Parameterized verification of round-based shared-memory systems

Nicolas Waldburger ¹
Nathalie Bertrand ¹, Nicolas Markey ¹, Ocan Sankur ¹

¹Univ Rennes, Inria, CNRS, IRISA, France
Round-based shared-memory algorithms

The considered algorithms

- Parallel, identical processes communicating via shared memory
- Asynchrony: some processes might be faster than others
- Non-atomic read & write combinations, no fault
- Round-based: There is a fresh copy of registers at each round
- Processes can be at different rounds; they may read to and write from registers of nearby rounds

---

Round-based shared-memory algorithms

The distributed systems considered

- **Parallel, identical** processes communicating via **shared memory**
- **Asynchrony**: some processes might be faster than others
- **Non-atomic** read & write combinations, no fault
- **Round-based**: There is a fresh copy of registers at each round
- Processes can be at different rounds; they may read to and write from registers of nearby rounds

```plaintext
int k := 0, bool p ∈ {0, 1}, (rg_b[r])_{b∈{0,1},r∈N} all initialized to no;
while true do
    read from rg_0[k] and rg_1[k];
    if rg_0[k] = yes and rg_1[k] = no then p := 0;
    else if rg_0[k] = no and rg_1[k] = yes then p := 1;
    write yes to rg_p[k];
    if k > 0 then
        read from rg_1−p[k−1];
        if rg_1−p[k−1] = no then return p;
    k := k+1;
```

**Algorithm 2**: Aspnes’ consensus algorithm\(^1\).

A model: round-based register protocols

Inspired by models for shared-memory systems without rounds\textsuperscript{23}.

- One model for all processes: a finite automaton
- Transitions are read actions, write actions and round increments
- A fixed number $d$ of registers per round (the total number of registers is hence unbounded)

\begin{figure}
\centering
\includegraphics[width=\textwidth]{diagram}
\end{figure}

\textsuperscript{2} Javier Esparza, Pierre Ganty, and Rupak Majumdar. Parameterized verification of asynchronous shared-memory systems. \textit{CAV’13}

\textsuperscript{3} Patricia Bouyer, Nicolas Markey, Mickael Randour, Arnaud Sangnier, and Daniel Stan. Reachability in networks of register protocols under stochastic schedulers. \textit{ICALP’16}
A limited visibility range

$k + 1$

$k$

$k - 1$

$k - v$

$k - v - 1$

$\vdots$

$v$ given in unary (in Aspnes’ consensus algorithm, $v = 1$)
Semantics of the model

From now on, let $d = 1$: one register per round.

<table>
<thead>
<tr>
<th>rounds</th>
<th>processes</th>
<th>registers</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td>$d_0$</td>
</tr>
<tr>
<td>1</td>
<td>$p$</td>
<td>$a$</td>
</tr>
<tr>
<td>2</td>
<td></td>
<td>$b$</td>
</tr>
<tr>
<td>3</td>
<td>$q$</td>
<td>$d_0$</td>
</tr>
</tbody>
</table>

processes are undistinguished
Semantics of the model

From now on, let $d = 1$: one register per round.

```
<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>$q$</td>
<td>$\times 1$</td>
</tr>
<tr>
<td>2</td>
<td>b</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>a</td>
<td></td>
<td>$p$</td>
<td>$\times 3$</td>
</tr>
<tr>
<td>0</td>
<td>d</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

```
<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>$r$</td>
<td>$\times 1$</td>
</tr>
<tr>
<td>2</td>
<td>b</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>a</td>
<td></td>
<td>$p$</td>
<td>$\times 3$</td>
</tr>
<tr>
<td>0</td>
<td>d</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

$((q, \text{write}(b), r), 3)$

"a process on round 3 moves from $q$ to $r$ and writes $b$ to the register of round 3"
Semantics of the model

From now on, let $d = 1$: one register per round.

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>$q_0 \times n$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>$p \times 3$</td>
<td>a</td>
<td>d_0</td>
</tr>
<tr>
<td>2</td>
<td></td>
<td>b</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td>d_0</td>
<td></td>
</tr>
</tbody>
</table>

Initial configuration of size $n$:

$((q, \text{write}(b), r), 3)$

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>$q_0 \times n$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
<td>a</td>
<td>d_0</td>
</tr>
<tr>
<td>2</td>
<td></td>
<td>b</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td>d_0</td>
<td></td>
</tr>
</tbody>
</table>
The (parameterized) safety problem

Is it true that, for all numbers of processes $n$ and all executions from the initial configuration of size $n$, an error state $q_{err}$ is avoided?

If the error state cannot be covered, the system is safe.

Dual problem: look for an execution covering the error.
A small example

The considered algorithms

Our model

The safety problem

Results

Conclusion

A small example

\[
\begin{align*}
q_0 & \xrightarrow{\text{write}(a)} q_1 \\
q_0 & \xrightarrow{\text{Inc}} q_2 \\
q_2 & \xrightarrow{\text{read}^{-1}(a)} q_5 \\
q_3 & \xrightarrow{\text{write}(a)} q_4 \\
q_5 & \xrightarrow{\text{read}^0(d_0)} q_6 \\
q_6 & \xrightarrow{\text{read}^0(b)} q_{\text{err}} \\
\end{align*}
\]

increment round

initial state

Increment

read a from register of previous round

read initial symbol \(d_0\) from register of current round

error state

\[d = 1\] (one register per round)

\[v = 1\] (processes can read one round back)
State $q_4$ can be covered from the initial configuration with one process:
A small example

State $q_4$ can be covered from the initial configuration with one process:

```
...  ...
1  q2  d0
0  q0  d0
```
A small example

State $q_4$ can be covered from the initial configuration with one process:
A small example

State $q_4$ can be covered from the initial configuration with one process:
A small example

State $q_6$ can be covered from the initial configuration with two processes:
A small example

State $q_6$ can be covered from the initial configuration with two processes:

\[
\begin{array}{c|c}
\vdots & \vdots \\
1 & d_0 \\
0 \times 2 & d_0 \\
\end{array}
\]
A small example

State $q_6$ can be covered from the initial configuration with two processes:

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$d_0$</td>
</tr>
<tr>
<td>0</td>
<td>$q_0$ $q_1$</td>
</tr>
</tbody>
</table>

writes $a$
A small example

State $q_6$ can be covered from the initial configuration with two processes:

\[\vdots\]

\[1\] $\downarrow q_2$

\[d_0\]

\[0\] $\downarrow q_0 \quad q_1$

\[a\]
A small example

State $q_6$ can be covered from the initial configuration with two processes:

\[
\begin{array}{c}
\vdots \\
1 & \overset{\text{reads } a}{\Rightarrow} & \text{reads } d_0 \\
0 & \overset{a}{\Rightarrow} & \vdots \\
\end{array}
\]
A small example

State $q_6$ can be covered from the initial configuration with two processes:
Theorem

Parameterized safety in round-based register protocols is PSPACE-complete.\(^4\)

\(^4\)Nathalie Bertrand, Nicolas Markey, Ocan Sankur, Nicolas Waldburger. Parameterized safety verification of round-based shared-memory systems. *ICALP’22*
Theorem

*Parameterized safety in round-based register protocols is PSPACE-complete.*

Ingredients of the polynomial-space algorithm

- **Copycat property** (thanks to non-atomicity)
- Thanks to copycat, define an **abstraction** where one only remembers which pairs (state,round) are populated by at least one process
- Exploit **limited visibility range**: reads and writes are local with respect to the round
- Rely on a **sliding window** along the rounds

---

4 Nathalie Bertrand, Nicolas Markey, Ocan Sankur, Nicolas Waldburger. Parameterized safety verification of round-based shared-memory systems. ICALP’22
Theorem

Parameterized safety in round-based register protocols is PSPACE-complete.\(^4\)

Ingredients of the polynomial-space algorithm

- **Copycat property** (thanks to non-atomicity)
- Thanks to copycat, define an abstraction where one only remembers which pairs (state,round) are populated by at least one process
- Exploit limited visibility range: reads and writes are local with respect to the round
- Rely on a sliding window along the rounds

This also allows to prove the following (tight) exponential upper bounds:

Exponential upper bounds

There exists an exponential upper bound on the number of processes and on the number of rounds needed to reach the error state.

\(^4\)Nathalie Bertrand, Nicolas Markey, Ocan Sankur, Nicolas Waldburger. Parameterized safety verification of round-based shared-memory systems. ICALP’22
Summary

- Round-based register protocols are a model for round-based shared-memory algorithms such as Aspnes’ consensus algorithm.
- The verification problem of parameterized safety is PSPACE-complete.

Future work

- Other problems on our model: parameterized TARGET, parameterized INEVITABILITY.
- Almost-sure reachability in round-based register protocols with stochastic schedulers (termination of Aspnes’ algorithm).
- Links with classical notions of fairness.
Classical notions of fairness are not satisfactory

$q_{err}$ is reached with probability 1 with a stochastic scheduler with two processes.

Consider the execution with two processes where one process goes to $q_1$ and back to $q_0$ on every round, while the other process stays on $q_0$ forever.

This execution is fair with respect to:
- Fairness on moves: no move is available infinitely often because $k$ increases
- Fairness on transitions: transition from $q_1$ to $q_{err}$ is never enabled.