The Read/Write Protocol Complex Is Collapsible

Fernando Benavides[1,2] and Sergio Rajsbaum[1(✉)]

[1] Instituto de Matemáticas, Universidad Nacional Autónoma de México, Ciudad Universitaria, 04510 Mexico City, Mexico
rajsbaum@im.unam.mx

[2] Departamento de Matemáticas y Estadística, Universidad de Nariño, San Juan de Pasto, Colombia
fandresbenavides@gmail.com

Abstract. The celebrated *asynchronous computability theorem* provides a characterization of the class of decision tasks that can be solved in a wait-free manner by asynchronous processes that communicate by writing and taking atomic snapshots of a shared memory. Several variations of the model have been proposed (immediate snapshots and iterated immediate snapshots), all equivalent for wait-free solution of decision tasks, in spite of the fact that the protocol complexes that arise from the different models are structurally distinct. The topological and combinatorial properties of these snapshot protocol complexes have been studied in detail, providing explanations for why the asynchronous computability theorem holds in all the models.

In reality concurrent systems do not provide processes with snapshot operations. Instead, snapshots are implemented (by a wait-free protocol) using operations that write and read individual shared memory locations. Thus, read/write protocols are also computationally equivalent to snapshot protocols. However, the structure of the read/write protocol complex has not been studied. In this paper we show that the read/write iterated protocol complex is collapsible (and hence contractible). Furthermore, we show that a distributed protocol that wait-free implements atomic snapshots in effect is performing the collapses.

1 Introduction

A *decision task* is the distributed equivalent of a function, where each process knows only part of the input, and after communicating with the other processes, each process computes part of the output. For example, in the *k-set agreement* task processes have to agree on at most k of their input values; when $k = 1$ we get the *consensus* task [8].

A central concern in distributed computability is studying which tasks are solvable in a distributed computing model, as determined by the type of communication mechanism available and the reliability of the processes. Early on it was shown

Full version in arXiv 1512.05427. Partially supported by UNAM-PAPIIT grant IN107714.

E. Kranakis et al. (Eds.): LATIN 2016, LNCS 9644, pp. 179–191, 2016.
DOI: 10.1007/978-3-662-49529-2_14

that consensus is not solvable even if only one process can fail by crashing, when asynchronous processes communicate by message passing [8] or even by writing and reading a shared memory [22]. A graph theoretic characterization of the tasks solvable in the presence of at most one process failure appeared soon after [3].

The *asynchronous computability theorem* [15] exposed that moving from tolerating one process failure, to any number of process failures, yields a characterization of the class of decision tasks that can be solved in a wait-free manner by asynchronous processes based on simplicial complexes, which are higher dimensional versions of graphs. In particular, n-set agreement is not wait-free solvable, even for $n+1$ processes [4,15,24].

Computability theory through combinatorial topology has evolved to encompass arbitrary malicious failures, synchronous and partially synchronous processes, and various communication mechanisms [13]. Still, the original wait-free model of the asynchronous computability theorem, where crash-prone processes that communicate wait-free by writing and reading a shared memory is fundamental. For instance, the question of solvability in other models (e.g. f crash failures), can in many cases be reduced to the question of wait-free solvability [7,14].

More specifically, in the *AS model* of [13] each process can write its own location of the shared-memory, and it is able to read the whole shared memory in one atomic step, called a *snapshot*. The characterization is based on the *protocol complex,* which is a geometric representation of the various possible executions of a protocol. Simpler variations of this model have been considered. In the *immediate snapshot* (IS) version [2,4,24], processes can execute a combined write-snapshot operation. The *iterated immediate snapshot* (IIS) model [6] is even simpler to analyze, and can be extended (IRIS) to analyze partially synchronous models [23]. Processes communicate by accessing a sequence of shared arrays, through immediate snapshot operations, one such operation in each array. The success of the entire approach hinges on the fact that the topology of the protocol complex of a model determines critical information about the solvability of the task and, if solvable, about the complexity of solution [17].

All these snapshot models, AS, IS, IIS and IRIS can solve exactly the same set of tasks. However, the protocol complexes that arise from the different models are structurally distinct. The combinatorial topology properties of these complexes have been studied in detail, providing insights for why some tasks are solvable and others are not in a model.

Results. In reality concurrent systems do not provide processes with snapshot operations. Instead, snapshots are implemented (by a wait-free protocol) using operations that write and read individual shared memory locations [1]. Thus, read/write protocols are also computationally equivalent to snapshot protocols. However, the structure of the read/write protocol complex has not been studied. Our results are the following.

1. The one-round read/write protocol complex is collapsible to the IS protocol, i.e. to a chromatic subdivision of the input complex. The collapses can be performed simultaneously in entire orbits of the natural symmetric group

action. We use ideas from [21], together with distributed computing techniques of partial orders.
2. Furthermore, the distributed protocol that wait-free implements immediate snapshots of [5,9] in effect is performing the collapses.
3. Finally, also the multi-round iterated read/write complex is collapsible. We use ideas from [10], together with carrier maps e.g. [13].

All omitted proofs are in the full version in http://arxiv.org/abs/1512.05427

Related Work. The one-round immediate snapshot protocol complex is the simplest, with an elegant combinatorial representation; it is a chromatic subdivision of the input complex [13,19], and so is the (multi-round) IIS protocol [6]. The multi-round (single shared memory array) IS protocol complex is harder to analyze, combinatorially it can be shown to be a pseudomanifold [2]. IS and IIS protocols are homeomorphic to the input complex. An AS protocol complex is not generally homeomorphic to the underlying input complex, but it is homotopy equivalent to it [12]. The span of [15] provides an homotopy equivalence of the (multi-round) AS protocol complex to the input complex [12], clarifying the basis of the obstruction method [11] for detecting impossibility of solution of tasks.

Later on stronger results were proved, about the collapsibility of the protocol complex. The one-round IS protocol complex is collapsible [20] and homeomorphic to closed balls. The structure of the AS is more complicated, it was known to be contractible [12,13], and then shown to be collapsible (one-round) to the IS complex [21]. The IIS (multi-round) version was shown to be collapsible too [10].

There are several wait-free implementations of atomic snapshots starting with [1], but we are aware of only two algorithms that implement immediate snapshots; the original of [5], and its recursive version [9].

2 Preliminaries

2.1 Distributed Computing Model

The basic model we consider is the one-round *read/write* model (WR), e.g. [16]. It consists of $n+1$ processes denoted by the numbers $[n] = \{0, 1, \ldots, n\}$. A process is a deterministic (possibly infinite) state machine. Processes communicate through a shared memory array $\mathsf{mem}[0 \ldots n]$ which consists of $n+1$ single-writer/multi-reader atomic registers. Each process accesses the shared memory by invoking the atomic operations $\mathsf{write}(x)$ or $\mathsf{read}(j)$, $0 \leq j \leq n$. The $\mathsf{write}(x)$ operation is used by process i to write value x to register i, and process i can invoke $\mathsf{read}(j)$ to read register $\mathsf{mem}[j]$, for any $0 \leq j \leq n$. Each process i has an input value, which may be its own id i. In its first operation, process i writes its input to $\mathsf{mem}[i]$, then it reads each of the $n+1$ registers, in an arbitrary order. Such a sequence of operations, consisting of a write followed by all the reads is abbreviated by $\mathsf{WScan}(x)$.

An *execution* consists of an interleaving of the operations of the processes, and we assume any interleaving of the operations is a possible execution. We may also

consider an execution where only a subset of processes participate, consisting of an interleaving of the operations of those processes. These assumptions represent a wait-free model where any number of processes may fail by crashing.

In more detail, an execution is described as a set of atomic operations together with the irreflexive and transitive partial order given by: *op* precedes *op*′ if *op* was completed before *op*′. If *op* does not precede *op*′ and viceversa, the operations are called *concurrent*. The set of values read in an execution α by process i is called the *local view* of i which is denoted by $view(i, \alpha)$. It consists of pairs (j, v), indicating that the value v was read from the j-th register. The set of all local views in the execution α is called the *view* of α and it is denoted by $view(\alpha)$. Let $\mathcal{E}$ be a set of executions of the WR model. Consider the equivalence relation on $\mathcal{E}$ given by: $\alpha \sim \alpha'$ if $view(\alpha) = view(\alpha')$. Notice that for every execution α there exists an equivalent *sequential* execution α' with no concurrent operations. In other words, if *op* and *op*′ are concurrent operations in α we can suppose that *op* was executed immediately before *op*′ without modifying any views. Thus, we often consider only sequential executions α, consisting of a linear order on the set of all write and read operations.

Two other models can be derived from the WR model. In the *iterated* WR model, processes communicate through a sequence of arrays. They all go through the sequence of arrays mem_0, $\mathsf{mem}_1 \ldots$ in the same order, and in the r-th round, they access the r-th array, mem_r, exactly as in the one-round version of the WR model. Namely, process i executes one write to $\mathsf{mem}_r[i]$ and then reads one by one all entries j, $\mathsf{mem}_r[j]$, in arbitrary order. In the *non-iterated, multi-round* version of the WR model, there is only one array mem, but processes can execute several rounds of writing and then reading one by one the entries of the array. The *immediate snapshot* model (IS) [4,24], consists of a subset of executions of the WR one round model. Namely, all the executions where the operations can be organized in *concurrency classes*, each one consisting a set of writes by the set of processes participating in the concurrency class, followed by a read to all registers by each of these processes. See Sect. 3.1.

2.2 Algorithm IS

Consider the recursive algorithm IS of [9] for the iterated WR model, presented in Fig. 1. Processes go trough a series of disjoint shared memory arrays $\mathsf{mem}_0, \mathsf{mem}_1, \ldots, \mathsf{mem}_n$. Each array mem_k is accessed by process i invoking $\mathsf{WScan}(i)$ in the recursive call $\mathrm{IS}(n+1-k)$. Process i executes $\mathrm{WScan}(i)$ (line (1)), by performing first $\mathsf{write}(i)$, followed by $\mathsf{read}(j)$ for each $j \in [n]$, in an arbitrary order. The set of values read (each one with its location) is what the invocation of $\mathsf{WScan}(i)$ returns. In line (2) the process checks if *view* contains $n+1-k$ id's, else $\mathrm{IS}(n-k)$ is again invoked on the next shared memory in line (3). It is important to note that in each recursive call $\mathrm{IS}(n+1-k)$ at least one process returns with $|view| = n+1-k$, given that $n+1-k$ processes invoked IS. For example, in the first recursive call $\mathrm{IS}(n+1)$ the last process to write reads $n+1$ values and terminates the algorithm.

Algorithm IS$(n+1)$
(1) $view \leftarrow \mathsf{WScan}(i)$
(2) **if** $|view| = n+1$ **then** return $view$
(3) **else** return IS(n).

Fig. 1. Code for process i

Every execution of the IS protocol can be represented by a finite sequence $\alpha = \alpha_0, \alpha_1, \ldots, \alpha_l$ with α_k an execution of the WR one round model where every process that takes a step in α_k invokes the recursive call with IS$(n+1-k)$. Since at least one process terminates the algorithm the length $l(\alpha) = l+1$ is at most $n+1$. The last returned local view in execution α for process i is denoted $view(i, \alpha)$, and the set of all local views is denoted by $view(\alpha)$.

Denote by $\mathcal{E}_l$ the set of views of all executions α with $l(\alpha) = l+1$. Then $\mathcal{E}_n \subseteq \cdots \subseteq \mathcal{E}_0$. In particular, $\mathcal{E}_0$ corresponds to the views of executions of the one round WR of Sect. 2.1. Also, $\mathcal{E}_n$ corresponds to the views of the immediate snapshot model, see Theorem 1 of [9].

3 Definition and Properties of the Protocol Complex

Here we define the protocol complex of the write/read model and other models, which arise from the sets $\mathcal{E}_i$ mentioned in the previous section.

3.1 Additional Properties About Executions

Recall from Sect. 2.1 that an execution can be seen as a linear order on the set of write and read operations. For a subset $I \subseteq [n]$ let

$$\mathcal{O}_I = \{w_i, r_i(j) \ : \ i \in I, \ j \in [n]\}.$$

with $I = \mathcal{O}_i = \emptyset$. A *wr-execution on* I is a pair $\alpha = (\mathcal{O}_I, \rightarrow_\alpha)$ with $\rightarrow_\alpha$ a linear order on $\mathcal{O}_I$ such that $w_i \rightarrow_\alpha r_i(j)$ for all $j \in [n]$. The set I is called the $\mathtt{Id}$ set of α which is denoted by $\mathtt{Id}(\alpha)$. Hence the view of i is $view(i, \alpha) = \{j \in I \ : \ w_j \rightarrow_\alpha r_i(j)\}$ and the view of α is $view(\alpha) = \{(i, view(i, \alpha)) \ : \ i \in I\}$. Note the chain $w_i \rightarrow_\alpha r_i(j_0) \rightarrow_\alpha \cdots \rightarrow_\alpha r_i(j_n)$ represents the invoking of WScan by the process i in the *wr*-execution α. Consider a *wr*-execution α and suppose that the order in which the process i reads the array $\mathsf{mem}[0 \ldots n]$ is given by $r_i(j_0) \rightarrow_\alpha \cdots \rightarrow_\alpha r_i(j_n)$. If every write operation w_k satisfies $w_k \rightarrow_\alpha r_i(j_0)$ or $r_i(j_n) \rightarrow_\alpha w_k$ then $view(i, \alpha)$ corresponds to an atomic snapshot.

As a consequence, every execution in the snapshot model and immediate snapshot model corresponds to an execution in the write/read model. For instance in the *wr*-execution

$$\alpha : w_2 \rightarrow r_2(0) \rightarrow w_0 \rightarrow r_0(0) \rightarrow r_0(1) \rightarrow r_0(2) \rightarrow w_1 \rightarrow \\ \rightarrow r_1(0) \rightarrow r_2(1) \rightarrow r_1(1) \rightarrow r_2(2) \rightarrow r_1(2)$$

the $view(0,\alpha) = \{0,2\}$ and $view(1,\alpha) = [2]$ are immediate snapshots, this means the processes 0 and 2 could have read the array instantaneously. In contrast, the $view(2,\alpha) = \{1,2\}$ does not correspond to a snapshot. For the following consider the process j such that $w_i \rightarrow_\alpha w_j$ for all i.

Proposition 1. *Let α be a wr-execution on I. Then there exists $j \in I$ such that $view(j,\alpha) = I$.*

Let α be a *wr*-execution. For $0 \leq k \leq n$, define $\mathtt{Id}_k(\alpha) = \{j \in \mathtt{Id}(\alpha) : |view(j,\alpha)| = n+1-k\}$. An *IS-execution* is a finite sequence $\alpha = \alpha_0, \ldots, \alpha_l$ such that α_0 is a *wr*-execution on $[n]$, and α_{k+1} is a *wr*-execution on $\mathtt{Id}(\alpha_k) - \mathtt{Id}_k(\alpha_k)$. Given an *IS*-execution α, Proposition 1 implies $l(\alpha) \leq n+1$. Moreover $\mathtt{Id}(\alpha_{k+1}) \subseteq \mathtt{Id}(\alpha_k)$ for all $0 \leq k \leq l-1$. Hence $|\mathtt{Id}(\alpha_k)| \leq n+1-k$. Executions α, α' are *equivalent* if $view(\alpha) = view(\alpha')$, denoted $\alpha \sim \alpha'$.

Lemma 1. *Let α and α' be IS-executions with $l(\alpha) = l(\alpha)$. Given $0 \leq k \leq l$, (1) If $\alpha \sim \alpha'$ then $\mathtt{Id}(\alpha_k) = \mathtt{Id}(\alpha'_k)$. (2) If $\alpha_k \sim \alpha'_k$ then $\alpha \sim \alpha'$.*

According to the behavior of the protocol in Fig. 1, the local view of i is defined as $view(i,\alpha) = view(i,\alpha_k)$, if $i \in \mathtt{Id}(\alpha_k) - \mathtt{Id}(\alpha_{k+1})$ and $view(i,\alpha) = view(i,\alpha_l)$ for $k = l$. Hence the view of α is defined as $view(\alpha) = \{(i, view(i,\alpha)) : i \in [n]\}$.

Lemma 2. *Let $\alpha = \alpha_0, \ldots, \alpha_{l+1}$ be an IS-execution, $l(\alpha) = l+2$. Then $view(\alpha) = view(\alpha')$ for some IS-execution α' such that $l(\alpha') = l+1$.*

The *wr*-execution $\alpha' = \alpha_0, \ldots, \alpha_{l-1}, \alpha'_l$ of the lemma is obtained by, α'_l such that

$$view(i,\alpha'_l) = \begin{cases} view(i,\alpha_l), & \text{if } i \in \mathtt{Id}_l(\alpha_l) \\ view(i,\alpha_{l+1}), & \text{if } i \notin \mathtt{Id}_l(\alpha_l). \end{cases}$$

It follows $\mathcal{E}_l = \{view(\alpha) : \alpha = \alpha_0, \ldots, \alpha_l\}$. Thus, Lemma 2 implies $\mathcal{E}_{l+1} \subseteq \mathcal{E}_l$. For example consider the *IS*-execution $\alpha = \alpha_0, \alpha_1, \alpha_2$ where $\alpha_0 : w_0 \rightarrow r_0(0) \rightarrow r_0(1) \rightarrow r_0(2) \rightarrow w_1 \rightarrow r_1(0) \rightarrow r_1(1) \rightarrow r_1(2) \rightarrow w_2 \rightarrow r_2(0) \rightarrow r_2(1) \rightarrow r_2(2)$. $\alpha_1 : w_0 \rightarrow r_0(0) \rightarrow r_0(1) \rightarrow r_0(2) \rightarrow w_1 \rightarrow r_1(0) \rightarrow r_1(1) \rightarrow r_1(2)$. $\alpha_2 : w_0 \rightarrow r_0(0) \rightarrow r_0(1) \rightarrow r_0(2)$.
So $view(\alpha) = \{(0,\{0\}),(1,\{0,1\}),(2,\{0,1,2\})\} \in \mathcal{E}_2 \subseteq \mathcal{E}_1 \subseteq \mathcal{E}_0$, Figs. 2 and 3.

3.2 Topological Definitions

The following are standard technical definitions, see [18,21]. A (abstract) *simplicial complex* Δ on a finite set V is a collection of subsets of V such that for any $v \in V$, $\{v\} \in \Delta$, and if $\sigma \in \Delta$ and $\tau \subseteq \sigma$ then $\tau \in \Delta$. The elements of V are called *vertices* and the elements of Δ *simplices*. The *dimension* of a simplex σ is $\dim(\sigma) = |\sigma| - 1$. For instance the vertices are 0-simplices. For the purposes of this paper, we adopt the convention that the *void complex* $\Delta = \emptyset$ is a simplicial complex which is different from the *empty complex* $\Delta = \{\emptyset\}$. Given a positive integer n, Δ^n denotes the standard simplicial complex whose vertex set is $[n]$

and every subset of $[n]$ is a simplex. From now on we identify a complex Δ with its collection of subsets. For every simplex τ we denote by $I(\tau)$ the set of all simplices ρ, $\tau \subseteq \rho$. A simplex τ of Δ is called *free* if there exists a maximal simplex σ such that $\tau \subseteq \sigma$ and no other maximal simplex contains τ. The procedure of removing every simplex of $I(\tau)$ from Δ is called a *collapse.*

Let Δ and Γ be simplicial complexes, Δ *is collapsible to* Γ if there exists a sequence of collapses leading from Δ to Γ. The corresponding procedure is denoted by $\Delta \searrow \Gamma$. In particular, if the collapse is elementary with free simplex τ, it is denoted by $\Delta \searrow_\tau \Gamma$. If Γ is the void complex, Δ is *collapsible.* The next definition from [21] gives a procedure to collapse a simplicial complex, by collapsing simultaneously by entire orbits of the group action on the vertex set. Let Δ be a simplicial complex with a simplicial action of a finite group G. A simplex τ is called G-free if it is free and for all $g \in G$ such that $g(\tau) \neq \tau$, $I(\tau) \cap I(g(\tau)) = \emptyset$. If τ is G-free, the procedure of removing every simplex $\rho \in \bigcup_{g \in G} I(g(\tau))$ is called a G-collapse of Δ.

Since, if τ is G-free then $g(\tau)$ is free as well, the above definition guarantees that all collapses in the orbit of τ can be done in any order *i.e.* every G-collapse is a collapse. A simplicial complex Δ is *G-collapsible to* Γ if there exist a sequence of G-collapses leading from Δ to Γ, it is denoted by $\Delta \searrow_G \Gamma$. In similar way, if the G-collapse is elementary with G-free simplex τ, the notation $\Delta \searrow_{G(\tau)} \Gamma$ will be used. In the case Γ is the void complex, Δ is called G-collapsible. For instance consider a 2-simplex σ, τ a 1-face of σ and the action of $\mathbb{Z}_3$ over σ, then τ is free but not $\mathbb{Z}_3$-free.

3.3 Protocol Complex

Let n be a positive integer. The abstract simplicial complex $\mathsf{WR}_l(\Delta^n)$ with $0 \leq l \leq n$ consists of the set of vertices $V = \{(i, view_i) \ : \ i \in view_i \subseteq [n]\}$. A subset $\sigma \subseteq V$ forms a simplex if only if there exist an IS-execution α of length $l+1$ such that $\sigma \subseteq view(\alpha)$.

The complex $\mathsf{WR}_0(\Delta^n)$ is called the *protocol complex* of the write/read model and it will be denoted by $\mathsf{WR}(\Delta^n)$. Protocol complexes for the particular cases $n = 1$ and $n = 2$ are shown in Fig. 2. In [21] a combinatorial description of the protocol complex View^n associated to the snapshot model is given. There every simplex of View^n is represented as a $2 \times t$ matrix. Every simplex $\sigma \in \mathsf{WR}(\Delta^n)$ can be expressed as

$$\mathrm{W}(\sigma) = \begin{pmatrix} V_1 & \cdots & V_{t-1} & [n] \\ I_1 & \cdots & I_{t-1} & I_t \end{pmatrix}$$

where $I_i \cap I_j = \emptyset$ with $i \neq j$ and $I_i \subseteq V_j$ for all $i \leq j$. Moreover we can associate a matrix for every simplex in the complex $\mathsf{WR}_l(\Delta^n)$. This representation implies that $\chi(\Delta^n)$ and View^n are subcomplexes of $\mathsf{WR}(\Delta^n)$. Figure 3 shows the complex $\mathsf{WR}_l(\Delta^2)$. From now on we will write WR_l instead of $\mathsf{WR}_l(\Delta^n)$ unless we specify the standard complex. Lemma 2 implies that every maximal simplex of WR_{l+1} is a simplex of WR_l, which implies that WR_{l+1} is a subcomplex of WR_l. From now on σ will denote a simplex of WR_l. For $0 \leq k \leq l$ let $\sigma_k^< = \{(i, view_i) \in$

$\sigma : |view_i| < n+1-k\}$. In a similar way $\sigma_k^{=}$ and $\sigma_k = \sigma_k^{<} \cup \sigma_k^{=}$ are defined. Therefore, the set of processes in σ which participate in the $l+1$ call recursive of Algorithm 1 is partitioned in those which read $n+1-l$ processes and those which read less than $n+1-l$ processes in the $l+1$ layer shared memory. Let us define $\mathtt{I}_\sigma^{<} = \bigcup_{i \in \mathtt{Id}(\sigma_l^{<})} view_i$ and $\mathtt{I}_\sigma = \bigcup_{i \in \mathtt{Id}(\sigma_l)} view_i$.

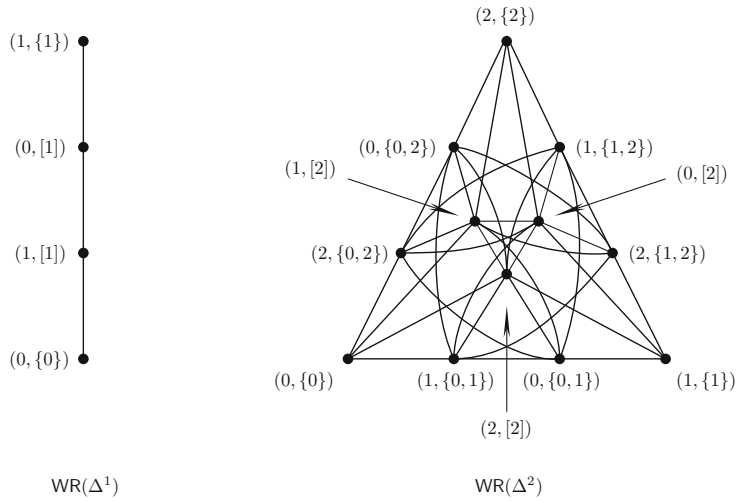

Fig. 2. Protocol complex for $n = 1$ and $n = 2$.

Theorem 1. *$\sigma \in \mathsf{WR}_{l+1}$ if only if $\mathtt{I}_\sigma^{<} \cap \mathtt{Id}(\sigma_l^{=}) = \emptyset$ and $|\mathtt{I}_\sigma^{<}| < n+1-l$.*

Proof. Suppose $\mathtt{I}_\sigma^{<} \cap \mathtt{Id}(\sigma_l^{=}) \neq \emptyset$ or $|\mathtt{I}_\sigma^{<}| = n+1-l$ and there exists an IS-execution $\alpha = \alpha_0, \ldots, \alpha_{l+1}$ such that $\sigma_l^{<} \subseteq view(\alpha_{l+1})$. Then there exist processes i and k such that $|view_i| < n+1-l$, $|view_k| = n+1-l$ and $k \in view_i$. This implies that k wrote in the $l+1$ shared memory, a contradiction. For the other direction, since $\mathtt{I}_\sigma^{<} \cap \mathtt{Id}(\sigma_l^{=}) = \emptyset$ and $|\mathtt{I}_\sigma^{<}| < n+1-l$ we can build an IS-execution $\alpha = \alpha_0, \ldots, \alpha_{l+1}$ such that $\sigma \subseteq view(\alpha)$. □

Notice that $\mathtt{I}_\sigma$ represents the set of processes which have been read in the $l+1$ recursive call of the algorithm in Fig. 1.

Corollary 1. *If $\sigma \notin \mathsf{WR}_{l+1}$ then*

1. $|\mathtt{I}_\sigma| = n+1-l$.
2. $\mathtt{I}_\sigma = \mathtt{I}_\tau$ *for all* $\sigma \subseteq \tau$.

Let $inv(\sigma) = \{(k, \mathtt{I}_\sigma) : k \in \mathtt{I}_\sigma \backslash \mathtt{I}_\sigma^{<}\}$ if $\mathtt{I}_\sigma^{<} \neq \mathtt{I}_\sigma$ else $inv(\sigma) = \{(k, \mathtt{I}_\sigma) : k \in \mathtt{I}_\sigma \backslash \mathtt{Id}(\sigma_l^{<})\}$. Notice that if $\sigma \notin \mathsf{WR}_{l+1}$ then $inv(\sigma) \neq \emptyset$.

For the simplices $\sigma^- = \sigma - inv(\sigma)$ and $\sigma^+ = \sigma \cup inv(\sigma)$.

Proposition 2. *If* $\sigma \notin \mathsf{WR}_{l+1}$ *then*

1. $\sigma^+ = \sigma^- \cup inv(\sigma)$.
2. $\sigma^- \subseteq \sigma \subseteq \sigma^+$.
3. $\sigma^- \notin \mathsf{WR}_{l+1}$.
4. *If* $\sigma^- \subseteq \tau \subseteq \sigma^+$ *then* $\sigma^- = \tau^-$ *and* $\sigma^+ = \tau^+$.
5. $(\sigma^-)^- = \sigma^-$.

Consider $\mathtt{I}^+_-(\sigma) = \{\tau \in \mathsf{WR}_l \ : \ \sigma^- \subseteq \tau \subseteq \sigma^+\}$. Item (3) above implies that $\mathtt{I}^+_-(\sigma) \cap \mathsf{WR}_{l+1} = \emptyset$ if $\sigma \notin \mathsf{WR}_{l+1}$. Moreover from (4) it is obtained that $\mathtt{I}^+_-(\sigma) \cap \mathtt{I}^+_-(\tau) = \emptyset$ or $\mathtt{I}^+_-(\sigma) = \mathtt{I}^+_-(\tau)$.

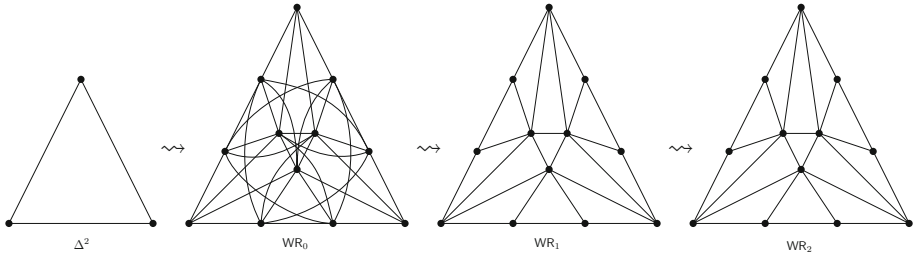

Fig. 3. Complexes WR_l.

4 Collapsibility

Let $S_{[n]}$ denote the permutation group of $[n]$. Notice that if the Id's of processes in a *wr*-execution on I are permuted according to $\pi \in S_{[n]}$ then we obtain a new linear order on $\pi(I)$. In other words if α is a *wr*-execution on I and $\pi \in S_{[n]}$ then $\alpha' = \pi(\alpha)$ is a *wr*-execution on $\pi(I)$. Moreover if $\sigma = view(\alpha)$ then $\pi(\sigma) = view(\pi(\alpha))$. This shows that there exists a natural group action on each simplicial complex WR_l.

Proposition 3. *Let* $\sigma \in \mathsf{WR}_l$ *be a simplex. Then*

1. $\pi(\sigma) \in \mathsf{WR}_l$.
2. $\pi(\sigma^-) = \pi(\sigma)^-$.
3. $\pi(\sigma^+) = \pi(\sigma)^+$.
4. $\pi(\mathtt{I}^+_-(\sigma)) = \mathtt{I}^+_-(\pi(\sigma))$.

For example in Fig. 2, $\sigma = \{(1,\{1,2\}),(2,\{0,2\}),(0,[2])\}$ and $\pi(0) = 1$, $\pi(1) = 2$ and $\pi(2) = 0$, then $\pi(\sigma) = \{(2,\{0,2\}),(0,\{0,1\}),(1,[2])\}$.

Theorem 2. *For every* $0 \leq l \leq n+1$,

1. WR_l *is collapsible to* WR_{l+1}.
2. WR_l *is* $S_{[n]}$*-collapsible to* WR_{l+1}.

Proof. Since $\sigma \in I_-^+(\sigma)$ for all simplices $\sigma \in \mathsf{WR}_l$, the intervals $I_-^+(\sigma)$ cover $L = \{\sigma \ : \ \sigma \in \mathsf{WR}_l,\ \sigma \notin \mathsf{WR}_{l+1}\}$. Also, Proposition 2 (4) implies that L can be decomposed as a disjoint union of intervals $I_-^+(\sigma_1), \ldots, I_-^+(\sigma_k)$ s.t. $\sigma_i = \sigma_i^+$ for all $1 \leq i \leq k$. Suppose $\dim(\sigma_{i+1})_l^< \leq \dim(\sigma_i)_l^<$ or if $\dim(\sigma_{i+1})_l^< = \dim(\sigma_i)_l^<$ then $\dim(\sigma_{i+1}) \leq \dim(\sigma_i)$. We will prove by induction on i, $1 \leq i \leq k$, that $\mathsf{WR}_l^i \searrow_{\sigma_i^-} \mathsf{WR}_l^{i+1}$ where $\mathsf{WR}_l^1 = \mathsf{WR}_l$ and $\mathsf{WR}_l^{k+1} = \mathsf{WR}_{l+1}$. If there exists a maximal simplex $\sigma \in \mathsf{WR}_l^i$ such that $\sigma_i \subseteq \sigma$ then $\sigma = \sigma_j$ for some $i \leq j \leq k$. Hence $(\sigma_i)_l^< \subseteq (\sigma_j)_l^<$ and therefore $\sigma_i = \sigma_j$. Now suppose there exists a maximal simplex $\sigma_j \in \mathsf{WR}_l^i$ with $i \leq j \leq k$ such that $\sigma_i^- \subseteq \sigma_j$. This implies that $(\sigma_i)_l^< = (\sigma_j)_l^<$ and $inv(\sigma_i) = inv(\sigma_j)$. Thus $\sigma_i = \sigma_i^- \cup inv(\sigma_i) \subseteq \sigma_j \cup inv(\sigma_j) = \sigma_j$ and therefore σ_i^- is free in WR_l^i. Therefore, $\mathsf{WR}_l = \mathsf{WR}_l^1 \searrow_{\sigma_1^-} \ldots \searrow_{\sigma_k^-} \mathsf{WR}_l^{k+1} = \mathsf{WR}_{l+1}$. Now if we specify in more detail the order of the sequence, the complex WR_l can be collapsed to WR_{l+1} in a $S_{[n]}$-equivariant way. First note that if $\pi(\sigma_i) \in \mathtt{I}_-^+(\sigma_j)$ for some $1 \leq j \leq k$, then Proposition 3 (3) and Proposition 2 (4) imply that $\pi(\sigma_i) = \sigma_j$. Moreover, $\dim(\sigma_i)_l^< = \dim \pi(\sigma_i)_l^<$ and $\dim(\sigma_i) = \dim \pi(\sigma_i)$. Hence the set $\{\sigma_1, \ldots, \sigma_k\}$ can be partitioned according to the equivalence relation given by: $\sigma_i \sim \sigma_j$ if there exists $\pi \in S_{[n]}$ such that $\pi(\sigma_i) = \sigma_j$. Let $\tau_1, \ldots, \tau_p$ be representatives of the equivalence classes which satisfy the order given in the proof of the item 1, then $\mathsf{WR}_l \searrow_{S_{[n]}(\tau_1^-)} \cdots \searrow_{S_{[n]}(\tau_p^-)} \mathsf{WR}_{l+1}$. □

Figure 4 illustrates the collapsing procedure $\mathsf{WR}_0 \searrow_{S_{[n]}} \mathsf{WR}_1$ for $n = 2$. In this case consider the simplexes $\sigma_1 = \{(1, \{1, 2\}), (2, \{0, 2\}), (0, [2])\}$ and $\sigma_2 = \{(0, \{0, 1\}), (1, [2]), (2, [2])\}$ then $\mathsf{WR}_0^1 \searrow_{S_{[n]}(\sigma_1^-)} \mathsf{WR}_0^2 \searrow_{S_{[n]}(\sigma_2^-)} \mathsf{WR}_0^3 = \mathsf{WR}_1$. And we have the following consequence.

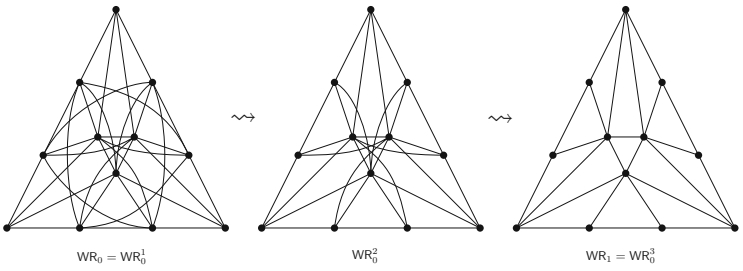

Fig. 4. $S_{[n]}$-collapse.

Corollary 2. *For every natural number n, the simplicial complex $\mathsf{WR}(\Delta^n)$ is $S_{[n]}$-collapsible to $\chi(\Delta^n)$.*

Multi-round Protocol Complex. A *carrier map* Φ from complex $\mathcal{C}$ to complex $\mathcal{D}$ assigns to each simplex σ a subcomplex $\Phi(\sigma)$ of $\mathcal{D}$ such that $\Phi(\tau) \subseteq \Phi(\sigma)$

if $\tau \subseteq \sigma$. The protocol complex of the iterated write/read model (see Fig. 5), $k \geq 0$, is $\mathsf{WR}^{(k+1)}(\Delta^n) = \bigcup_{\sigma \in \mathsf{WR}^{(k)}(\Delta^n)} \mathsf{WR}(\sigma)$.

Corollary 3. *For all* $k \geq 1$, $\mathsf{WR}^{(k)}(\Delta^n) \searrow \chi^{(k)}(\Delta^n)$.

The collapsing procedure consists first in collapsing, in parallel, each subcomplex $\mathsf{WR}(\sigma)$ where σ is a maximal simplex of $\mathsf{WR}^{(k-1)}(\Delta^n)$ as in Theorem 2. An illustration is in Fig. 6, applied to the simplexes $\sigma_1 = \{(0, \{0, 1\}), (1, \{0, 1\}), (2, [2])\}$ and $\sigma_2 = \{(0, \{0, 1\}), (1, [2]), (2, [2])\}$ of $\mathsf{WR}(\Delta^2)$. Second, we collapse $\chi(\mathsf{WR}^{(k-1)}(\Delta^n))$ to $\chi^{(k)}(\Delta^n)$.

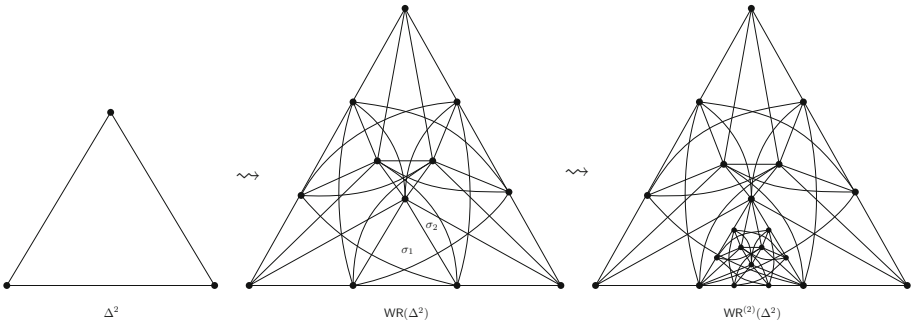

Fig. 5. Complexes of the iterated model; in $\mathsf{WR}^{(2)}(\Delta^2)$ only $\mathsf{WR}(\sigma_1)$ is depicted.

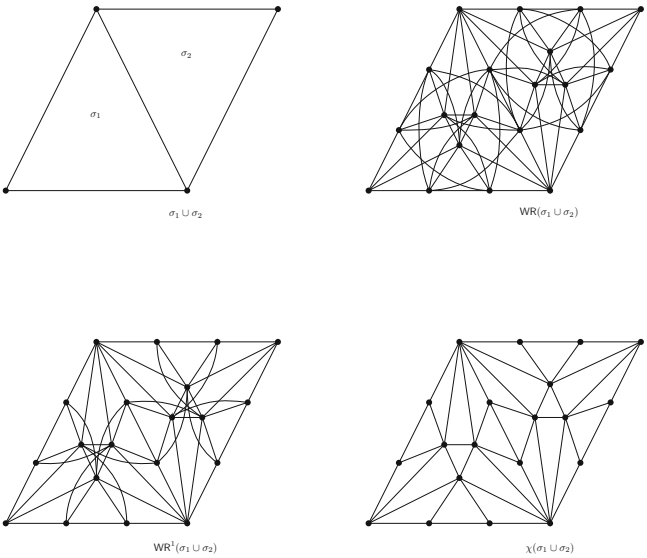

Fig. 6. First collapsing.

References

1. Afek, Y., Attiya, H., Dolev, D., Gafni, E., Merritt, M., Shavit, N.: Atomic snapshots of shared memory. J. ACM **40**(4), 873–890 (1993)
2. Attiya, H., Rajsbaum, S.: The combinatorial structure of wait-free solvable tasks. SIAM J. Comput. **31**(4), 1286–1313 (2002)
3. Biran, O., Moran, S., Zaks, S.: A combinatorial characterization of the distributed 1-solvable tasks. J. Algorithms **11**(3), 420–440 (1990)
4. Borowsky, E., Gafni, E.: Generalized FLP impossibility result for t-resilient asynchronous computations. In: Proceedings of the 25th Annual ACM Symposium on Theory of Computing, STOC, pp. 91–100. ACM, New York (1993)
5. Borowsky, E., Gafni, E.: Immediate atomic snapshots and fast renaming. In: Proceedings of the 12th ACM Symposium on Principles of Distributed Computing, PODC, pp. 41–51. ACM, New York (1993)
6. Borowsky, E., Gafni, E.: A simple algorithmically reasoned characterization of wait-free computation (extended abstract). In: Proceedings of the Sixteenth Annual ACM Symposium on Principles of Distributed Computing, PODC 1997, pp. 189–198. ACM, New York (1997)
7. Borowsky, E., Gafni, E., Lynch, N., Rajsbaum, S.: The BG distributed simulation algorithm. Distrib. Comput. **14**(3), 127–146 (2001)
8. Fischer, M., Lynch, N.A., Paterson, M.S.: Impossibility of distributed commit with one faulty process. J. ACM **32**(2), 374–382 (1985)
9. Gafni, E., Rajsbaum, S.: Recursion in distributed computing. In: Dolev, S., Cobb, J., Fischer, M., Yung, M. (eds.) SSS 2010. LNCS, vol. 6366, pp. 362–376. Springer, Heidelberg (2010)
10. Goubault, E., Mimram, S., Tasson, C.: Iterated chromatic subdivisions are coll apsible. Appl. Categorical Struct. **23**(6), 777–818 (2015)
11. Havlicek, J.: Computable obstructions to wait-free computability. Distrib. Comput. **13**(2), 59–83 (2000)
12. Havlicek, J.: A note on the homotopy type of wait-free atomic snapshot protocol complexes. SIAM J. Comput. **33**(5), 1215–1222 (2004)
13. Herlihy, M., Kozlov, D., Rajsbaum, S.: Distributed Computing Through Combinatorial Topology. Elsevier, Imprint Morgan Kaufmann, Boston (2013)
14. Herlihy, M., Rajsbaum, S.: Simulations and reductions for colorless tasks. In: Proceedings of the ACM Symposium on Principles of Distributed Computing, PODC 2012, pp. 253–260. ACM, New York (2012)
15. Herlihy, M., Shavit, N.: The topological structure of asynchronous computability. J. ACM **46**(6), 858–923 (1999)
16. Herlihy, M., Shavit, N.: The Art of Multiprocessor Programming. Morgan Kaufmann Publishers Inc., San Francisco (2008)
17. Hoest, G., Shavit, N.: Towards a topological characterization of asynchronous complexity. In: Proceedings of the 16th ACM Symposium Principles of Distributed Computing, PODC, pp. 199–208. ACM, New York (1997)
18. Jonsson, J.: Simplicial Complexes of Graphs. Lecture Notes in Mathematics. Springer, Heidelberg (2008). doi:10.1007/978-3-540-75859-4
19. Kozlov, D.N.: Chromatic subdivision of a simplicial complex. Homology Homotopy Appl. **14**(2), 197–209 (2012)
20. Kozlov, D.N.: Topology of the immediate snapshot complexes. Topology Appl. **178**, 160–184 (2014)

21. Kozlov, D.N.: Topology of the view complex. Homology Homotopy Appl. **17**(1), 307–319 (2015)
22. Loui, M.C., Abu-Amara, H.H.: Memory requirements for agreement among unreliable asynchronous processes **4**, 163–183 (1987). JAI Press
23. Rajsbaum, S., Raynal, M., Travers, C.: The iterated restricted immediate snapshot model. In: Hu, X., Wang, J. (eds.) COCOON 2008. LNCS, vol. 5092, pp. 487–497. Springer, Heidelberg (2008)
24. Saks, M., Zaharoglou, F.: Wait-free k-set agreement is impossible: the topology of public knowledge. SIAM J. Comput. **29**(5), 1449–1483 (2000)