin(Tx1)
read(Src, Dst)
[amount validation]
[new amount calculation]
src_balance: 100
dstA_balance: 10

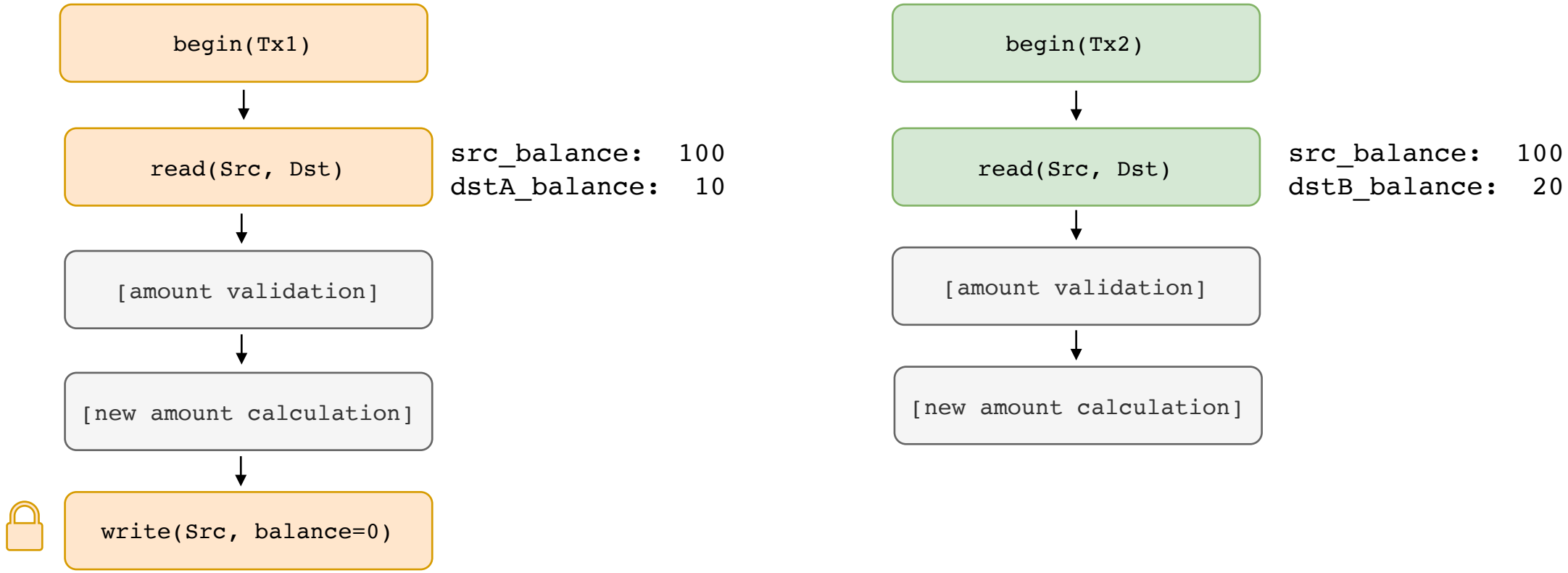

begin(Tx2)
read(Src, Dst)
[amount validation]
src_balance: 100
dstB_balance: 20


begin(Tx1)
read(Src, Dst)
[amount validation]
[new amount calculation]
src_balance: 100
dstA_balance: 10


begin(Tx2)
read(Src, Dst)
[amount validation]
[new amount calculation]
src_balance: 100
dstB_balance: 20

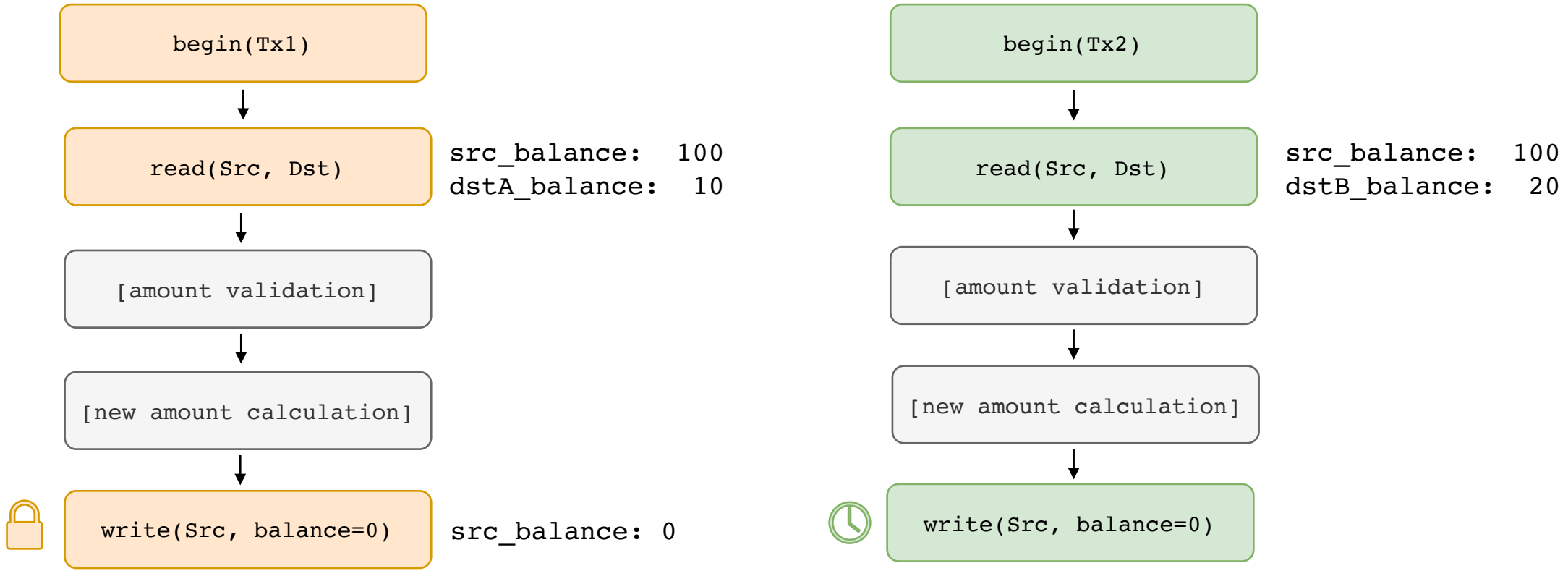

begin(Tx1)
read(Src, Dst)
src_balance: 100
dstA_balance: 10
[amount validation]
[new amount calculation]

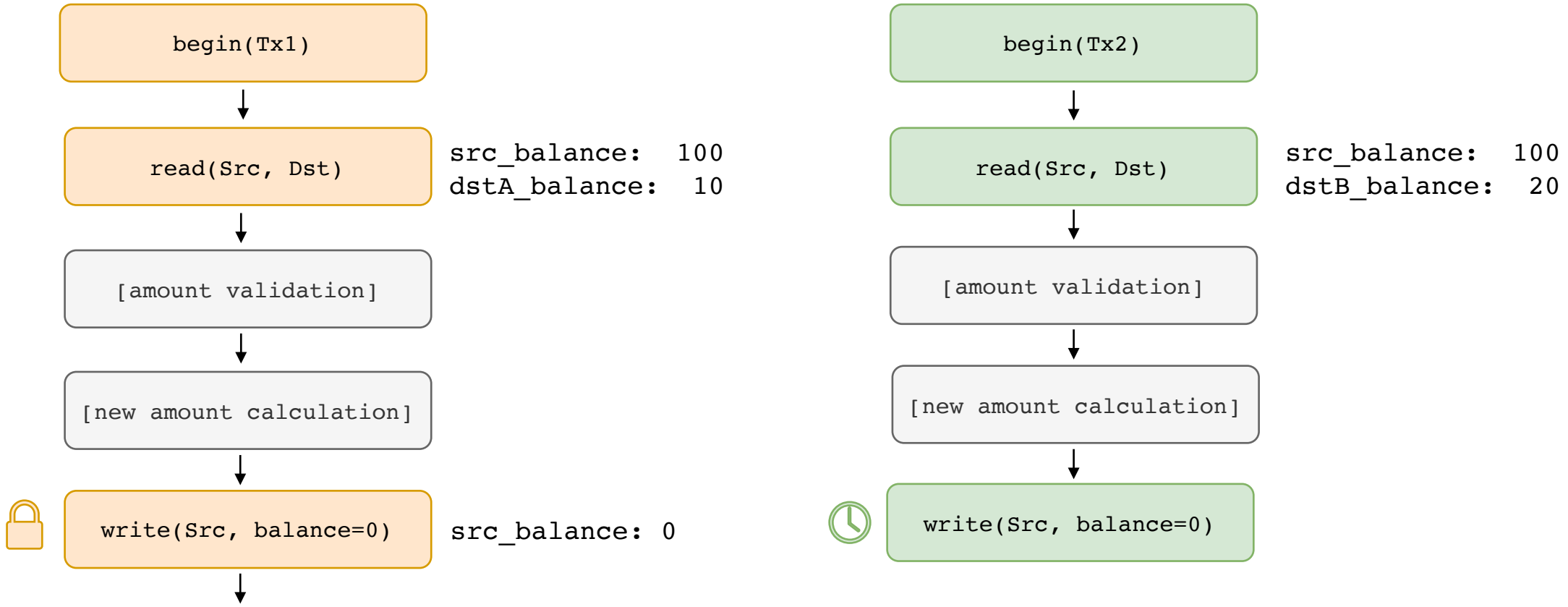

begin(Tx2)
read(Src, Dst)
src_balance: 100
dstB_balance: 20
[amount validation]
[new amount calculation]

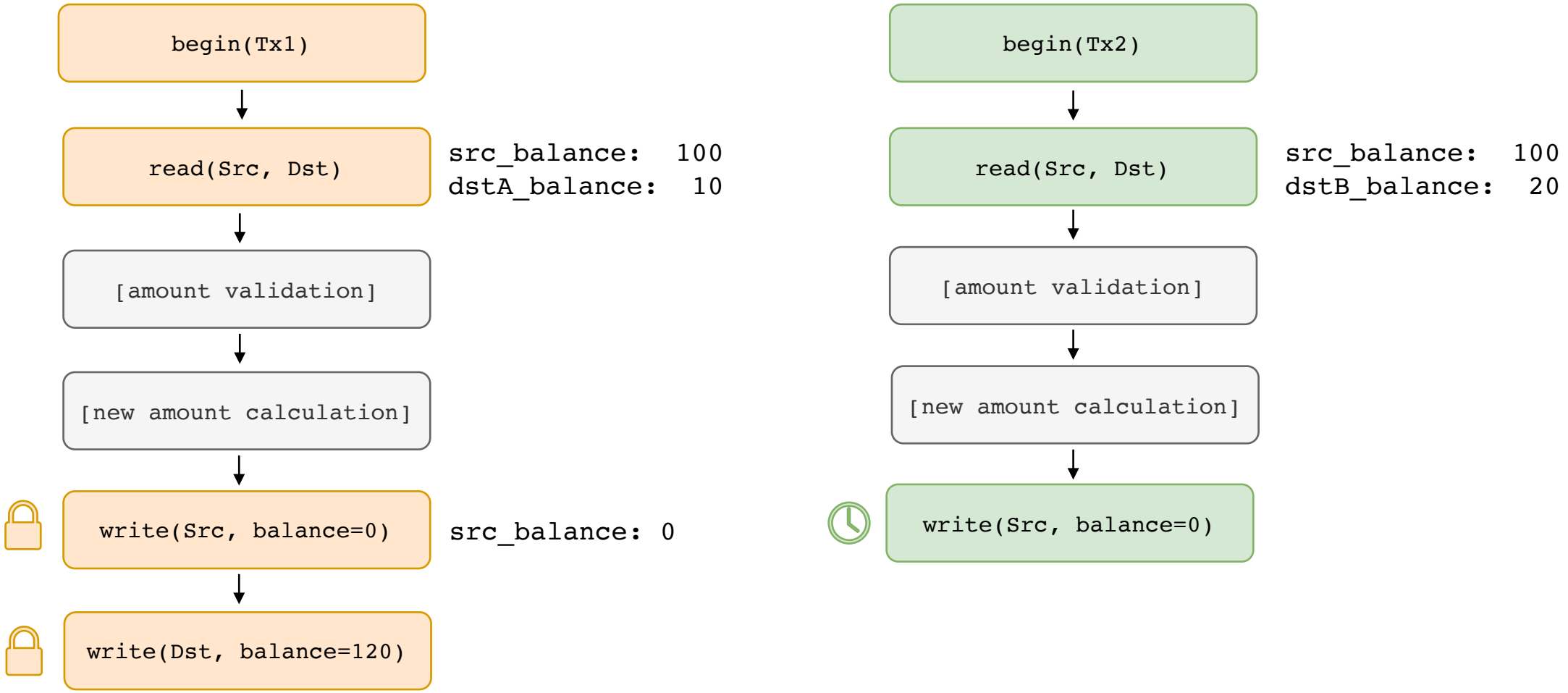

begin(Tx1)
read(Src, Dst)
src_balance: 100
dstA_balance: 10
[amount validation]
[new amount calculation]
write(Src, balance=0)
begin(Tx2)
read(Src, Dst)
src_balance: 100
dstB_balance: 20
[amount validation]
[new amount calculation]

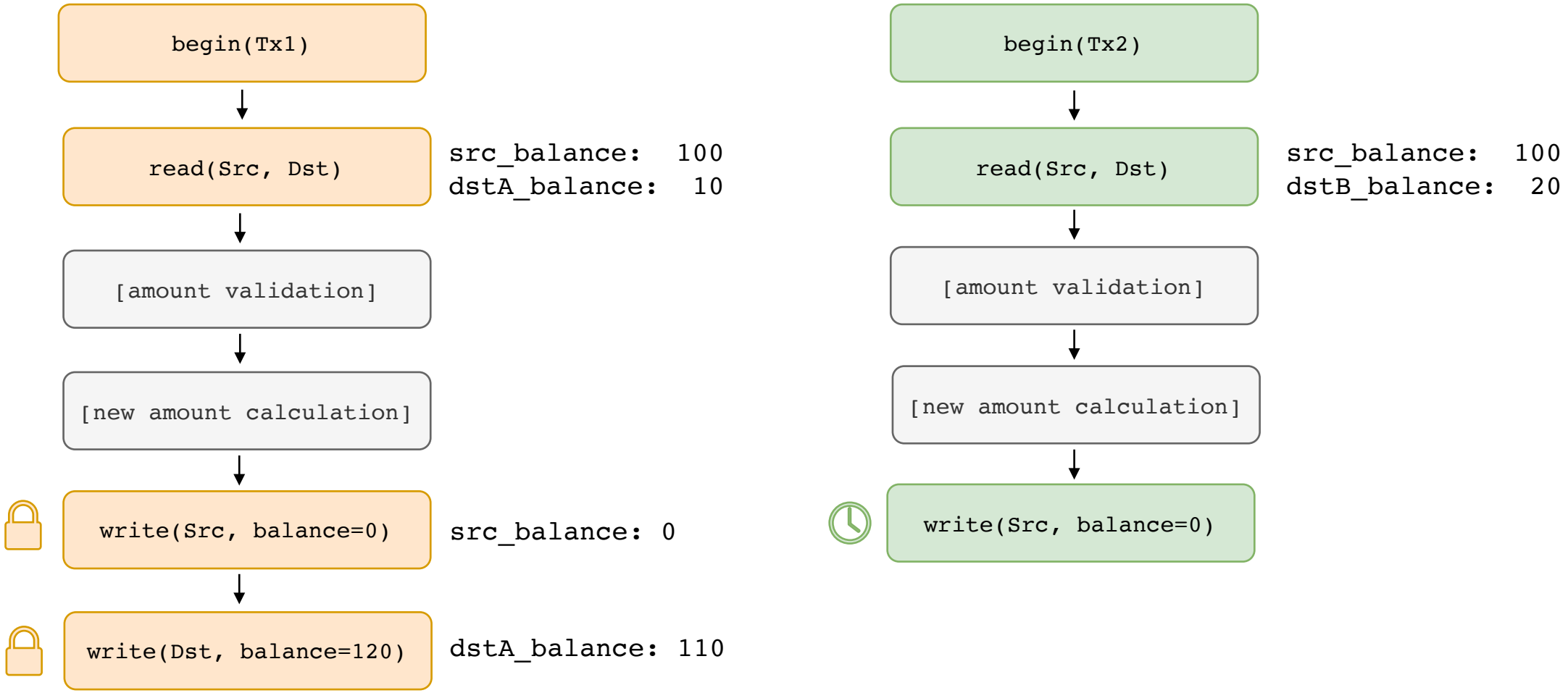

begin(Tx1)
read(Src, Dst)
src_balance: 100
dstA_balance: 10
[amount validation]
[new amount calculation]
write(Src, balance=0)
src_balance: 0
begin(Tx2)
read(Src, Dst)
src_balance: 100
dstB_balance: 20
[amount validation]
[new amount calculation]

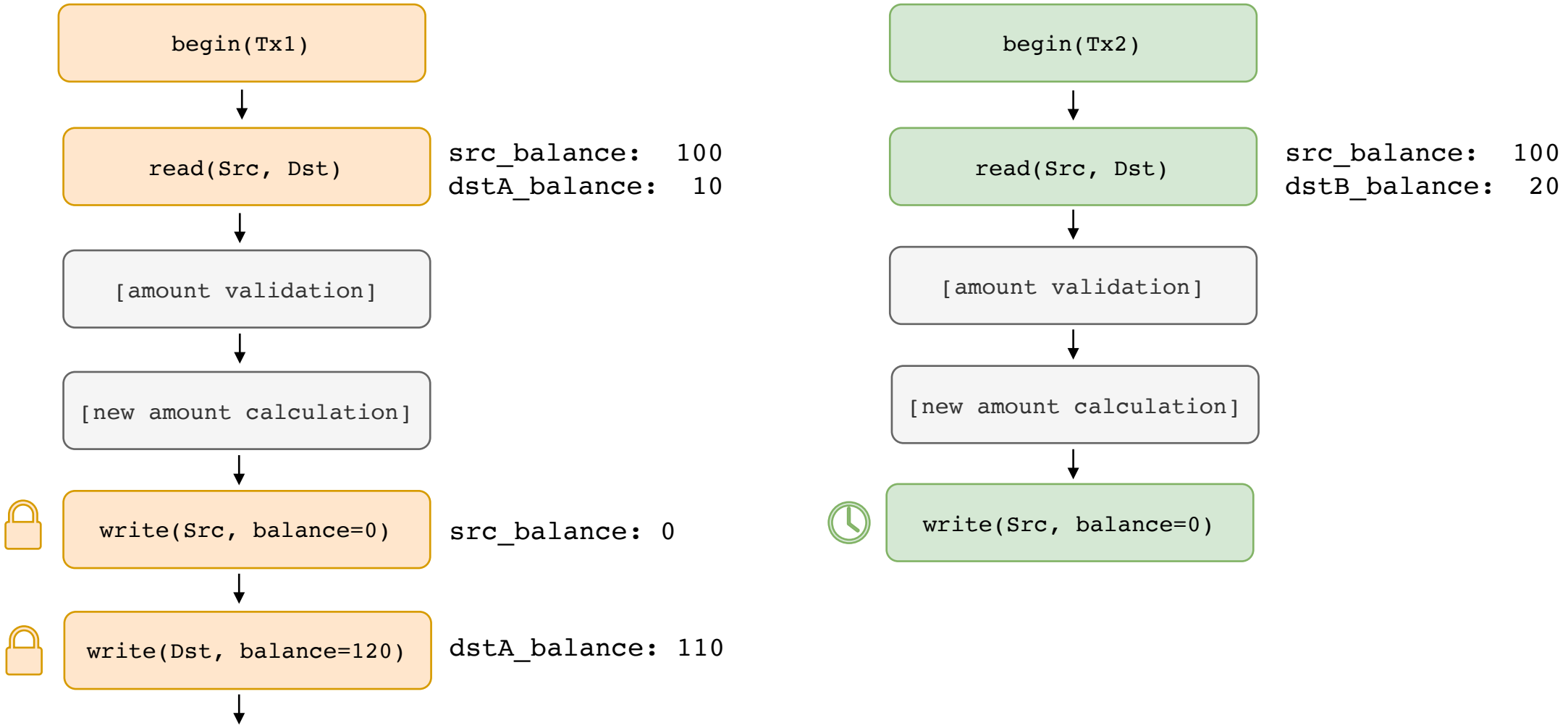

begin(Tx1)
read(Src, Dst)
src_balance: 100
dstA_balance: 10
[amount validation]
[new amount calculation]
write(Src, balance=0)
src_balance: 0
begin(Tx2)
read(Src, Dst)
src_balance: 100
dstB_balance: 20
[amount validation]
[new amount calculation]

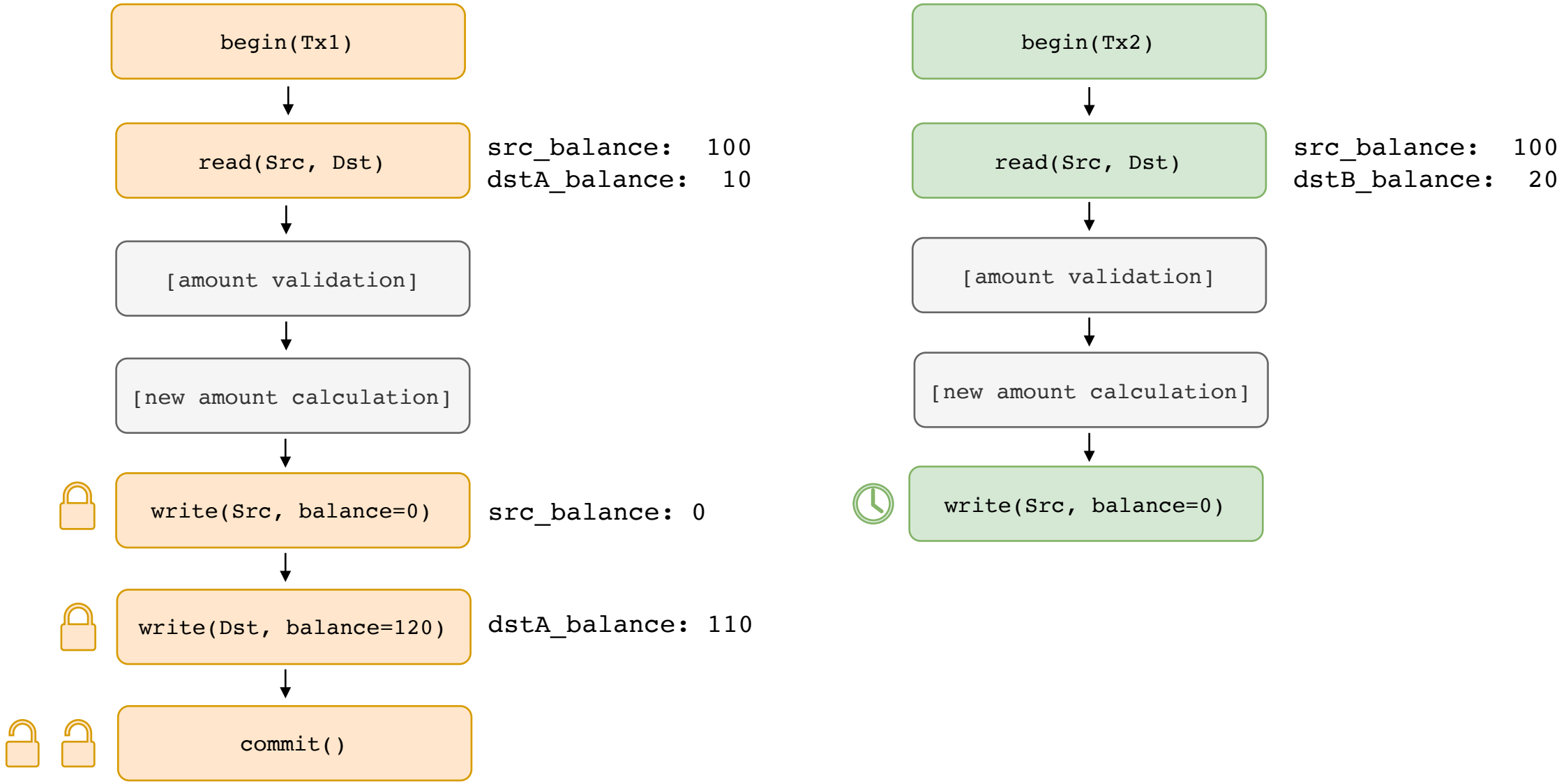

begin(Tx1)
read(Src, Dst)
src_balance: 100
dstA_balance: 10
[amount validation]
[new amount calculation]
write(Src, balance=0)
src_balance: 0
begin(Tx2)
read(Src, Dst)
src_balance: 100
dstB_balance: 20
[amount validation]
[new amount calculation]
write(Src, balance=0)

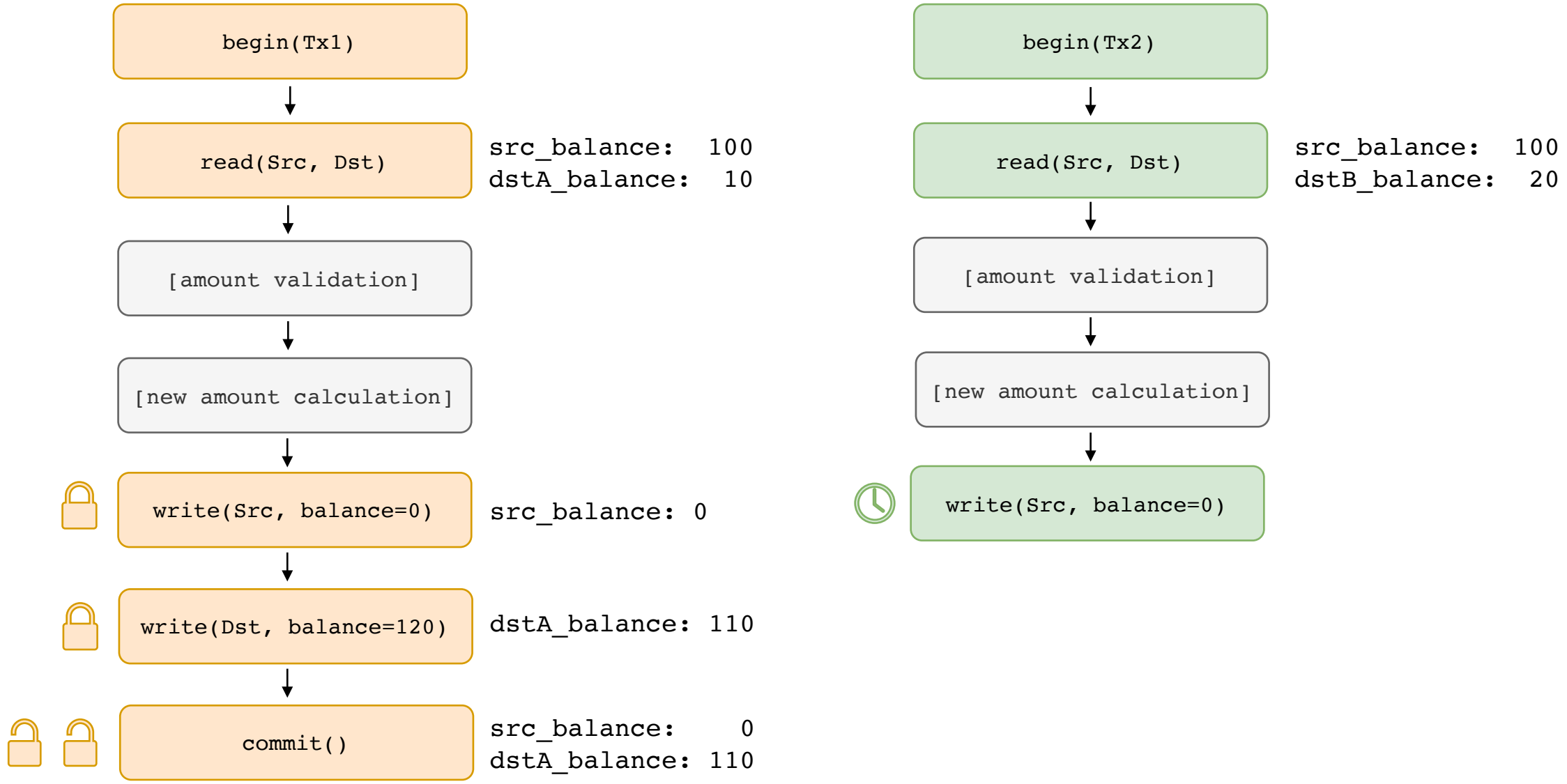

begin(Tx1)
read(Src, Dst)
src_balance: 100
dstA_balance: 10
[amount validation]
[new amount calculation]
write(Src, balance=0)
src_balance: 0
begin(Tx2)
read(Src, Dst)
src_balance: 100
dstB_balance: 20
[amount validation]
[new amount calculation]
write(Src, balance=0)

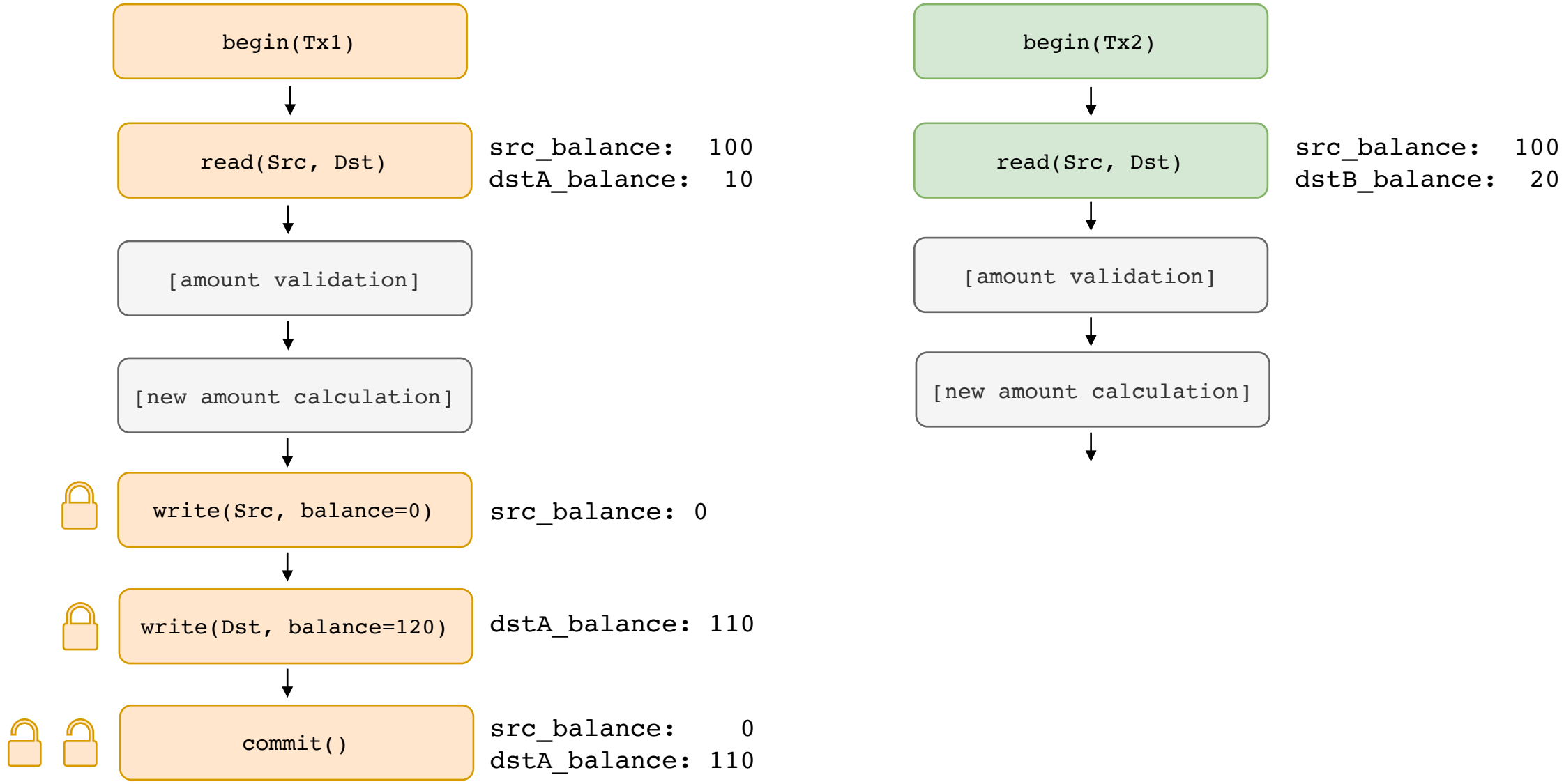

begin(Tx1)
read(Src, Dst)
src_balance: 100
dstA_balance: 10
[amount validation]
[new amount calculation]
write(Src, balance=0)
src_balance: 0
write(Dst, balance=120)
begin(Tx2)
read(Src, Dst)
src_balance: 100
dstB_balance: 20
[amount validation]
[new amount calculation]
write(Src, balance=0)

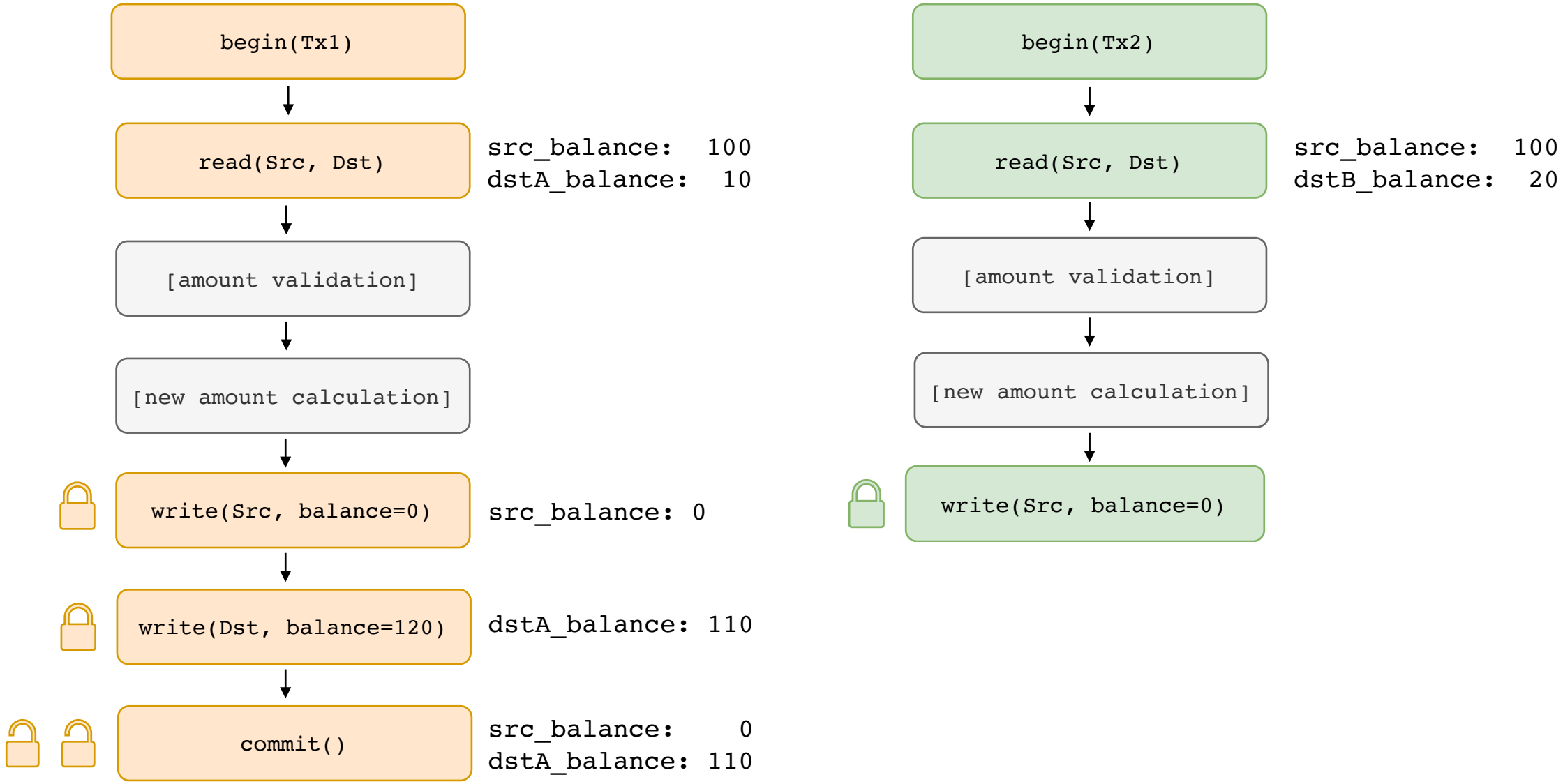

begin(Tx1)
read(Src, Dst)
src_balance: 100
dstA_balance: 10
[amount validation]
[new amount calculation]
write(Src, balance=0)
src_balance: 0
write(Dst, balance=120)
dstA_balance: 110
begin(Tx2)
read(Src, Dst)
src_balance: 100
dstB_balance: 20
[amount validation]
[new amount calculation]
write(Src, balance=0)

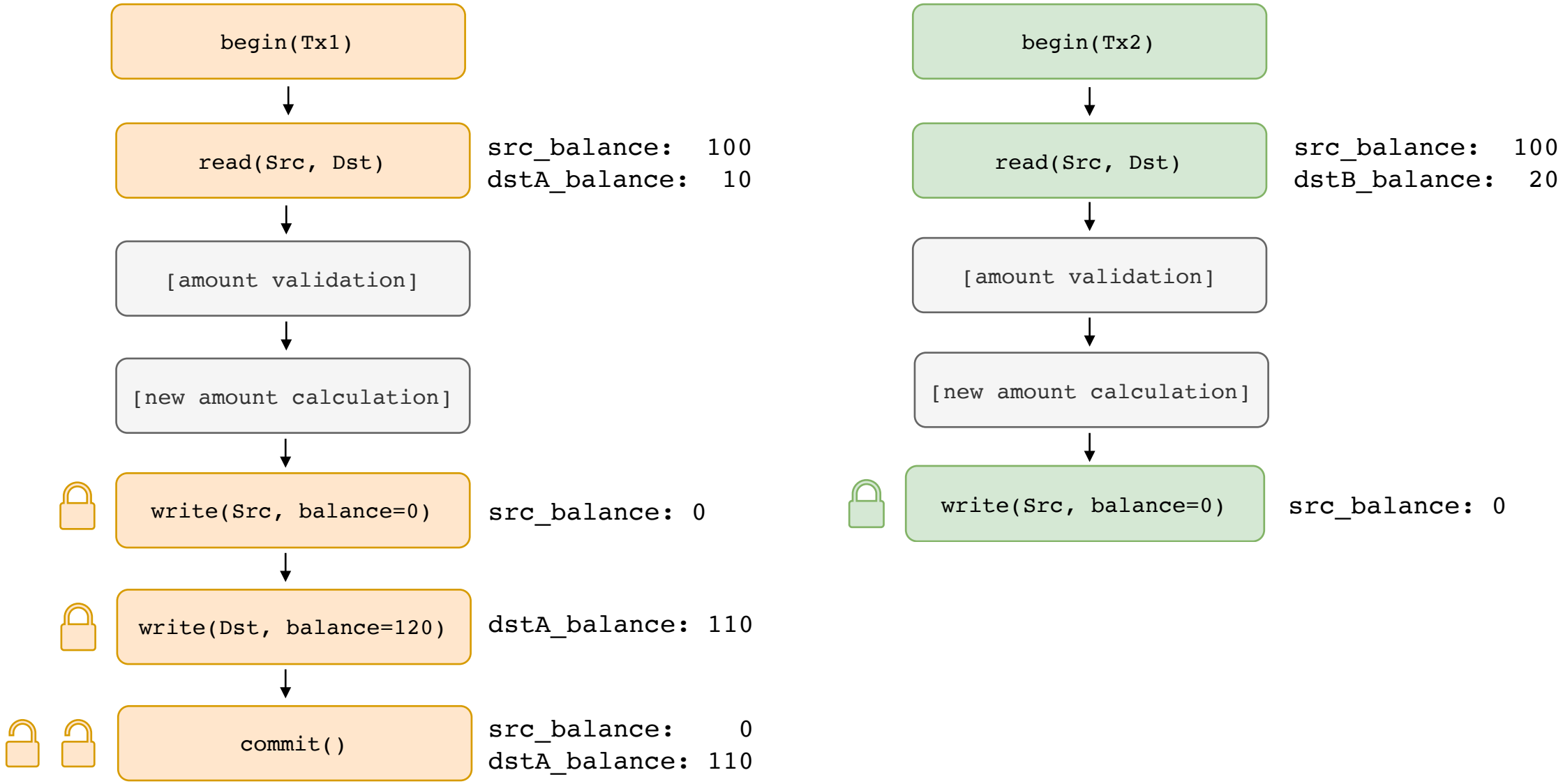

begin(Tx1)
read(Src, Dst)
src_balance: 100
dstA_balance: 10
[amount validation]
[new amount calculation]
write(Src, balance=0)
src_balance: 0
write(Dst, balance=120)
dstA_balance: 110
begin(Tx2)
read(Src, Dst)
src_balance: 100
dstB_balance: 20
[amount validation]
[new amount calculation]
write(Src, balance=0)

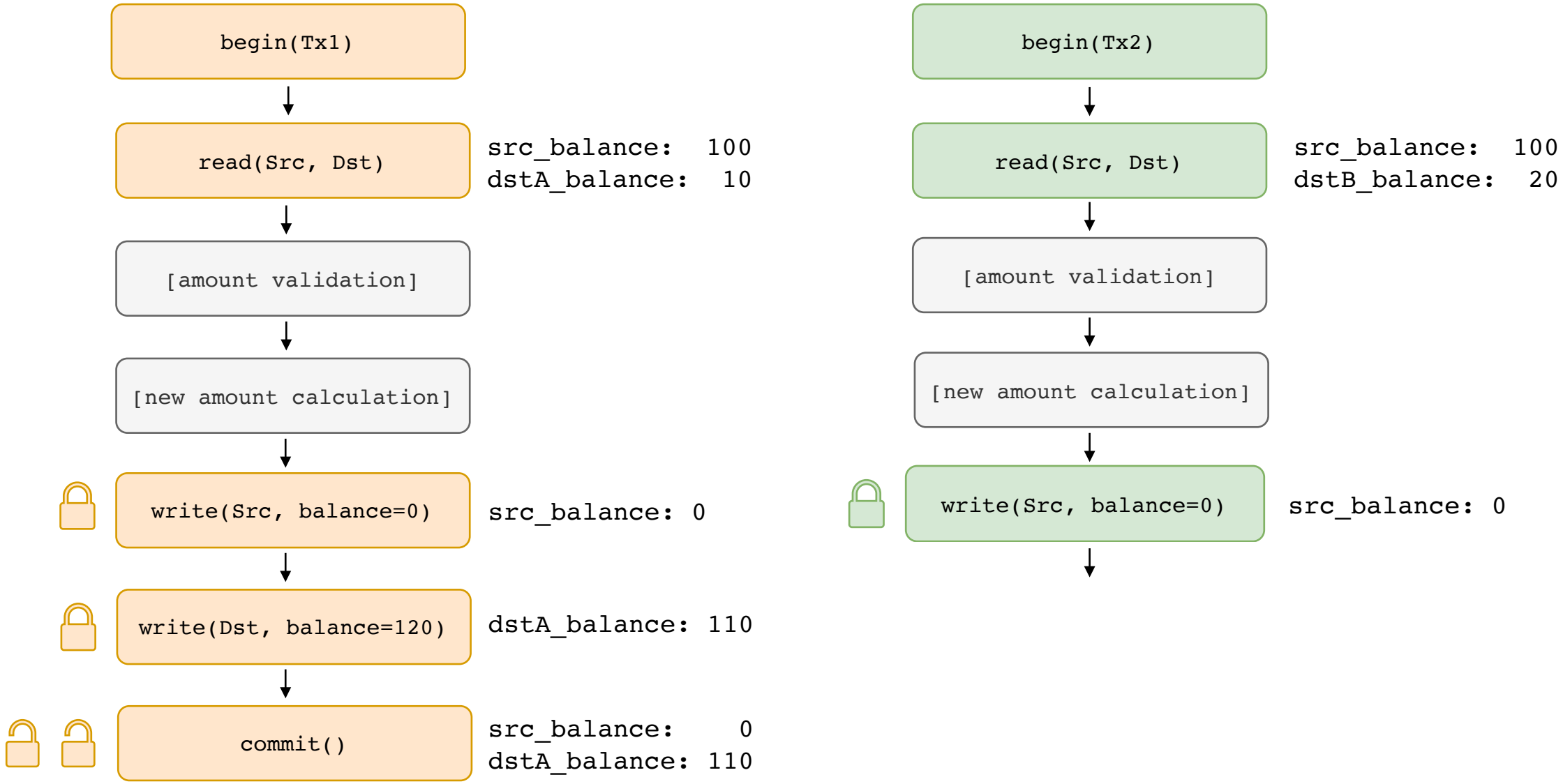

begin(Tx1)
read(Src, Dst)
src_balance: 100
dstA_balance: 10
[amount validation]
[new amount calculation]
write(Src, balance=0)
src_balance: 0
write(Dst, balance=120)
dstA_balance: 110
commit()
begin(Tx2)
read(Src, Dst)
src_balance: 100
dstB_balance: 20
[amount validation]
[new amount calculation]
write(Src, balance=0)

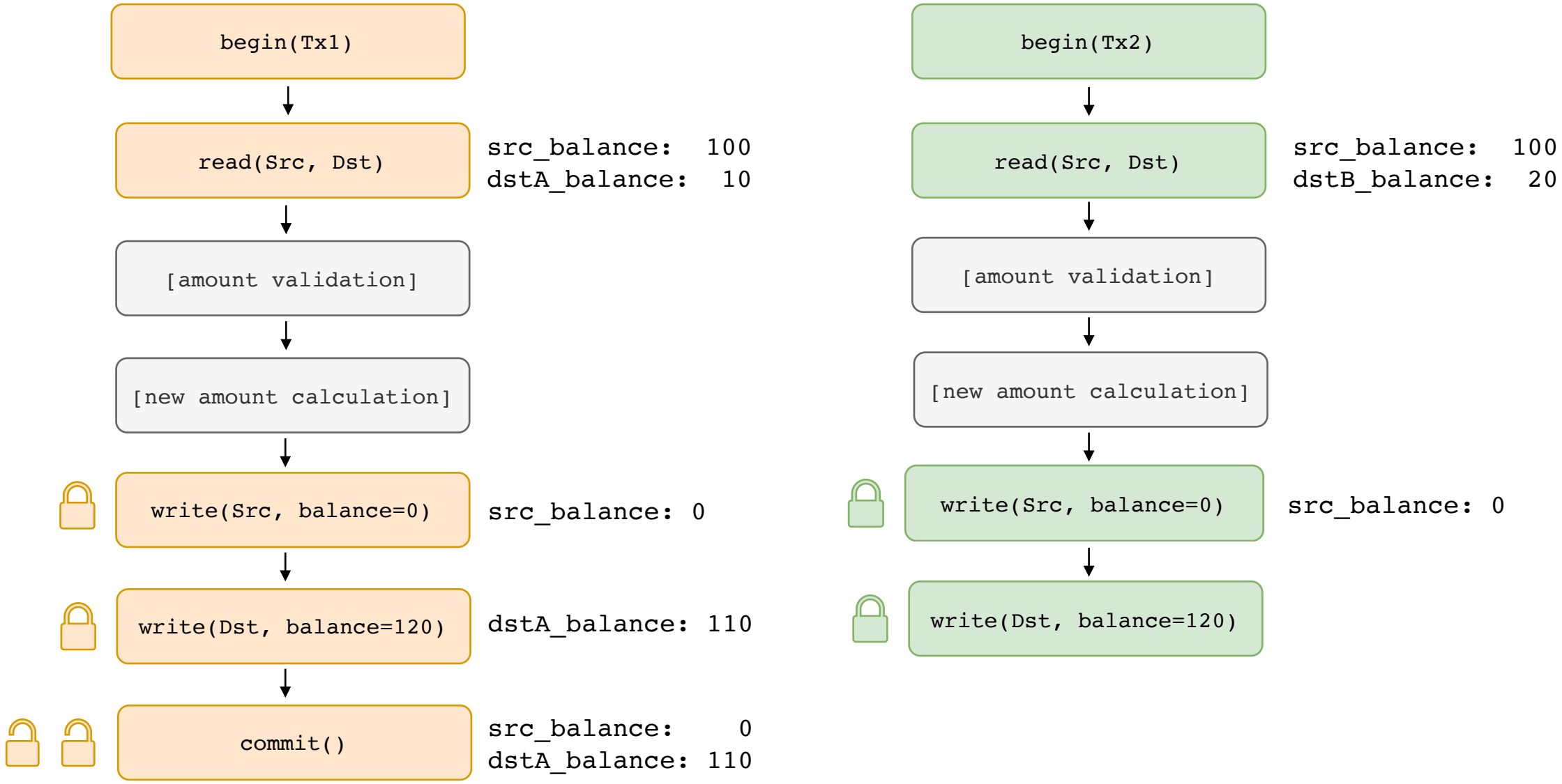

begin(Tx1)
read(Src, Dst)
src_balance: 100
dstA_balance: 10
[amount validation]
[new amount calculation]
write(Src, balance=0)
src_balance: 0
write(Dst, balance=120)
dstA_balance: 110
commit()
src_balance: 0
dstA_balance: 110
begin(Tx2)
read(Src, Dst)
src_balance: 100
dstB_balance: 20
[amount validation]
[new amount calculation]
write(Src, balance=0)

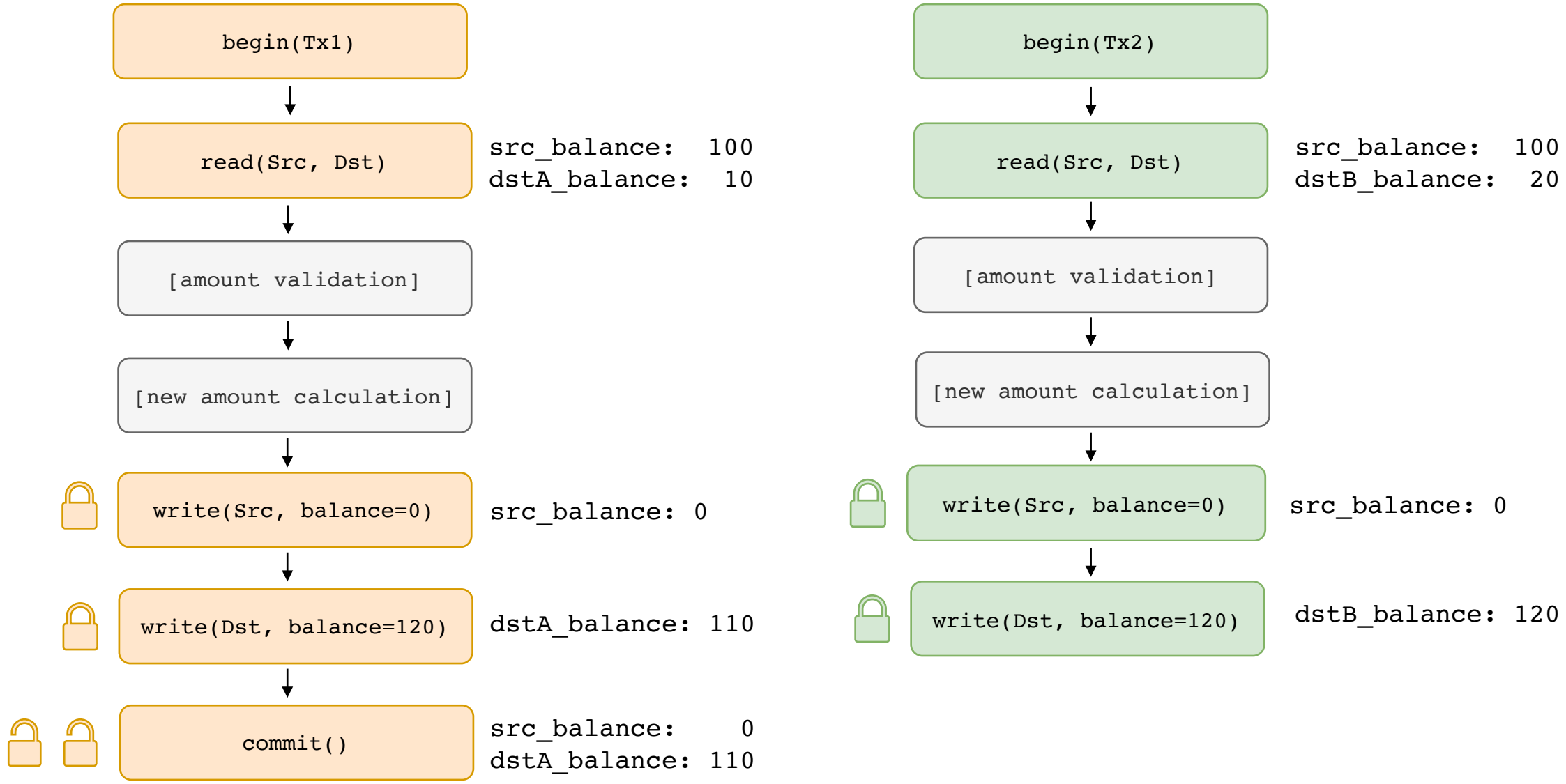

begin(Tx1)
read(Src, Dst)
src_balance: 100
dstA_balance: 10
[amount validation]
[new amount calculation]
write(Src, balance=0)
src_balance: 0
write(Dst, balance=120)
dstA_balance: 110
commit()
src_balance: 0
dstA_balance: 110
begin(Tx2)
read(Src, Dst)
src_balance: 100
dstB_balance: 20
[amount validation]
[new amount calculation]

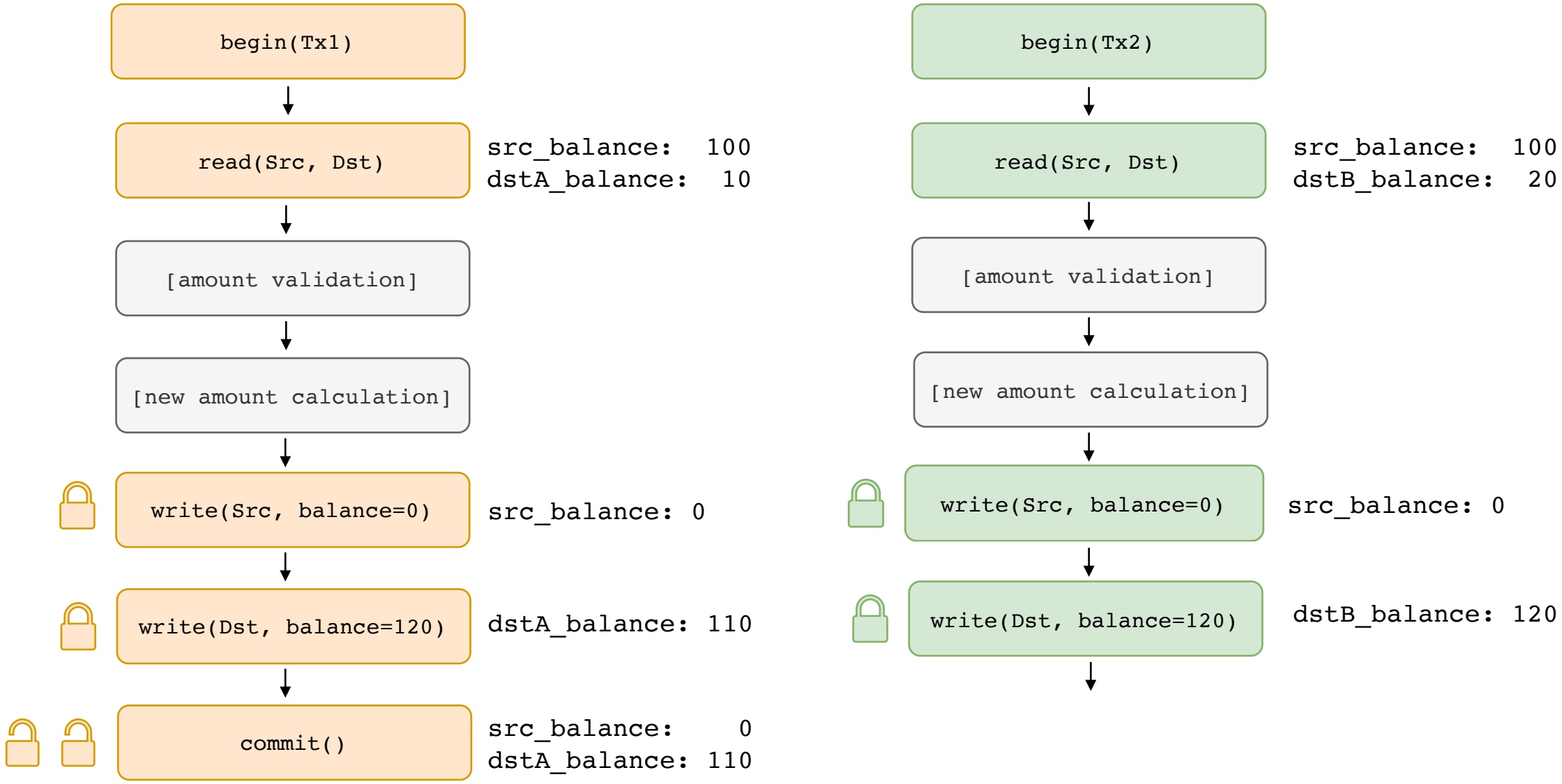

begin(Tx1)
read(Src, Dst)
src_balance: 100
dstA_balance: 10
[amount validation]
[new amount calculation]
write(Src, balance=0)
src_balance: 0
write(Dst, balance=120)
dstA_balance: 110
commit()
src_balance: 0
dstA_balance: 110
begin(Tx2)
read(Src, Dst)
src_balance: 100
dstB_balance: 20
[amount validation]
[new amount calculation]
write(Src, balance=0)

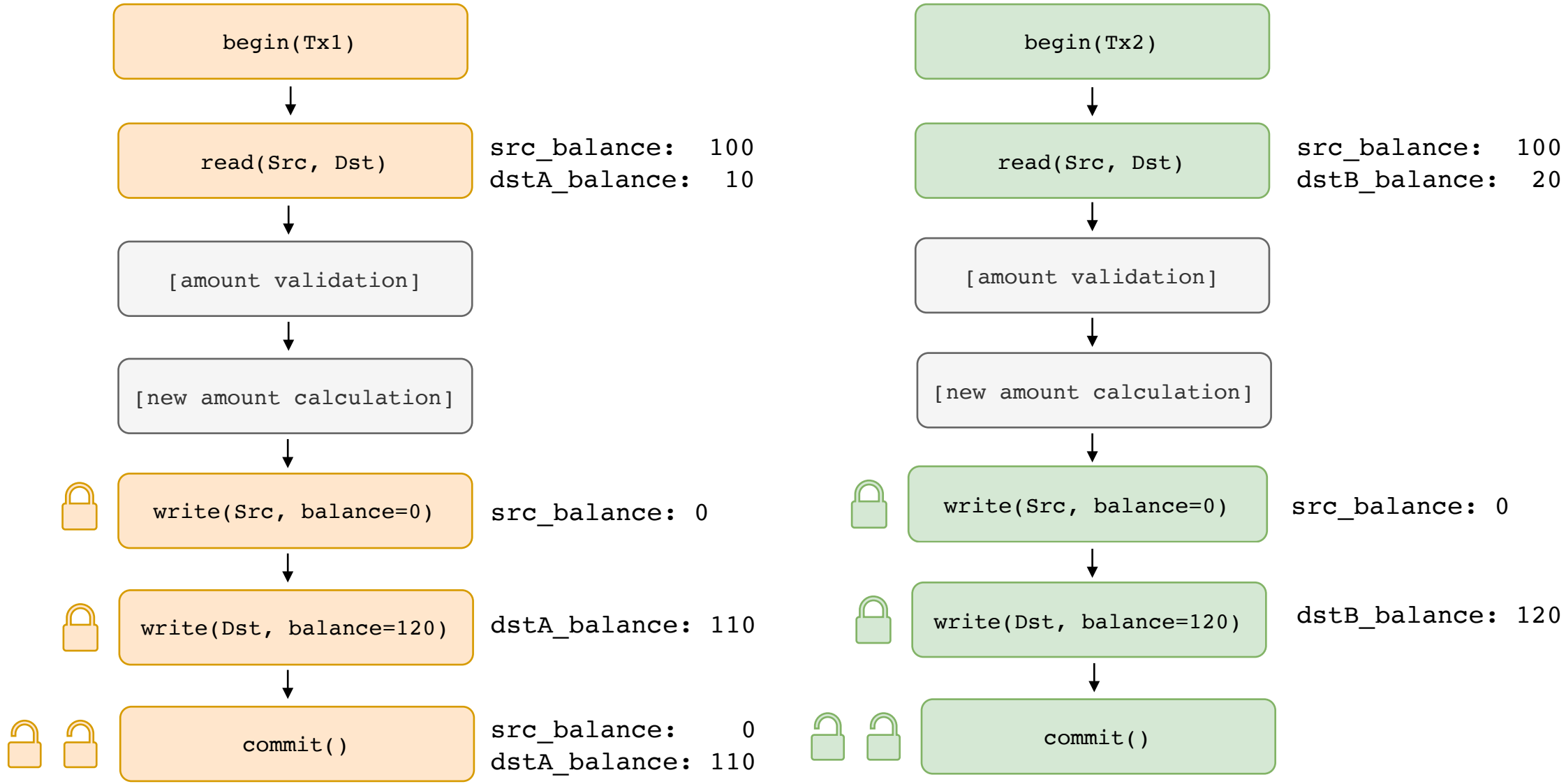

begin(Tx1)
read(Src, Dst)
src_balance: 100
dstA_balance: 10
[amount validation]
[new amount calculation]
write(Src, balance=0)
src_balance: 0
write(Dst, balance=120)
dstA_balance: 110
commit()
src_balance: 0
dstA_balance: 110
begin(Tx2)
read(Src, Dst)
src_balance: 100
dstB_balance: 20
[amount validation]
[new amount calculation]
write(Src, balance=0)
src_balance: 0

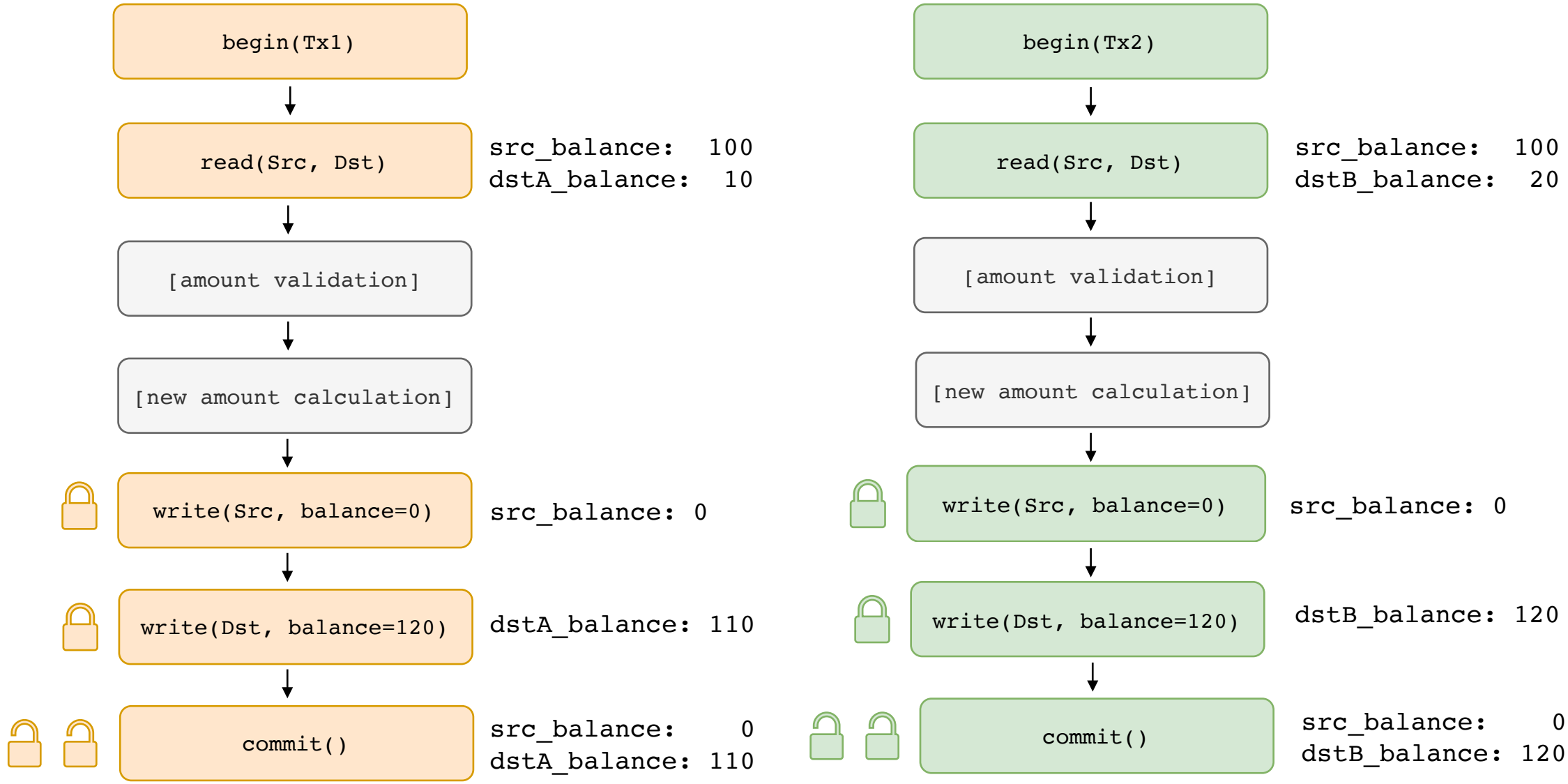

begin(Tx1)
read(Src, Dst)
src_balance: 100
dstA_balance: 10
[amount validation]
[new amount calculation]
write(Src, balance=0)
src_balance: 0
write(Dst, balance=120)
dstA_balance: 110
commit()
src_balance: 0
dstA_balance: 110
begin(Tx2)
read(Src, Dst)
src_balance: 100
dstB_balance: 20
[amount validation]
[new amount calculation]
write(Src, balance=0)
src_balance: 0


begin(Tx1)
read(Src, Dst)
src_balance: 100
dstA_balance: 10
[amount validation]
[new amount calculation]
write(Src, balance=0)
src_balance: 0
write(Dst, balance=120)
dstA_balance: 110
commit()
src_balance: 0
dstA_balance: 110
begin(Tx2)
read(Src, Dst)
src_balance: 100
dstB_balance: 20
[amount validation]
[new amount calculation]
write(Src, balance=0)
src_balance: 0
write(Dst, balance=120)


begin(Tx1)
read(Src, Dst)
src_balance: 100
dstA_balance: 10
[amount validation]
[new amount calculation]
write(Src, balance=0)
src_balance: 0
write(Dst, balance=120)
dstA_balance: 110
commit()
src_balance: 0
dstA_balance: 110
begin(Tx2)
read(Src, Dst)
src_balance: 100
dstB_balance: 20
[amount validation]
[new amount calculation]
write(Src, balance=0)
src_balance: 0
write(Dst, balance=120)
dstB_balance: 120


begin(Tx1)
read(Src, Dst)
src_balance: 100
dstA_balance: 10
[amount validation]
[new amount calculation]
write(Src, balance=0)
src_balance: 0
write(Dst, balance=120)
dstA_balance: 110
commit()
src_balance: 0
dstA_balance: 110
begin(Tx2)
read(Src, Dst)
src_balance: 100
dstB_balance: 20
[amount validation]
[new amount calculation]
write(Src, balance=0)
src_balance: 0
write(Dst, balance=120)
dstB_balance: 120


begin(Tx1)
read(Src, Dst)
src_balance: 100
dstA_balance: 10
[amount validation]
[new amount calculation]
write(Src, balance=0)
src_balance: 0
write(Dst, balance=120)
dstA_balance: 110
commit()
src_balance: 0
dstA_balance: 110
begin(Tx2)
read(Src, Dst)
src_balance: 100
dstB_balance: 20
[amount validation]
[new amount calculation]
write(Src, balance=0)
src_balance: 0
write(Dst, balance=120)
dstB_balance: 120
commit()


begin(Tx1)
read(Src, Dst)
src_balance: 100
dstA_balance: 10
[amount validation]
[new amount calculation]
write(Src, balance=0)
src_balance: 0
write(Dst, balance=120)
dstA_balance: 110
commit()
src_balance: 0
dstA_balance: 110
begin(Tx2)
read(Src, Dst)
src_balance: 100
dstB_balance: 20
[amount validation]
[new amount calculation]
write(Src, balance=0)
src_balance: 0
write(Dst, balance=120)
dstB_balance: 120
commit()
src_balance: 0
dstB_balance: 120

Exploitation

Exploitation

- Multithreaded `for` loops

Exploitation

- Multithreaded `for` loops
- BurpSuite

Exploitation

- Multithreaded `for` loops
- BurpSuite
  - Intruder

Exploitation

- Multithreaded `for` loops
- BurpSuite
  - Intruder
  - Turbo Intruder

Exploitation

- Multithreaded `for` loops
- BurpSuite
  - Intruder
  - Turbo Intruder
    - Last-byte sync (HTTP 1.1)

Exploitation

- Multithreaded `for` loops
- BurpSuite
  - Intruder
  - Turbo Intruder
    - Last-byte sync (HTTP 1.1)
    - Single packet attack (HTTP 2.0)

How Can We Fix This?

How Can We Fix This?

- Optimistic locking

Optimistic Locking

Optimistic Locking

- Introduce a logical clock (ex. `version` column)
- Verify the `version` whenever a change is made

Optimistic Locking

```
CREATE TABLE users(
  id INT PRIMARY KEY NOT NULL AUTO_INCREMENT,
  name TEXT NOT NULL,
  balance INT NOT NULL
  version INT NOT NULL AUTO_INCREMENT);
```

Optimistic Locking

```
CREATE TABLE users(
  id INT PRIMARY KEY NOT NULL AUTO_INCREMENT,
  name TEXT NOT NULL,
  balance INT NOT NULL
  version INT NOT NULL AUTO_INCREMENT);
```

```
UPDATE users
SET balance = 100
WHERE id = 1 AND version = $last_seen_version
```

How Can We Fix This?

- Optimistic locking
- Pessimistic locking

Pessimistic Locking

Pessimistic Locking

- Application-level locking

Pessimistic Locking

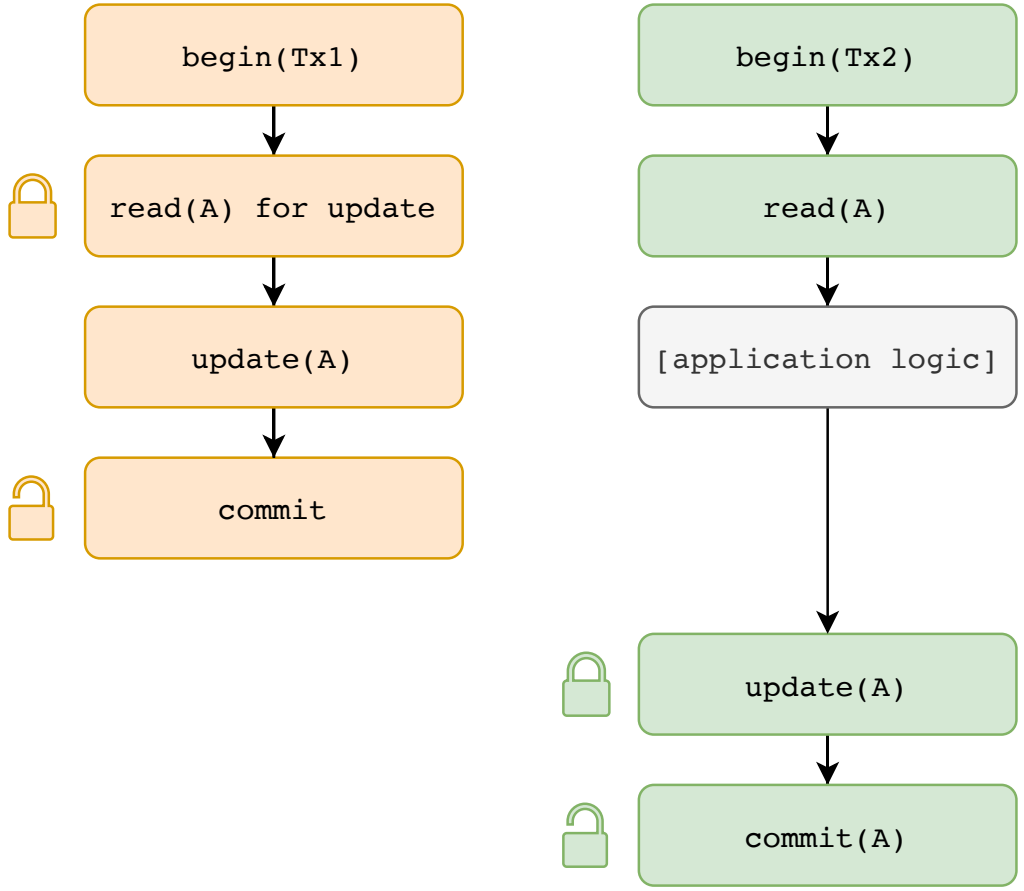
- Application-level locking
- Database-level locking (`FOR SHARE` / `FOR UPDATE`)

Pessimistic Locking

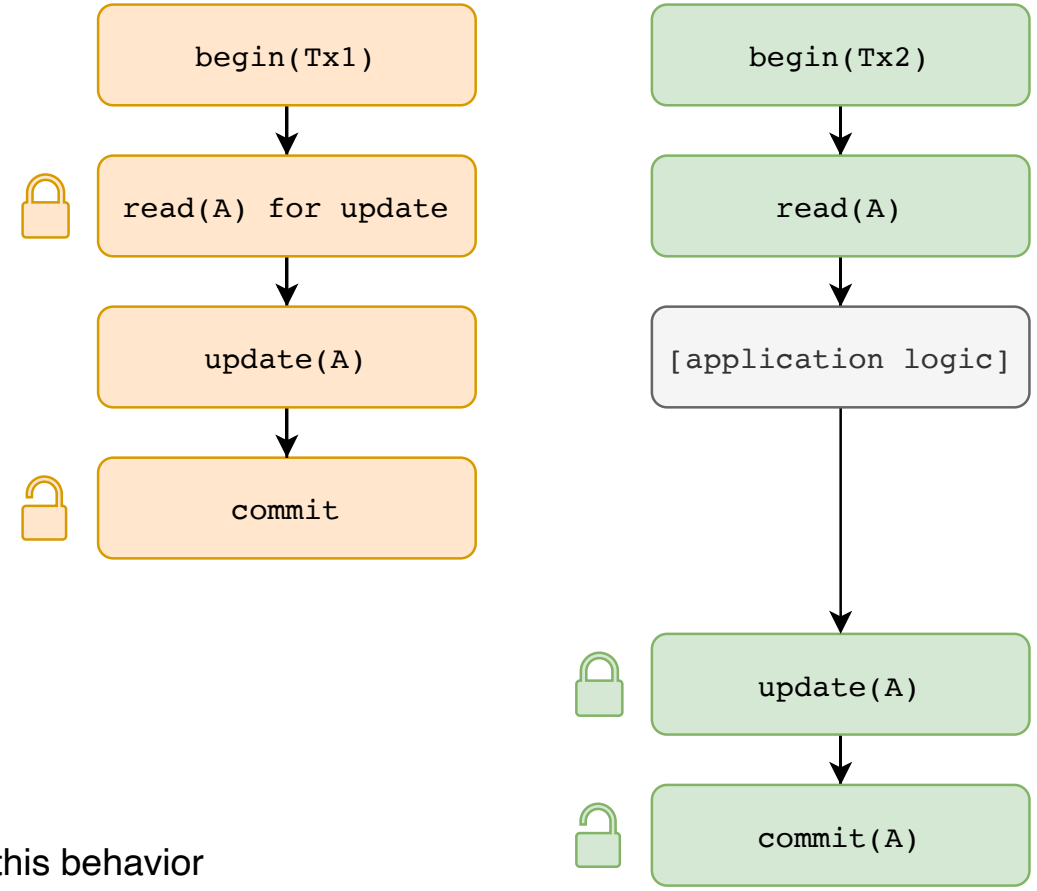
- Application-level locking
- Database-level locking (`FOR SHARE` / `FOR UPDATE`)

```
SELECT * FROM users WHERE id = 1 FOR UPDATE
```

Manual Locking - Lost Update



Manual Locking - Lost Update



* higher isolation levels may prevent this behavior

How Can We Fix This?

- Optimistic locking
- Pessimistic locking
- Transaction isolation

Transaction Isolation Levels

Transaction Isolation Levels

- Serializable

Transaction Isolation Levels

- Serializable
- Repeatable Reads

Transaction Isolation Levels

- Serializable
- Repeatable Reads
- Read Committed

Transaction Isolation Levels

- Serializable
- Repeatable Reads
- Read Committed
- Read Uncommitted

Transaction Isolation Levels

- Serializable
- Repeatable Reads
- Read Committed
- Read Uncommitted

	Dirty Reads	Non-Repeatable Reads	Phantom Reads
Serializable	✗	✗	✗
Repeatable Read	✗	✗	✓
Read Committed	✗	✓	✓
Read Uncommited	✓	✓	✓

Transaction Isolation Levels

- Serializable
- Repeatable Reads
- Read Committed (the usual default)
- Read Uncommitted

	Dirty Reads	Non-Repeatable Reads	Phantom Reads
Serializable	✗	✗	✗
Repeatable Read	✗	✗	✓
Read Committed	✗	✓	✓
Read Uncommited	✓	✓	✓

Detection

Detection

```
rules:
  - id: sql-tx-no-isolation
    message: "SQL transaction without isolation level"
    languages:
      - go
    severity: WARNING
    patterns:
      - pattern: $CONN.Exec($CTX, $BEGIN)
      - metavariable-regex:
          metavariable: $BEGIN
          regex: ("begin transaction"|"BEGIN TRANSACTION"|"begin"|"BEGIN")
```

Detection

```
rules:
  - id: pgx-tx-missing-options
    message: "Postgres transaction options not set"
    languages:
      - go
    severity: WARNING
    patterns:
      - pattern: $CONN.BeginTx($CTX)
```

Detection

```
rules:
  - id: pgx-tx-missing-isolation
    message: "Postgres transaction isolation level not set"
    languages:
      - go
    severity: WARNING
    patterns:
      - pattern: $CONN.BeginTx($CTX, $OPTS)
      - metavariable-pattern:
          metavariable: $OPTS
          patterns:
            - pattern-not
                $PGX.TxOptions{..., IsoLevel:$LVL, ...}
```

Real-World Exploits

- FlexCoin [1]
- Poloniex [2]

- [1] https://web.archive.org/web/20160408190656/http://www.flexcoin.com/
- [2] https://bitcointalk.org/index.php?topic=499580

Takeaways

Takeaways

- Databases introduce another layer of concurrency

Takeaways

- Databases introduce another layer of concurrency
- Test it yourself

Takeaways

- Databases introduce another layer of concurrency
- Test it yourself
- Familiarize yourself with the database you are using


OWASP 2024
GLOBAL
AppSec
CONGRESS CENTRE
LISBON
JUNE 24-28
THANK YOU