t-Resilient Immediate Snapshot and its Relation with Agreement Problems

Carole DELPORTE[†], Hugues FAUCONNIER[†]

Sergio RAJSBAUM[‡], Michel RAYNAL[⋆,◇]

[†]IRIF, Université Paris 7 Diderot, Paris, France

[‡]Instituto de Matemáticas, UNAM, México

[⋆]Institut Universitaire de France

[◇]IRISA, Université de Rennes, France

Summary

- Basic wait-free read/write model
- Immediate snapshot and iterated model
- t-Resilient k-Immediate Snapshot
- Impossibility results
- Relation with x-Set agreement
- Conclusion

The AIM is: understand (again and again ...)

Understand t-resilience and its impact on Immediate Snapshot and Agreement

i.e., Relations linking synchronization problems and agreement problems

Enrich the map of our understanding of distributed computability

Part 1

Basic wait-free model

Immediate snapshot object

and iterated model

Computing entities

- n asynchronous sequential processes $p_1, \ldots, p_n$
- Asynchrony = each process proceeds at its own speed, which can be arbitrary and remains always unknown to the other processes
- Up to t processes may crash, $1 \le t \le n-1$
  - ⋆ $t = n-1$: wait-free model
  - ⋆ $1 \le t < n-1$: t-crash model
- Terminology: given a run

  a process that crashes is $faulty$, otherwise it is $correct$

Communication and notations

- SWMR atomic registers: one $REG[i]$ per process p_i
  $REG[i]$: written by p_i, read by all
- $\mathcal{CARW}_{n,t}[t = n - 1]$: wait-free model
- $\mathcal{CARW}_{n,t}[1 \leq t < n - 1]$: t-crash model
- Capital letters: for shared objects
- Small letters: for local variables

k-Set agreement

One-shot object that provides the processes with a single operation denoted $\text{propose}_k()$, which returns/decides a value

Each process is assumed to propose a value

Specification:

- Termination: $\text{propose}_k()$ by a correct process terminates
- Validity: A decided value is a proposed value
- Agreement: At most k different values are decided

Consensus is 1-set agreement

Well-known computability results

- Consensus: impossible in $\mathcal{CARW}_{n,t}[t \geq 1]$
- k-Set agreement: impossible in $\mathcal{CARW}_{n,t}[t \geq k]$
- $(2n-1)$-Renaming: possible in $\mathcal{CARW}_{n,t}[t \leq n-1]$
- Immediate snapshot: possible in $\mathcal{CARW}_{n,t}[t = n-1]$

Immediate snapshot object

One-shot object that provides the processes with a single operation denoted write_snapshot(), which returns/decides a $view$ (set of pairs $\langle i, v_i \rangle$)

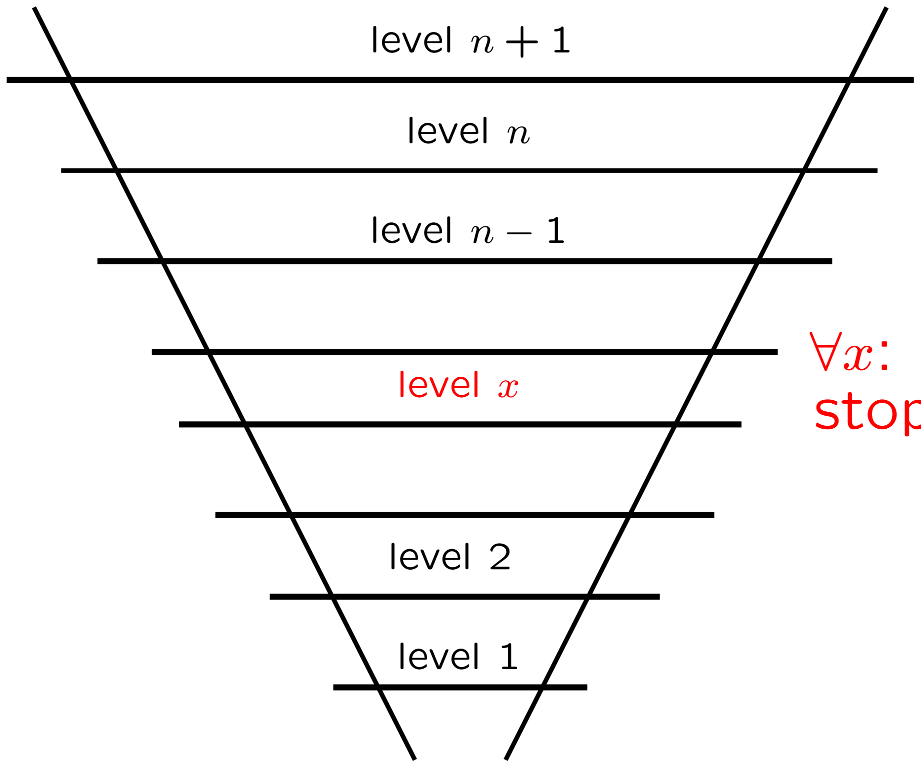
Specification:

- Termination: write_snapshot() by a correct terminates
- Self-inclusion: $\forall\ i :\ \langle i, v \rangle \in view_i$
- Validity: $\forall\ i : (\langle j, v \rangle \in view_i) \Rightarrow p_j$ invoked write_snapshot(v)
- Containment: $\forall\ i, j :\ (view_i \subseteq view_j) \vee (view_j \subseteq view_i)$
- Immediacy:
  $\forall\ i, j : (\langle i, v \rangle \in view_j) \wedge (\langle j, v' \rangle \in view_i) \ \Rightarrow\ (view_i = view_j)$

One-shot ~~immediate~~ Snapshot object

- A snapshot has two operations $\mathsf{write}(v)$ and $\mathsf{snapshot}()$
- Defined by
  Termination, Self-inclusion, Validity, and Containment
  ~~immediacy~~
- $\mathsf{write\_snapshot}(v)$ encapsulates $\mathsf{write}(v) \oplus \mathsf{snapshot}()$
- On atomicity:
  - ⋆ A snapshot object is atomic,
  - ⋆ An immediate snapshot object is not atomic
    Immediacy captures concurrent operations:
    "if I see you and you see me, we see the same"

Immediate snapshot in the wait-free model (BG'93)



$\forall x$: at most x processes stop at levels $y \leq x$

Immediate snapshot in the wait-free read/write model

$REG[1..n]$ init to $[\perp, .., \perp]$

$LEVEL[1..n]$ init to $[(n+1), .., (n+1)]$

operation write_snapshot(v_i) **is**
  $REG[i] \leftarrow v_i$;
  repeat
    $LEVEL[i] \leftarrow LEVEL[i] - 1$;
    for $j \in \{1, \ldots, n\}$ **do** $level_i[j] \leftarrow LEVEL[j]$ **end for**;
    $seen_i \leftarrow \{j : level_i[j] \leq level_i[i]\}$
  until $(|seen_i| \geq level_i[i])$ **end repeat**;
  $view_i \leftarrow \{\langle j, REG[j]\rangle$ such that $j \in seen_i\}$
  return($view_i$)
end operation.

The iterated immediate snapshot (wait-free) model

$KIS[1..)$ sequence of immediate snapshot objects

$KIS[r]$: object used at round r by the processes

Model: sequence of asynchronous rounds

$r_i \leftarrow 0$; $\ell s_i \leftarrow$ initial local state of p_i (including its input);
repeat forever % asynchronous IS-based rounds
  $r_i \leftarrow r_i + 1$;
  $view_i \leftarrow KIS[r_i]$.write_snapshot$(\ell s_i)$;
  $ls_i \leftarrow \delta(ls_i, view_i)$; % new local state
end repeat.

Power and limits of the IIS (wait-free) model

- Algorithmic foundation of distributed iterated models
  structured sequence of rounds

- Equivalent to the usual read/write wait-free model

  Borowsky E. and Gafni E., A simple algorithmically reasoned characterization of wait-free computations. *Proc. PODC'97*, pp. 189-198, 1997

- IIS enriched with a (non-trivial) failure detector FD is weaker than $\mathcal{CARW}_{n,t}[t = n - 1, FD]$

  Rajsbaum, S., Raynal, M., and Travers, C., An impossibility about failure detectors in the iterated immediate snapshot model. *IPL*, 108(3):160-164 2008

- Possible extension:

  Iterated Restricted Immediate Snapshot model

  Rajsbaum S., Raynal M., and Travers C., The iterated restricted immediate snapshot model. *Proc. 14th Annual Int'l Conference on Computing and Combinatorics*, Springer LNCS 5092, pp. 487-497, 2008

t-Resilient k-Immediate Snapshot

- The IIS model considers $t = n-1$ (wait-free)
- Consider a t-crash model: $1 \leq t < n-1$
  - ⋆ Define an associated immediate snapshot object
    Notion of k-immediate snapshot object (k-IS)
    which could be used in the t-crash iterated model
  - ⋆ Design algorithms for k-IS in $\mathcal{CARW}_{n,t}[1 \leq t < n-1]$
- In short: How to benefit from the fact that at least $n-t$ processes never crash when designing a k-IS object?

Definition: k-immediate snapshot object

It is an immediate snapshot object with a "natural" property on the size on the set of pairs obtained by a process

- Termination: write_snapshot() by a correct terminates
- Self-inclusion: $\forall\ i:\ \langle i, v\rangle \in view_i$
- Validity: $\forall\ i: (\langle j, v\rangle \in view_i) \Rightarrow p_j$ invoked write_snapshot(v)
- Containment: $\forall\ i, j:\ (view_i \subseteq view_j) \vee (view_j \subseteq view_i)$
- Immediacy: $\forall\ i, j: (\langle i, v\rangle \in view_j) \wedge (\langle j, v'\rangle \in view_i) \ \Rightarrow\ (view_i = view_j)$

- Output size: for any p_i: $|view_i| \geq n - k$

On the size of the returned set

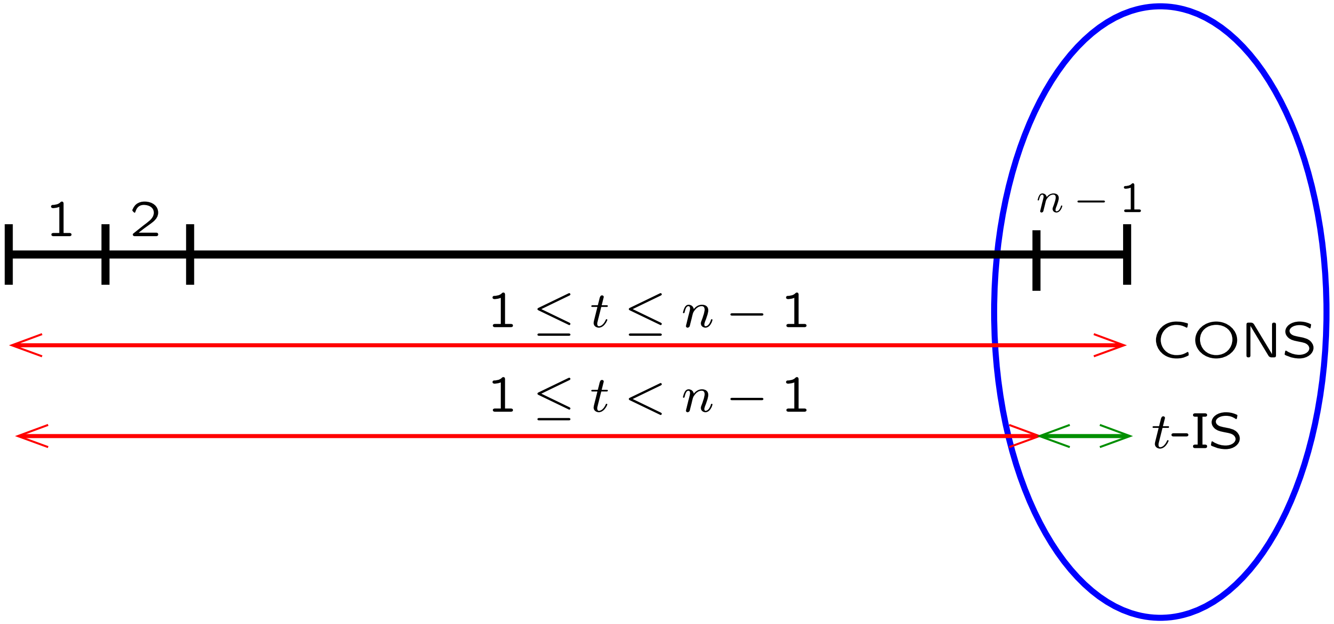
- Immediate snapshot object
  - ⋆ Any set $view$ is such that $|view| \geq 1$
  - ⋆ Can be implemented in $\mathcal{CARW}_{n,t}[t = n - 1]$ [BG93]
- k-immediate snapshot object
  - ⋆ Any set $view$ is such that $|view| \geq n - k$
    (more information obtained by a process)
  - ⋆ $(n - 1)$-IS object = basic immediate snapshot
  - ⋆ $\boxed{\text{Can } k\text{-IS be implemented when } t < n - 1?}$

It is impossible to implement t-resilient t-immediate snapshot in $\mathcal{CARW}_{n,t}[1 \leq t < n-1]$

t-Resilient immediate snapshot is impossible.
C. Delporte and H. Fauconnier, S. Rajsbaum, M. Raynal
Proc. 23nd Int'l Colloquium on Structural Information and Communication Complexity (SIROCCO'16), Springer LNCS 9988, pp. 177-191 (2016)

Two impossibility results

- Consensus impossibility in $\mathcal{CARW}_{n,t}[t \geq 1]$
- t-IS impossibility in $\mathcal{CARW}_{n,t}[t < n-1]$



Preliminary theorems

- A property associated with k-IS objects:

  Let $\ell \geq n-k$ be the size of the smallest view ($view$) obtained by a process.
  There is a set S of processes such that $|S| = \ell$ and each process of S obtains $view$ or crashes during its invocation of $\text{write\_snapshot}_k()$

- A simple impossibility associated with k-IS objects:

  k-IS cannot be implemented in $\mathcal{CARW}_{n,t}[k < t]$

- A stronger impossibility associated with k-IS objects:

  If $k < n-1$:
  k-IS cannot be implemented in $\mathcal{CARW}_{n,t}[1 \leq t < n]$

Relations between k-Immediate snapshot and x-set Agreement in $\mathcal{CARW}_{n,t}[t < n-1]$

Understand their relative impossibility

- Are all/some x-SA (resp. k-IS) objects "more impossible" than all/some k-IS (resp. x-SA)?
- Are all/some x-SA (resp. k-IS) objects "less impossible" than all/some k-IS (resp. x-SA)?
- Which is their cartography (possibility, impossibility, reductions)?
- Compare/rank impossibility classes
- Etc.

From k-IS to x-SA

System model: $\mathcal{CARW}_{n,t}[1 \leq t \leq k < n-1, k\text{-IS}]$, $n = 9$

From k-IS to x-SA with $x = \max(1, t + k - (n-2))$

$k \rightarrow$ $t \downarrow$	1 1	2 2	3 3	$< n/2$ 4	$n-4$ 5 $\geq n/2$	$n-3$ 6	$n-2$ 7	$n-1$ 8
1	1-SA	1-SA	1-SA	1-SA	1-SA	1-SA	1-SA	2-SA
2		1-SA	1-SA	1-SA	1-SA	1-SA	2-SA	3-SA
3			1-SA	1-SA	1-SA	2-SA	3-SA	4-SA
4 $\boxed{< n/2}$				1-SA	2-SA	3-SA	4-SA	5-SA
5 $\boxed{\geq n/2}$					3-SA	4-SA	5-SA	6-SA
6						5-SA	6-SA	7-SA
$7 = n-4$							7-SA	8-SA
$8 = n-1$								9-SA

From k-IS to x-SA: reduction algorithm

System model: $\mathcal{CARW}_{n,t}[1 \leq t \leq k < n-1, k\text{-IS}]$

$$\boxed{x = \max(1, k+t-(n-2))}$$

One k-IS object: KIS
An array of SWMR atomic registers: $VIEW[1..n]$ init $\perp$

operation propose$_x(v)$ **is**
$view_i \leftarrow KIS$.write_snapshot$_k(v)$;
$VIEW[i] \leftarrow view_i$;
wait($|\{\ j$ such that $VIEW[j] \neq \perp\}| = n-t$);
let $view$ **be** the smallest of the previous $(n-t)$ views;
return(smallest proposed value in $view$)
end operation.

From CONS (1-SA) to k-IS

System model: $\mathcal{CARW}_{n,t}[1 \leq t \leq k < n-1, \text{CONS}]$

$n = 9$ processes

$k \rightarrow$ $t \downarrow$	1 1	2 2	3 3	.. 4	.. 5	$n-3$ 6	$n-2$ 7	$n-1$ 8
1	1-IS	2-IS	3-IS	4-IS	5-IS	$(n-3)$-IS	$(n-2)$-IS	$(n-1)$-IS
2		2-IS	3-IS	4-IS	5-IS	$(n-3)$-IS	$(n-2)$-IS	$(n-1)$-IS
3			3-IS	4-IS	5-IS	$(n-3)$-IS	$(n-2)$-IS	$(n-1)$-IS
$4 < n/2$				4-IS	5-IS	$(n-3)$-IS	$(n-2)$-IS	$(n-1)$-IS
$5 \geq n/2$					5-IS	$(n-3)$-IS	$(n-2)$-IS	$(n-1)$-IS
$6 = n-3$						$(n-3)$-IS	$(n-2)$-IS	$(n-1)$-IS
$7 = n-2$							$(n-2)$-IS	$(n-1)$-IS
$8 = n-1$								$(n-1)$-IS

From CONS to k-IS (1)

System model: $\mathcal{CARW}_{n,t}[1 \leq t \leq k < n-1, \text{CONS}]$

- $REG[1..n]$: array of SWMR atomic registers
- $CONS[(n-t)..n]$: consensus objects (tolerating t crashes)
  - ⋆ Reduction of k-IS to CONS: based on an iteration
  - ⋆ Aim of iteration ℓ: obtain a view with $(n-t+\ell)$ pairs

From CONS to k-IS (2)

System model: $\mathcal{CARW}_{n,t}[1 \le t \le k < n-1, \text{CONS}]$

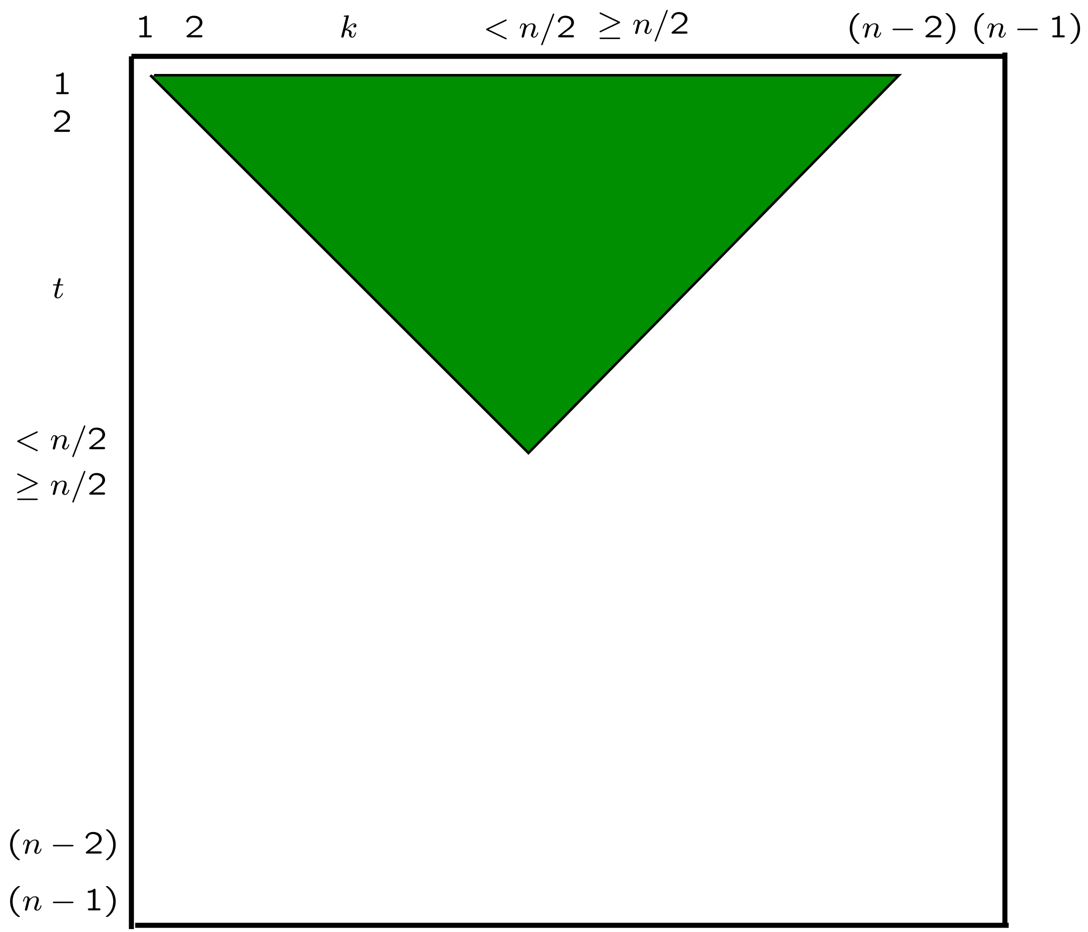
operation write_snapshot$_k(v)$ **is**
$REG[i] \leftarrow v$; $view_i \leftarrow \emptyset$; $dec_i \leftarrow \emptyset$; $\ell \leftarrow -1$; launch $T1$ and $T2$.

task $T1$ **is**
repeat $\ell \leftarrow \ell + 1$;
wait$\big(\exists$ a set aux_i: $(dec_i \subset aux_i) \wedge (|aux_i| = n - t + \ell)$
$\wedge\ (aux_i \subseteq \{\langle j, REG[j]\rangle$ such that $REG[j] \neq \bot\})\big)$;
$dec_i \leftarrow CONS[n-t+\ell]$.propose$_1(aux_i)$;
if $(\langle i, v_i\rangle \in dec_i) \wedge (view_i = \emptyset)$ **then** $view_i \leftarrow dec_i$ **end if**
until $(\ell = t)$ **end repeat**
end task $T1$.

task $T2$ **is** wait$(view_i \neq \emptyset)$; return$(view_i)$ **end task** $T2$.

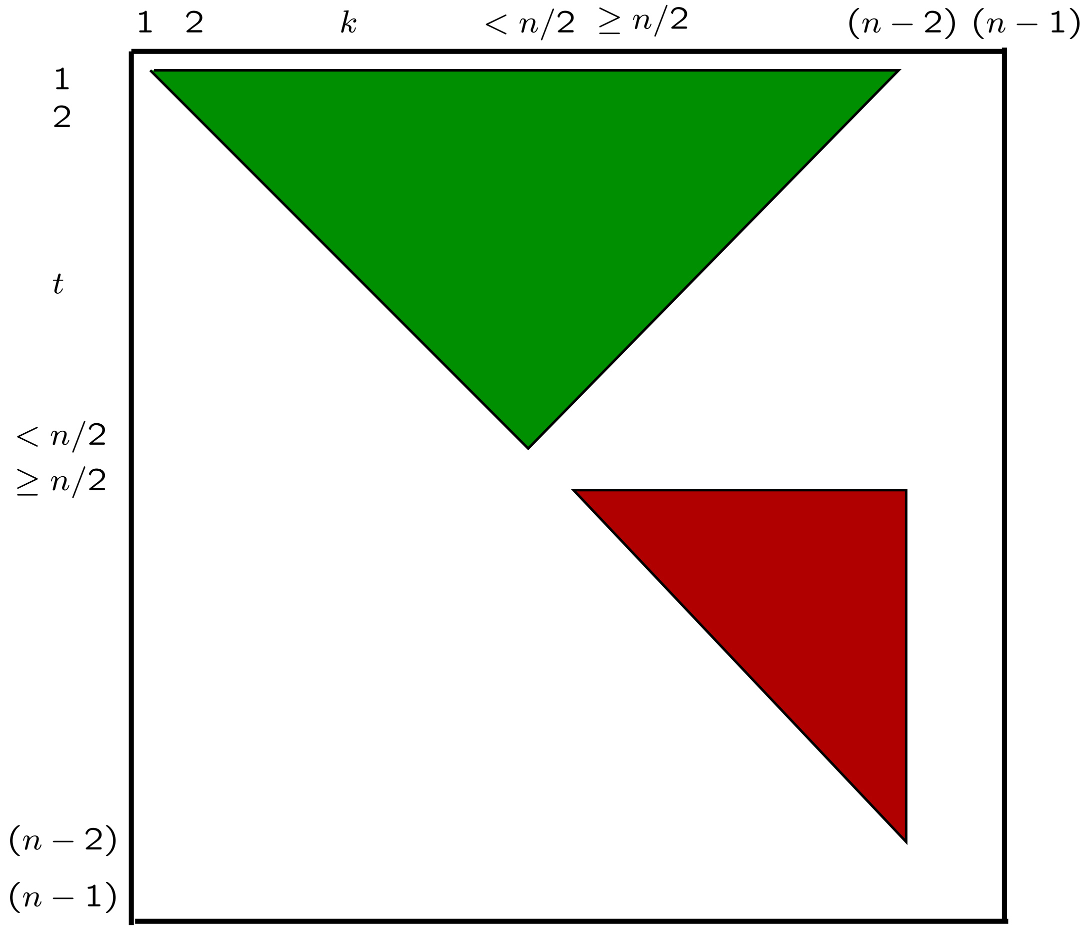
Combining the previous results

k-IS and CONS: equivalent when $(t < n/2) \wedge (t + k \leq n - 1)$



When CONS is stronger than k-IS

CONS stronger than k-IS when $(n/2 \leq t \leq k < n-1)$



CONCLUSION

What has been learned

- Impossibility of t-resilient k-immediate snapshot objects
- A model with less failures does not necessarily help!
  - ⋆ The assumption "at most $t < n - 1$ processes may crash" does not provide us with additional computational power to implement a t-IS object
  - ⋆ $\Rightarrow$ limits of the t-crash iterated model
- A computability map of objects impossible to implement in the wait-free read/write models
- Relations between agreement and synchronization

Open problems at the heart of DC computability

- Direction "from k-IS to x-SA"
  Is it possible to implement x-SA objects, with $1 \leq x < t + k - (n - 2)$, in t-crash n-process systems enriched with k-IS objects?

  Conjecture: the answer to this question is *no*

- Direction "from x-SA to k-IS"
  Which k-IS objects can be implemented from x-SA objects in a t-crash n-process read/write system?

  Conjecture: x-SA objects ($x > 1$) allows to build k-IS objects only for the pairs (t, k) satisfying $x \leq t+k-(n-2)$ (see the first table)