Read-Write Protocols for General Tasks

Companion slides for
Distributed Computing Through Combinatorial Topology
Maurice Herlihy & Dmitry Kozlov & Sergio Rajsbaum
General Tasks

So far, focused on colorless tasks …

only sets of input and output value matter

Not who is assigned which values ….
FetchAndIncrement

```plaintext
shared int v // shared counter

fetchAndIncrement(): int
    atomic     // next 2 steps are atomic
    prior: int := v
    v := v + 1
    return prior
```

Not colorless because outputs are unique
fetch&inc output complex

path-connected but has “holes”
Output: (3,2)-Set Agreement
Weak Symmetry-Breaking

If all participate …

At least one 0, at least one 1.
Renaming

**Input**

Unique names from \( \{0 \ldots N\}, \ N \gg n \)

**Output**

Unique names from \( \{0 \ldots M\}, \ N \gg M \)

Output complex called *rook complex*
Colored Complexes
Review: Task Specification

\[(\mathcal{I}, \mathcal{O}, \Delta)\]

Input complex

Output complex

Carrier map

\[\Delta: \mathcal{I} \rightarrow 2^\mathcal{O}\]
Immediate Snapshot Executions

Restricted form of Read-Write memory

(Equivalent to regular R-W memory)

But we will not prove it yet.
Write

Single-writer, multi-reader variables
Snapshot

Single-writer, multi-reader variables
Immediate Snapshot Executions

Pick a set of processes

write together

snapshot together

Repeat with another set
Each process writes, then takes snapshot

<table>
<thead>
<tr>
<th></th>
<th>P</th>
<th>Q</th>
<th>R</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>write</td>
<td></td>
<td>snap</td>
</tr>
<tr>
<td>P??</td>
<td>write</td>
<td>snap</td>
<td></td>
</tr>
<tr>
<td>PQ?</td>
<td>write</td>
<td>snap</td>
<td></td>
</tr>
<tr>
<td>PQR</td>
<td>write</td>
<td>snap</td>
<td></td>
</tr>
</tbody>
</table>

Each process has a view
Example Executions

<table>
<thead>
<tr>
<th>P</th>
<th>Q</th>
<th>R</th>
</tr>
</thead>
<tbody>
<tr>
<td>write</td>
<td></td>
<td></td>
</tr>
<tr>
<td>snap</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>write</td>
<td></td>
</tr>
<tr>
<td></td>
<td>snap</td>
<td></td>
</tr>
<tr>
<td></td>
<td>write</td>
<td></td>
</tr>
<tr>
<td></td>
<td>snap</td>
<td></td>
</tr>
<tr>
<td>P??</td>
<td>PQ?</td>
<td>PQR</td>
</tr>
</tbody>
</table>

Moving last process one round earlier,
Changes this view from PQ? to PQR
Example Executions

<table>
<thead>
<tr>
<th>P</th>
<th>Q</th>
<th>R</th>
</tr>
</thead>
<tbody>
<tr>
<td>write</td>
<td></td>
<td></td>
</tr>
<tr>
<td>snap</td>
<td></td>
<td></td>
</tr>
<tr>
<td>write</td>
<td>write</td>
<td></td>
</tr>
<tr>
<td>snap</td>
<td>snap</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>write</td>
</tr>
<tr>
<td></td>
<td></td>
<td>snap</td>
</tr>
</tbody>
</table>

Changes this view from P?? to PQR

Moving last processes one round earlier,
Protocol Complex

Process (color) & view
Protocol Complex

<table>
<thead>
<tr>
<th>P</th>
<th>Q</th>
<th>R</th>
</tr>
</thead>
<tbody>
<tr>
<td>write</td>
<td>write</td>
<td>write</td>
</tr>
<tr>
<td>snap</td>
<td>snap</td>
<td>snap</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>P</th>
<th>Q</th>
<th>R</th>
</tr>
</thead>
<tbody>
<tr>
<td>write</td>
<td></td>
<td></td>
</tr>
<tr>
<td>snap</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>P</th>
<th>Q</th>
<th>R</th>
</tr>
</thead>
<tbody>
<tr>
<td>write</td>
<td>write</td>
<td></td>
</tr>
<tr>
<td>snap</td>
<td>snap</td>
<td></td>
</tr>
</tbody>
</table>
Standard Chromatic Subdivision

$\text{Ch}(\sigma)$
Combinatorial Definition (I)

Input simplex $\sigma$

Protocol complex $\mathcal{IS}(\sigma)$

Vertex $(P_i, \sigma_i)$

Process name view $\sigma_i \subseteq \sigma$
Combinatorial Definition (II)

Each process's write appears in its own view

\[ P_i \in \text{names}(\sigma_i). \]

Snapshots are ordered

\[ \sigma_i \subseteq \sigma_j \text{ or vice-versa} \]

Each snapshot ordered immediately after write

if \[ P_j \in \text{names}(\sigma_j), \text{ then } \sigma_i \subseteq \sigma_j. \]

If \[ P_j \text{ saw } P_i \text{ write } \ldots \]
then \[ P_j\text{'s snapshot not later than } P_i\text{'}s. \]
2 Input Simplexes, 1 Round

$\text{Ch}(\sigma_0) \cup \text{Ch}(\sigma_0)$
Multi-layer protocol complexes

\( \mathcal{I} \)

\( \text{Ch}(\mathcal{I}) \)

\( \text{Ch}^2(\mathcal{I}) \)
Summary

Tasks and protocols described as …

\textit{colored complexes.}

Protocols are layered immediate snapshot
This work is licensed under a Creative Commons Attribution-ShareAlike 2.5 License.

• You are free:
  – to Share — to copy, distribute and transmit the work
  – to Remix — to adapt the work

• Under the following conditions:
  – Attribution. You must attribute the work to “Distributed Computing through Combinatorial Topology” (but not in any way that suggests that the authors endorse you or your use of the work).
  – Share Alike. If you alter, transform, or build upon this work, you may distribute the resulting work only under the same, similar or a compatible license.

• For any reuse or distribution, you must make clear to others the license terms of this work. The best way to do this is with a link to
  – http://creativecommons.org/licenses/by-sa/3.0/.

• Any of the above conditions can be waived if you get permission from the copyright holder.

• Nothing in this license impairs or restricts the author's moral rights.
Distributed Computing through Combinatorial Topology