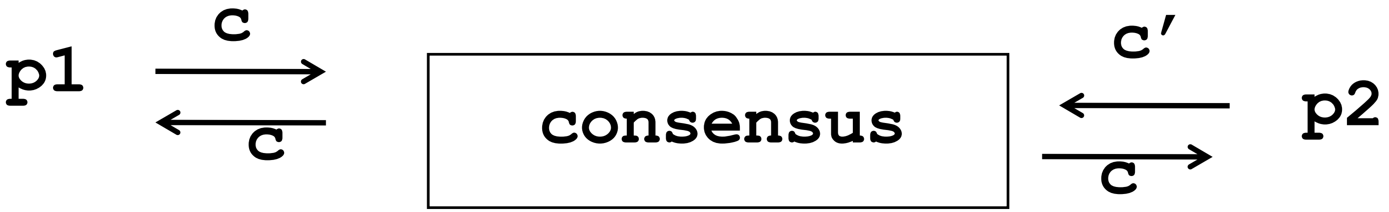
Generalized Universality

Consensus

Processes propose each a value and ***agree*** on one

output = ***propose(***input***)***



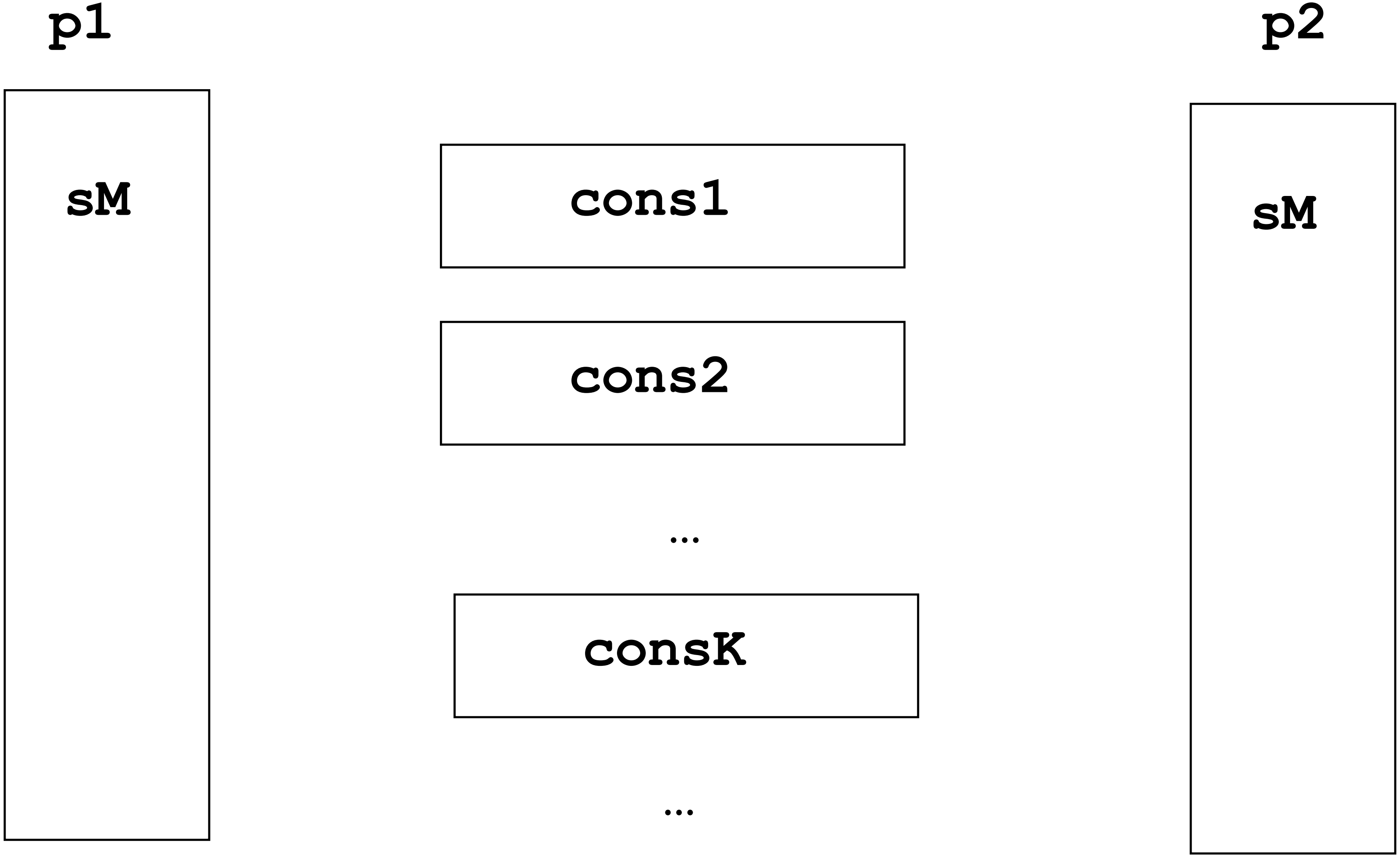
Universal Construction

Every process holds a copy of the - simulated - machine

Every process holds a list of commands for the machine

All processes share a list of consensus objects

Universal Construction



p1

sM

cons1

cons2

...

consK

...

p2

sM

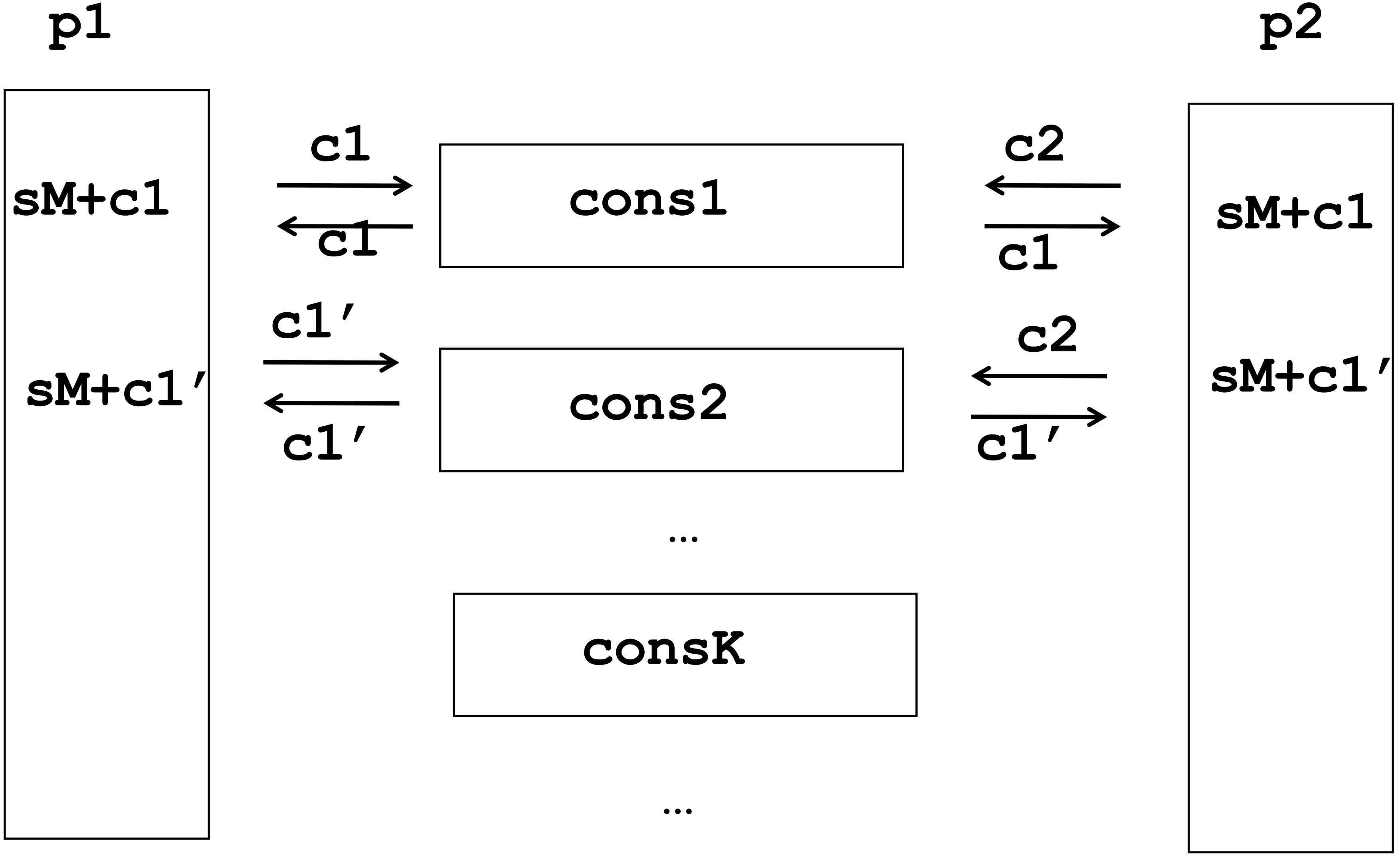
Universal Construction

- while(true)

- c = commands.next()
- cons = Consensus.next()

- c’ = cons.***propose***(c)
- sM.***perform***(c’)

Universal Construction



What if consensus is not available [FLP,CHT,DFG]

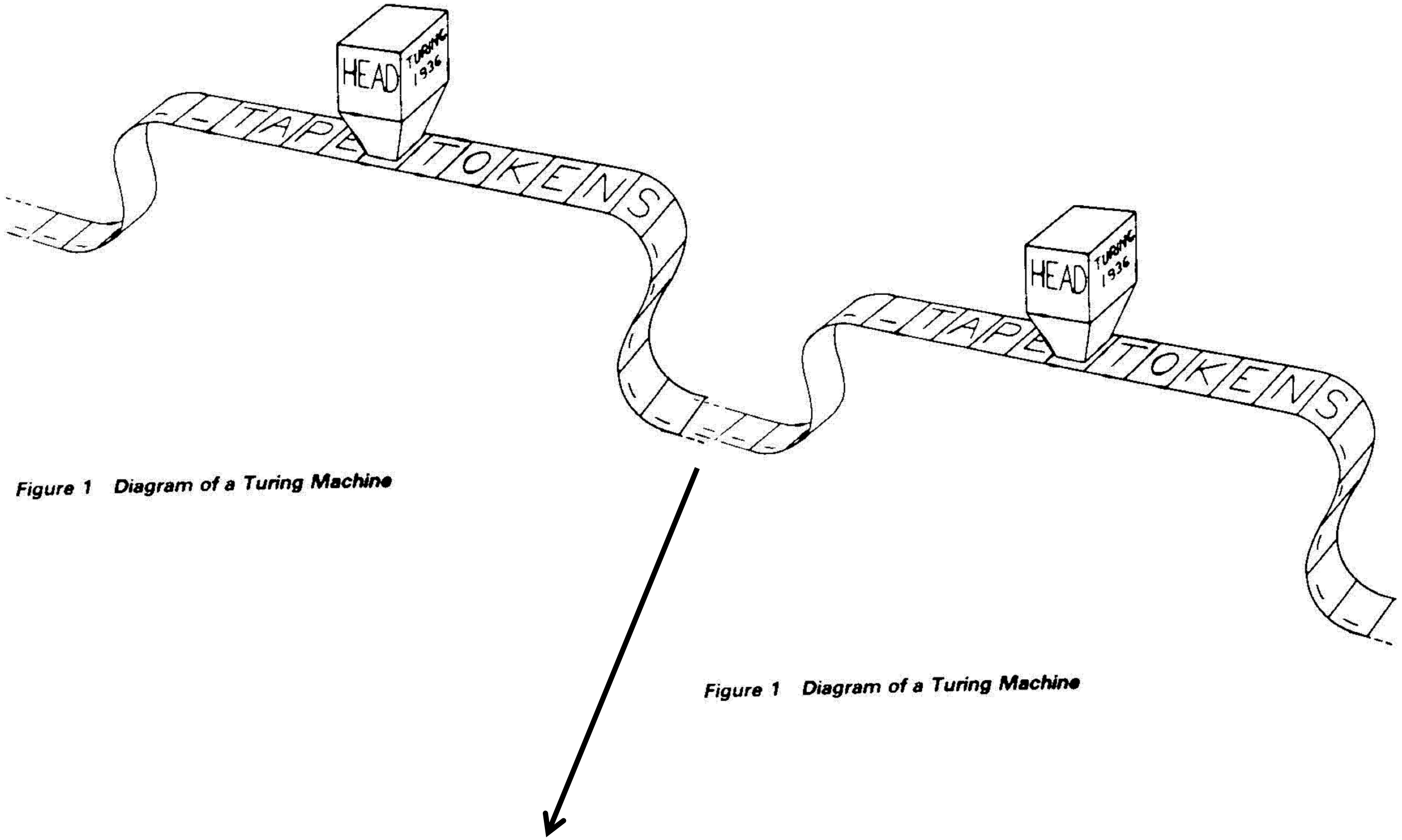


Figure 1 **Diagram of a Turing Machine**

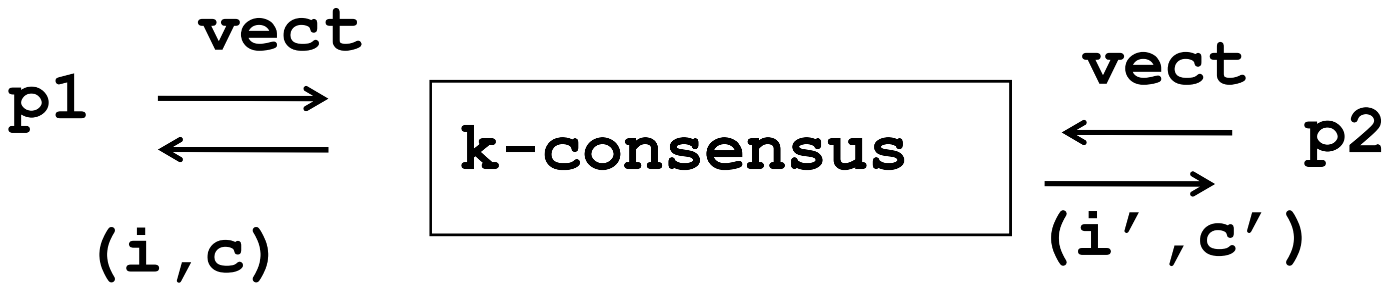
Figure 1 **Diagram of a Turing Machine**

Consensus

K-Consensus

Every process proposes a vector of k values and returns a value at some position (Chauduri et al)

(i,c) = ***propose(***kVect***)***



p1

vect

(i,c)

k-consensus

vect

p2

(i',c')

K-Consensus

- ***Validity***: the value returned at any position has been proposed at that position

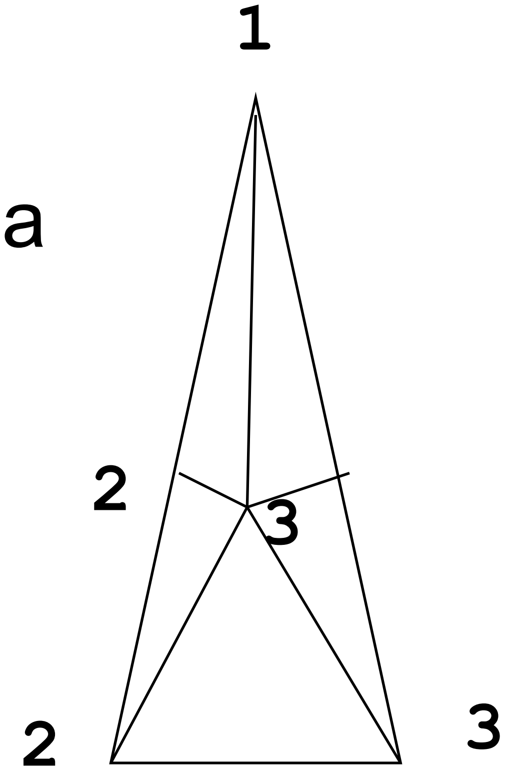
- ***Agreement***: no two values returned at the same position are different

- ***Termination***: every correct process that proposes eventually returns

k+1-consensus is strictly weaker than k-consensus in any system of at least k+1 processes

(Godel prize 2004 – HS,BG,SZ 93)

Sperner’s Lemma

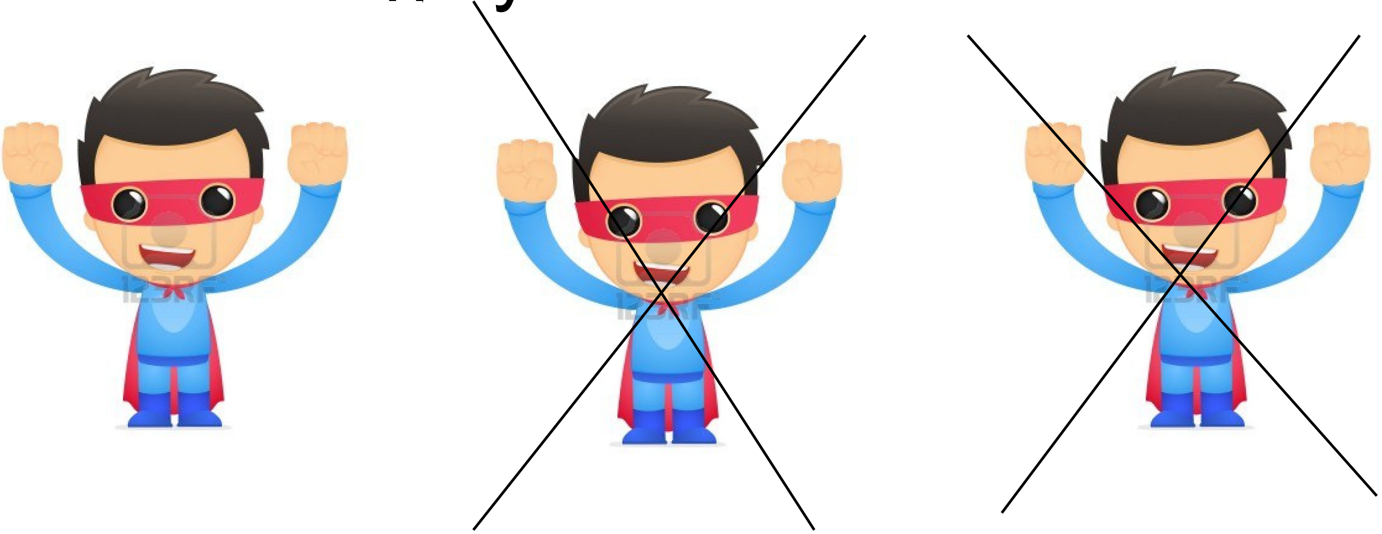


For any distributed computing task T, there is a k such that T ⇔ k-consensus (FDGT 2010)

What form of universality with K-consensus?

With consensus

We implement a highly-available state machine



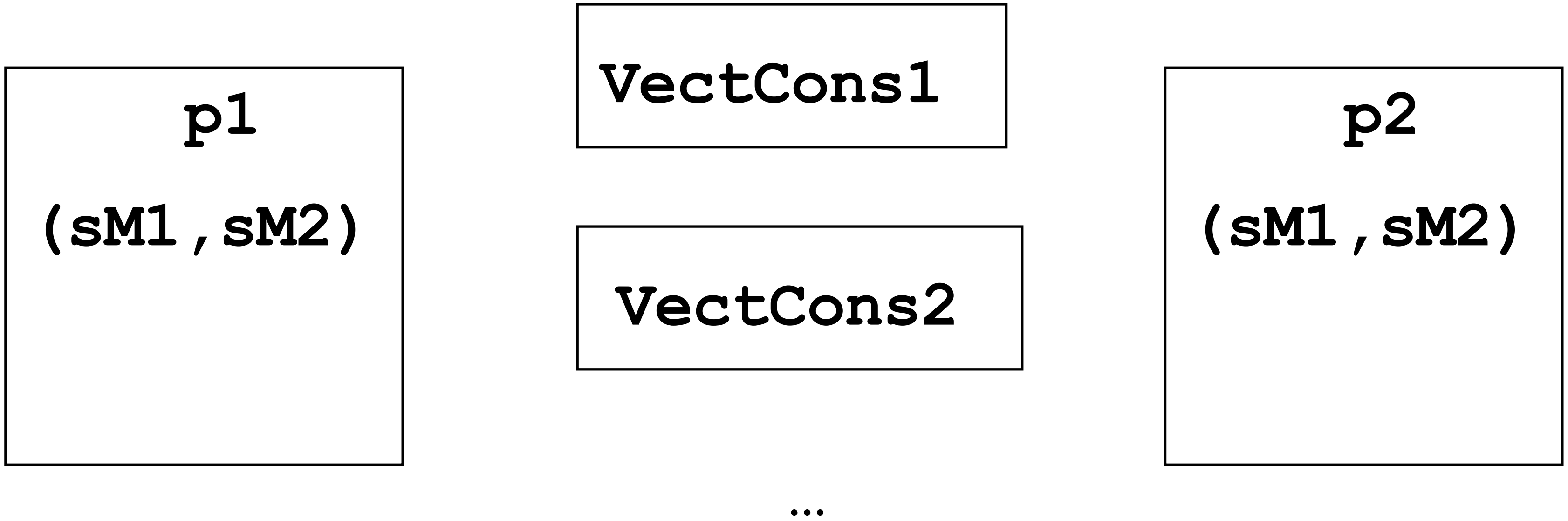
With k-consensus

We implement k state machines of which ***at least one*** is highly-available

Generalized Universality

Generalized Universality

Every process holds a copy of each of the machines sM(i) - and a lists of commands for each



p1

(sM1,sM2)

VectCons1

VectCons2

...

p2

(sM1,sM2)

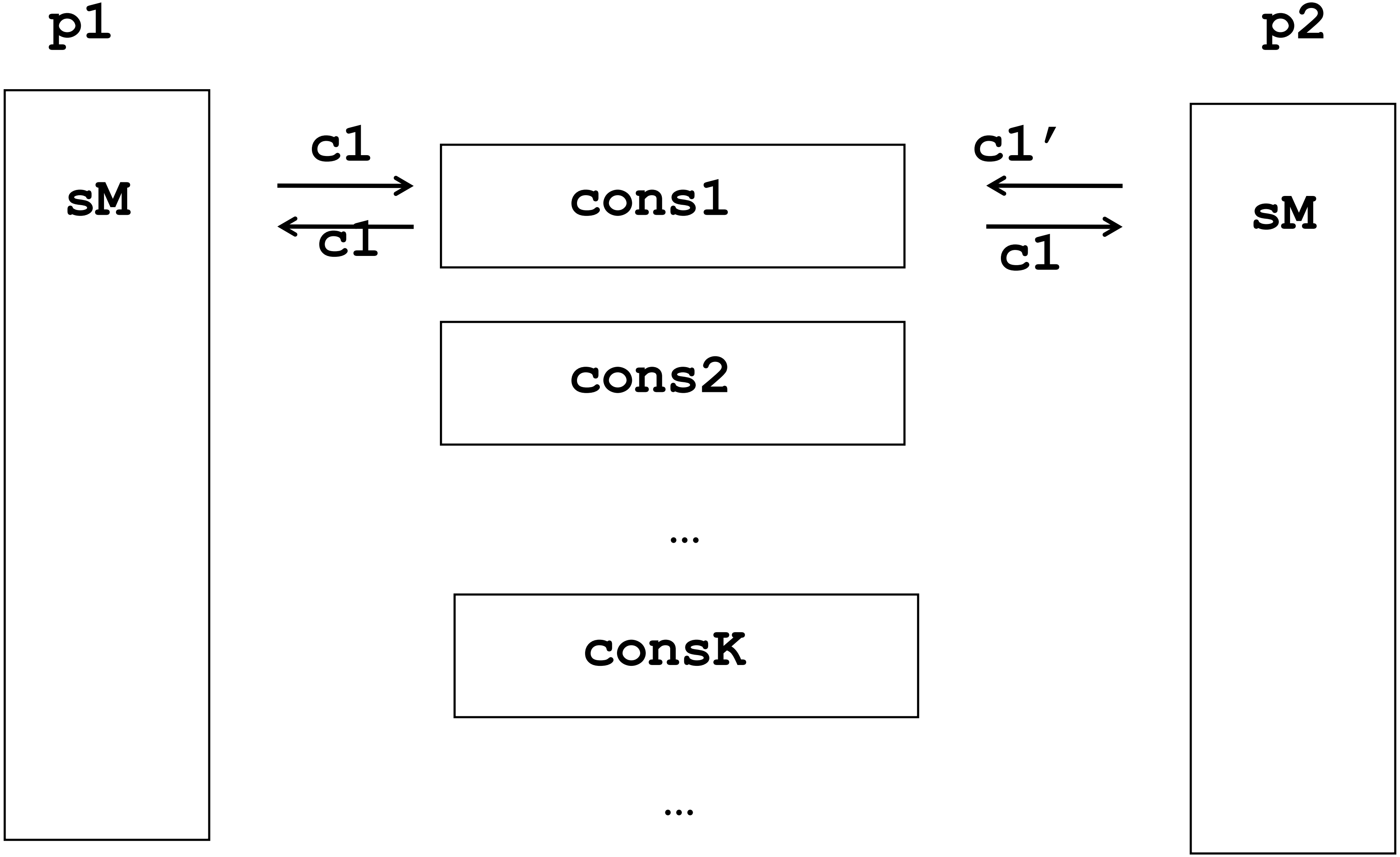
The processes share a list of k-vector consensus objects

Universal Construction

```
while(true)
    c = commands.next()
    cons = consensus.next()

    c’ = cons.propose(c)
    sM.perform(c’)
```

Universal Construction

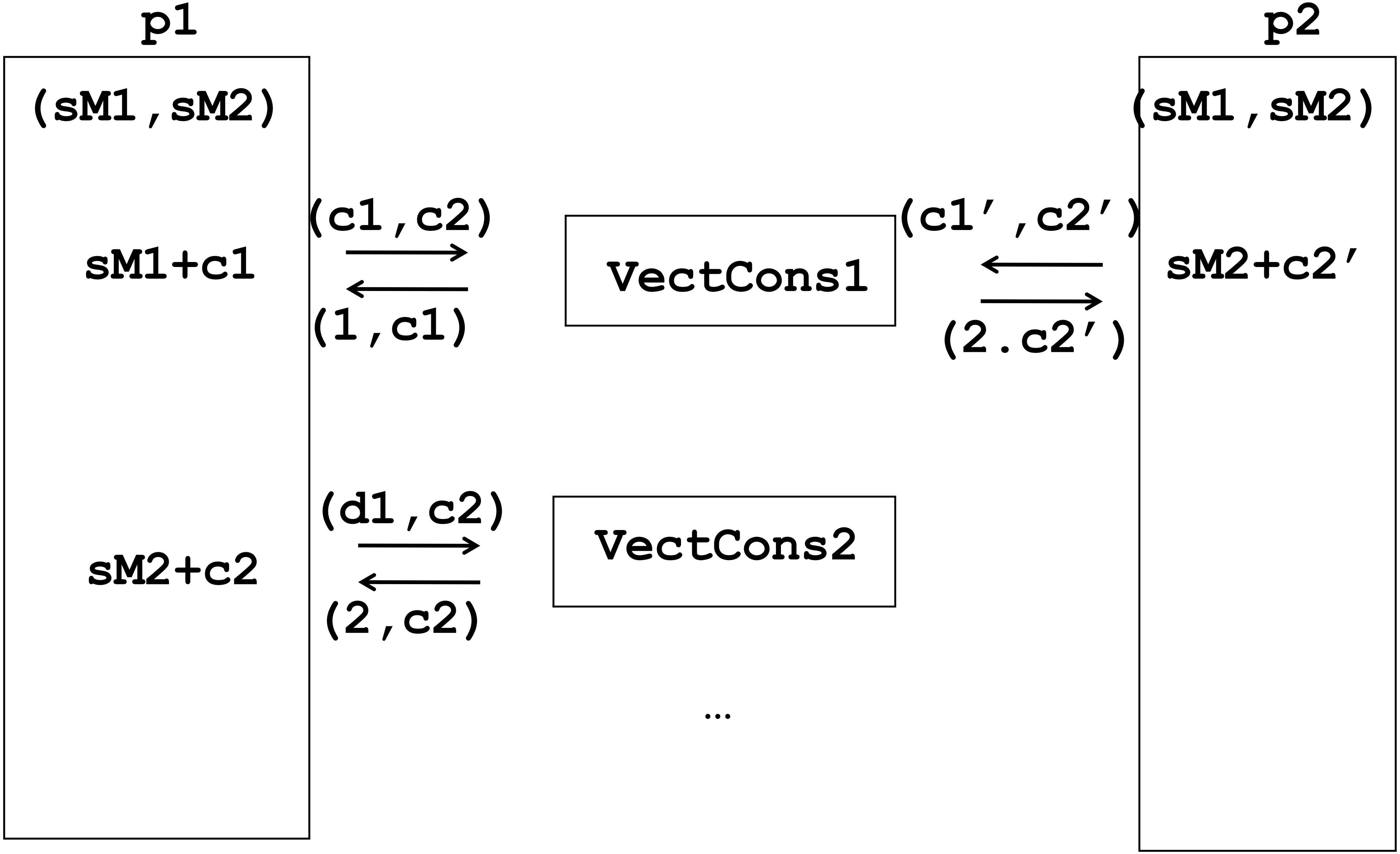


Generalized Universality?

```
while(true)
    for j = 1 to k: com(j) = commands(j).next()
    kVectC = kVectCons.next()

    (c,i) = kVectC.propose(com)
    sM(i).perform(c)
```

Problem with safety



p1

p2

(sM1,sM2)

(sM1,sM2)

(c1,c2)

(c1′,c2′)

sM1+c1

VectCons1

sM2+c2′

(1,c1)

(2.c2′)

(d1,c2)

VectCons2

sM2+c2

(2,c2)

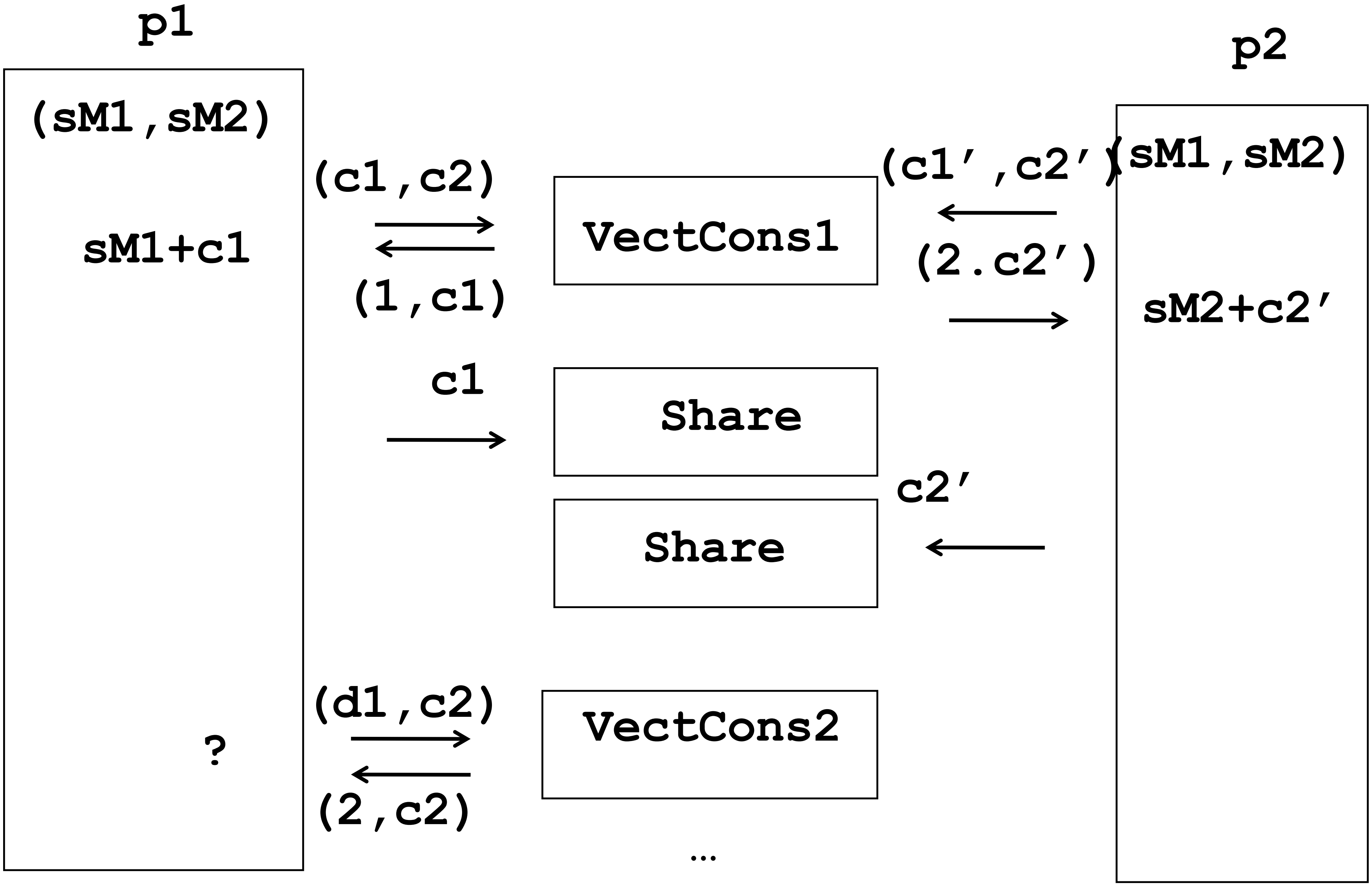
…

Generalized Universality

- while(true)
- for j = 1 to k: com(j) = commands(j).next()
- kVectC = kVectCons.next()

- (c,i) = kVectC.propose(com)
- ***check other processes for any missing c'***
- sM(i).perform(c)
- ***inform other processes about c***

Generalized Universality



p1

(sM1,sM2)

sM1+c1

?

(c1,c2)

(1,c1)

c1

(d1,c2)

(2,c2)

VectCons1

Share

Share

VectCons2

…

(c1′,c2′)

(2.c2′)

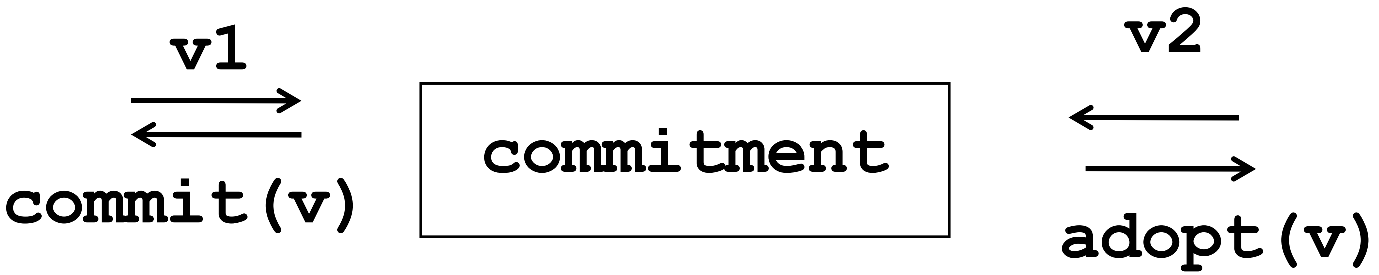
c2′

p2

(sM1,sM2)

sM2+c2′

1st key idea (ensuring safety)



v1

commit(v)

commitment

v2

adopt(v)

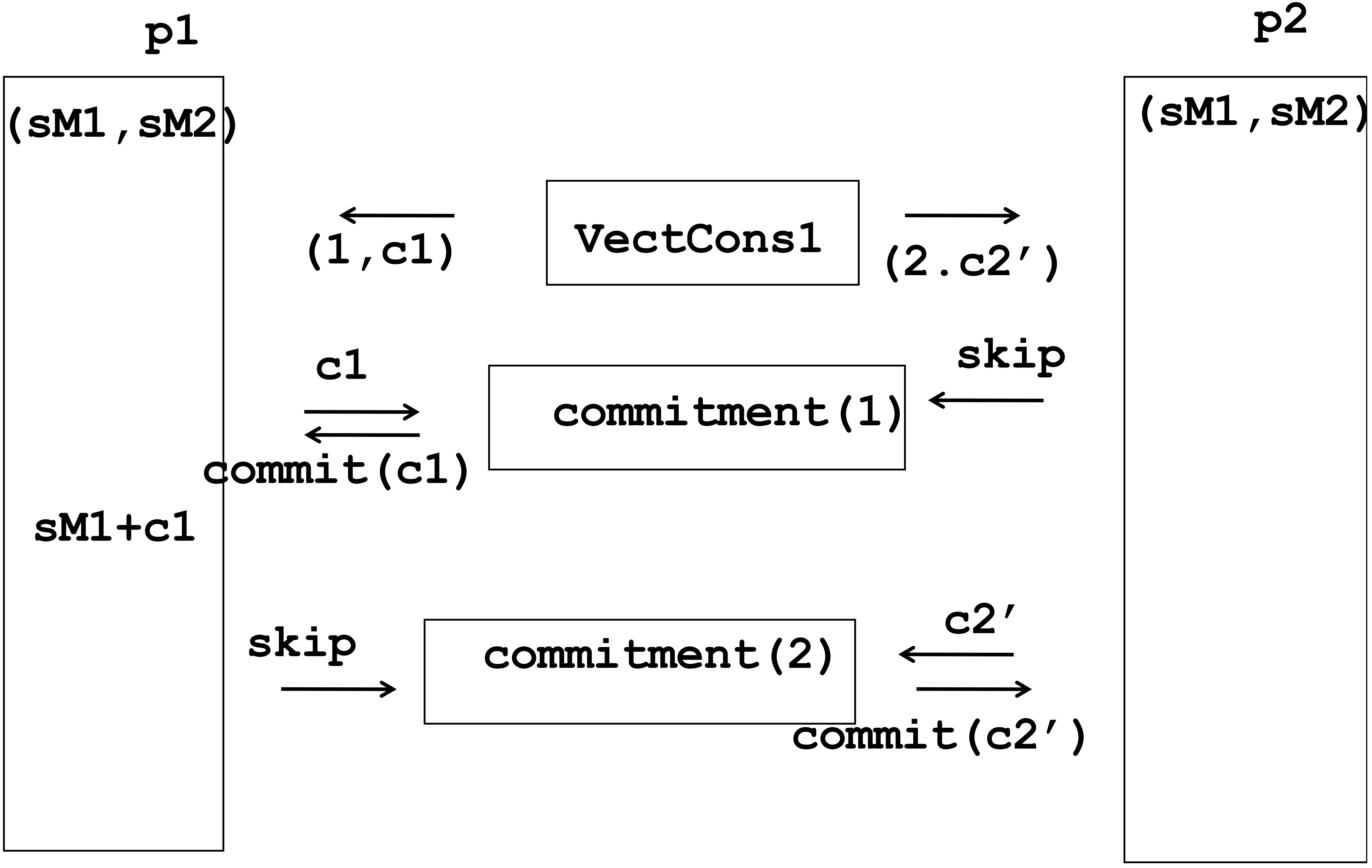
```
write (v)
if there is only v, write (commit, v)
   if there is only (commit, v), return(commit, v)
      if there is (commit, v’), return(adopt, v’)
else return(adopt, v)
```

Commitment

- ***Invariant (1)***: if a value v is committed then no other value is returned

- ***Invariant (2)***: if all processes propose the same value then the value is committed

Generalized Universality



p1

p2

(sM1,sM2)

(sM1,sM2)

VectCons1

(1,c1)

(2.c2′)

c1

skip

commitment(1)

commit(c1)

sM1+c1

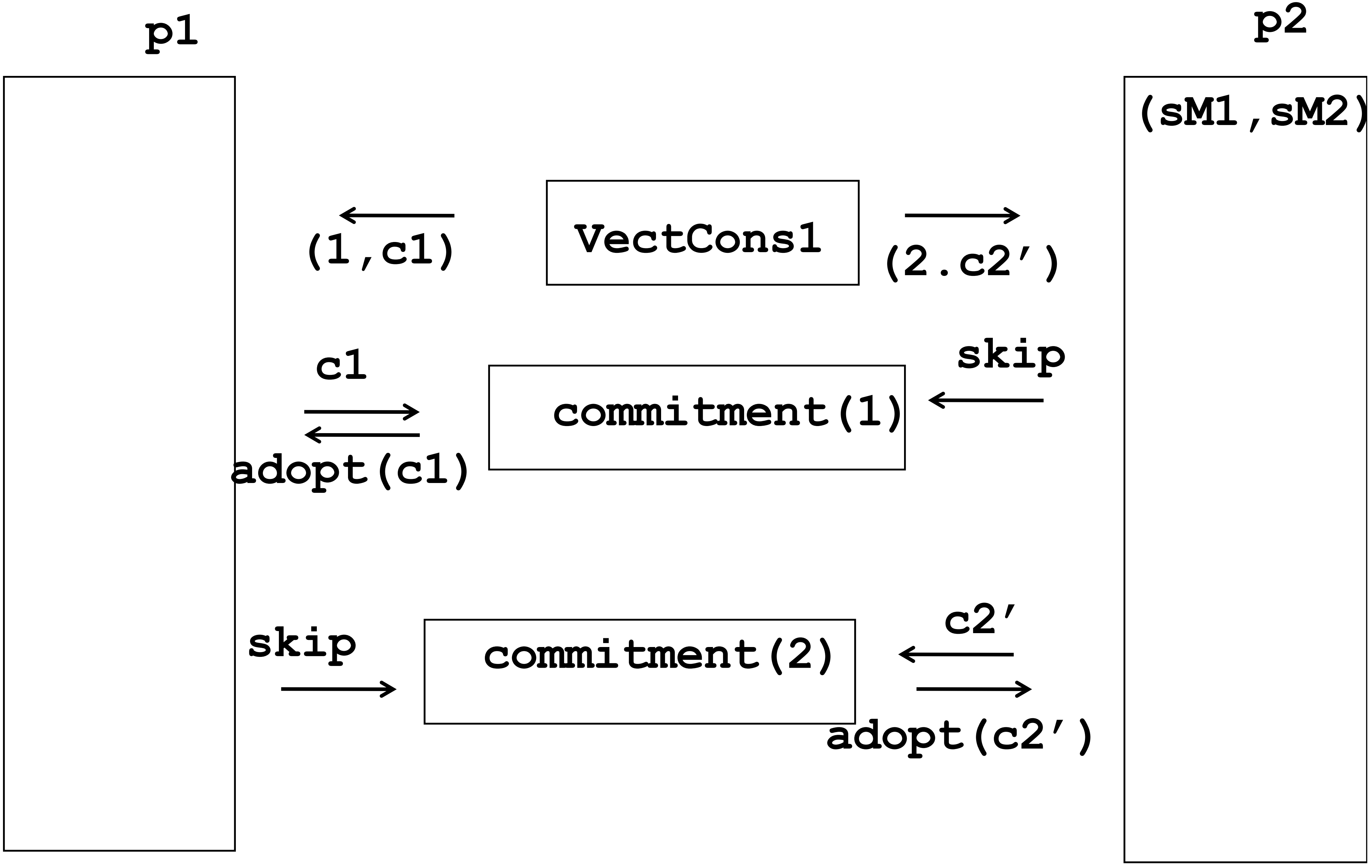
skip

c2′

commitment(2)

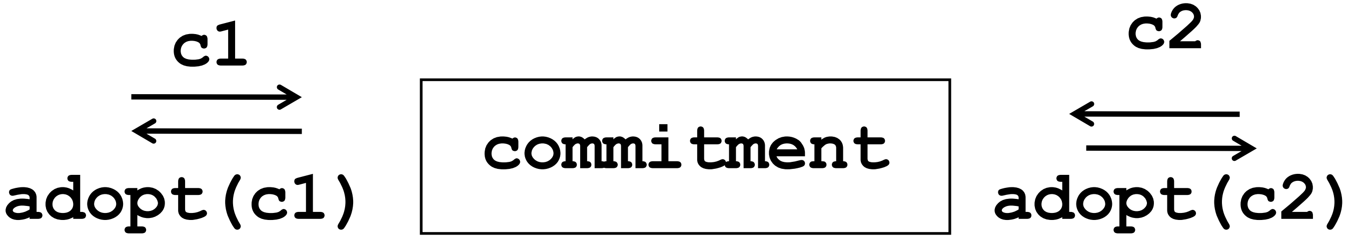
commit(c2′)

Problem with liveness



2nd key idea (ensuring liveness)

Exploit success first



`c1`

`adopt(c1)`

`commitment`

`c2`

`adopt(c2)`

Can it be that no command is committed? i.e., if every commitment box has one process proposes skip

Generalized universality (step 0)

- newCom = commands.next()

- while(true)

-   kVectC = kVectCons.next()

Generalized universality (step 1)

- …
- (c,i) = kVectC.propose(newCom)
- …

Generalized universality (step1-2)

- …
- (c,i) = kVectC.propose(newCom)
- vect(i) = commitment(i,c)
- …

Generalized universality (step1-2-2’)

- …

- (c,i) = kVectC.propose(newCom)

- vect(i) = commitment(i,c)

- for j = 1 to k except i:
    - vect(j) = commitment(j,newCom(j))

  …

Generalized universality (step 3)

```
…
for i = 1 to k
    if  ok(vect(i)) then
        sM(i).perform(vect(i))
        newCom(i) = commands(i).next()
    else
        newCom(i) = vect(i)
```

Generalized universality (step 3’)

```
…
for i = 1 to k
▪ If older(newCom(i),vect(i)) then
                          sM(i).perform(newCom(i))
▪ If no(vect(i)) then newCom(i) = vect(i)
▪ else
▪ sM(i).perform(vect(i))
▪ If  vect(i) = newCom(i) then
    ▪ newCom(i) = commands(i).next()
▪ add(newCom(i),vect(i))
```

Commitment

- ***Safety***: a process does not perform a command unless all others know the command

- ***Liveness***: at least one process executes a command in every round

NB. Every correct process executes at least one command every two rounds