Generalized Universality

E. Gafni, UCLA

R. Guerraoui, EPFL
Act 1
Classical Universality

Act 2
Modern Universality

Act 3
Generalized Modern Universality
Algorithm

A finite set of precise instructions

The only intelligence required is to compute the instructions

Must always produce a result
Which machine enables to compute everything?

Universality
Act 1
Classical Universality

Act 2
Modern Universality

Act 3
Generalized Universality
The Network is the Computer
Algorithm

A finite set of precise instructions

The only intelligence required is to compute the instructions

Must always produce a result

NB. Despite concurrency and failures
Which network enables to compute everything?

- Linearizable
- Highly-available
Universality of consensus [L-L-S-H-CT]

Message Passing?

Register?

Test&Set?

Consensus  C&S  Abcast
Consensus

Processes propose each a value and agree on one of those values

\[
\text{same-output} = \text{propose}(\text{input})
\]
Universal construction

A state machine of which each process holds a copy

A list of commands local to each process

A list of consensus objects shared by the processes
Universal construction

- while(true)

- c = commands.next()

- cons = Consensus.next()

- c' = cons.propose(c)

- sM.perform(c')
Act 1
Classical Universality

Act 2
Modern Universality

Act 3
Generalized Universality
Generalized Universality

Consensus is the particular case of k-consensus
K-consensus [C, AGK]

- Every process proposes a vector of $k$ values and returns a value at some position

- \[(\text{value}, \text{position}) = \text{propose}(\text{kVect})\]
K-consensus

- **Validity**: the value returned at some 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
What form of universality with K-consensus?

With consensus
Processes implement a highly-available state machine

With k-consensus
Processes implement k state machines of which at least one is highly-available

Generalized Universality
Act 1
Classical Universality

Act 2
Modern Universality

Act 3
Generalized Universality
Generalized universality

k state machines: each process holding a copy of each (sM(i))

k lists of commands local to each process

A list of k-vector consensus objects (kVectCons)

Reads and writes in shared memory
Universal construction

- while(true)
- \( c = \text{commands.next()} \)
- \( \text{cons} = \text{consensus.next()} \)
- \( c' = \text{cons.propose}(c) \)
- \( \text{sM.perform}(c') \)
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)
Generalized universality?

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

- (c,i) = kVectC.propose(com)
- read shared memory and update any missing c’
- sM(i).perform(c)
- write (c,i) in shared memory
Key idea

- **Safety (commitment)**: a process does not perform a command unless all others know the command.
Commitment

write (c) at level 1
let V1 be the set of values at level 1
if V1 has only c, write (commit, c) at level 2

let V2 be the set of values at level 2
if V2 has only (commit, c) then return(commit, c)
if V2 has some (commit, c’) then return(adopt, c’)
else return (adopt, c)
Commitment

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

- **Invariant (2):** if all processes propose the same command then the command is committed
Generalized universality (step 0)

- `newCom = commands.next()`
- `while(true)`
- `kVectC = kVectCons.next()`
Generalized universality (step 1)

- ...

- \((c, i) = \text{kVectC.propose(newCom)}\)

- ...
Generalized universality (step1-2)

- ...

- \((c,i) = \text{kVectC.propose(newCom)}\)

- \(\text{vect}(i) = \text{commitment}(i,c)\)

- ...

- ...
Generalized universality (step1-2-2’) 

\[ \text{(c,i)} = \text{kVectC.propose(newCom)} \]

\[ \text{vect(i)} = \text{commitment(i,c)} \]

\[ \text{for } j = 1 \text{ to } k \text{ except } i: \]
\[ \quad \text{vect}(j) = \text{commitment(newCom}(j)) \]

...
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($i$)
Key ideas

- **Safety (commitment)**: a process does not perform a command unless all others know the command.

- **Liveness (success first)**: at least one process executes a command in every round.
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))
3 Key ideas

- **Safety (commitment)**: a process does not perform a command unless all others know the command.

- **Liveness (success first)**: at least one process executes a command in every round.

- **Safety (old promises)**: a process might execute two commands at the same round.
Act 1
Classical Universality

Act 2
Modern Universality

Act 3
Generalized Universality
Generalized Universality

Consensus is the particular case of k-consensus

What if consensus is not available? Mistakes? Partitions?
Generalized Universality

With consensus
Processes implement a highly-available state machine

With k-consensus
Processes implement k state machines of which at least one is highly-available