Language and tool support for (available) distributed systems

Kevin De Porre, Jim Bauwens,
Matteo Marra, Carmen Torres Lopez,
Elisa Gonzalez Boix

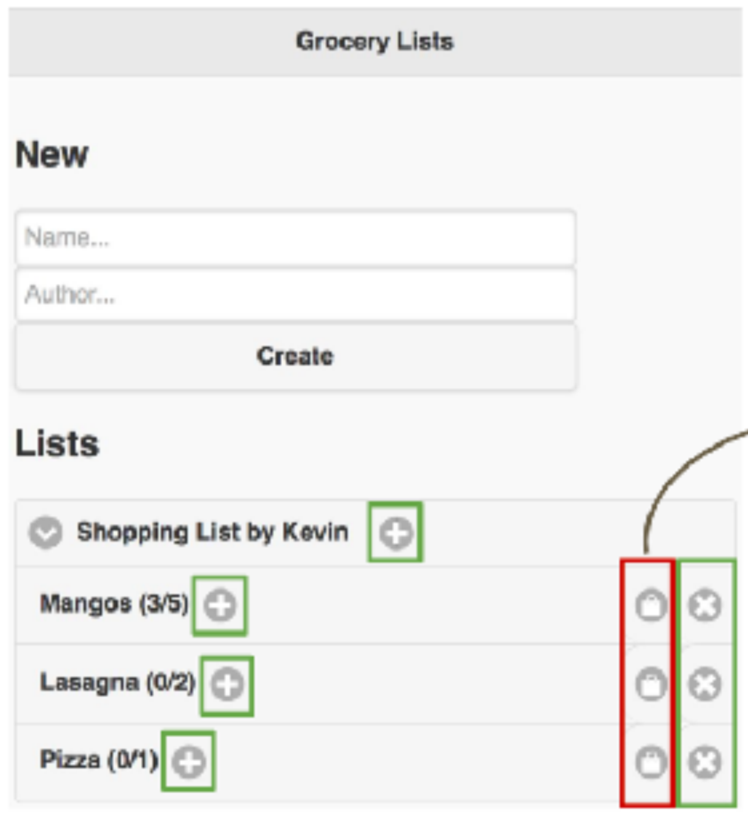
Shonan Meeting 2019 “Programming Languages for Distributed Systems”.
27/05/2019

Research Areas

- Abstractions to share distributed data
  - Script
  - SECROs
- Tools for debugging distributed systems
  - Map/Reduce debugging
  - Multiverse debugging

Fault Tolerant Distributed Systems

- Wirelessly connected
- Run on mobile/fixed hardware
- Often collaborative
- Store data in a decentralized way



Grocery Lists

New

Name...

Author...

Create

Lists

Shopping List by Kevin

Mangos (3/5)

Lasagna (0/2)

Pizza (0/1)

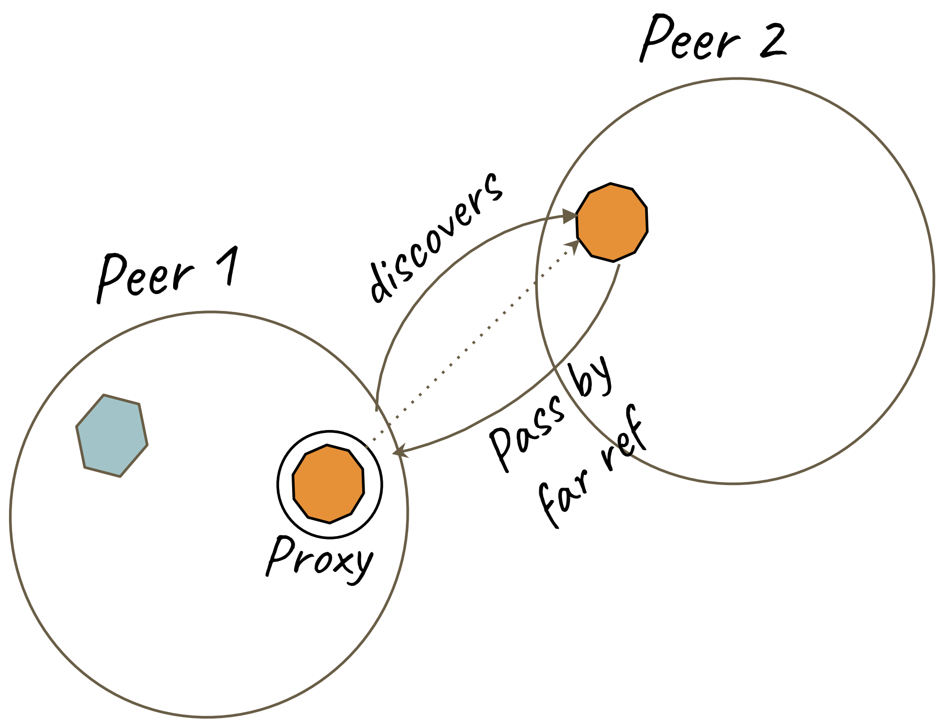
CScript

- Extension to JS for consistent and available replicated objects

```
service GroceryService {
    rep list = new GroceryList();
    rep inventory = new Inventory();

    constructor(name, author)
        this.name = name;
        this.author = author;
    }

    add(item, qty) {
        return this.list.add(item, qty);
    }

    delete (itemName) {
        return this.list.delete(itemName);
    }

    buy(itemName, qty) { /* ... */ }
}
```

CScript's Strongly Consistent Replicated Objects

- Objects reside at the event loop owning them
- Passed by far reference to other event loops:
    - Operations on remote objects via asynchronous message passing.
- Owning actor serializes operations (as processes messages in order).



CScript’s Available Replicated Objects

- Goal:
  - A generic replicated data type with same guarantees as CRDTs:
    - Eventual consistency
    - Strong convergence
  - Without requiring commutative operations

Strongly Eventually Consistent Replicated Object (SECRO)

- General-purpose replicated data type
- No restrictions on operations
- Relies on programmers to specify concurrent behaviour by means of state validators:
  - preconditions
  - postconditions

Kevin De Porre, Florian Myter, Christophe De Troyer, Christophe Scholliers, Wolfgang De Meuter and Elisa Gonzalez Boix. Putting Order in Strong Eventual Consistency To appear in 19th International Conference on Distributed Applications and Interoperable Systems (DAIS '19)

Grocery App: List

```
class GroceryList extends SECRO {
   constructor() {
       super();
       this.items = new Map();
   }

   add(item, qty) {
      const description =
        this.items.getOrElse(item.name, {requested: 0, bought: 0});
      description.requested += qty;
      this.items.set(item.name, description);
   }

   post add(oState, state, args, res) {
       const [item]    = args,
             addedQty = item.requested,
             resQty    = state.items.getOrElse(item.name, 0).requested;
       return resQty >= addedQty;
   }

   delete(itemName) {
       this.items.delete(itemName);
   }
}
```

SECRO’s algorithm

Shopping List

3 x pizza

5 x mango

[ ]

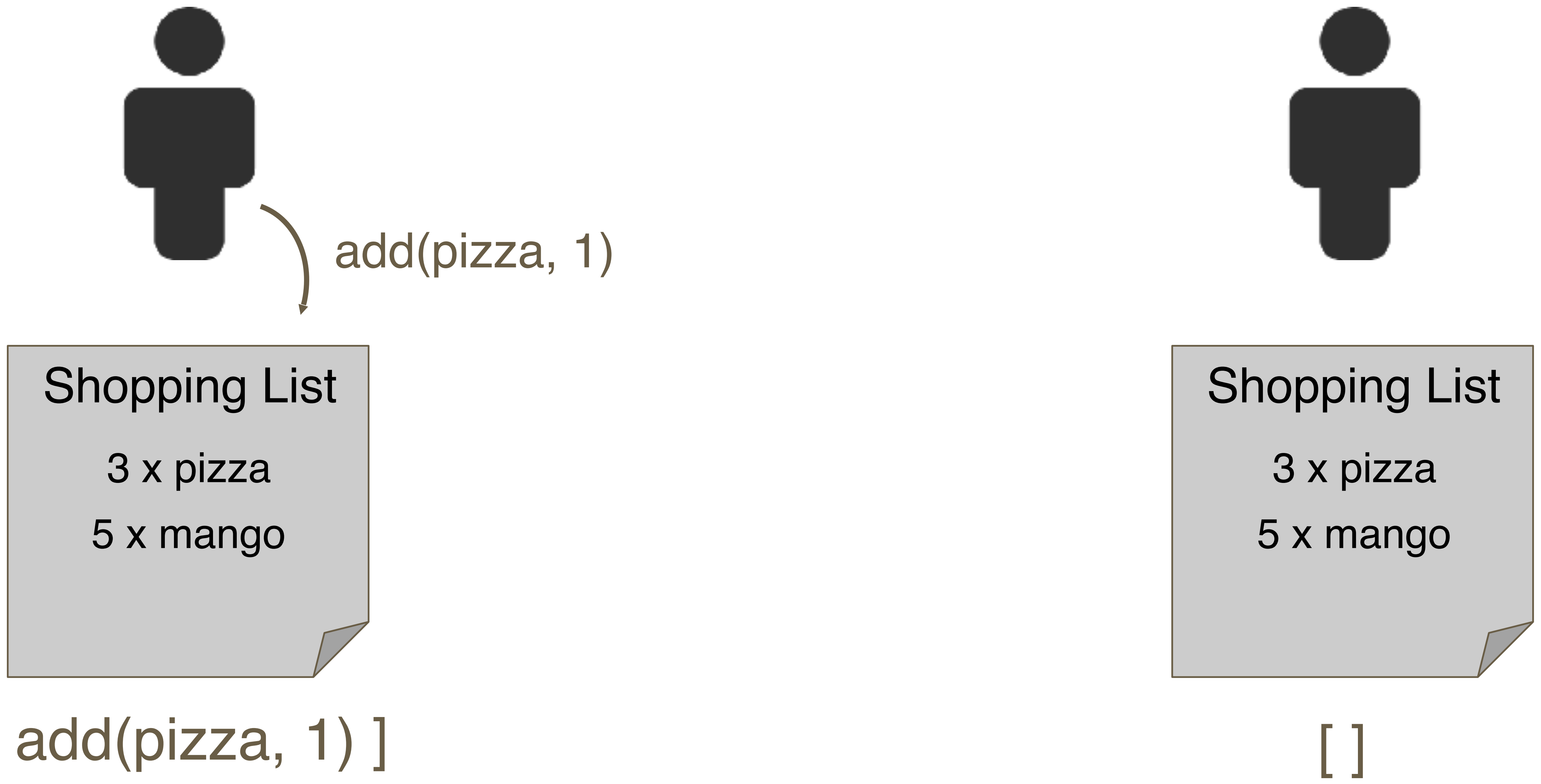
Shopping List

3 x pizza

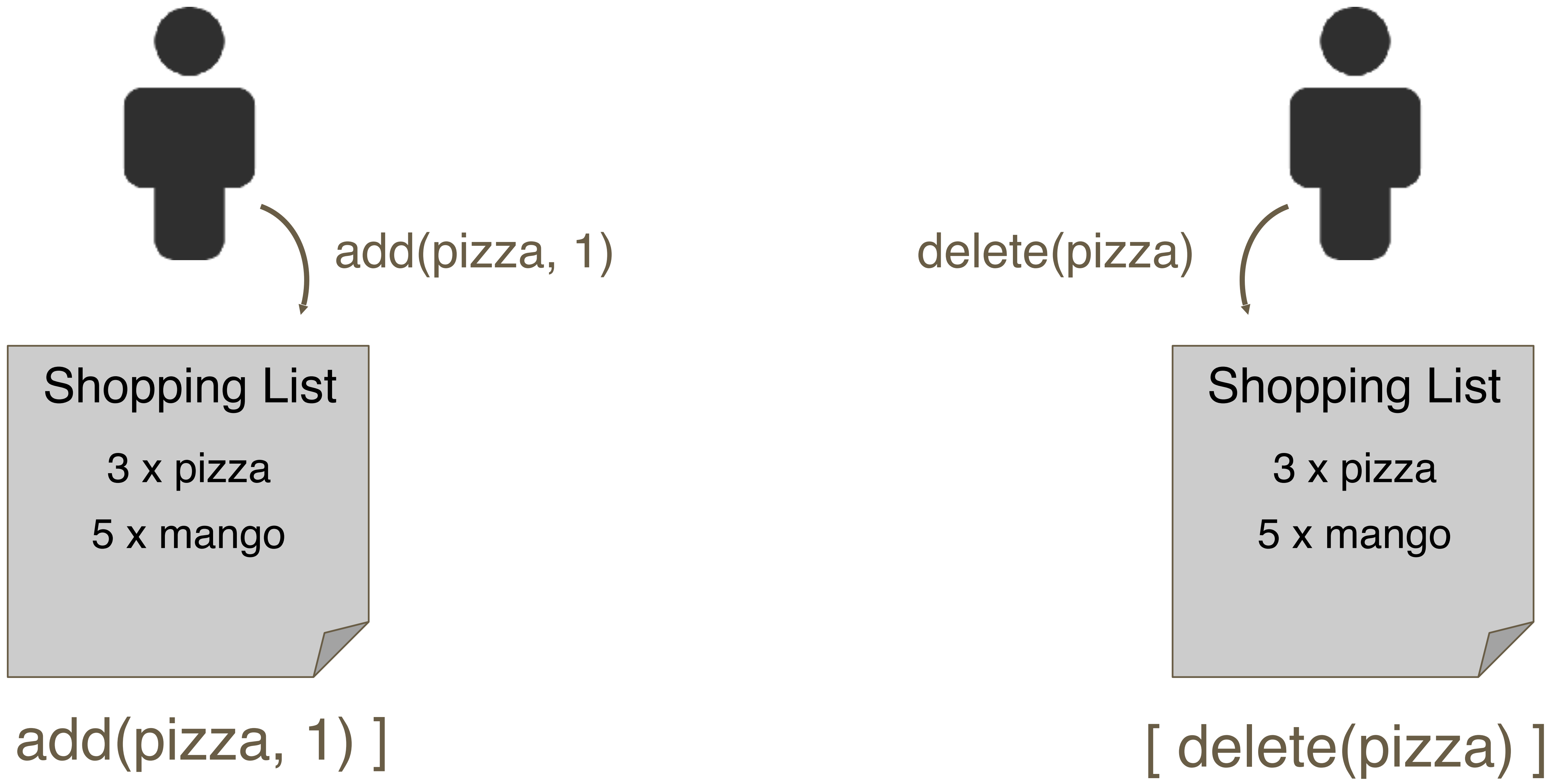
5 x mango

[ ]

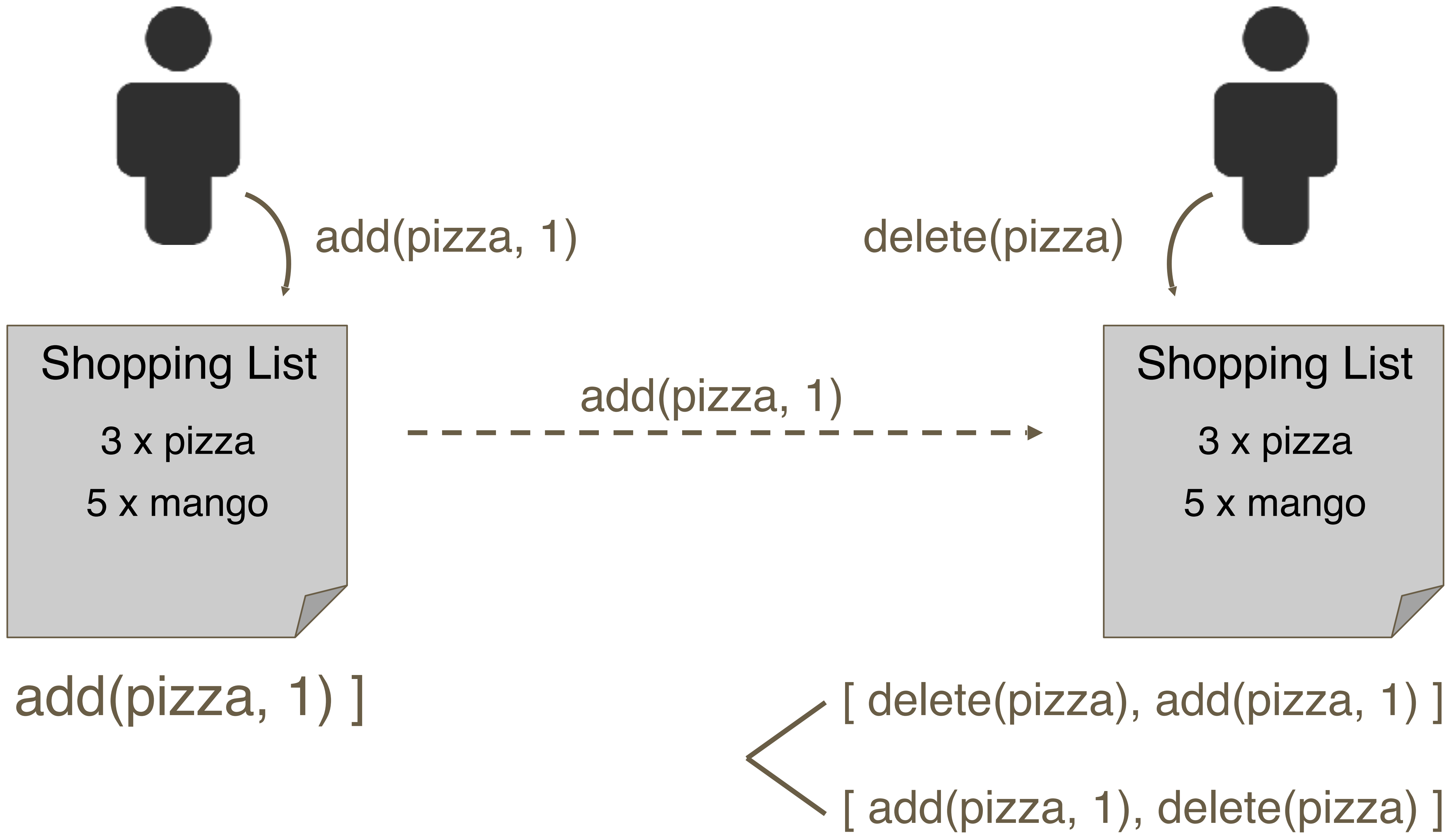
SECRO’s algorithm



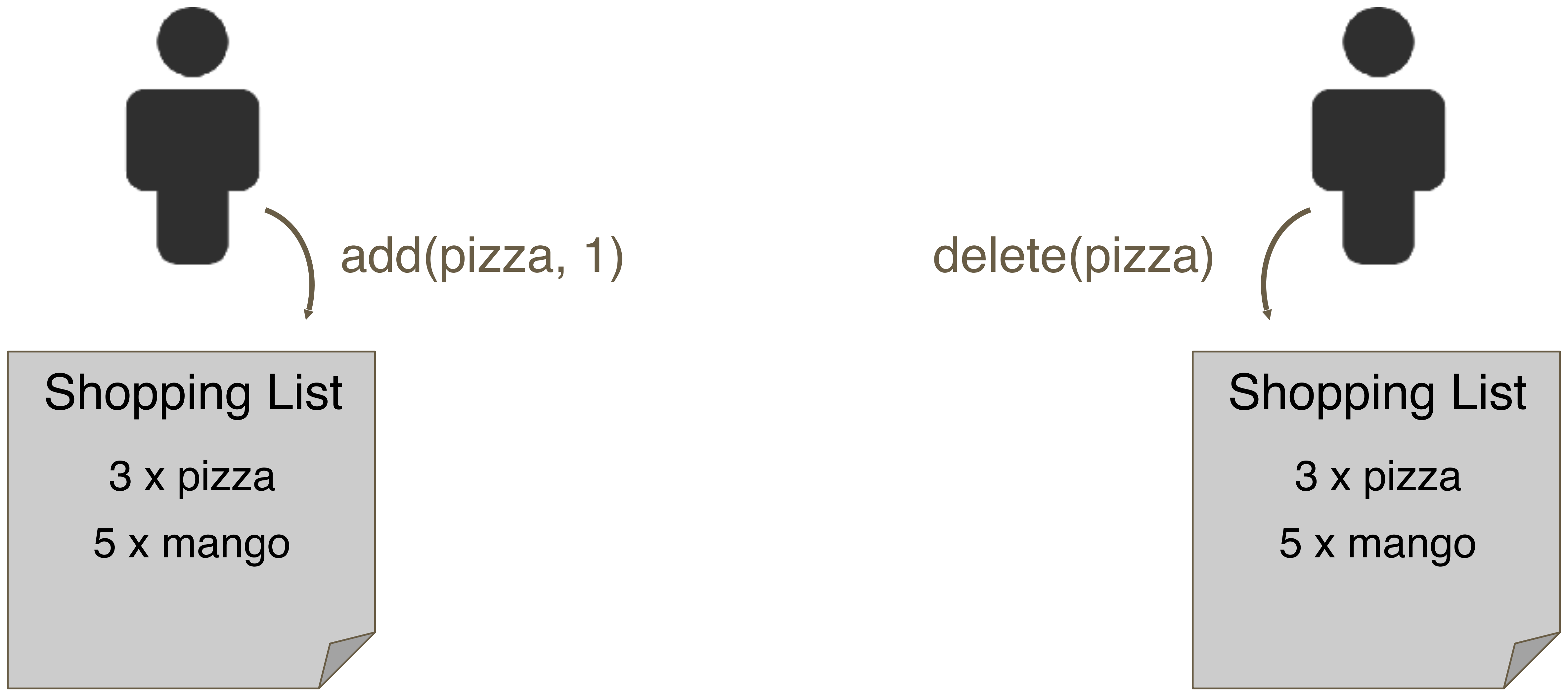
SECRO's algorithm



SECRO's algorithm



SECRO’s algorithm

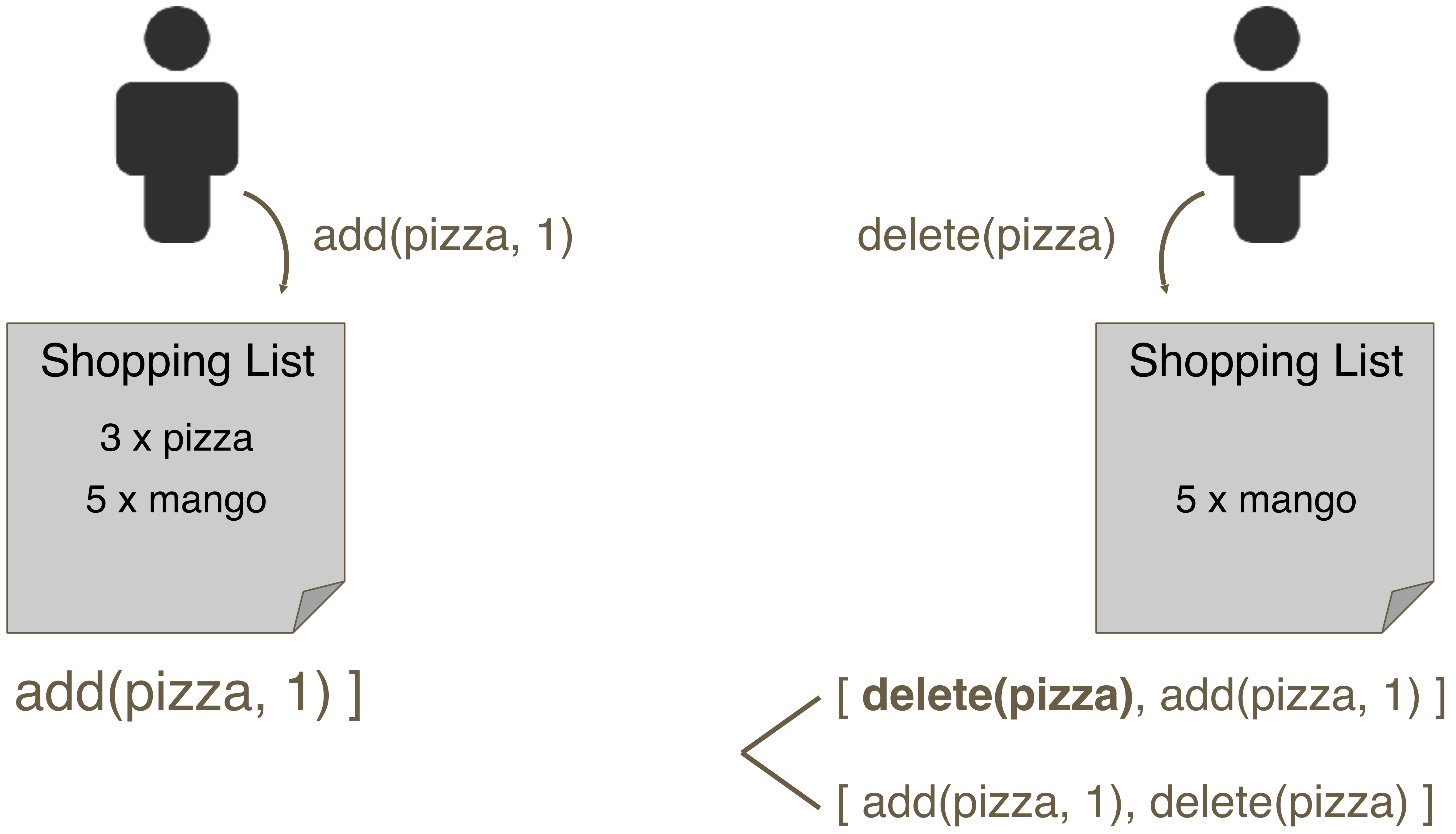


[add(pizza, 1)]

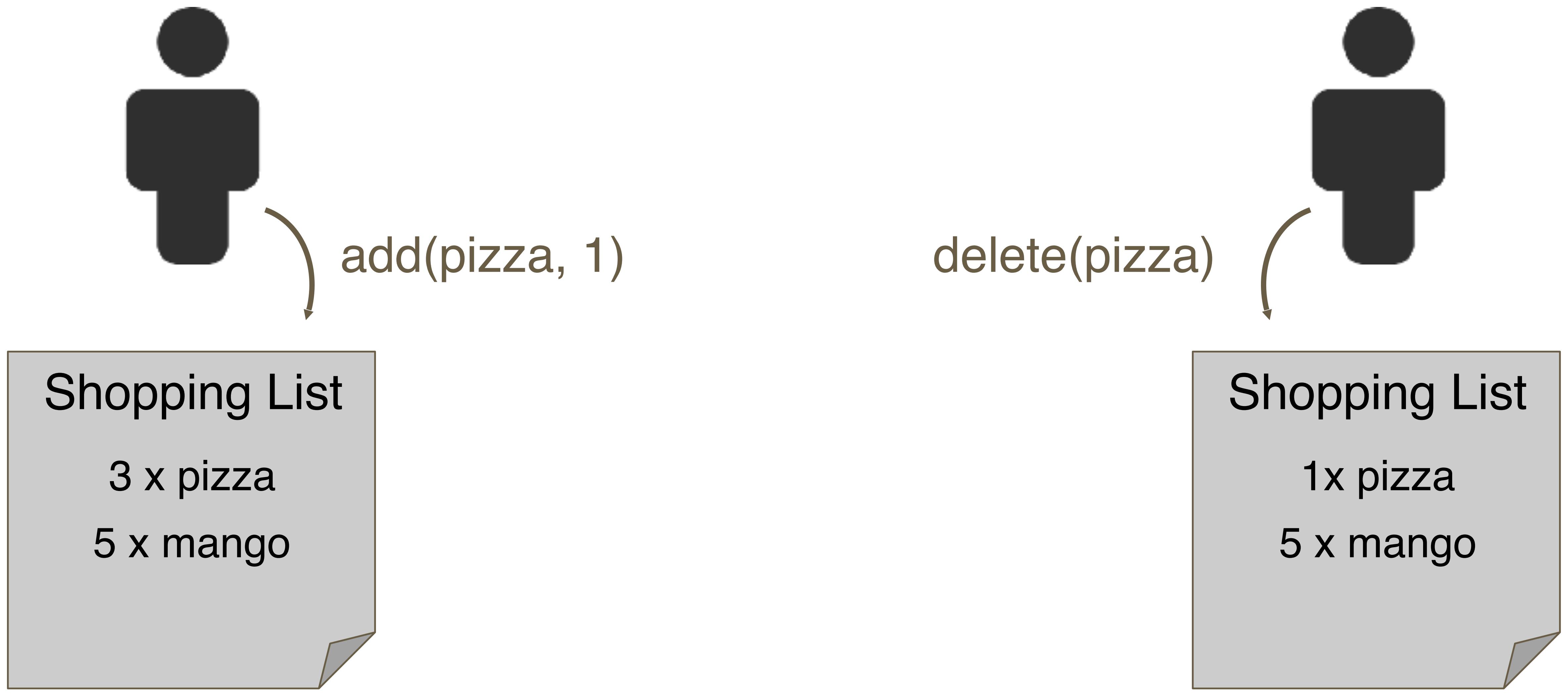
[delete(pizza), add(pizza, 1)]

[add(pizza, 1), delete(pizza)]

SECRO's algorithm



SECRO’s algorithm



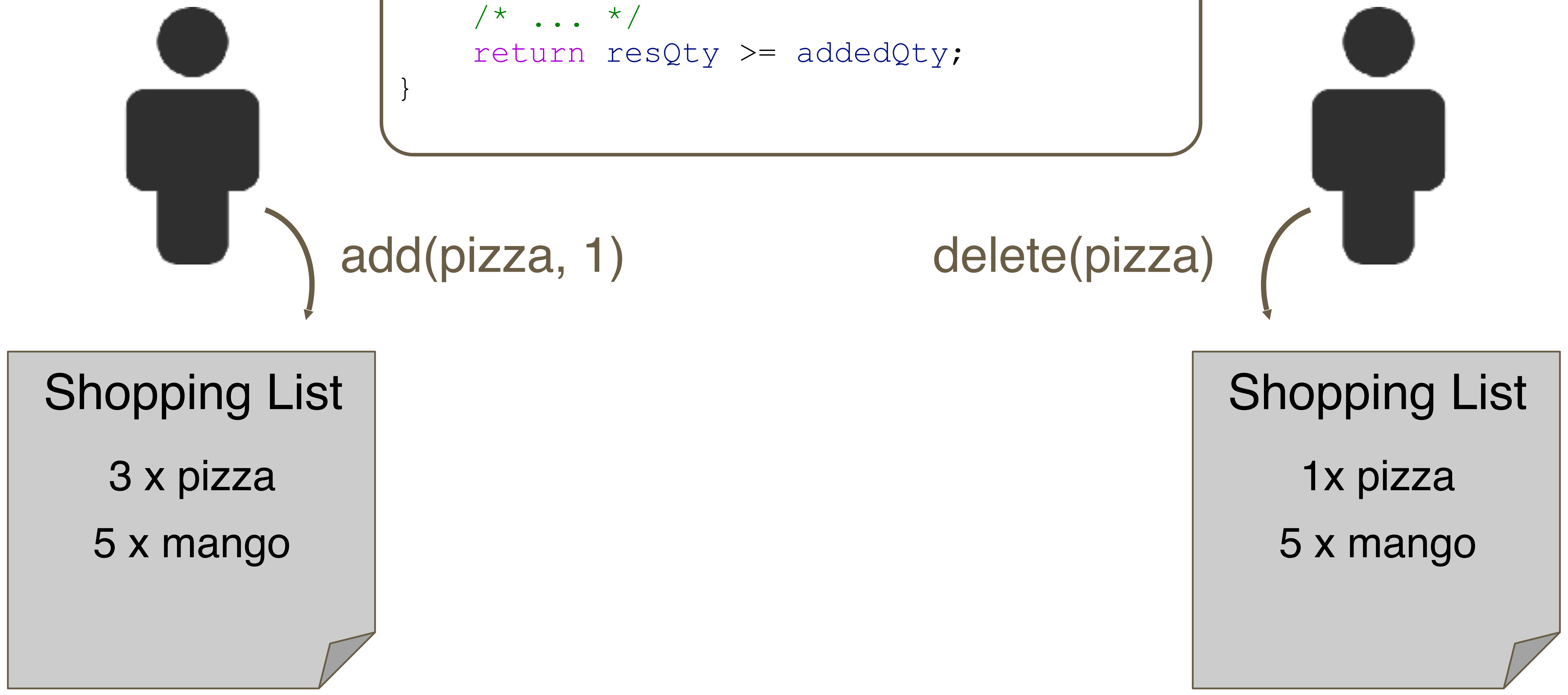
[add(pizza, 1)]

[delete(pizza), **add(pizza, 1)**]

[add(pizza, 1), delete(pizza)]

SECRO's algorithm

```
post add(oState, state, args, res) {
     /* ... */
     return resQty >= addedQty;
}
```



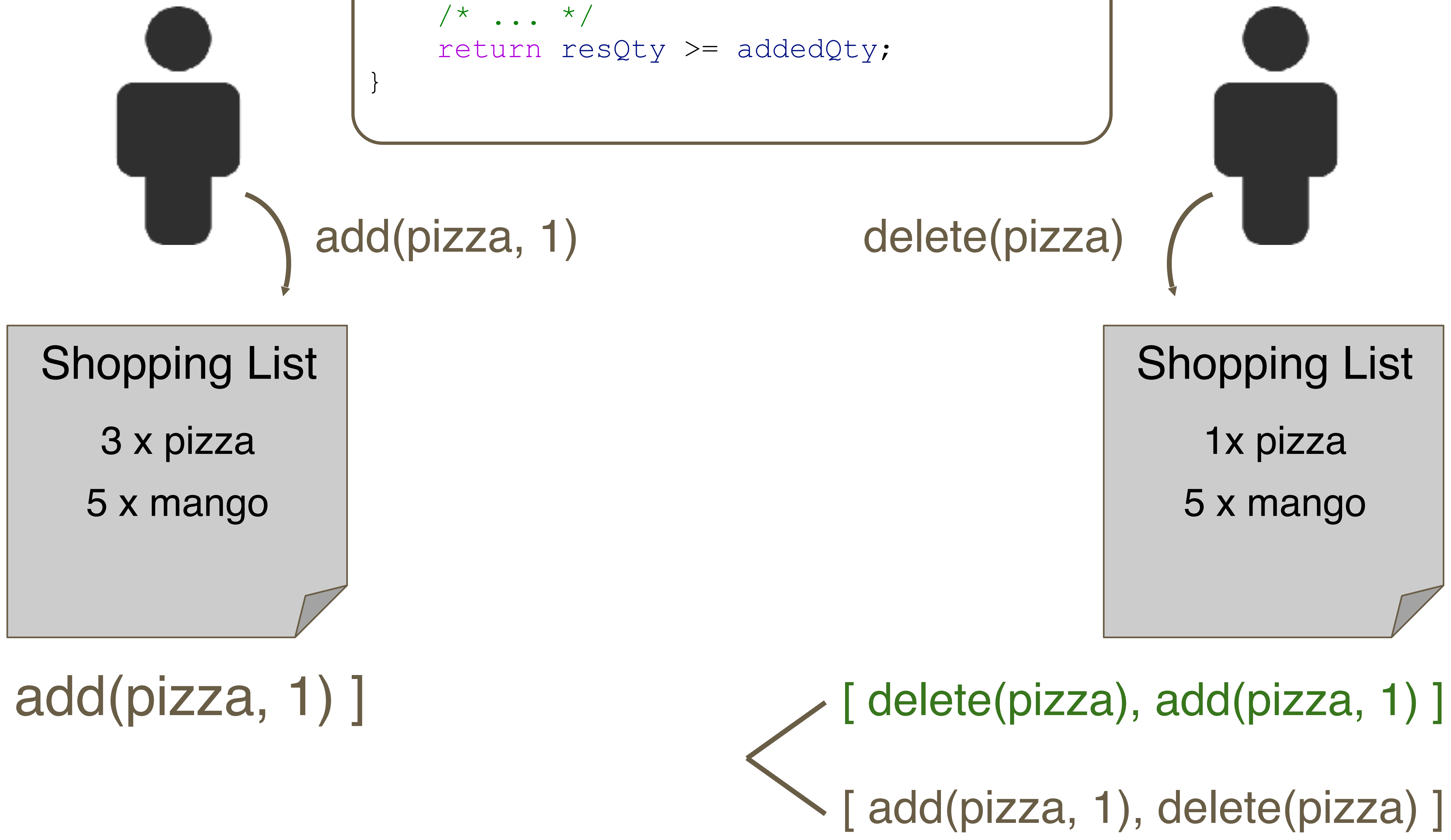
[add(pizza, 1)]

[delete(pizza), **add(pizza, 1)**]

[add(pizza, 1), delete(pizza)]

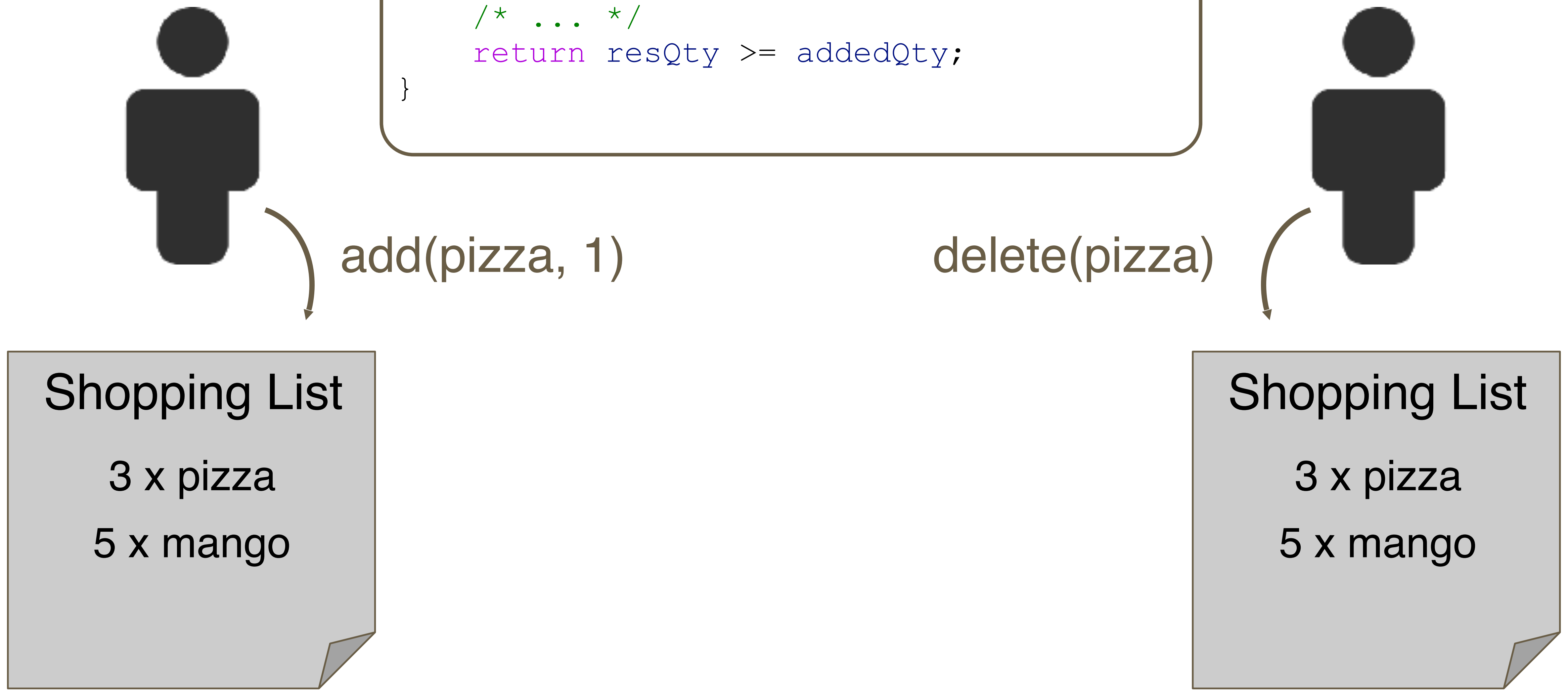
SECRO's algorithm

```
post add(oState, state, args, res) {
    /* ... */
    return resQty >= addedQty;
}
```



SECRO's algorithm

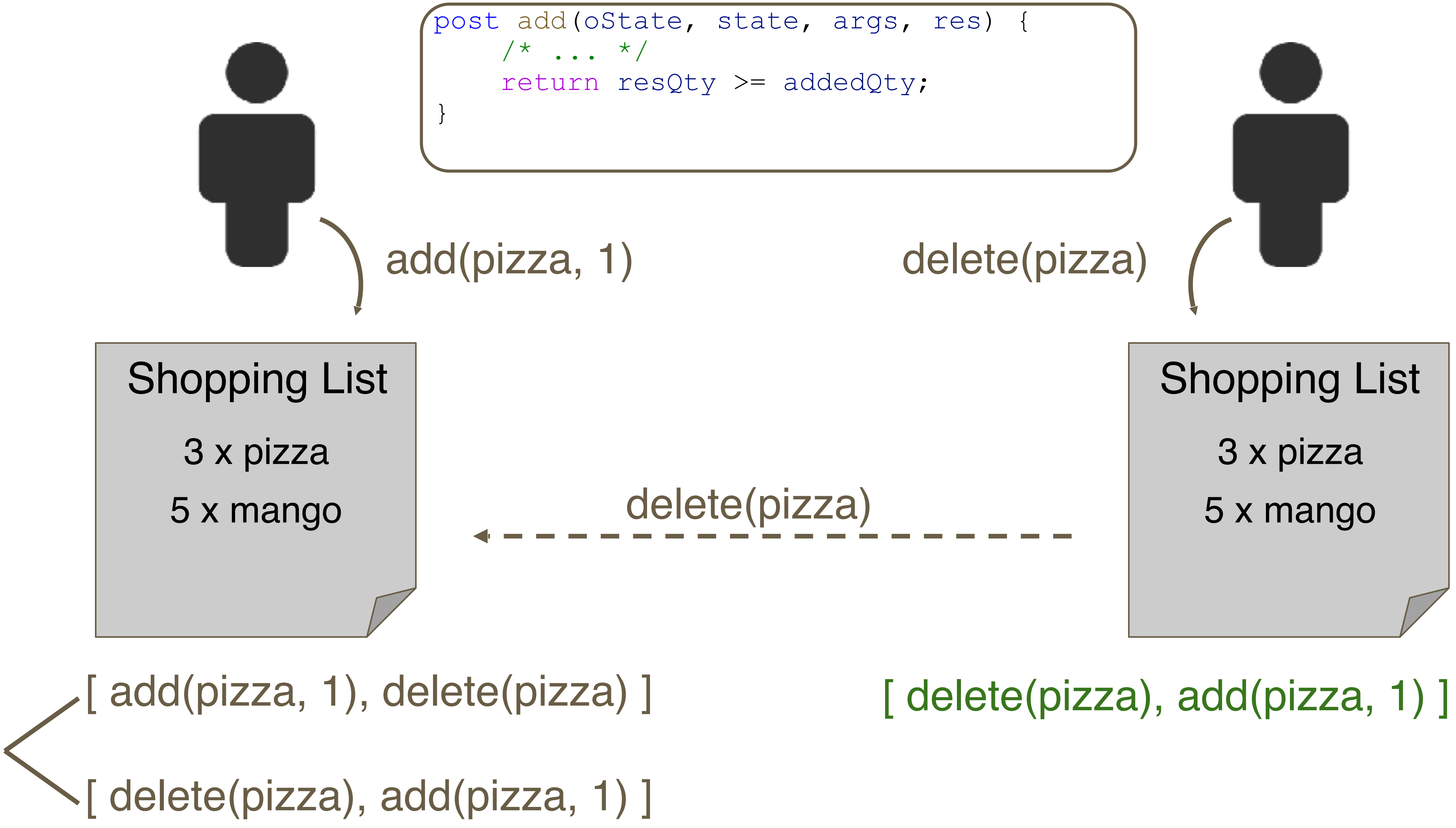
```
post add(oState, state, args, res) {
    /* ... */
    return resQty >= addedQty;
}
```



[add(pizza, 1)]

[delete(pizza), add(pizza, 1)]

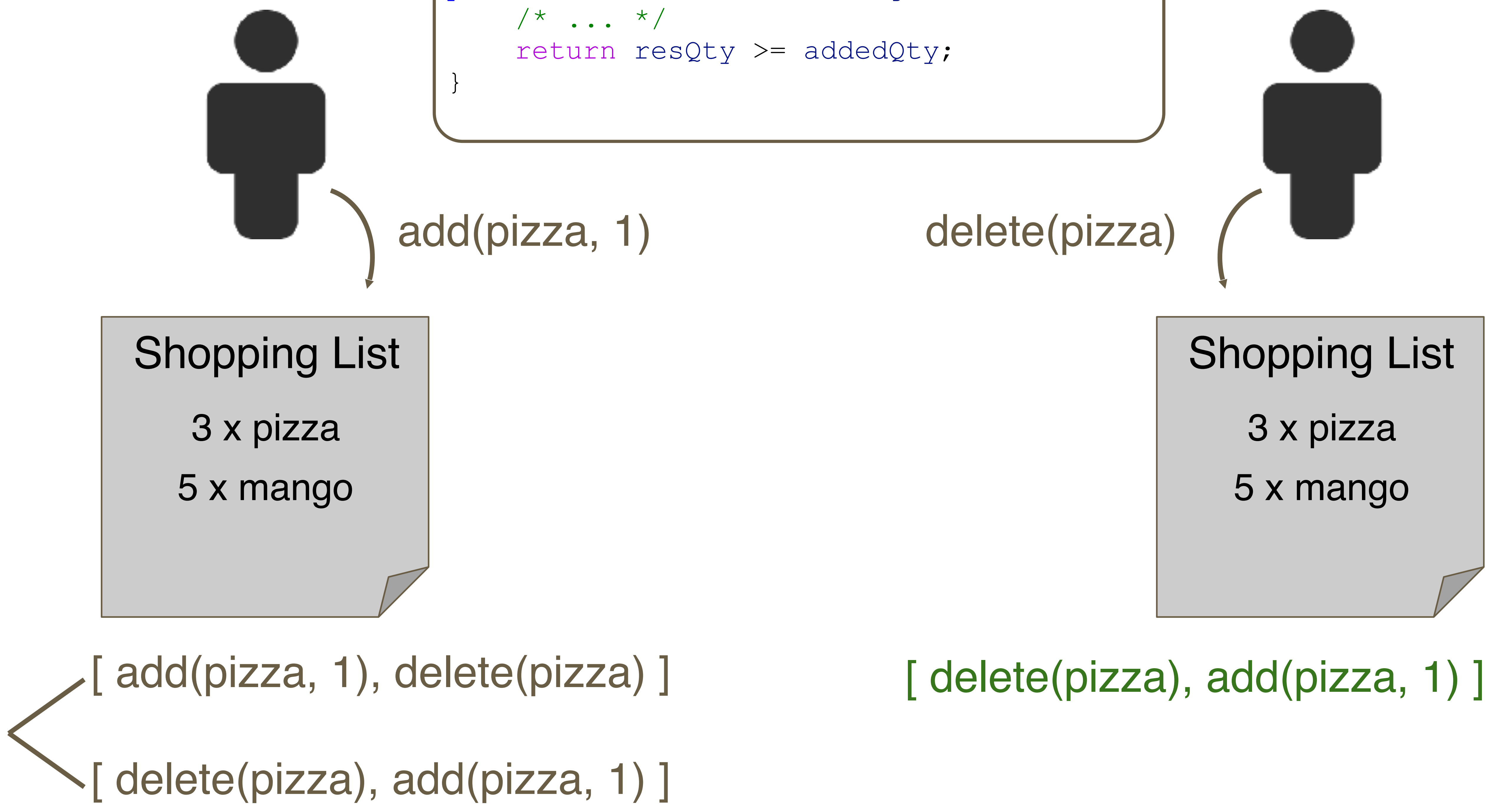
SECRO's algorithm



```
post add(oState, state, args, res) {
    /* ... */
    return resQty >= addedQty;
}
```

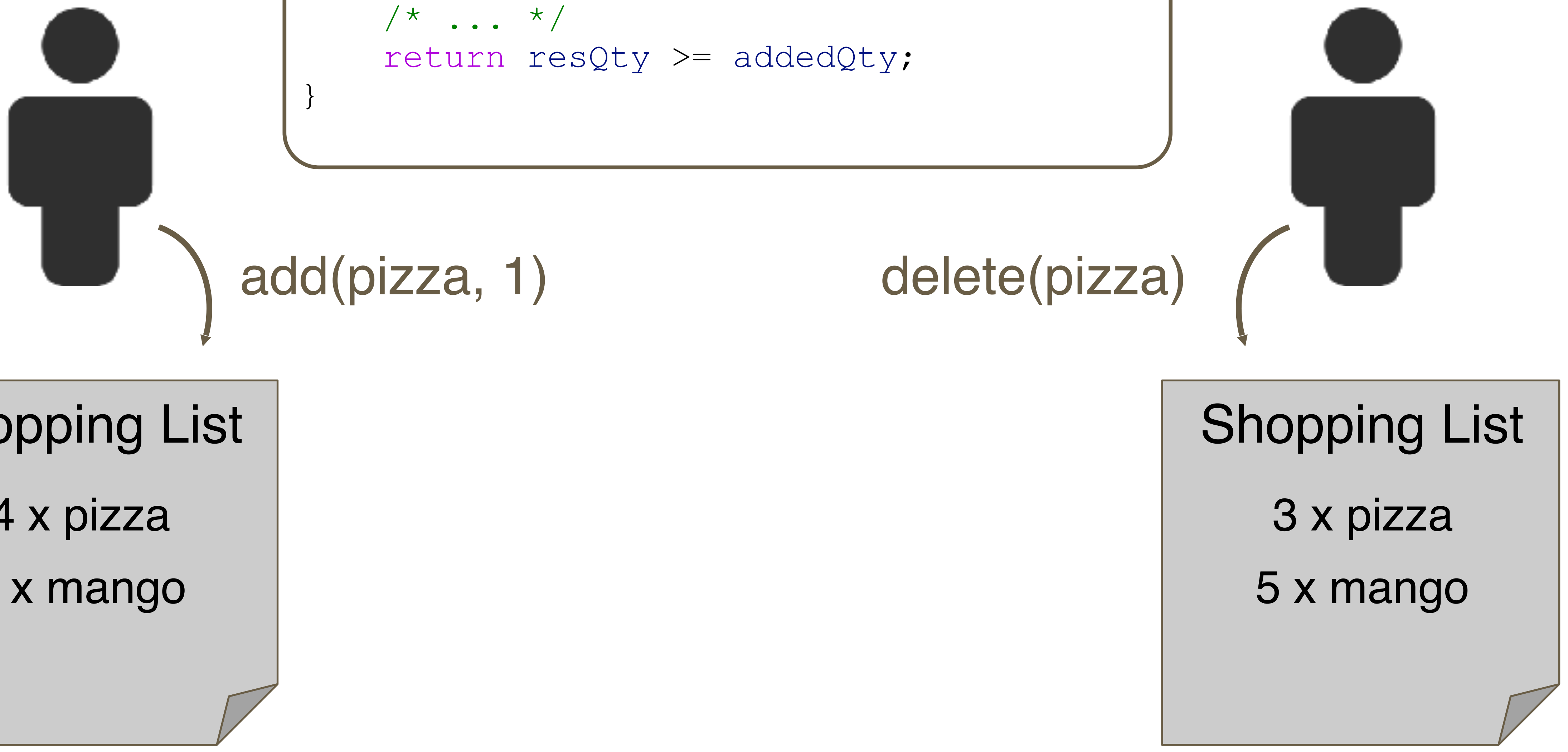
SECRO's algorithm

```
post add(oState, state, args, res) {
    /* ... */
    return resQty >= addedQty;
}
```


add(pizza, 1)
delete(pizza)
Shopping List
3 x pizza
5 x mango
Shopping List
3 x pizza
5 x mango
[add(pizza, 1), delete(pizza)]
[delete(pizza), add(pizza, 1)]
[delete(pizza), add(pizza, 1)]

SECRO’s algorithm

```
post add(oState, state, args, res) {
     /* ... */
     return resQty >= addedQty;
}
```



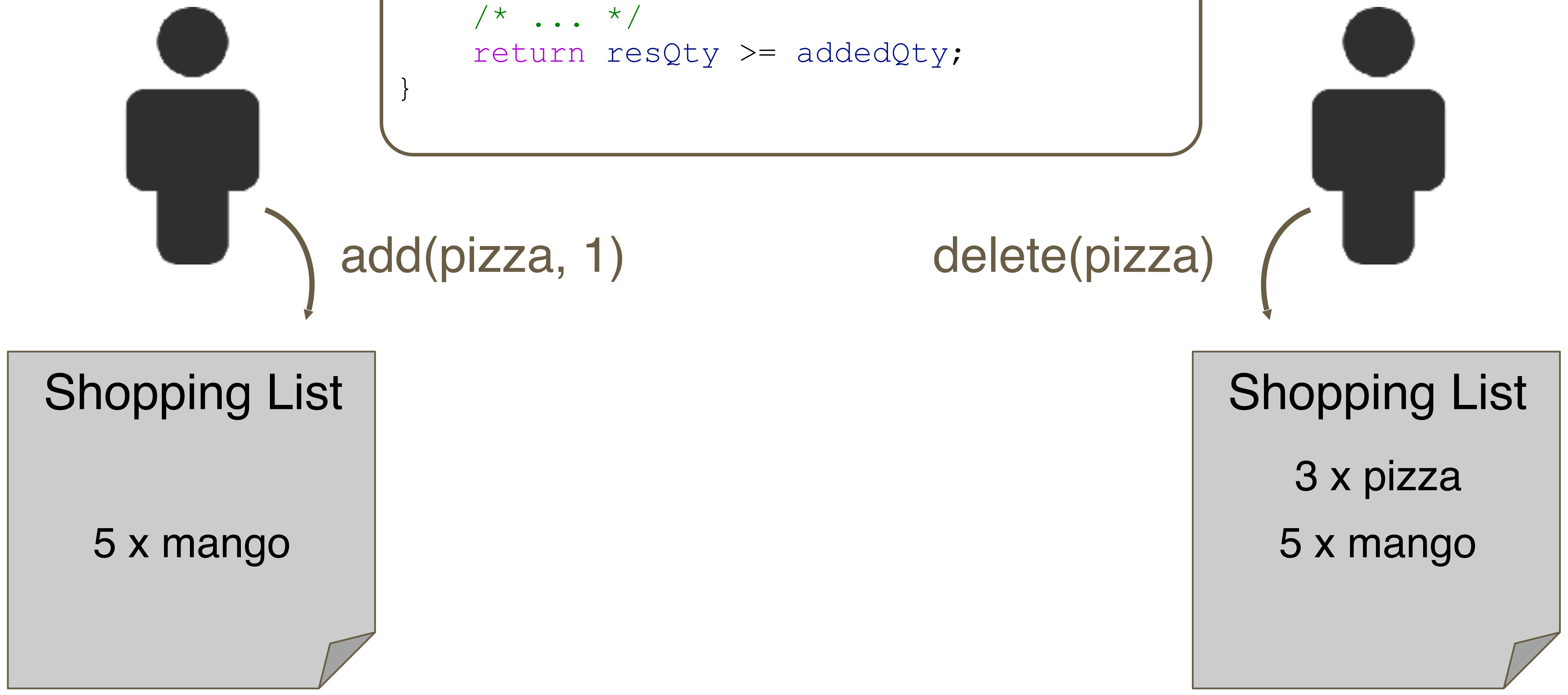
[**add(pizza, 1)**, delete(pizza)]

[delete(pizza), add(pizza, 1)]

[delete(pizza), add(pizza, 1)]

SECRO’s algorithm

```
post add(oState, state, args, res) {
     /* ... */
     return resQty >= addedQty;
}
```



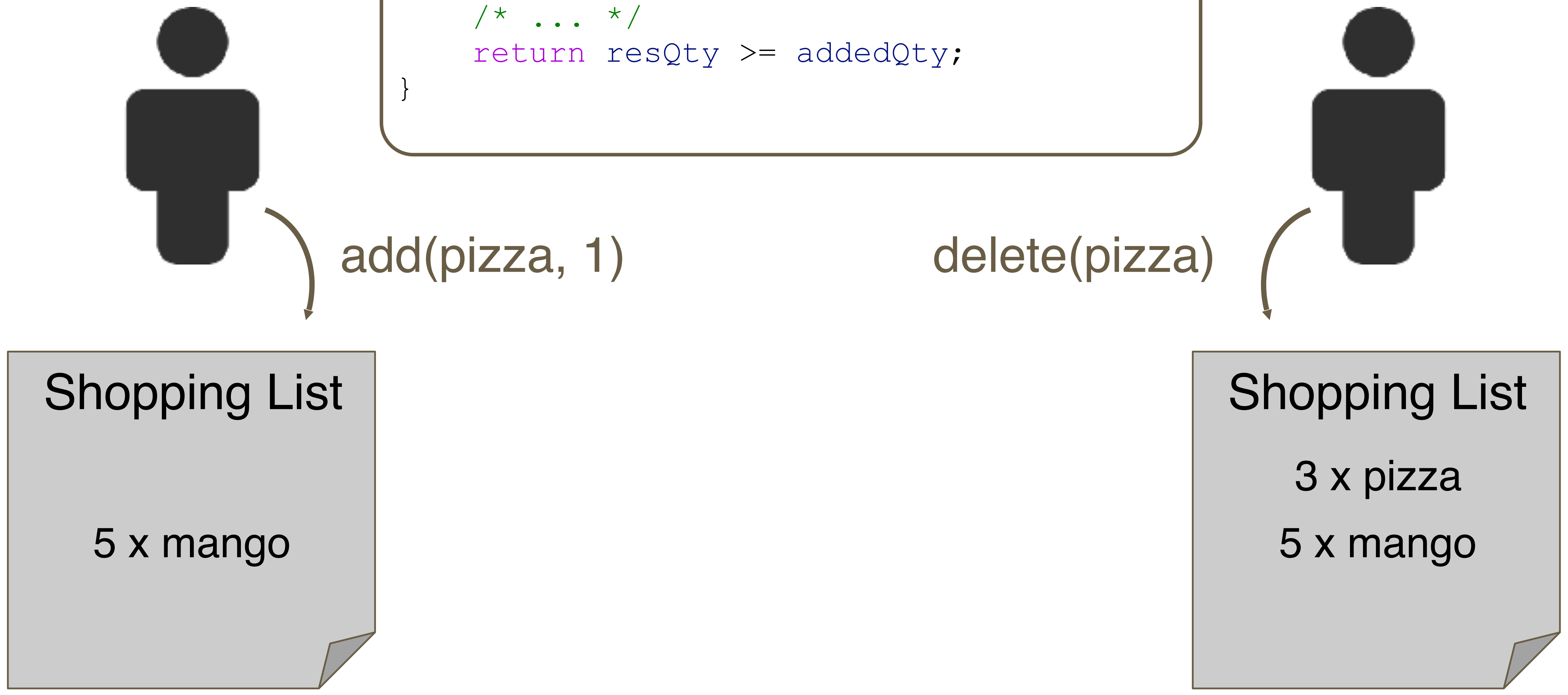
[add(pizza, 1), **delete(pizza)**]

[delete(pizza), add(pizza, 1)]

[delete(pizza), add(pizza, 1)]

SECRO’s algorithm

```
post add(oState, state, args, res) {
    /* ... */
    return resQty >= addedQty;
}
```



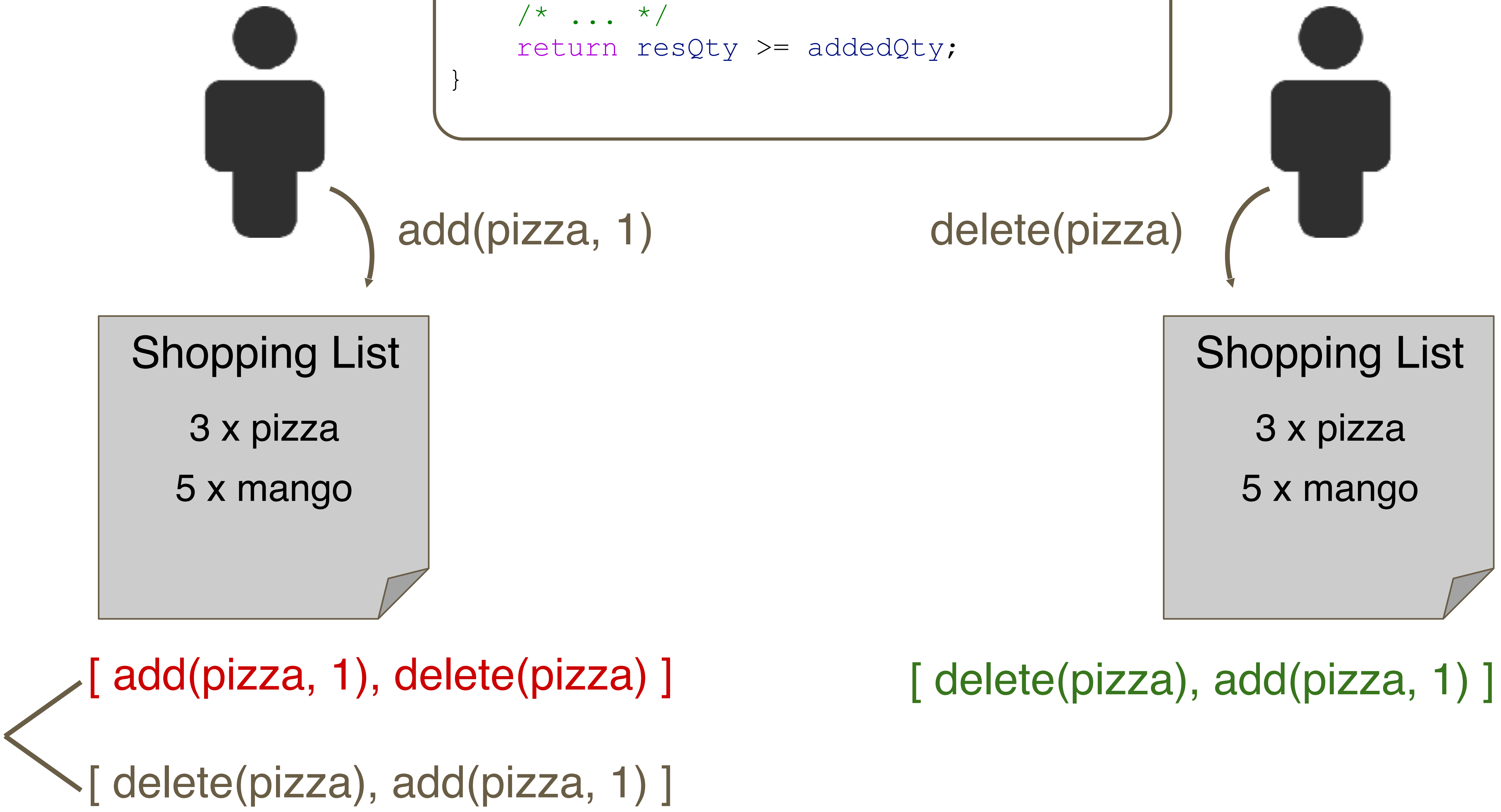
[add(pizza, 1), delete(pizza)]

[delete(pizza), add(pizza, 1)]

[delete(pizza), add(pizza, 1)]

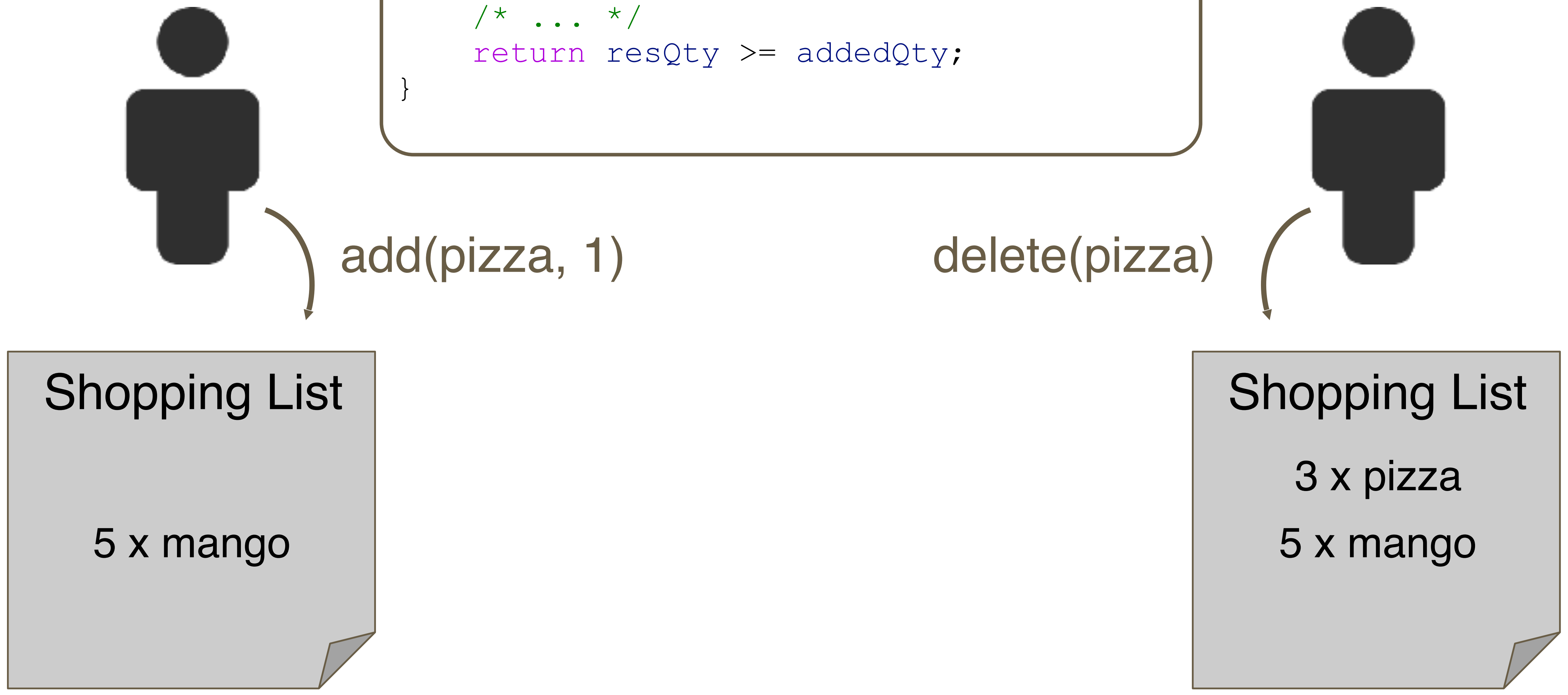
SECRO’s algorithm

```
post add(oState, state, args, res) {
    /* ... */
    return resQty >= addedQty;
}
```


add(pizza, 1)
delete(pizza)
Shopping List
3 x pizza
5 x mango
Shopping List
3 x pizza
5 x mango
[add(pizza, 1), delete(pizza)]
[delete(pizza), add(pizza, 1)]
[delete(pizza), add(pizza, 1)]

SECRO’s algorithm

```
post add(oState, state, args, res) {
     /* ... */
     return resQty >= addedQty;
}
```



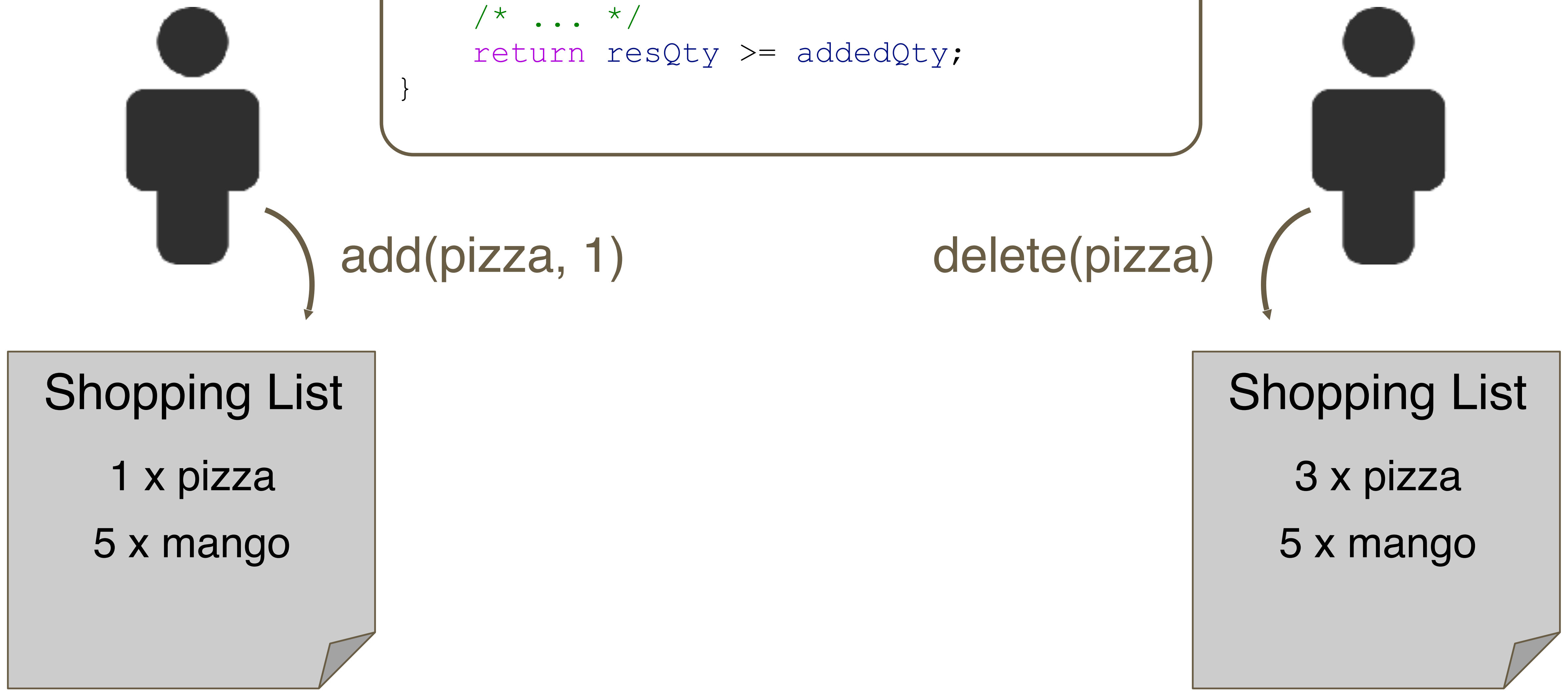
[add(pizza, 1), delete(pizza)]

[**delete(pizza)**, add(pizza, 1)]

[delete(pizza), add(pizza, 1)]

SECRO's algorithm

```
post add(oState, state, args, res) {
    /* ... */
    return resQty >= addedQty;
}
```



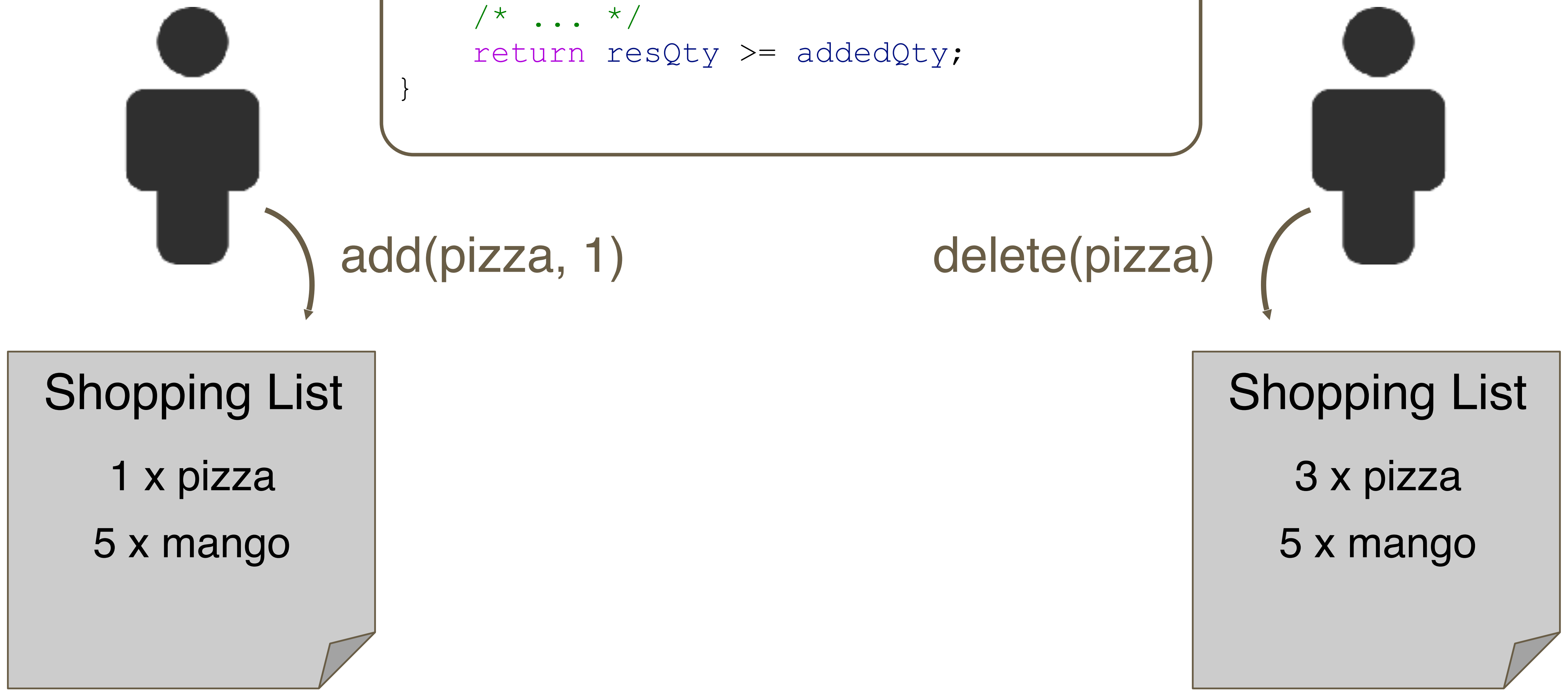
[add(pizza, 1), delete(pizza)]

[delete(pizza), **add(pizza, 1)**]

[delete(pizza), add(pizza, 1)]

SECRO's algorithm

```
post add(oState, state, args, res) {
    /* ... */
    return resQty >= addedQty;
}
```



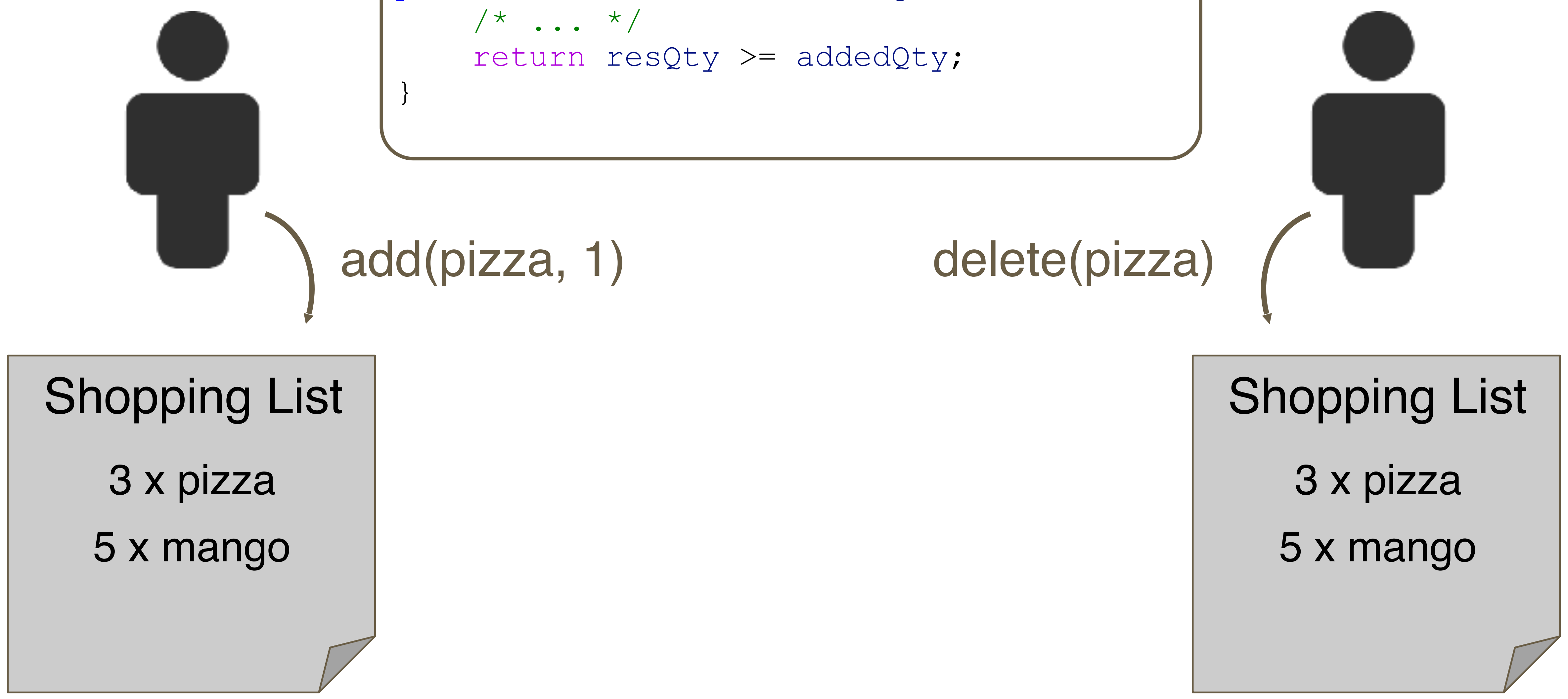
[add(pizza, 1), delete(pizza)]

[delete(pizza), add(pizza, 1)]

[delete(pizza), add(pizza, 1)]

SECRO's algorithm

```
post add(oState, state, args, res) {
    /* ... */
    return resQty >= addedQty;
}
```



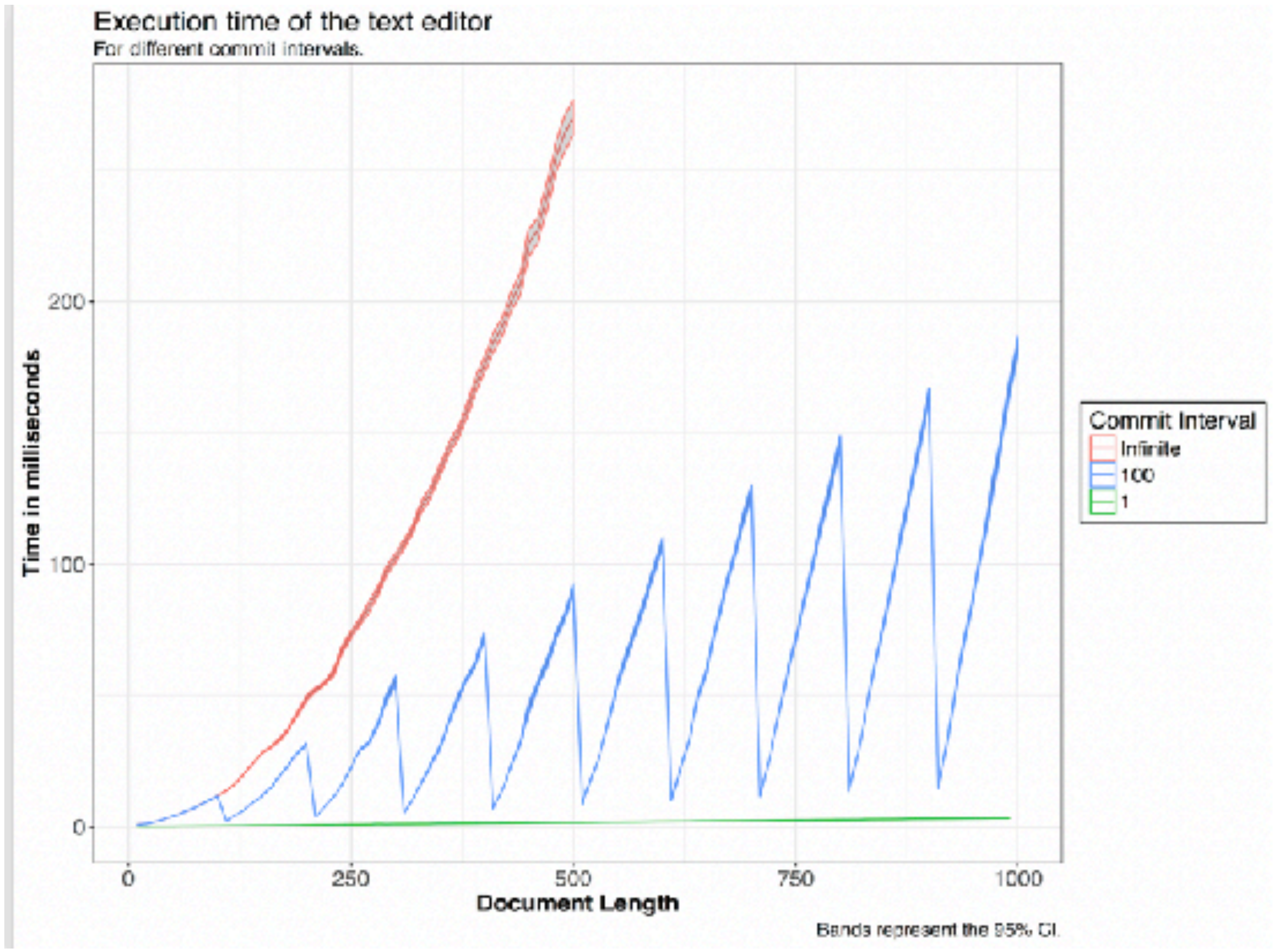
[add(pizza, 1), delete(pizza)]

[delete(pizza), add(pizza, 1)]

[delete(pizza), add(pizza, 1)]

SECROs in practice

- Collaborative Text Editor, compared to JSCN CRDTs
- Memory efficient but slower
  - Commit operation

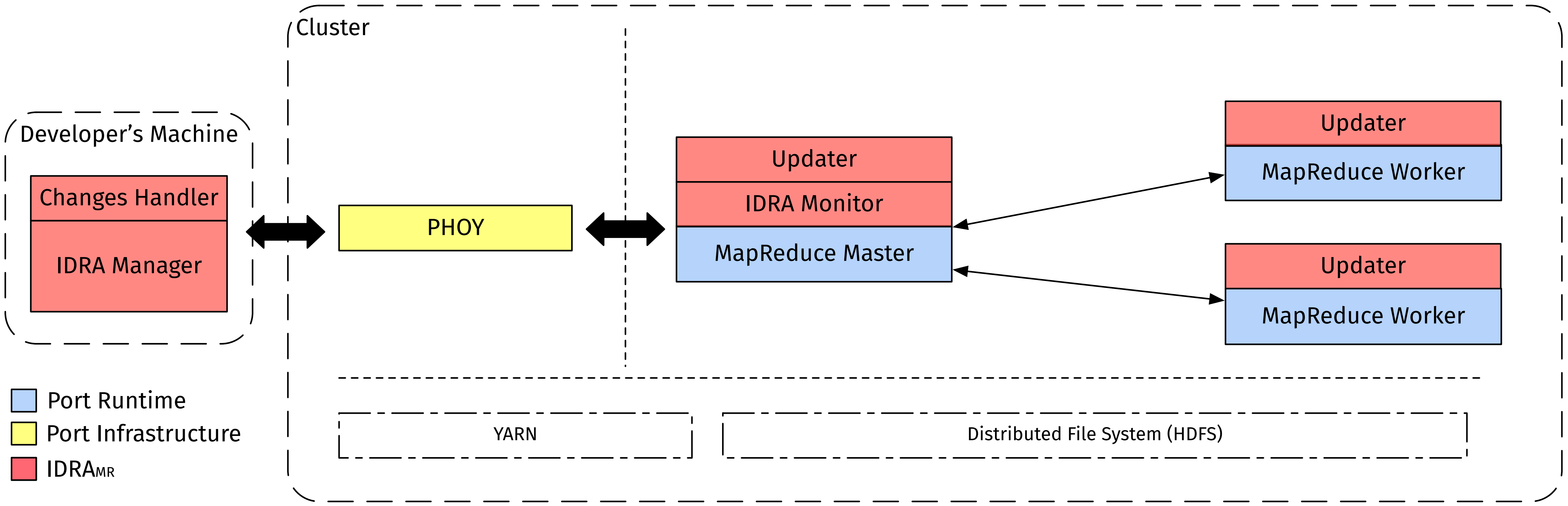


Future Directions

- Language support for application-level invariants
- Discover conflict patterns using static analysis
- ∀ pattern: determine a correct ordering
  - Based on invariants
  - Commit no longer needed

Distributed Debugging

- A Debuggeable Live Map/Reduce Framework

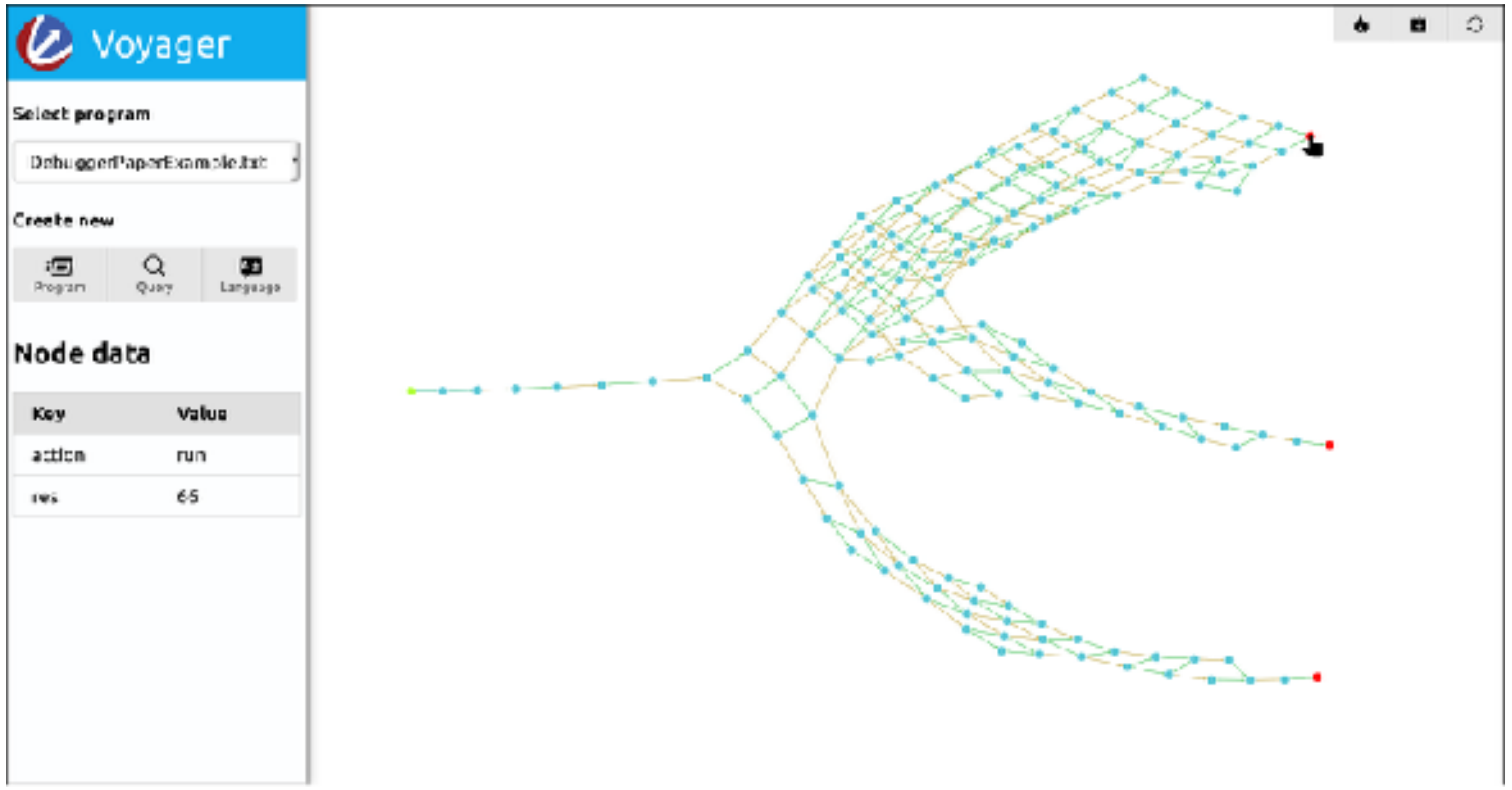


M. Marra, G. Polito and E. Gonzalez Boix Out-of-place Debugging: A debugging architecture to reduce debugging interference. In The Art, Science and Engineering of Programming, 3(2), 2018.

Distributed Debugging

- Multiverse debugging:

  1. Observe *all* possible paths of the program execution
  2. Interactively explore execution paths using breakpoints and stepping operations



Node data

Key	Value
action	run
res	65

Demo of the Voyager debugger on Thursday

Carmen Torres Lopez, Robbert Gurdeep Sigh, Stefan Marr, Elisa Gonzalez Boix, Christophe Scholliers. Multiverse Debugging: Non-deterministic Debugging for Non-deterministic Programs. To Appear in ECOOP 2019