Synchrony Weakened by Message Adversaries vs Asynchrony Restricted by Failure Detectors

Michel RAYNAL[⋆,†] **Julien STAINER**[†]

[⋆] Institut Universitaire de France

[†] IRISA, Université de Rennes, France

Table of content

Buzzwords or fundamental concepts??

Synchronous system, Failure, process crash, Lossy link, Asynchronous system, Distributed oracle, Reliable broadcast, Message adversary, Survivor set, Quorum, FLP, Consensus,Safety, Failure detector, Eventual leader, Total order broadcast, Core set, Weakest failure detector, Uniform property, Message pattern, Eventual synchrony, Recurrent link, Quiescent communication, Indulgent algorithm, Assumption coverage, Progress condition, DC Problem, Graceful degradation, Source, Dynamic system, Solo execution, t-Resilience, Iterated model, Wait-freedom,

..., etc., ...

A jungle? Some unity? jewels inside?

Is it possible to add “some order” to better understand?

RETURNING to The BASICS

From SEQUENTIAL COMPUTING

to DISTRIBUTED COMPUTING

Sequential computing

- Basic computation unit: function $f(x)$

$$x \longrightarrow \boxed{\quad f \quad} \longrightarrow y = f(x)$$

- Hierarchy: FSA $\subset$ Pushdown automata $\subset$ Turing machines
- Equivalences (examples):
  - $\star$ Regular languages $\simeq$ FSA $\simeq$ ND-FSA
  - $\star$ Turing machines $\simeq$ Lambda calculus $\simeq$ Post’s system

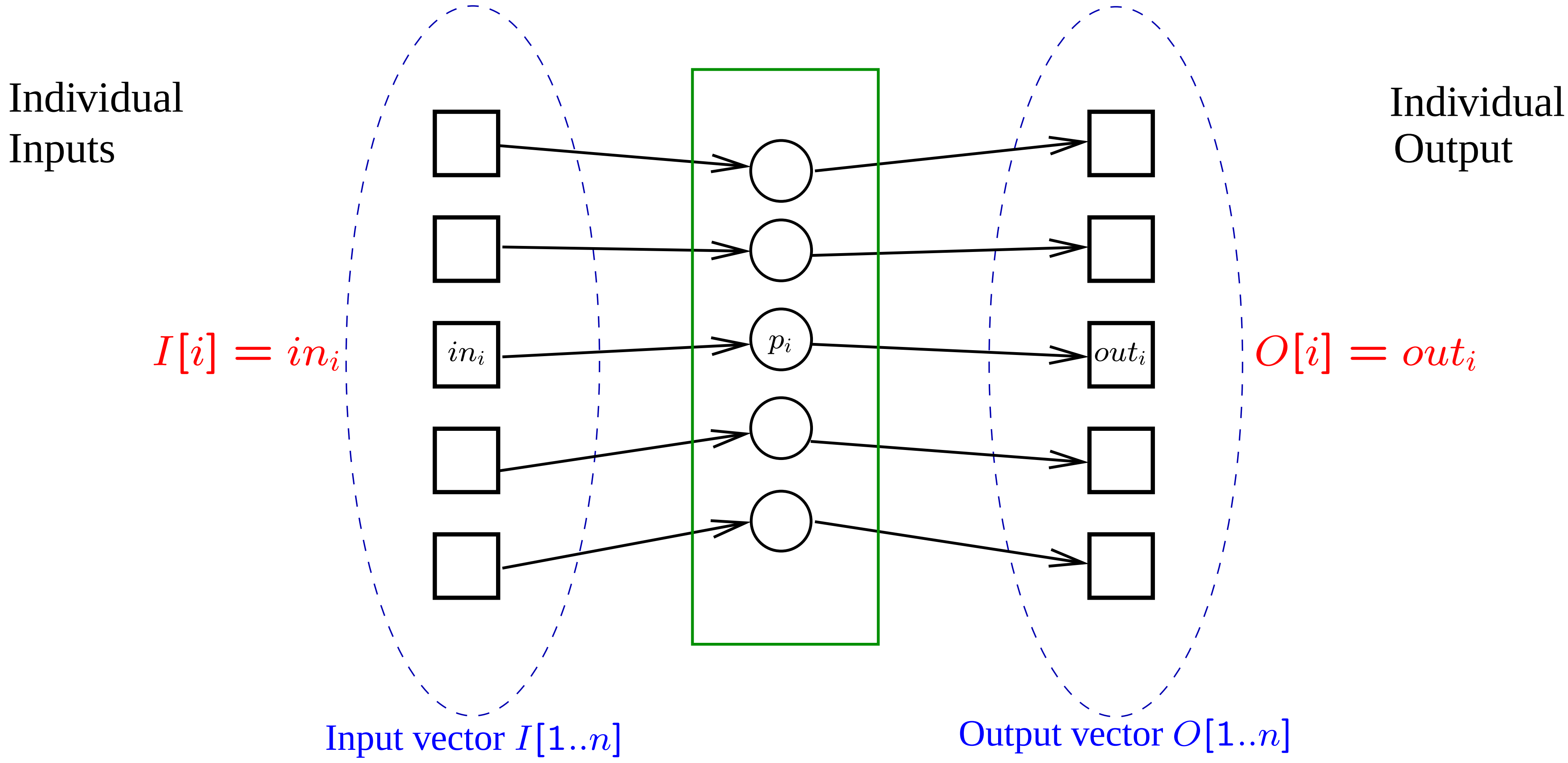
The world of distributed systems

- Time: Synchronous vs asynchronous systems
- Communication: shared memory vs message-passing
- Evolution: Static vs dynamic
- Failures
  - ⋆ What is concerned: process, link, or both
  - ⋆ Types of failures (crash, crash/recovery, omission, arbitrary)

This generates a multiplicity of DC models

Basic computation unit in DC: The notion of a task

The DC counterpart of a function



Formal definition

- A decision task T is a triple $(\mathcal{I}, \mathcal{O}, \Delta)$
  - $\star$ $\mathcal{I}$: set of input vectors (of size n)
  - $\star$ $\mathcal{O}$: set of output vectors (of size n)
  - $\star$ Δ: relation from $\mathcal{I}$ into $\mathcal{O}$: $\forall I \in \mathcal{I} : \Delta(I) \subseteq \mathcal{O}$
- $I[i]$: private input of p_i
- $O[i]$: private output of p_i
- $\forall I \in \mathcal{I}$:

  $\Delta(I)$ defines the set of output vectors that can be decided from the input vector I

Solving a task

A distributed algorithm A is a set of n local automata (Turing machines) that cooperate through specific communication objects (e.g., message-passing network, shared memory, etc.)

The set of automata is fixed (not a dynamic system with churn, etc.)

An algorithm A solves a task T if in any run

- $\forall \ I \in \mathcal{I}$ such that each p_i starts with (proposes) $in_i = I[i]$
- $\exists \ O \in \Delta(I)$ such that $out_j = O[j]$ for each process p_j that that computes (decides) an output out_j

- Consensus and k-set agreement
  - ⋆ Binary consensus:

    $\mathcal{I} = \{$all vectors of 0 and 1$\}$

    $\mathcal{O} = \big\{\{0,\ldots,0\},\{1,\ldots,1\}\big\}$

    Let $X_0 = \{0,\ldots,0\}$ and $X_0 = \{1,\ldots,1\}$

    Δ(any vector but $X_O, X_1) = \mathcal{O}$
    $\Delta(X_0) = \{0,\ldots,0\}$ and $\Delta(X_1) = \{1,\ldots,1\}$.

- Renaming, Weak symmetry breaking
- k-Simultaneous consensus, Etc.

Type of a task

- Colorless: In any run, the input (output) value of a process can be the input (output) of any other process
  - ⋆ Example: consensus, k-set agreement
- Colored: symmetry breaking tasks
  - ⋆ Example: Renaming problem, Weak symmetry breaking

AIM of the PAPER

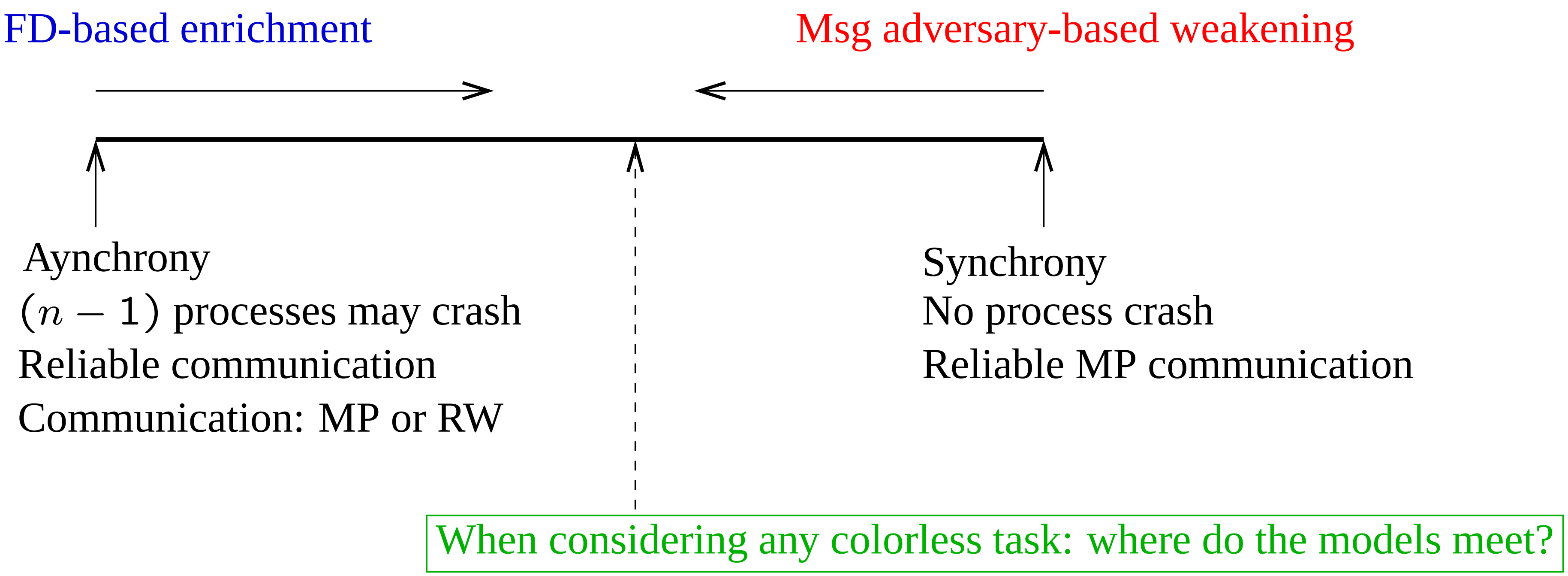
- A lot of papers:

  have introduced new models and investigated which pbs can be solved in each of these models

- This paper:

  does not introduce new DC model, but establish a hierarchy and equivalences between existing models

the basic figure



On the side of asynchronous models

the paper considers the following models

Asynchronous models

- n asynchronous processes
- up to $(n-1)$ processes may crash
- Communication: reliable and
  - ⋆ Asynchronous msg-passing, point-to-point complete network
  - ⋆ Or Read/Write shared memory
- notation:
  - ⋆ MP: $\mathcal{AMP}_{n,n-1}[fd : \emptyset]$ vs $\mathcal{AMP}_{n,n-1}[fd : \text{FD}]$
  - ⋆ RW: $\mathcal{ARW}_{n,n-1}[fd : \emptyset]$ vs $\mathcal{ARW}_{n,n-1}[fd : \text{FD}]$

Eventual leader failure detector Ω

- Let $\mathcal{C}$ = the set of non-faulty processes
- Each process p_i has a read-only local variable $leader_i$ such that
  - ⋆ $leader_i$ always contains a process identity (validity), and
  - ⋆ there is an unknown but finite time τ and a process identity $\ell \in \mathcal{C}$ such that $\forall \tau' \geq \tau : (i \in \mathcal{C}) \Rightarrow (leader_i^{\tau'} = \ell)$ (eventual convergence)
- Notation: $\mathcal{AMP}_{n,n-1}[fd : \Omega]$ and $\mathcal{ARW}_{n,n-1}[fd : \Omega]$

- Chandra T., Hadzilacos V. and Toueg S., The weakest failure detector for solving consensus. *Journal of the ACM*, 43(4):685-722, 1996

Quorum failure detector Σ

- Each process p_i has a read-only local variable qr_i such that
  - ⋆ qr_i always contains a non-$\emptyset$ set of process identities (validity)
  - ⋆ $\forall \tau, \tau', \forall i, j$: $qr_i^\tau \cap qr_j^{\tau'} \neq \emptyset$ (intersection property)
  - ⋆ $\forall i \in \mathcal{C} : \exists \tau : \forall \tau' \geq \tau : qr_i^{\tau'} \subseteq \mathcal{C}$ (liveness property)
- Notation: $\mathcal{AMP}_{n,n-1}[fd : \Sigma]$

- Delporte-Gallet C., Fauconnier H., and Guerraoui R., Tight failure detection bounds on atomic object implementations. *Journal of the ACM*, 57(4), Article 22, 2010

Asynchronous shared memory models

- Basic model: $\mathcal{ARW}_{n,n-1}[fd : \emptyset]$
  - $\star$ n asynchronous processes
  - $\star$ up to $(n-1)$ may crash
  - $\star$ communication through atomic read/write registers
- Enrched model $\mathcal{ARW}_{n,n-1}[fd : \Omega]$

On the side of synchronous models

the paper considers the following models

Basic reliable synchronous model

- n processes
- no process failure
- Synchronous msg-passing, point-to-point complete network
- Round-based computation:
  - $\star$ at every round, each process sends a msg to all
  - $\star$ $\forall$ msg: received in the very same round in which it is sent
- Notation $\mathcal{SMP}_n[adv : \emptyset]$
- Remark: due to synchrony assumption, the progress condition in this model is inherently wait-freedom

The notion of a message adversary

- Power of an adversary:

  at any round the adversary can suppress messages

- Weakening the power of an adversary:

  The power of an adversary can be restricted by imposing constraints (properties) on it behavior

  - ⋆ at one extreme it is not allowed to suppress messages,
  - ⋆ at the other extreme it is allowed to suppress all messages at every round
  - ⋆ and in between: it exists plenty of adversaries!

The T-connectivity adversary

- T-interval connectivity: for any T consecutive rounds there a connected subgraph on which the adversary does not suppress messages
- $T = 1$: the minimal communication graph left by the adversary at every round is connected (it is consequently a spanning tree) but it change arbitrarily at every round
- notation: $\mathcal{SMP}_n[adv$:T-$connectivity]$

  Any computable function can be computed in this synchr model

- Kuhn F., Lynch N.A., and Oshman R., Distributed computation in dynamic networks. *Proc. 42nd ACM Symposium on Theory of Computing (STOC'10)*, ACM press, pp. 513-522, 2010

Afek-Gafni's message adversaries

- TOUR (tournament): at every round, the adversary can suppress one message on each link but not both
- PAIRS: (1) At each round, the adversary can suppress all messages except one message, and (2) on k consecutive rounds (e.g., $k = \frac{n(n-1)}{2}$) each link is selected for the non-suppression
- TP: At each round, there is a directed path connecting all processes on which messages are not suppressed
- $\mathcal{SMP}_n[adv : \text{TOUR}]$, $\mathcal{SMP}_n[adv : \text{PAIRS}]$, $\mathcal{SMP}_n[adv : \text{TP}]$ have the same computability power for task solvability

- Afek Y. and Gafni E., Asynchrony from synchrony. *Proc. Int'l Conference on Distributed Computing and Networking (ICDCN'13)*, Springer LNCS 7730, pp. 225-239, 2013.

CONTENT of the PAPER

Model equivalences?

- Distributed computing models:
  - $\star$ Asynchrony (RW or MP), process crashes, reliable communication, possibly enriched with failure detectors
  - $\star$ Synchrony (MP), reliable processes, message losses (adversaries)
- Afek-Gafni 2013: $\boxed{\mathcal{SMP}_n[adv : \text{TOUR}] \simeq_T \mathcal{ARW}_{n,n-1}[fd : \emptyset]}$
- More generally: How all these models are related??

Introduction of two new adversaries

- SOURCE:
  There are a process p_s and a round r_0, such that, at every round $r \geq r_0$, the adversary does not suppress the message sent by p_s to the other processes

- QUORUM: Given any pair of processes p_i and p_j:
  - $\star$ whatever the synchronous rounds r_i and r_j executed by p_i and p_j, there is a process p_k whose messages to p_i at round r_i and to p_j at round r_j are not eliminated by the adversary (intersection property)
  - $\star$ there is at least one process whose messages are infinitely often received by each other process (liveness property)

Content of the paper: Hierarchy and equivalences

$\mathcal{SMP}_n[adv:\emptyset] \simeq_M \mathcal{AMP}_{n,0}[fd:\emptyset] \simeq_M \mathcal{ARW}_{n,0}[fd:\emptyset]$

Section 6

$\mathcal{SMP}_n[adv:\text{SOURCE, QUORUM}] \simeq_T \mathcal{AMP}_{n,n-1}[fd:\Sigma,\ \Omega]$

$\mathcal{SMP}_n[adv:\text{QUORUM}] \simeq_T \mathcal{AMP}_{n,n-1}[fd:\Sigma]$

Section 5

$\mathcal{SMP}_n[adv:\text{SOURCE, TOUR}] \simeq_T \mathcal{ARW}_{n,n-1}[fd:\Omega]$

Section 3

$\mathcal{SMP}_n[adv:\text{TOUR}] \simeq_T \mathcal{ARW}_{n,n-1}[fd:\emptyset]$

Afek-Gafni

$\mathcal{SMP}_n[adv:\text{SOURCE}] \simeq_T \mathcal{AMP}_{n,n-1}[fd:\Omega]$

Section 4

$\mathcal{SMP}_n[adv:\infty] \simeq_T \mathcal{AMP}_{n,n-1}[fd:\emptyset]$

Section 2

Content of the paper

- The paper contains plenty reductions
- A few are easy, the others are not!
- On model-dependent notions
  - $\star$ notion of non-faulty process in $\mathcal{AMP}_{n,n-1}[fd : \text{FD}]$ vs
  - $\star$ notion of terminating process in $\mathcal{SMP}_n[adv : \text{AD}]$

A remark on consensus

- The weakest FD to solve consensus in $\mathcal{ARW}_{n,n-1}[fd : \emptyset]$ is Ω
- we have (this equivalence is for colorless tasks)

$$\mathcal{SMP}_n[adv : \text{SOURCE, TOUR}] \simeq_T \mathcal{ARW}_{n,n-1}[fd : \Omega]$$

- But, this does not allow us to conclude that the adversary defined by the constraints SOURCE + TOUR is the weakest adversary to solve consensus in $\mathcal{SMP}_n[adv : \emptyset]$

A FEW REDUCTIONS

Reductions involving QUORUM

- QUORUM: Given any pair of processes p_i and p_j:
  - $\star$ whatever the synchronous rounds r_i and r_j executed by p_i and p_j, there is a process p_k whose messages to p_i at round r_i and to p_j at round r_j are not eliminated by the adversary (intersection property)
  - $\star$ there is at least one process whose messages are infinitely often received by each other process (liveness property)

- $$\left[\forall\, i, j : \forall\, r_i, r_j : (\{k : k \xrightarrow{r_i} i\} \cap \{k : k \xrightarrow{r_j} j\} \neq \emptyset)\right] \\ \wedge\, (\mathcal{SC} \neq \emptyset)$$

- QUORUM in $\mathcal{SMP}_n[adv : \emptyset]$captures Σ in $\mathcal{AMP}_{n,n-1}[fd : \emptyset]$

From $\mathcal{AMP}_{n,n-1}[fd : \Sigma]$ to $\mathcal{SMP}_n[adv : \text{QUORUM}]$ (1)

initialization

$r_i \leftarrow 0$;
$sim\_rec\_msgs_i[1,\ldots,n] \leftarrow [\perp,\ldots,\perp]$;
$(msgs\_to\_send_i[1,\ldots,n], ls\_state_i)$
$\leftarrow$ simulate$(sim_rec_msgs_i)$;
for each $r > 0$
do $rec\_msgs_i[r][1,\ldots,n] \leftarrow [\perp,\ldots,\perp]$ **end for**.

when (r, m) **received from** p_j: $rec\_msgs_i[r][j] \leftarrow m$.

From $\mathcal{AMP}_{n,n-1}[fd : \Sigma]$ to $\mathcal{SMP}_n[adv : \text{QUORUM}]$ (2)

repeat forever

$r_i \leftarrow r_i + 1$;

for each $j \in \{1, \dots, n\}$

do $\text{send}(r_i, msgs\_to\_send_i[j])$ to p_j **end for**;

repeat $cur\_qr_i \leftarrow qr_i$

until $(\forall j \in cur\_qr_i \setminus \{i\} \ : \ rec\_msgs_i[r_i][j] \neq \perp)$

end repeat;

for each $j \in cur\_qr_i$

do $sim\_rec\_msgs_i[j] \leftarrow rec\_msgs_i[r_i][j]$ **end for**;

$(msgs\_to\_send_i[1, \dots, n], ls\_state_i)$

$\leftarrow \text{simulate}(sim\_rec\_msgs_i)$;

$sim\_rec\_msgs_i[1, \dots, n] \leftarrow [\perp, \dots, \perp]$

end repeat.

From $\mathcal{SMP}_n[adv : \text{QUORUM}]$ to $\mathcal{AMP}_{n,n-1}[fd : \Sigma]$ (1)

initialization

$ls\_state_i \leftarrow$ initial state of the local simulated algorithm;
$msgs\_to\_rec_i \leftarrow \emptyset$;
$msgs\_received_i \leftarrow \emptyset$;
$(msgs\_to\_send_i, ls\_state_i) \leftarrow \mathsf{simulate}(ls\_state_i, msgs\_to\_rec_i)$;
$rec\_from_i \leftarrow \{1, \ldots, n\}$;
$view_i \leftarrow msgs\_to\_send_i$.

$view_i$ = { messages that have been sent, to p_i's knowledge }

when qr_i **is read**: $\mathsf{return}(rec\_from_i)$.

From $\mathcal{SMP}_n[adv : \text{QUORUM}]$ to $\mathcal{AMP}_{n,n-1}[fd : \Sigma]$ (2)

round $r = 1, 2, \cdots$ **do**:

send$(i, view_i)$ to each other process;

$rec\_msgs_i \leftarrow$ set of pairs $(j, view\_j)$ received during this round;

$view_i \leftarrow view_i \cup \left(\cup_{(j,view\_j) \in rec\_msgs_i} view\_j\right)$;

$rec\_from_i \leftarrow \left\{j \in \{1, \ldots, n\} \ : \ \exists (j, view\_j) \in rec\_msgs_i\right\} \cup \{i\}$;

if $(msgs\_to\_send_i \in \cap_{(j,view\_j) \in rec\_msgs_i} view\_j)$ **then**

$msgs\_to\_rec_i \leftarrow msgs\_to\_rec_i \ \cup$

$\{(j, i, m) \ : \ (j, view\_j) \in rec\_msgs_i \wedge (j, i, m) \in view\_j\}$;

$(msgs\_to\_send_i, ls\_state_i) \leftarrow$

simulate$(ls_state_i, msgs_to_rec_i \setminus msgs_received_i)$;

$msgs\_received_i \leftarrow msgs\_to\_rec_i$;

$view_i \leftarrow view_i \cup msgs\_to\_send_i$

end if.

Theorem: Let T be a colorless task:

$$T \text{ can be solved in } \mathcal{SMP}_n[adv : \text{QUORUM}]$$
$$\Longleftrightarrow$$
$$T \text{ be solved in } \mathcal{AMP}_{n,n-1}[fd : \Sigma]$$

Simulators and simulated processes (1)

- Model $\mathcal{AMP}_{n,n-1}[fd:\emptyset]$: correct vs faulty processes
- Model $\mathcal{SMP}_n[adv:\emptyset]$: strongly vs weakly correct processes
  - ⋆ Strongly correct: a process whose an infinite number of messages are eventually received (directly or indiredctly) by the other processes
  - ⋆ Weakly correct: the other processes
- From $\mathcal{SMP}_n[adv:\text{QUORUM}]$ to $\mathcal{AMP}_{n,n-1}[fd:\Sigma]$
  - ⋆ Strongly correct simulator $\rightarrow$ correct simulated process
  - ⋆ Weakly correct simulator $\rightarrow$ faulty simulated process

Simulators and simulated processes (2)

- From $\mathcal{AMP}_{n,n-1}[fd : \emptyset]$ to $\mathcal{SMP}_n[adv : \emptyset]$

  ⋆ Faulty simulator $\longrightarrow$ weakly correct process

  ⋆ Correct simulator $\longrightarrow$

  ∗ strongly correct process or
  ∗ weakly correct process

  This depends on the reduction

From RW + Omega to SOURCE + TOUR (1)

From $\mathcal{ARW}_{n,n-1}[fd : \Omega]$ to $\mathcal{SMP}_n[adv$: SOURCE, TOUR$]$

- $r_i \leftarrow$ 0: simulated round number
- $ls\_state_i \leftarrow$ simulated initial state at p_i
- $msgs\_to\_send_i[1..n] \leftarrow$ initial msgs to send to each process
- $\forall r > 0 : MEM[i][r][1..n]$ init to $[\perp, ..., \perp]$

From RW + Omega to SOURCE + TOUR (2)

From $\mathcal{ARW}_{n,n-1}[fd : \Omega]$ to $\mathcal{SMP}_n[adv : \text{SOURCE, TOUR}]$

repeat forever

$r_i \leftarrow r_i + 1$;

repeat $leader\_val_i \leftarrow MEM[leader_i][r_i][i]$

until $(leader\_val_i \neq \perp) \vee (leader_i = i)$

end repeat;

$MEM[i][r_i] \leftarrow msgs\_to\_send_i$;

$rec\_msgs_i[1..n] \leftarrow MEM[1..n][r_i][i]$;

$(msgs\_to\_send_i, ls\_state_i) \leftarrow \text{simulate}(ls\_state_i, rec\_msgs_i)$

end repeat.

The other reductions

- They are (much) more involved
- See the paper
- Proofs: not always easy

To conclude : what do have we learn?

- Main result: A hierarchy and a set of non-trivial equivalences
- A question: Is it possible to discover a unifying model that would includes a lot of known specific DC models
- The ultimate goal (a DC's Holy Grail??)

What is the the Grand Unified Model of DC!

(similar to the "Grand Unified Theory" in Physics)