Rich Specifications for Ethereum Smart Contract Verification

Christian Bräm
Marco Eilers
Peter Müller
Robin Sierra
Alexander J. Summers

ETH zürich

THE UNIVERSITY OF BRITISH COLUMBIA

Smart contract verification vs. OOP

```
balances: map(address, int)
minter: address

def transfer(to: address, amount: int):
  from = msg.sender
  assert self.balances[from] >= amount
  self.balances[to] += amount
  self.balances[from] -= amount
  to.notify(from, self, amount)

def mint(to: address, amount: int):
  assert msg.sender == self.minter
  self.balances[to] += amount
```

Smart contract verification vs. OOP

```
balances: map(address, int)
minter: address

def transfer(to: address, amount: int):
  from = msg.sender
  assert self.balances[from] >= amount
  self.balances[to] += amount
  self.balances[from] -= amount
  to.notify(from, self, amount)

def mint(to: address, amount: int):
  assert msg.sender == self.minter
  self.balances[to] += amount
```

- Calls to unverified, untrusted code

Smart contract verification vs. OOP

```
balances: map(address, int)
minter: address

def transfer(to: address, amount: int):
  from = msg.sender
  assert self.balances[from] >= amount
  self.balances[to] += amount
  self.balances[from] -= amount
  to.notify(from, self, amount)

def mint(to: address, amount: int):
  assert msg.sender == self.minter
  self.balances[to] += amount
```

- Calls to unverified, untrusted code
  - Rules out techniques like separation logic

Smart contract verification vs. OOP

```
balances: map(address, int)
minter: address

def transfer(to: address, amount: int):
  from = msg.sender
  assert self.balances[from] >= amount
  self.balances[to] += amount
  self.balances[from] -= amount
  to.notify(from, self, amount)

def mint(to: address, amount: int):
  assert msg.sender == self.minter
  self.balances[to] += amount
```

- Calls to unverified, untrusted code
  - Rules out techniques like separation logic
- Execution model
  - Encapsulation

Smart contract verification vs. OOP

```
balances: map(address, int)
minter: address

def transfer(to: address, amount: int):
  from = msg.sender
  assert self.balances[from] >= amount
  self.balances[to] += amount
  self.balances[from] -= amount
  to.notify(from, self, amount)

def mint(to: address, amount: int):
  assert msg.sender == self.minter
  self.balances[to] += amount
```

- Calls to unverified, untrusted code
  - Rules out techniques like separation logic
- Execution model
  - Encapsulation
  - Caller-dependent behavior

Smart contract verification vs. OOP

```
balances: map(address, int)
minter: address

def transfer(to: address, amount: int):
  from = msg.sender
  assert self.balances[from] >= amount
  self.balances[to] += amount
  self.balances[from] -= amount
  to.notify(from, self, amount)

def mint(to: address, amount: int):
  assert msg.sender == self.minter
  self.balances[to] += amount
```

- Calls to unverified, untrusted code
  - Rules out techniques like separation logic
- Execution model
  - Encapsulation
  - Caller-dependent behavior
- Typical application domain
  - Resources and resource transfers

Goal

Go *beyond* pre- and postconditions, invariants.

Goal

Go *beyond* pre- and postconditititions, invariants.

Specification constructs for

- sound *and precise* reasoning about re-entrancy
  - in hostile environment

Goal

Go *beyond* pre- and postconditititions, invariants.

Specification constructs for

- sound *and precise* reasoning about re-entrancy
  - in hostile environment
- access control
  - enables reasoning about invariants between different contracts

Goal

Go *beyond* pre- and postconditions, invariants.

Specification constructs for

- sound *and precise* reasoning about re-entrancy
  - in hostile environment
- access control
  - enables reasoning about invariants between different contracts
- resource transfers
  - simple specification of common cases

This talk: OOPSLA 2021

Reasoning about (re-entrant) calls

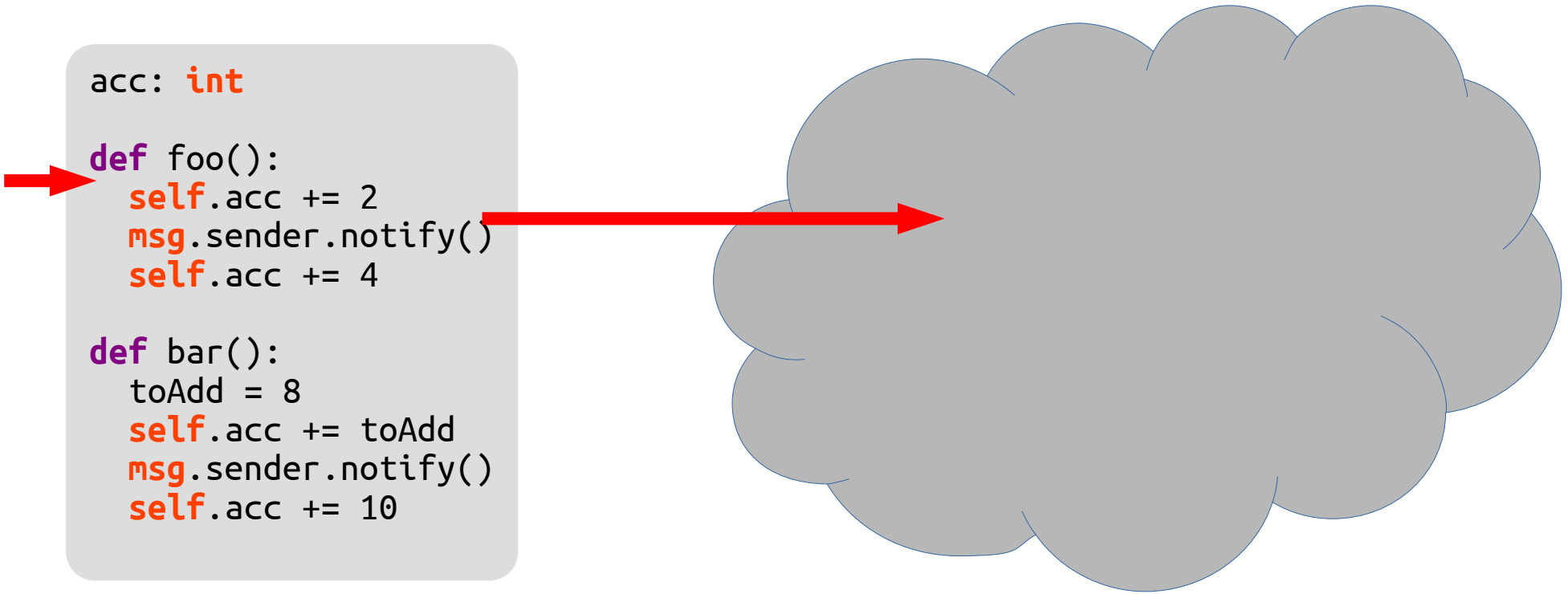
```
acc: int

def foo():
  self.acc += 2
  msg.sender.notify()
  self.acc += 4

def bar():
  toAdd = 8
  self.acc += toAdd
  msg.sender.notify()
  self.acc += 10
```

Reasoning about (re-entrant) calls

```
acc: int

def foo():
  self.acc += 2
  msg.sender.notify()
  self.acc += 4

def bar():
  toAdd = 8
  self.acc += toAdd
  msg.sender.notify()
  self.acc += 10
```

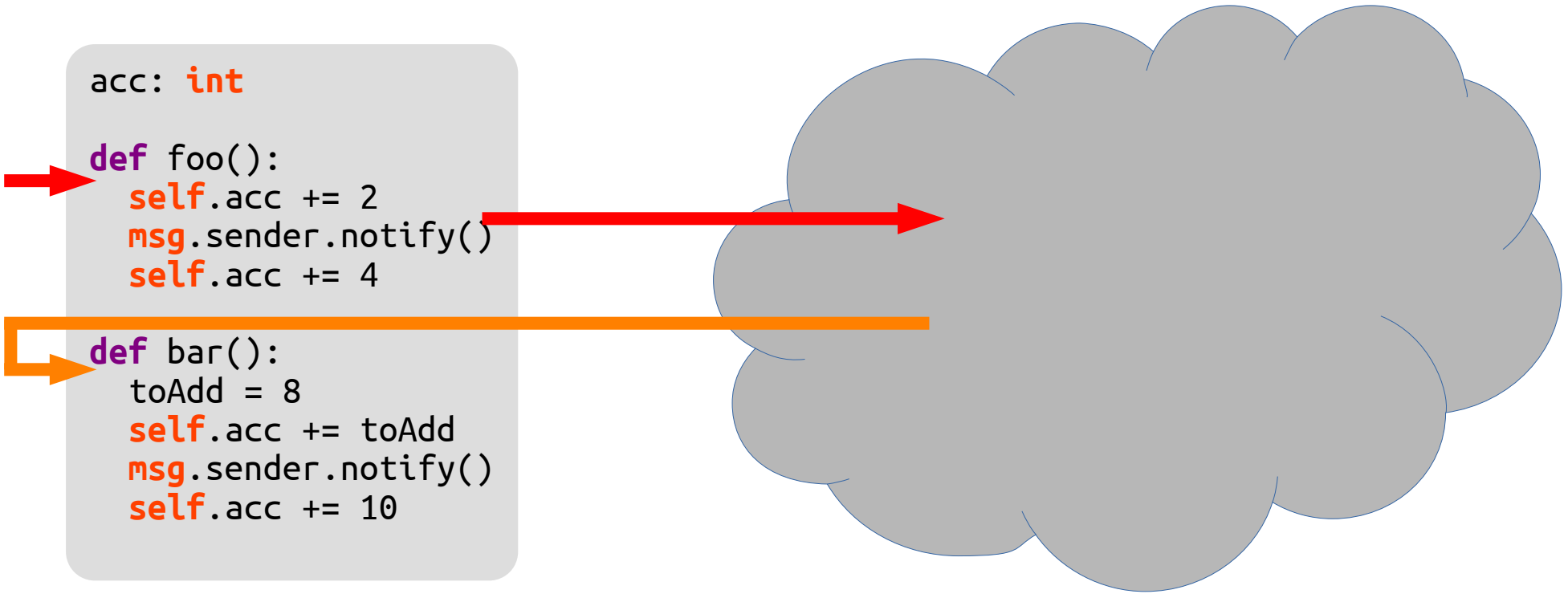
Reasoning about (re-entrant) calls

```
acc: int

def foo():
  self.acc += 2
  msg.sender.notify()
  self.acc += 4

def bar():
  toAdd = 8
  self.acc += toAdd
  msg.sender.notify()
  self.acc += 10
```

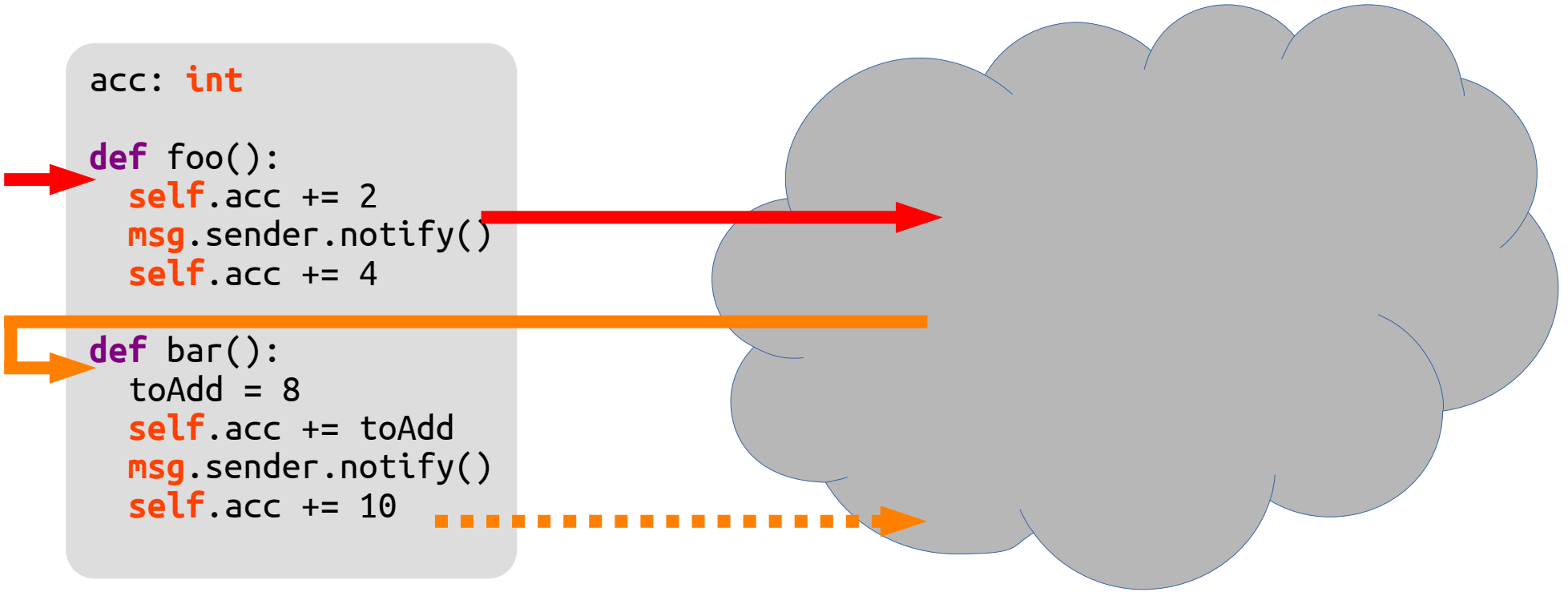


Reasoning about (re-entrant) calls

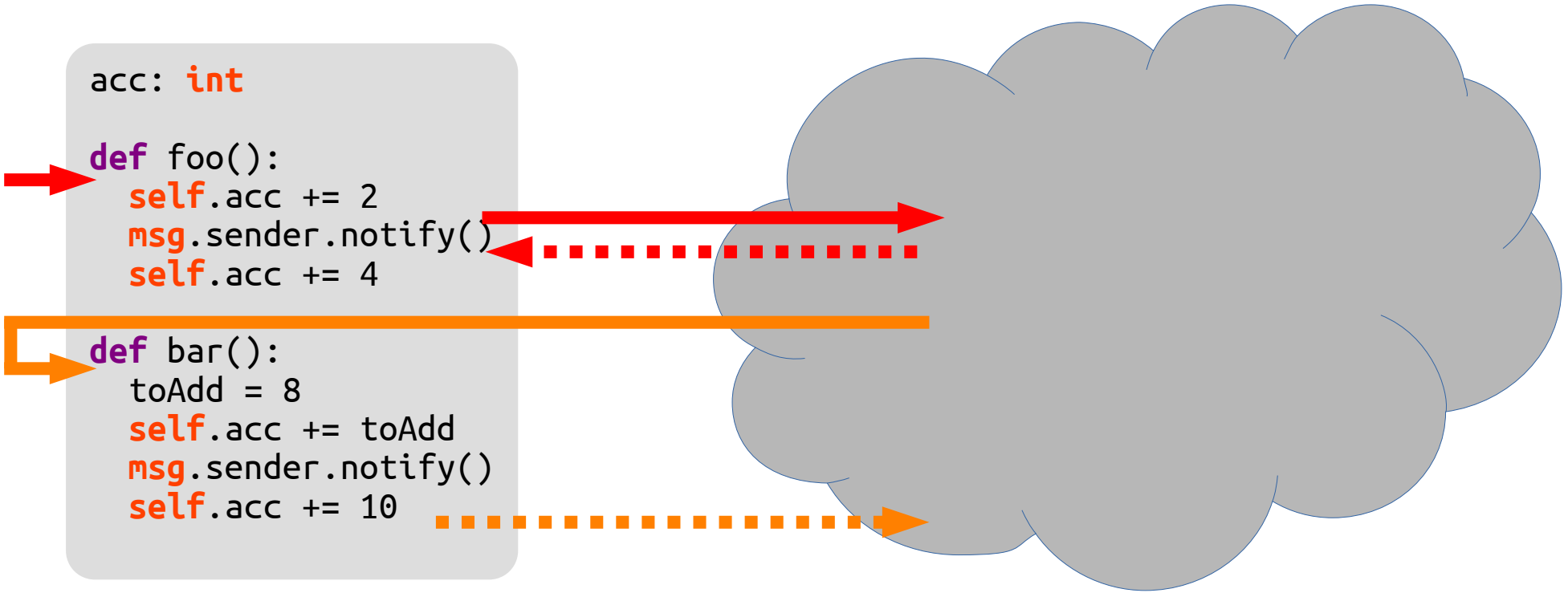
```
acc: int

def foo():
  self.acc += 2
  msg.sender.notify()
  self.acc += 4

def bar():
  toAdd = 8
  self.acc += toAdd
  msg.sender.notify()
  self.acc += 10
```



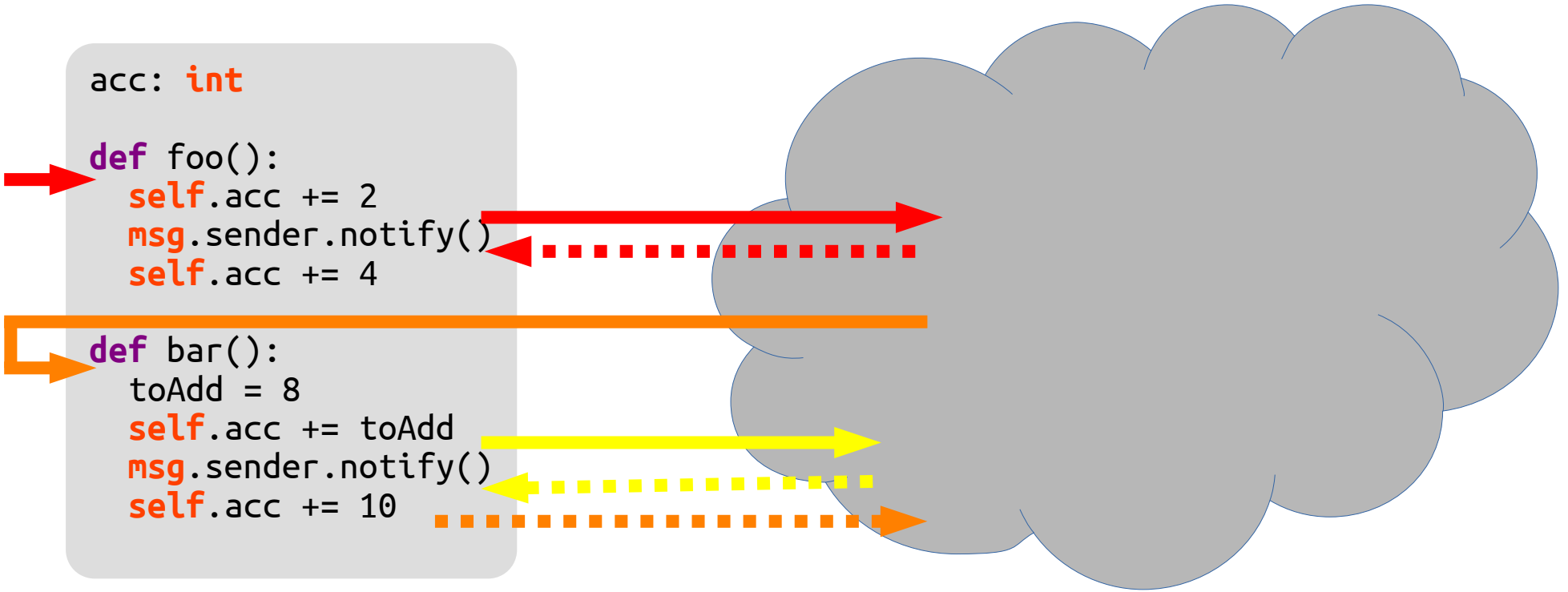
Reasoning about (re-entrant) calls



Reasoning about (re-entrant) calls



Reasoning about (re-entrant) calls



Reasoning about (re-entrant) calls

```
acc: int

def foo():
  self.acc += 2
  msg.sender.notify()
  self.acc += 4

def bar():
  toAdd = 8
  self.acc += toAdd
  msg.sender.notify()
  self.acc += 10
```

Reasoning about (re-entrant) calls

```
acc: int

def foo():
  self.acc += 2
  msg.sender.notify()
  self.acc += 4

def bar():
  toAdd = 8
  self.acc += toAdd
  msg.sender.notify()
  self.acc += 10
```

- Every local segment:

Reasoning about (re-entrant) calls

```
acc: int

def foo():
  self.acc += 2
  msg.sender.notify()
  self.acc += 4

def bar():
  toAdd = 8
  self.acc += toAdd
  msg.sender.notify()
  self.acc += 10
```

- Every local segment:

  self.acc % 2 == 0

Reasoning about (re-entrant) calls

```
acc: int

def foo():
  self.acc += 2
  msg.sender.notify()
  self.acc += 4

def bar():
  toAdd = 8
  self.acc += toAdd
  msg.sender.notify()
  self.acc += 10
```

- Every local segment:

  self.acc % 2 == 0 &&

  self.acc >= old(self.acc)

Reasoning about (re-entrant) calls

```
acc: int

def foo():
  self.acc += 2
  msg.sender.notify()
  self.acc += 4

def bar():
  toAdd = 8
  self.acc += toAdd
  msg.sender.notify()
  self.acc += 10
```

- Every local segment:

  self.acc % 2 == 0 &&

  self.acc >= old(self.acc)

  - *Segment constraint*

Reasoning about (re-entrant) calls

```
acc: int

def foo():
  self.acc += 2
  msg.sender.notify()
  self.acc += 4

def bar():
  toAdd = 8
  self.acc += toAdd
  msg.sender.notify()
  self.acc += 10
```

- Every local segment:

  self.acc % 2 == 0 &&

  self.acc >= old(self.acc)
  - *Segment constraint*

Reasoning about (re-entrant) calls

```
acc: int

def foo():
  self.acc += 2
  msg.sender.notify()
  self.acc += 4

def bar():
  toAdd = 8
  self.acc += toAdd
  msg.sender.notify()
  self.acc += 10
```

- Every local segment:

  self.acc % 2 == 0 &&

  self.acc >= old(self.acc)
  - *Segment constraint*

- *Similar concept: Function constraints*

Call-independent properties: access control

```
balances: map(address, int)
minter: address

def transfer(to: address, amount: int):
  from = msg.sender
  assert self.balances[from] >= amount
  self.balances[to] += amount
  self.balances[from] -= amount
  to.notify(from, self, amount)

def mint(to: address, amount: int):
  assert msg.sender == self.minter
  self.balances[to] += amount
```

Call-independent properties: access control

```
balances: map(address, int)
minter: address

def transfer(to: address, amount: int):
  from = msg.sender
  assert self.balances[from] >= amount
  self.balances[to] += amount
  self.balances[from] -= amount
  to.notify(from, self, amount)

def mint(to: address, amount: int):
  assert msg.sender == self.minter
  self.balances[to] += amount
```

- Balances conceptually belong to different addresses

Call-independent properties: access control

```
balances: map(address, int)
minter: address

def transfer(to: address, amount: int):
  from = msg.sender
  assert self.balances[from] >= amount
  self.balances[to] += amount
  self.balances[from] -= amount
  to.notify(from, self, amount)

def mint(to: address, amount: int):
  assert msg.sender == self.minter
  self.balances[to] += amount
```

- Balances conceptually belong to different addresses
  - Only *a* may decrease self.balances[*a*]

Call-independent properties: access control

```
balances: map(address, int)
minter: address

def transfer(to: address, amount: int):
  from = msg.sender
  assert self.balances[from] >= amount
  self.balances[to] += amount
  self.balances[from] -= amount
  to.notify(from, self, amount)

def mint(to: address, amount: int):
  assert msg.sender == self.minter
  self.balances[to] += amount
```

- Balances conceptually belong to different addresses
  - Only *a* may decrease self.balances[*a*]
- Expressed as property of every local segment

Call-independent properties: access control

```
balances: map(address, int)
minter: address

def transfer(to: address, amount: int):
    from = msg.sender
    assert self.balances[from] >= amount
    self.balances[to] += amount
    self.balances[from] -= amount
    to.notify(from, self, amount)

def mint(to: address, amount: int):
    assert msg.sender == self.minter
    self.balances[to] += amount
```

- Balances conceptually belong to different addresses
  - Only *a* may decrease self.balances[*a*]

- Expressed as property of every local segment:

  ∀ a. msg.sender != a ==>

  self.balances[a] >=

  old(self.balances[a])

Inter-contract invariants

```
contract TokenClient:
…

def payout():
  self.token.transfer(self.beneficiary,
                      self.owedAmount)
```

Inter-contract invariants

```
contract TokenClient:
…

def payout():
  self.token.transfer(self.beneficiary,
                      self.owedAmount)
```

```
interface Token:

balances: map(address, int)

def transfer(to: address,
             amount: int):
  …
```

Inter-contract invariants

```
contract TokenClient:
…

def payout():
  self.token.transfer(self.beneficiary,
                      self.owedAmount)

# invariant: self.owedAmount <=
#            self.token.balances[self]
```

```
interface Token:

balances: map(address, int)

def transfer(to: address,
             amount: int):
  …
```

Inter-contract invariants

```
contract TokenClient:
…

def payout():
  self.token.transfer(self.beneficiary,
                      self.owedAmount)

# invariant: self.owedAmount <=
#            self.token.balances[self]
```

```
interface Token:

balances: map(address, int)

def transfer(to: address,
             amount: int):
  …
```

```
contract Token:
…

def steal(from: address):
  self.balances[from] = 0
```

Inter-contract invariants

```
contract TokenClient:
…

def payout():
  self.token.transfer(self.beneficiary,
                      self.owedAmount)

# invariant: self.owedAmount <=
#            self.token.balances[self]
```

```
interface Token:

balances: map(address, int)

def transfer(to: address,
             amount: int):
  …

# segment constraint:
# self.balances[a] is private
# for all a
```

```
contract Token:
…

def steal(from: address):
  self.balances[from] = 0
```

Inter-contract invariants

```
contract TokenClient:
…

def payout():
  self.token.transfer(self.beneficiary,
                      self.owedAmount)

# invariant: self.owedAmount <=
#            self.token.balances[self]
```

```
interface Token:

balances: map(address, int)

def transfer(to: address,
             amount: int):
  …

# segment constraint:
# self.balances[a] is private
# for all a
```

```
contract Token:
…

def steal(from: address):
  self.balances[from] = 0
```

Tokens are Resources

```
balances: map(address, int)
minter: address

def transfer(to: address, amount: int):
  from = msg.sender
  assert self.balances[from] >= amount
  self.balances[to] += amount
  self.balances[from] -= amount
  to.notify(from, self, amount)

def mint(to: address, amount: int):
  assert msg.sender == self.minter
  self.balances[to] += amount
```

Tokens are Resources

```
balances: map(address, int)
minter: address

def transfer(to: address, amount: int):
  from = msg.sender
  assert self.balances[from] >= amount
  self.balances[to] += amount
  self.balances[from] -= amount
  to.notify(from, self, amount)

def mint(to: address, amount: int):
  assert msg.sender == self.minter
  self.balances[to] += amount
```

- Every segment:
  - Tokens are not duplicated
  - Noone can take tokens from owner
  - Noone can destroy other's tokens
  - ....

Tokens are Resources

```
balances: map(address, int)
minter: address




def transfer(to: address, amount: int):
  ...

def mint(to: address, amount: int):
  ...
```

- Every segment:
  - Tokens are not duplicated
  - Noone can take tokens from owner
  - Noone can destroy other's tokens
  - ....

Tokens are Resources

```
balances: map(address, int)
minter: address

# resource Token



def transfer(to: address, amount: int):
  ...

def mint(to: address, amount: int):
  ...
```

- Every segment:
  - Tokens are not duplicated
  - Noone can take tokens from owner
  - Noone can destroy other's tokens
  - ....

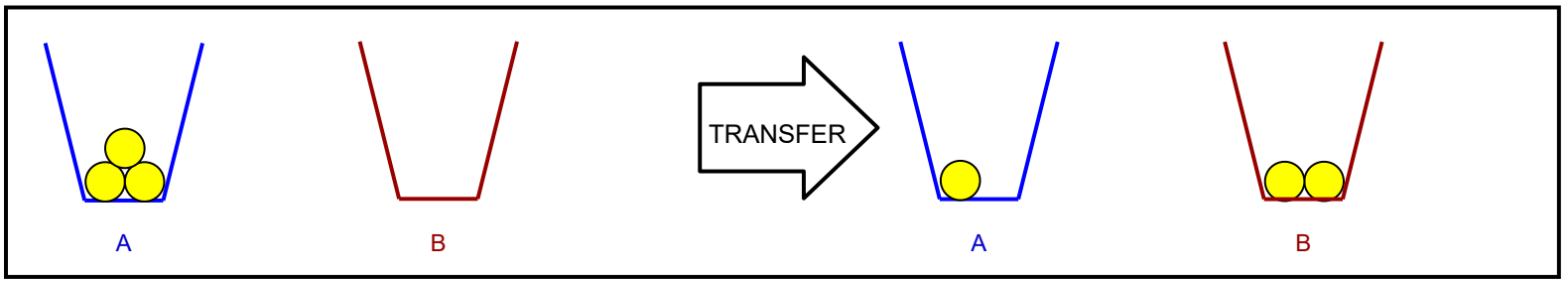
Tokens are Resources

```
balances: map(address, int)
minter: address

# resource Token
# inv: balances[Token] = self.balances


def transfer(to: address, amount: int):
  ...

def mint(to: address, amount: int):
  ...
```

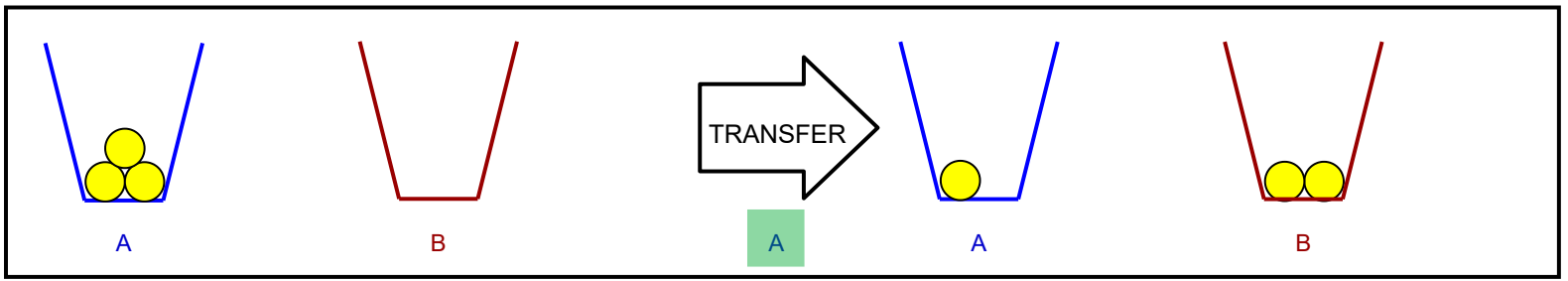
- Every segment:
  - Tokens are not duplicated
  - Noone can take tokens from owner
  - Noone can destroy other's tokens
  - ....

Resource Operations

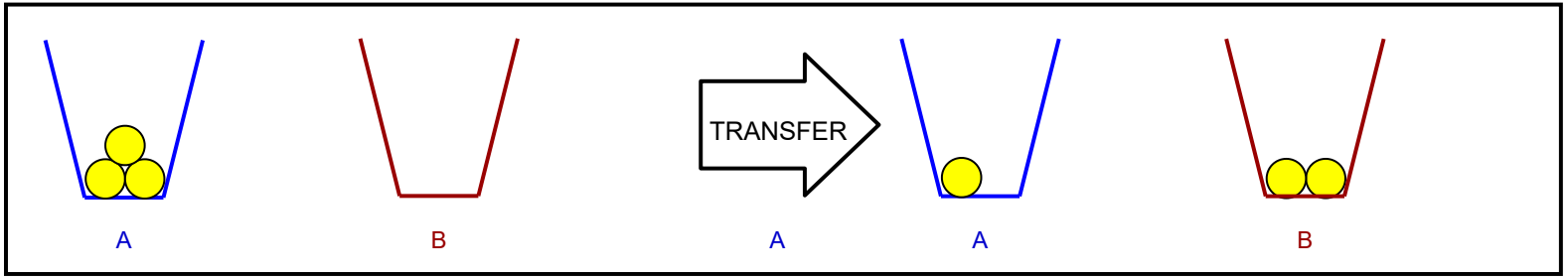
Resource Operations

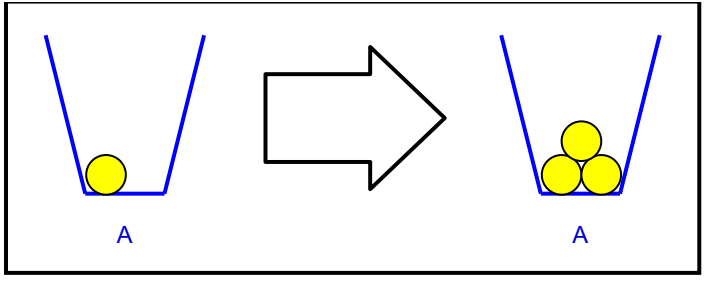


Resource Operations

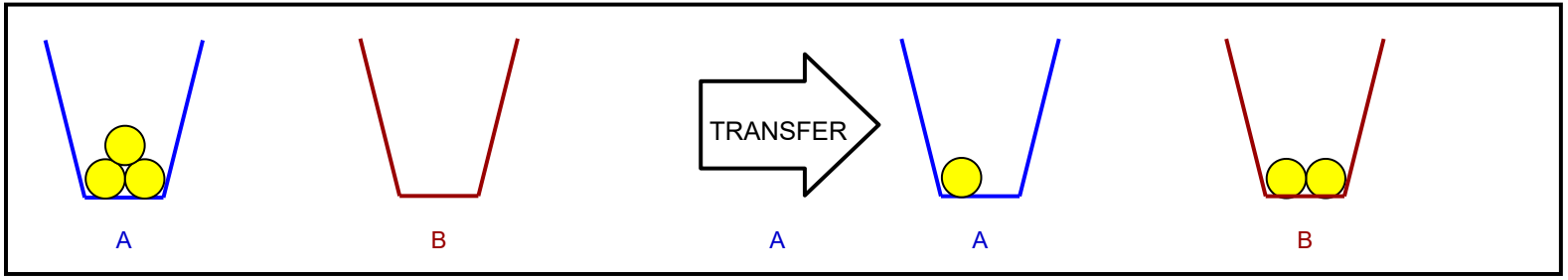


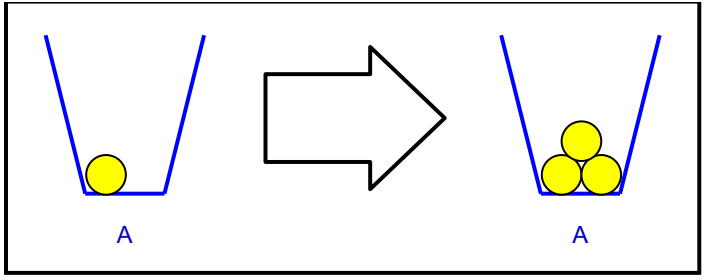
Resource Operations

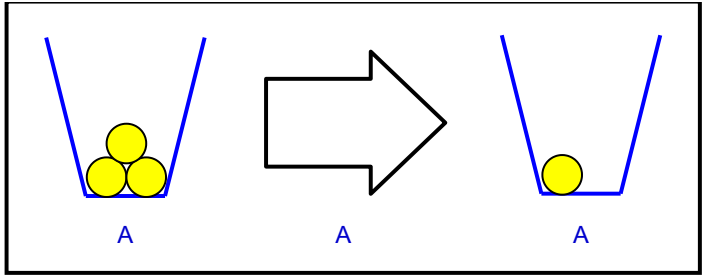




Resource Operations







Tokens are Resources

```
balances: map(address, int)
minter: address

# resource Token
# inv: balances[Token] = self.balances



def transfer(to: address, amount: int):
  from = msg.sender
  assert self.balances[from] >= amount
  self.balances[to] += amount
  self.balances[from] -= amount

  to.notify(from, self, amount)
```

- Every segment:
  - Tokens are not duplicated
  - Noone can take tokens from owner
  - Noone can destroy other's tokens
  - ....

Tokens are Resources

```
balances: map(address, int)
minter: address

# resource Token
# inv: balances[Token] = self.balances



def transfer(to: address, amount: int):
  from = msg.sender
  assert self.balances[from] >= amount
  self.balances[to] += amount
  self.balances[from] -= amount
  # transfer[Token](from, to, amount)
  to.notify(from, self, amount)
```

- Every segment:
  - Tokens are not duplicated
  - Noone can take tokens from owner
  - Noone can destroy other's tokens
  - ....

Tokens are Resources

```
balances: map(address, int)
minter: address

# resource Token
# inv: balances[Token] = self.balances



def transfer(to: address, amount: int):
  from = msg.sender
  assert self.balances[from] >= amount
  self.balances[to] += amount
  self.balances[from] -= amount
  # transfer[Token](from, to, amount)
  to.notify(from, self, amount)
```

- Every segment:
  - Tokens are not duplicated
  - Noone can take tokens from owner
  - Noone can destroy other's tokens
  - ....

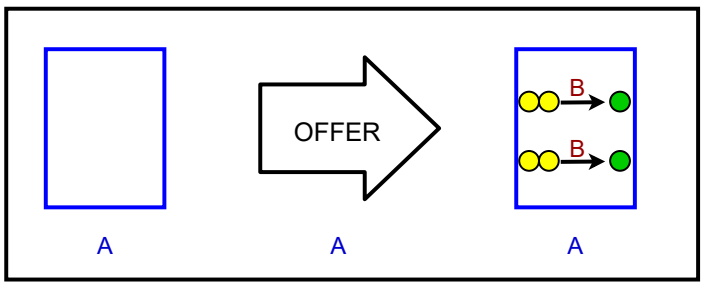
- Function transfer:
  - Receiver has amount added
  - Sender has amount subtracted
  - All others unaffected
  - ... modulo call to outside

Tokens are Resources

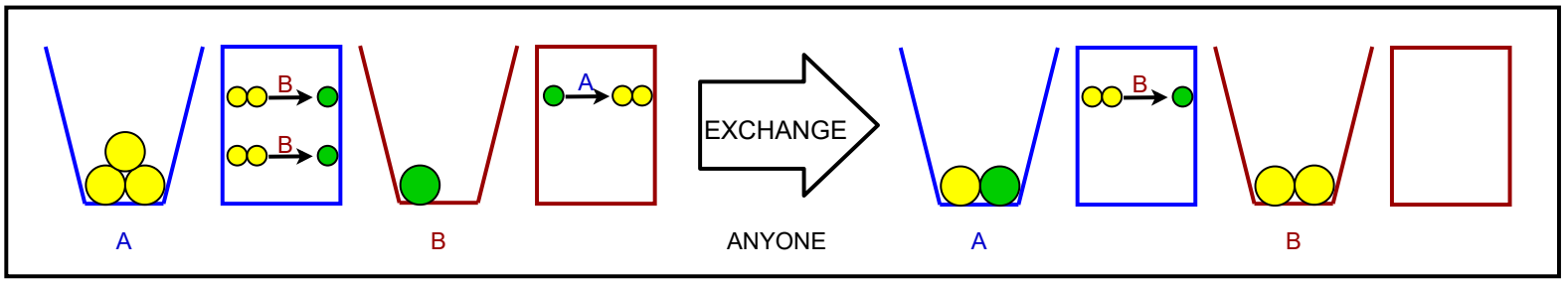
```
balances: map(address, int)
minter: address

# resource Token
# inv: balances[Token] = self.balances

# performs:
#   transfer[Token](from, to, amount)
def transfer(to: address, amount: int):
  from = msg.sender
  assert self.balances[from] >= amount
  self.balances[to] += amount
  self.balances[from] -= amount
  # transfer[Token](from, to, amount)
  to.notify(from, self, amount)
```

- Every segment:
  - Tokens are not duplicated
  - Noone can take tokens from owner
  - Noone can destroy other's tokens
  - ....
- Function transfer:
  - Receiver has amount added
  - Sender has amount subtracted
  - All others unaffected
  - ... modulo call to outside

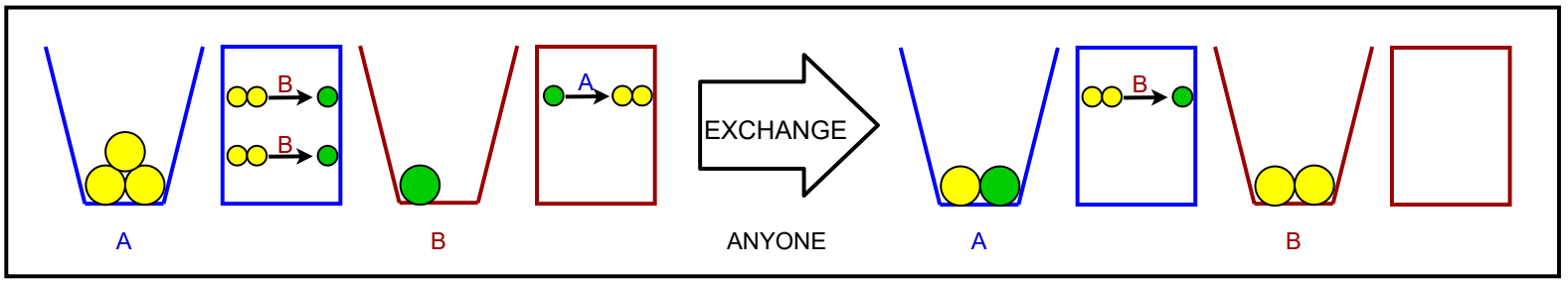
Resource Operations



Resource Operations

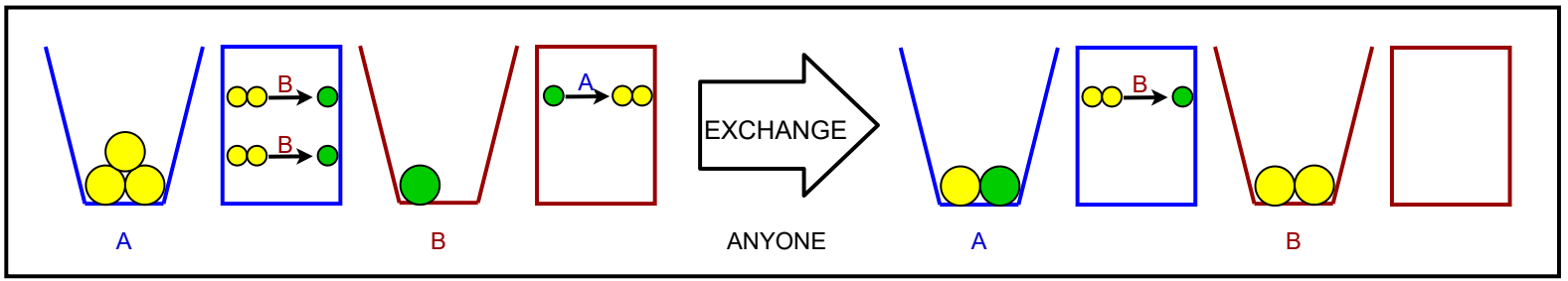


Resource Operations



Other concepts:

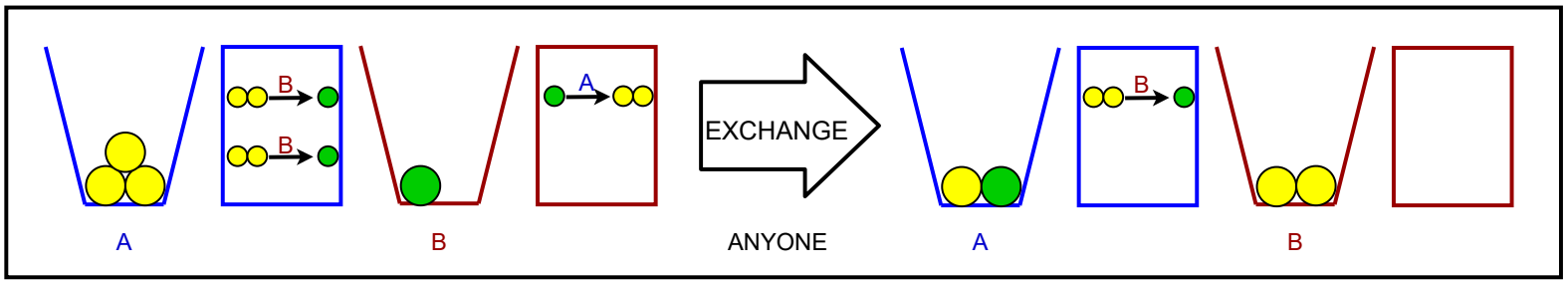
Resource Operations



Other concepts:

- Possession vs. ownership
  - A has a resource that conceptually belongs to B

Resource Operations



Other concepts:

- Possession vs. ownership
  - A has a resource that conceptually belongs to B
- Trust: consciously weaken guarantees

Our tool: 2vyper

https://github.com/viperproject/2vyper

Our tool: 2vyper

- Supports entire Vyper language modulo bugs
  - different Vyper versions

https://github.com/viperproject/2vyper

Our tool: 2vyper

- Supports entire Vyper language modulo bugs
  - different Vyper versions
- All standard specification constructs
  - pre/post, quantifiers, sums, …
- … *plus* segment constraints, inter-contract invariants, resources

https://github.com/viperproject/2vyper

Our tool: 2vyper

```
#@ ensures: implies(_value > old(self.balanceOf[msg.sender]), revert())
#@ seg_constr: implies(success(), event(Transfer(msg.sender, _to, _value)))
@public
def transfer(_to: address, _value: uint256) -> bool:
    self.balanceOf[msg.sender] -= _value
    #@ transfer[token](_value, to=_to)
    self.balanceOf[_to] += _value
    log.Transfer(msg.sender, _to, _value)
    return True
```

```
Verification failed
Errors:
Invariant not preserved by transfer. Assertion (allocated[token]() == self.balanceOf) might not hold.
(ERC20_alloc.vy@112.1, via invariant at ERC20_alloc.vy@68.15)
Counterexample:
   _to = 0x00000000000000000000000000000000000000ea
   _value = 1
   block.coinbase = 0x0000000000000000000000000000000000000000
   block.difficulty = 0
   block.number = 0
   block.prevhash = [ _ -> 0 ]: 32
   block.timestamp = 0
   chain.id = 38
   msg.gas = 574
   msg.sender = 0x00000000000000000000000000000000000000eb
```

Evaluation

Application	Contract	LOC	Ann.	IF LOC	IF Ann.	T
auction	auction	63	30	-	-	12.72
auction_token	auction_token††	96	37	88	33	23.67
DAO	DAO*	17	2	-	-	5.13
ERC20	ERC20	98	31	67	25	11.51
ERC721	ERC721	178	32	-	-	15.95
ERC1363	ERC1363	142	31	88	33	22.59
ICO	gv_option_token	98	26	86	36	10.03
	gv_token	121	24	107	49	15.21
	gv_option_program*	86	12	154	67	29.19
	ico_alloc*	159	30	261	116	101.86
Mana	token*	18	3	50	25	2.78
	crowdsale*	42	14	50	25	5.77
	continuous_sale*	36	8	50	25	3.88
VerX_overview	escrow	60	11	65	33	6.36
	crowdsale	41	9	65	33	6.35
safe_remote_purchase	safe_remote_purchase	71	29	-	-	16.84
serenuscoin	serenuscoin	103	4	-	-	6.40
Uniswap V1	Uniswap†	398	115	105	45	112.81

Evaluation

Application	Contract	LOC	Ann.	IF LOC	IF Ann.	T
auction	auction	63	30	-	-	12.72
auction_token	auction_token††	96	37	88	33	23.67
DAO	DAO*	17	2	-	-	5.13
ERC20	ERC20	98	31	67	25	11.51
ERC721	ERC721	178	32	-	-	15.95
ERC1363	ERC1363	142	31	88	33	22.59
ICO	gv_option_token	98	26	86	36	10.03
	gv_token	121	24	107	49	15.21
	gv_option_program*	86	12	154	67	29.19
	ico_alloc*	159	30	261	116	101.86
Mana	token*	18	3	50	25	2.78
	crowdsale*	42	14	50	25	5.77
	continuous_sale*	36	8	50	25	3.88
VerX_overview	escrow	60	11	65	33	6.36
	crowdsale	41	9	65	33	6.35
safe_remote_purchase	safe_remote_purchase	71	29	-	-	16.84
serenuscoin	serenuscoin	103	4	-	-	6.40
Uniswap V1	Uniswap†	398	115	105	45	112.81

Evaluation

Application	Contract	LOC	Ann.	IF LOC	IF Ann.	T
auction	auction	63	30	-	-	12.72
auction_token	auction_token††	96	37	88	33	23.67
DAO	DAO*	17	2	-	-	5.13
ERC20	ERC20	98	31	67	25	11.51
ERC721	ERC721	178	32	-	-	15.95
ERC1363	ERC1363	142	31	88	33	22.59
ICO	gv_option_token	98	26	86	36	10.03
	gv_token	121	24	107	49	15.21
	gv_option_program*	86	12	154	67	29.19
	ico_alloc*	159	30	261	116	101.86
Mana	token*	18	3	50	25	2.78
	crowdsale*	42	14	50	25	5.77
	continuous_sale*	36	8	50	25	3.88
VerX_overview	escrow	60	11	65	33	6.36
	crowdsale	41	9	65	33	6.35
safe_remote_purchase	safe_remote_purchase	71	29	-	-	16.84
serenuscoin	serenuscoin	103	4	-	-	6.40
Uniswap V1	Uniswap†	398	115	105	45	112.81

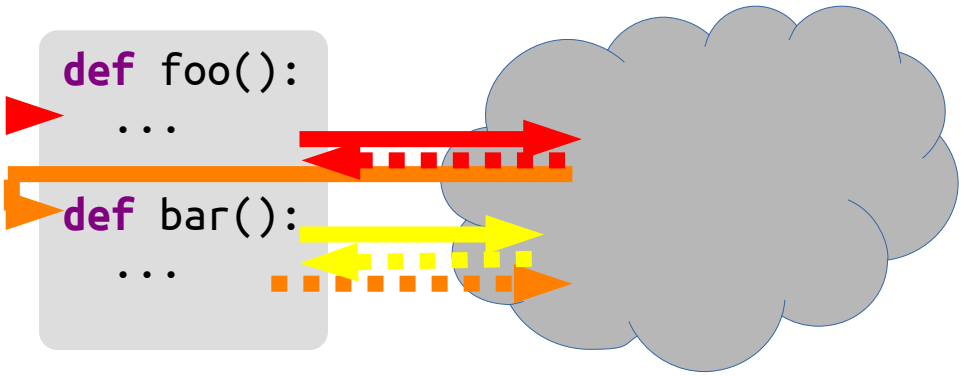
Evaluation

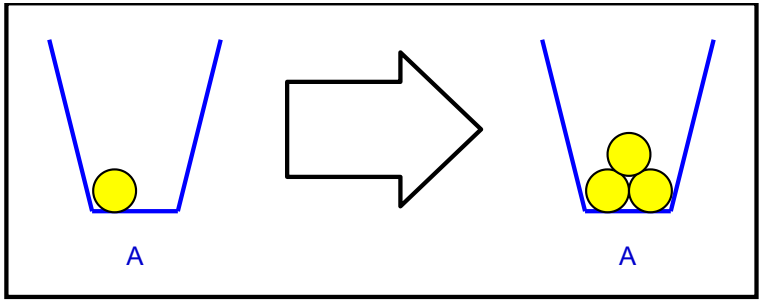
Application	Contract	LOC	Ann.	IF LOC	IF Ann.	T
auction	auction	63	30	-	-	12.72
auction_token	auction_token††	96	37	88	33	23.67
DAO	DAO*	17	2	-	-	5.13
ERC20	ERC20	98	31	67	25	11.51
ERC721	ERC721	178	32	-	-	15.95
ERC1363	ERC1363	142	31	88	33	22.59
ICO	gv_option_token	98	26	86	36	10.03
	gv_token	121	24	107	49	15.21
	gv_option_program*	86	12	154	67	29.19
	ico_alloc*	159	30	261	116	101.86
Mana	token*	18	3	50	25	2.78
	crowdsale*	42	14	50	25	5.77
	continuous_sale*	36	8	50	25	3.88
VerX_overview	escrow	60	11	65	33	6.36
	crowdsale	41	9	65	33	6.35
safe_remote_purchase	safe_remote_purchase	71	29	-	-	16.84
serenuscoin	serenuscoin	103	4	-	-	6.40
Uniswap V1	Uniswap†	398	115	105	45	112.81

Evaluation

Application	Contract	LOC	Ann.	IF LOC	IF Ann.	T
auction	auction	63	30	-	-	12.72
auction_token	auction_token††	96	37	88	33	23.67
DAO	DAO*	17	2	-	-	5.13
ERC20	ERC20	98	31	67	25	11.51
ERC721	ERC721	178	32	-	-	15.95
ERC1363	ERC1363	142	31	88	33	22.59
ICO	gv_option_token	98	26	86	36	10.03
	gv_token	121	24	107	49	15.21
	gv_option_program*	86	12	154	67	29.19
	ico_alloc*	159	30	261	116	101.86
Mana	token*	18	3	50	25	2.78
	crowdsale*	42	14	50	25	5.77
	continuous_sale*	36	8	50	25	3.88
VerX_overview	escrow	60	11	65	33	6.36
	crowdsale	41	9	65	33	6.35
safe_remote_purchase	safe_remote_purchase	71	29	-	-	16.84
serenuscoin	serenuscoin	103	4	-	-	6.40
Uniswap V1	Uniswap†	398	115	105	45	112.81

Thank you!





2vyper

VIPER

Z3

https://github.com/viperproject/2vyper