Generalized Universality
Consensus

Processes propose each a value and agree on one

output = \textit{propose}(\text{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
Universal Construction

- while(true)
  - \( c = \text{commands.next()} \)
  - \( \text{cons} = \text{Consensus.next()} \)
  - \( c' = \text{cons}.\text{propose}(c) \)
  - \( \text{sM}.\text{perform}(c') \)
Universal Construction

\[ \begin{array}{c}
\text{p1} \\
\text{sM+c1} \\
\text{sM+c1'}
\end{array} \quad \begin{array}{c}
\xrightarrow{c1} \\
\xleftarrow{c1'}
\end{array} \quad \begin{array}{c}
\text{cons1} \\
\text{cons2} \\
... \\
\text{consK}
\end{array} \quad \begin{array}{c}
\xrightarrow{c2} \\
\xleftarrow{c1'}
\end{array} \quad \begin{array}{c}
\text{sM+c1} \\
\text{sM+c1'}
\end{array} \quad \text{p2} \]
What if consensus is not available
[FLP, CHT, DFG]
K-Consensus

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

\[(i, c) = \text{propose}(k\text{Vect})\]
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)

For any distributed computing task T, there is a k such that T $\Leftrightarrow$ 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

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

\[ p_1 \quad \xrightarrow{c_1} \quad \text{cons1} \quad \xleftarrow{c_1} \quad \text{sM} \quad \xrightarrow{c_1'} \quad \text{cons2} \quad \xleftarrow{c_1'} \quad sM \]

\[ ... \quad \xrightarrow{c_1} \quad \text{consK} \quad \xleftarrow{c_1} \quad ... \]

\[ p_2 \]
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

\[ (sM_1, sM_2) \]

\[ sM_1 + c_1 \]

\[ sM_2 + c_2 \]

\[ \frac{\text{VectCons}_1}{\text{VectCons}_2} \]

\[ (c_1', c_2') \]

\[ (sM_1, sM_2) \]

\[ sM_2 + c_2' \]
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

\[(sM_1, sM_2)\]

\[sM_1 + c_1\]

\[c_1\]

\[(d_1, c_2)\]

\[(1, c_1)\]

\[\text{Share}\]

\[\text{VectCons}_1\]

\[\text{VectCons}_2\]

\[\text{Share}\]

\[(2, c_2)\]

\[(c_1', c_2')\]

\[sM_2 + c_2'\]
1st key idea (ensuring safety)

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

\[(sM_1, sM_2)\]

\[(1, c_1)\]

commit\((c_1)\)

\[sM_1 + c_1\]

commit\((1)\)

commit\((2, c_2')\)

commit\((c_2')\)
Problem with liveness

(1, c1) \rightarrow VectCons1 \rightarrow (2, c2')

c1 \leftrightarrow c1

commitment(1) \leftrightarrow skip

skip \rightarrow commitment(2) \rightarrow c2'

adopt(c1) \leftrightarrow adopt(c2')
2nd key idea (ensuring liveness)

*Exploit success first*

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)

- ... 
- $(c, i) = \text{kVectC}.propose(\text{newCom})$
- ...
Generalized universality (step1-2)

- ... 

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

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

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

- ... 

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

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

- for \(j = 1\) to \(k\) except \(i\):
  - \(\text{vect}(j) = \text{commitment}(j, \text{newCom}(j))\)
  - ...

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