Checking Presence Reachability Properties on Parameterized Shared-Memory Systems
Aim: automated verification of distributed algorithms.

Parameterized verification: arbitrarily large systems.

Two models in this talk:

- Simple model: shared-memory systems with finite memory
- More complex model: round-based shared-memory systems
A model for shared-memory systems

- Arbitrary number of processes
- Processes are *identical* agents
- No identifiers: processes are *anonymous*
- Modelled by a single, common *finite automaton*

A model for shared-memory systems

Finite number of shared registers, each register has a value from finite set of symbols $\Sigma$

1. a
2. b
3. $d_0$

Registers are initialized to value $d_0$

No atomic read/write combinations

Semantics

A configuration:

<table>
<thead>
<tr>
<th>$q \times 2$</th>
<th>$p \times 1$</th>
<th>$a$</th>
<th>$b$</th>
<th>$d_0$</th>
</tr>
</thead>
</table>

How many processes are on each state  
Content of the registers
Semantics

\[ q \times 2 \quad p \times 1 \quad a \quad b \quad d_0 \quad (q, \text{write}_3(a), r) \]

\[ p \times 1 \quad r \times 1 \quad a \quad b \quad a \]

\[ (p, \text{read}_1(a), r) \]

\[ q \times 1 \quad r \times 2 \quad a \quad b \quad a \]
Semantics

Initial configurations:

\[
\begin{array}{ccc}
  q \times 2 & p \times 1 & a \ b \ d_0 \\
\end{array}
\]

\[
(q, \text{write}_3(a), r)
\]

\[
\begin{array}{ccc}
  q \times 1 & p \times 1 & r \times 1 \\
\end{array}
\]

\[
\begin{array}{ccc}
  a & b & a \\
\end{array}
\]

\[
\begin{array}{ccc}
  q \times 1 & r \times 2 & a \ b \ a \\
\end{array}
\]

\[
(p, \text{read}_1(a), r)
\]

Can be arbitrarily large

Registers are initialized to \(d_0\)

Initial configurations:

\[
\begin{array}{ccc}
  q_1 \times n_1 & q_2 \times n_2 & \cdots \\
  d_0 & d_0 & d_0 \\
\end{array}
\]

with \(n_1, n_2, \ldots \geq 0\) and \(q_1, q_2, \ldots\) initial states
A small example

A single register

\[ q_0 \rightarrow \text{write}(c) \rightarrow A \rightarrow \text{read}(a) \rightarrow q_f \]

\[ B \rightarrow \text{read}(d_0) \rightarrow C \rightarrow \text{read}(b) \rightarrow q_f \]

\[ \text{read}(c) \rightarrow A \rightarrow \text{write}(b) \rightarrow q_f \]

\[ \text{write}(a) \rightarrow C \rightarrow \text{read}(b) \rightarrow q_f \]
A small example

Two processes

Initial value

Initial value $d_0$
A small example
A small example

\[d_0\]

\[q_0 \rightarrow \text{write}(c) \rightarrow A\]

\[B \rightarrow \text{read}(d_0) \rightarrow C\]

\[C \rightarrow \text{read}(b) \rightarrow \text{write}(a) \rightarrow q_f\]

\[A \rightarrow \text{read}(a) \rightarrow \text{write}(b) \rightarrow q_f\]
A small example

\[
\begin{align*}
q_0 & \xrightarrow{\text{write}(c)} A \\
B & \xrightarrow{\text{read}(d_0)} C \\
C & \xrightarrow{\text{read}(d_0)} B \\
A & \xrightarrow{\text{read}(c)} q_f \\
q_f & \xrightarrow{\text{write}(b)} A \\
A & \xrightarrow{\text{read}(a)} q_f \\
C & \xrightarrow{\text{read}(b)} C \\
A & \xrightarrow{\text{write}(a)} A
\end{align*}
\]
A small example

\[
\begin{align*}
q_0 & \rightarrow \text{write}(c) \\
A & \rightarrow \text{read}(a) \\
B & \rightarrow \text{read}(d_0) \\
C & \rightarrow \text{read}(d_0) \\
q_f & \rightarrow \text{write}(b)
\end{align*}
\]
A small example

\[ q_0 \xrightarrow{\text{write}(c)} A \]
\[ B \xrightarrow{\text{read}(d_0)} q_0 \]
\[ C \xrightarrow{\text{read}(d_0)} A \]
\[ A \xrightarrow{\text{read}(a)} q_f \]
\[ q_f \xrightarrow{\text{write}(b)} C \]
\[ C \xleftarrow{\text{read}(b)} A \]
\[ A \xleftarrow{\text{read}(c)} q_f \]

\( q_f \) is covered ✓
Reachability problems

COVER: \[ \exists n, \exists \gamma_0, \exists \rho: \gamma_0 \rightarrow^{*} \gamma, \gamma(q_f) > 0? \]

**Parameterized:** arbitrarily many processes

**An initial configuration**

**A least one process on** $q_f$: “error state”

**Execution**
Reachability problems

COVER: \( \exists n, \exists \gamma_0, \exists \rho: \gamma_0 \rightarrow^* \gamma, \gamma(q_f) > 0 \) ?

TARGET: \( \exists n, \exists \gamma_0, \exists \rho: \gamma_0 \rightarrow^* \gamma, \forall q \neq q_f, \gamma(q) = 0 \) ?

All processes “synchronize” on \( q_f \)
Reachability problems

COVER: \[ \exists n, \exists \gamma_0, \exists \rho: \gamma_0 \rightarrow^* \gamma, \gamma(q_f) > 0 ? \]

TARGET: \[ \exists n, \exists \gamma_0, \exists \rho: \gamma_0 \rightarrow^* \gamma, \forall q \neq q_f, \gamma(q) = 0 ? \]

PRP²: \[ \exists n, \exists \gamma_0, \exists \rho: \gamma_0 \rightarrow^* \gamma, \gamma \models \phi ? \]

 Presence Reachability Problem

with \( \phi \in B(\{\#q = 0, \#q > 0\}, \{\text{reg}_i = d, \text{reg}_i \neq d\}) \)

\#q = \text{number of processes on } q

Reachability problems

**COVER:** \( \exists n, \gamma_0, \rho: \gamma_0 \rightarrow^* \gamma, \gamma(q_f) > 0 ? \)

**TARGET:** \( \exists n, \gamma_0, \rho: \gamma_0 \rightarrow^* \gamma, \forall q \neq q_f, \gamma(q) = 0 ? \)

**PRP**\(^2\): \( \exists n, \gamma_0, \rho: \gamma_0 \rightarrow^* \gamma, \gamma \vdash \phi ? \)

with \( \phi \in \mathcal{B}(\{\#q = 0, \#q > 0\}, \{\text{reg}_i = d, \text{reg}_i \neq d\}) \)

Examples:
- \( \phi = \``\#q_f > 0'' \) (COVER), \( \phi = \``\bigvee_{q \neq q_f} \#q = 0'' \) (TARGET)
- \( \phi = \``(\#q_1 > 0) \lor ([\#q_2 = 0] \land [\text{reg}_1 = d_0])'' \)

A process may “copy” the behavior of another process on the same state.
A process may “copy” the behavior of another process on the same state.

Monotonicity

\textit{write(b)}
Monotonicity

A process may “copy” the behavior of another process on the same state.

write(b)
Monotonicity

A process may “copy” the behavior of another process on the same state.

Diagram:
- A state represented by a red circle labeled "read(a)"
- An action symbol represented by a black circle labeled "a"
A process may “copy” the behavior of another process on the same state.
Monotonicity

A process may “copy” the behavior of another process on the same state.

\[ \text{read}(a) \]
**Monotonicity**

Abstraction: remember whether there is at least one process on a given state.

*Sound and complete* for PRP because:
- Monotonicity property (thanks to non-atomicity)
- Arbitrarily many properties in initial configurations
- Interested in parameterized properties (does there exists $n$ such that…?)
- $\phi$ cannot count processes
Some processes

Initial value $d_0$

$q_0$ → write$(c)$ → $A$

$A$ → read$(a)$ → $q_f$

$B$ → read$(d_0)$ → $C$

$C$ → read$(c)$ → $A$

$C$ → read$(b)$ → $q_f$

$B$ → read$(d_0)$ → $C$

$C$ → write$(a)$ → $C$

$C$ → write$(b)$ → $q_f$
Back to the example

\[ d_0 \]

\[ q_0 \quad \text{write}(c) \quad A \quad \text{read}(a) \quad q_f \]

\[ B \quad \text{read}(d_0) \quad C \quad \text{read}(b) \quad \text{write}(b) \]

\[ \text{read}(d_0) \quad \text{write}(a) \]
Back to the example

$q_0$ -> $A$ (write(c))

$B$ -> $C$ (read(d_0))

$C$ -> $q_f$ (write(b))

$q_f$ -> $A$ (read(a))

$B$ -> $q_f$ (read(b))

$C$ -> $q_f$ (read(b))

$d_0$ -> $q_0$ (read(d_0))
Back to the example

Graph:
- State $q_0$ transitions to $A$ with $write(c)$.
- State $A$ transitions to $q_f$ with $read(a)$.
- State $B$ transitions to $C$ with $read(d_0)$.
- State $C$ transitions to $B$ with $write(b)$.
- State $C$ transitions to $q_f$ with $read(b)$.
- State $q_f$ transitions to $A$ with $write(b)$.

Symbols:
- $c$
- $d_0$
- $a$
- $b$
- $f$
Back to the example

$q_0 \xrightarrow{\text{write}(c)} A \xrightarrow{\text{read}(a)} q_f$

$B \xrightarrow{\text{read}(d_0)} C \xrightarrow{\text{read}(b)} q_f$

$\text{read}(c) \xrightarrow{\text{write}(b)} q_f$

$\text{read}(b) \xrightarrow{\text{write}(a)} q_f$

$a$
Positive instance of COVER

Back to the example
Back to the example

Negative instance of TARGET

\[ q_0 \xrightarrow{\text{write}(c)} A \]

\[ B \xrightarrow{\text{read}(d_0)} C \]

\[ C \xrightarrow{\text{read}(d_0)} B \]

\[ A \xrightarrow{\text{read}(a)} B \]

\[ A \xrightarrow{\text{read}(b)} C \]

\[ A \xrightarrow{\text{write}(b)} A \]

\[ A \xrightarrow{\text{read}(a)} A \]

\[ A \xrightarrow{\text{write}(a)} A \]

\[ A \xrightarrow{\text{read}(c)} A \]

\[ A \xrightarrow{\text{write}(c)} A \]

\[ q_f \]

\[ a \text{ cannot be the final register value} \]

\[ b \text{ cannot be the final register value} \]
Positive instance of PRP with
\[ \phi = (\text{reg} = c \lor \text{reg} = d_0) \land (\#A > 0 \land \#q_f > 0 \land \#C = 0) \lor (\#A = 0 \land \#q_f > 0 \land \#C = 0) \]
Positive instance of PRP with

\[ \phi = (\text{reg} = c \lor \text{reg} = d_0) \land \left( [\#A > 0 \land \#q_f > 0 \land \#C = 0] \lor [\#A = 0 \land \#q_f > 0 \land \#C = 0] \right) \]
Positive instance of PRP with
\[ \phi = (\text{reg} = c \lor \text{reg} = d_0) \land ([\#A > 0 \land \#q_f > 0 \land \#C = 0] \lor [\#A = 0 \land \#q_f > 0 \land \#C = 0]) \]
Back to the example

Positive instance of PRP with
\[ \phi = (\text{reg} = c \lor \text{reg} = d_0) \land \left( [\#A > 0 \land \#q_f > 0 \land \#C = 0] \lor [\#A = 0 \land \#q_f > 0 \land \#C = 0] \right) \]
Positive instance of PRP with

\[ \phi = (\text{reg} = c \lor \text{reg} = d_0) \land \left(\#A > 0 \land \#q_f > 0 \land \#C = 0\right) \lor \left(\#A = 0 \land \#q_f > 0 \land \#C = 0\right) \]
Positive instance of PRP with
\[ \phi = (\text{reg} = c \lor \text{reg} = d_0) \land ([\#A > 0 \land \#q_f > 0 \land \#C = 0] \lor [\#A = 0 \land \#q_f > 0 \land \#C = 0]) \]
Positive instance of PRP with
\[
\phi = (\text{reg} = c \lor \text{reg} = d_0) \land \left( [\#A > 0 \land \#q_f > 0 \land \#C = 0] \lor \left[ \#A = 0 \land \#q_f > 0 \land \#C = 0 \right] \right)
\]
Positive instance of PRP with

\[ \phi = (\text{reg} = c \lor \text{reg} = d_0) \land (\#A > 0 \land \#q_f > 0 \land \#C = 0) \lor (\#A = 0 \land \#q_f > 0 \land \#C = 0) \]
Positive instance of PRP with

$$\phi = (\text{reg} = c \lor \text{reg} = d_0) \land \left( [\#A > 0 \land \#q_f > 0 \land \#C = 0] \lor [\#A = 0 \land \#q_f > 0 \land \#C = 0] \right)$$
NP-completeness of COVER

COVER: $\exists n, \exists \gamma_0, \exists \rho: \gamma_0 \rightarrow^* \gamma, \gamma(q_f) > 0$?
NP-completeness of COVER

COVER: $\exists n, \exists \gamma_0, \exists \rho: \gamma_0 \rightarrow^{*} \gamma, \gamma(q_f) > 0$?

Reduction from 3-SAT:

<table>
<thead>
<tr>
<th>$x$</th>
<th>$d_0$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\neg x$</td>
<td>$d_0$</td>
</tr>
</tbody>
</table>

Check $x$: $\rightarrow$ write$_x(T)$ $\rightarrow$ read$_{\neg x}(d_0)$

Check $\neg x$: $\rightarrow$ write$_{\neg x}(T)$ $\rightarrow$ read$_x(d_0)$
NP-completeness of COVER

COVER: \( \exists n, \exists \gamma_0, \exists \rho: \gamma_0 \rightarrow^\ast \gamma, \gamma(q_f) > 0 \) ?

Reduction from 3-SAT:

\[
\begin{array}{c|c}
\hline
x & d_0 \\
\hline
\neg x & d_0 \\
\hline
\end{array}
\]

Check \( x \):
\[
\rightarrow \text{write}_x(T) \rightarrow \text{read}_{\neg x}(d_0)
\]

Check \( \neg x \):
\[
\rightarrow \text{write}_{\neg x}(T) \rightarrow \text{read}_x(d_0)
\]

\[
\begin{array}{c}
\text{write}_{\neg x}(T)\\
\rightarrow
\end{array}
\quad
\begin{array}{c}
\text{Check clause 1} \\
\rightarrow \ldots \rightarrow \text{Check clause } m \\
\rightarrow q_f
\end{array}
\]

Directly relies on initialization of registers!
COVER when registers are uninitialized

COVER: $\exists n, \exists \gamma_0, \exists \rho: \gamma_0 \to^* \gamma, \gamma(q_f) > 0$?

COVER drops down to PTIME when registers are uninitialized\(^3\).

Saturation algorithm:

COVER when registers are uninitialized

COVER:  \( \exists n, \exists \gamma_0, \exists \rho: \gamma_0 \rightarrow^* \gamma, \gamma(q_f) > 0 \) ?

COVER drops down to PTIME when registers are uninitialized\(^3\).

Saturation algorithm:

COVER when registers are uninitialized

COVER: \( \exists n, \exists \gamma_0, \exists \rho: \gamma_0 \rightarrow^* \gamma, \gamma(q_f) > 0 \) ?

COVER drops down to PTIME when registers are uninitialized\(^3\).

Saturation algorithm:

COVER when registers are uninitialized

COVER: \[\exists n, \exists \gamma_0, \exists \rho: \gamma_0 \rightarrow^* \gamma, \gamma(q_f) > 0?\]

COVER drops down to PTIME when registers are uninitialized\(^3\).

Saturation algorithm:

TARGET when registers are uninitialized

TARGET: \( \exists n, \exists \gamma_0, \exists \rho: \gamma_0 \rightarrow^* \gamma, \forall q \neq q_f, \gamma(q) = 0 \) ?

TARGET is still NP-complete when registers are uninitialized. Reduction from 3-SAT:

∀x
\( \text{write}_x(\text{true}) \)

\( \text{write}_x(\text{false}) \)
∀x

Check clause 1

\( \cdots \)

Check clause m

\( q_f \)
TARGET with a single register

TARGET: \[ \exists n, \exists \gamma_0, \exists \rho: \gamma_0 \rightarrow^* \gamma, \forall q \neq q_f, \gamma(q) = 0 \]

TARGET is PTIME when only one register.
For simplicity: the register is uninitialized. Algorithm inspired from broadcast protocols\(^4\).

TARGET with a single register

TARGET: \[ \exists n, \exists \gamma_0, \exists \rho: \gamma_0 \rightarrow^* \gamma, \forall q \neq q_f, \gamma(q) = 0 \? \]

TARGET is PTIME when only one register.
For simplicity: the register is uninitialized. Algorithm inspired from broadcast protocols\(^4\).

Compute *coverable states* (the state can be covered from initial configurations)
and *backwards coverable states* (\(q_f\) may be reached from some configuration containing the state).

TARGET with a single register

TARGET: \( \exists n, \exists \gamma_0, \exists \rho: \gamma_0 \rightarrow^* \gamma, \forall q \neq q_f, \gamma(q) = 0 \) ?

TARGET is PTIME when only one register.
For simplicity: the register is uninitialized. Algorithm inspired from broadcast protocols\(^4\).

Compute *coverable states* (the state can be covered from initial configurations) and *backwards coverable states* (\( q_f \) may be reached from some configuration containing the state).

\[ \text{\textbullet} \quad \text{= coverable} \]
\[ \text{\textbullet} \quad \text{= backwards coverable} \]

TARGET with a single register

TARGET: \( \exists n, \exists \gamma_0, \exists \rho: \gamma_0 \rightarrow^* \gamma, \ \forall q \neq q_f, \ \gamma(q) = 0 \) ?

TARGET is PTIME when only one register.
For simplicity: the register is uninitialized. Algorithm inspired from broadcast protocols\(^4\).

Compute *coverable states* (the state can be covered from initial configurations)
and *backwards coverable states* (\(q_f\) may be reached from some configuration containing the state).

![Diagram showing coverable and backwards coverable states]

Iteratively remove all states that are not
TARGET with a single register

TARGET: \[ \exists n, \exists \gamma_0, \exists \rho: \gamma_0 \rightarrow^* \gamma, \forall q \neq q_f, \gamma(q) = 0 ? \]

TARGET is PTIME when only one register.
For simplicity: the register is uninitialized. Algorithm inspired from broadcast protocols\(^4\).

Compute *coverable states* (the state can be covered from initial configurations)
and *backwards coverable states* (*q_f* may be reached from some configuration containing the state).

Iteratively remove all states that are not \(= \text{backwards coverable}\)

\[= \text{coverable}\]

TARGET with a single register

TARGET: \( \exists n, \exists \gamma_0, \exists \rho: \gamma_0 \rightarrow^* \gamma, \ \forall q \neq q_f, \ \gamma(q) = 0 \) ?

TARGET is PTIME when only one register.
For simplicity: the register is uninitialized. Algorithm inspired from broadcast protocols\(^4\).

Compute *coverable states* (the state can be covered from initial configurations)
and *backwards coverable states* (\(q_f\) may be reached from some configuration containing the state).

Iteratively remove all states that are not

\(\checkmark\) = coverable

\(\checkmark\) = backwards coverable

TARGET with a single register

TARGET: \[ \exists n, \exists \gamma_0, \exists \rho: \gamma_0 \rightarrow^* \gamma, \forall q \neq q_f, \gamma(q) = 0? \]

TARGET is PTIME when only one register. For simplicity: the register is uninitialized. Algorithm inspired from broadcast protocols\textsuperscript{4}.

Compute *coverable states* (the state can be covered from initial configurations) and *backwards coverable states* (\(q_f\) may be reached from some configuration containing the state).

The algorithm is generalizable to PRP when the formula is in Disjunctive Normal Form (DNF).

**DNF-PRP:** \[ \exists n, \exists \gamma_0, \exists \rho: \gamma_0 \rightarrow^* \gamma, \gamma \models \phi, \]

\(\phi\) in DNF: \[ \phi = \bigvee_i (t_{i,1} \land t_{i,2} \land \cdots \land t_{i,m_i}), \]
\[ t_{i,j} \in \{\#q = 0, \#q > 0\} \cup \{\text{reg}_i = d, \text{reg}_i \neq d\} \]

### Summary of complexity results

<table>
<thead>
<tr>
<th></th>
<th>COVER</th>
<th>TARGET</th>
<th>DNF-PRP</th>
<th>PRP</th>
</tr>
</thead>
<tbody>
<tr>
<td>General case</td>
<td>NP-complete</td>
<td>NP-complete</td>
<td>NP-complete</td>
<td>NP-complete</td>
</tr>
<tr>
<td>Uninitialized</td>
<td>PTIME-complete</td>
<td>NP-complete</td>
<td>NP-complete</td>
<td>NP-complete</td>
</tr>
<tr>
<td>One register</td>
<td>PTIME-complete</td>
<td>PTIME-complete</td>
<td>PTIME-complete</td>
<td>NP-complete</td>
</tr>
</tbody>
</table>

Round-based shared-memory systems
Round-based shared-memory systems

Model inspired by round-based algorithms from the literature\textsuperscript{678}.

Process progress in asynchronous rounds, each round having its own finite set of registers.

The round-based model

- Read transitions now mention from which round they are reading, relatively to the current round of the process
- A new type of transitions: *round increments*, which send the process to the next round

Example with one register per round:
A limited visibility range

- The process may write to these registers:
  - \( \text{reg}_1[k+1] \) to \( \text{reg}_{\text{dim}}[k+1] \)
  - \( \text{reg}_1[k] \) to \( \text{reg}_{\text{dim}}[k] \)
  - \( \text{reg}_1[k-1] \) to \( \text{reg}_{\text{dim}}[k-1] \)
  - \( \text{reg}_1[k-v-1] \) to \( \text{reg}_{\text{dim}}[k-v-1] \)
  - \( \text{reg}_1[k-v] \) to \( \text{reg}_{\text{dim}}[k-v] \)

- The process may read from these registers:
  - \( \text{reg}_1[k+1] \) to \( \text{reg}_{\text{dim}}[k+1] \)
  - \( \text{reg}_1[k] \) to \( \text{reg}_{\text{dim}}[k] \)
  - \( \text{reg}_1[k-1] \) to \( \text{reg}_{\text{dim}}[k-1] \)
  - \( \text{reg}_1[k-v-1] \) to \( \text{reg}_{\text{dim}}[k-v-1] \)
  - \( \text{reg}_1[k-v] \) to \( \text{reg}_{\text{dim}}[k-v] \)
### Semantics

<table>
<thead>
<tr>
<th></th>
<th>( p \times 1 )</th>
<th></th>
<th>( q \times 3 )</th>
<th></th>
<th>( r \times 1 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>( d_0 )</td>
<td>2</td>
<td>( b )</td>
<td>1</td>
<td>( a )</td>
</tr>
<tr>
<td>1</td>
<td>( a )</td>
<td>2</td>
<td>( b )</td>
<td>0</td>
<td>( d_0 )</td>
</tr>
<tr>
<td>3</td>
<td>( d_0 )</td>
<td></td>
<td>( p \times 1 )</td>
<td></td>
<td>( r \times 1 )</td>
</tr>
</tbody>
</table>

\((p, \text{read}^{-1}(b), r), 3\)

*here with one register per round*
here with one register per round

\[ (q, \text{write}(b), r), 1 \]
Initial configurations:

\[ n_1, n_2, \ldots \geq 0 \text{ and } q_1, q_2, \ldots \text{ initial states} \]
# Abstraction

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

Initial configurations:

- $(q, \text{write}(b), r), 1$

- with $n_1, n_2, ... \geq 0$ and $q_1, q_2, ...$ initial states
An example of round-based register protocol

Increment round

A

write(a)

write(b)

read\(^{-1}\)(a)

read\(^{-1}\)(d_0)

read\(^{-1}\)(b)

read\(^0\)(d_0)

read\(^0\)(b)

C

B

A

q_0

D

E

q_f

\[ \text{write}(b) \]

\[ \text{read}^{-1}(a) \]

\[ \text{read}^{-1}(d_0) \]

\[ \text{write}(a) \]

\[ \text{read}^{-1}(b) \]

\[ \text{read}^0(d_0) \]

\[ \text{read}^0(b) \]

<table>
<thead>
<tr>
<th>[ 0 ]</th>
<th>[ q_0 ]</th>
</tr>
</thead>
<tbody>
<tr>
<td>[ 1 ]</td>
<td>[ d_0 ]</td>
</tr>
<tr>
<td>[ 2 ]</td>
<td>[ d_0 ]</td>
</tr>
</tbody>
</table>

\[ \vdots \]

\[ \vdots \]

\[ \vdots \]

\[ d_0 \]

\[ d_0 \]

\[ d_0 \]

\[ = \text{round 0} \]

\[ = \text{round 1} \]

\[ = \text{round 2} \]
An example of round-based register protocol

Increment round

\[ \begin{array}{c|c|c}
\text{Round} & \text{Register State} & \text{Value} \\
\hline
0 & q_0 & d_0 \\
1 & q_0 & d_0 \\
2 & q_0 & d_0 \\
\end{array} \]

= round 0

= round 1

= round 2
An example of round-based register protocol

\[ \begin{align*}
q_0 & \xrightarrow{\text{write}(a)} A \\
& \xleftarrow{\text{read}^{-1}(a)} B \\
& \xleftarrow{\text{read}^{-1}(d_0)} A \\
& \xrightarrow{\text{write}(b)} C
\end{align*} \]

Increment round

\[
\begin{array}{c|c|c|c}
\text{Round} & q_0 & A & d_0 \\
\hline
0 & q_0 & & d_0 \\
1 & q_0 & A & a \\
2 & & & \\
\end{array}
\]

- \( \text{red}^{-1} \) = round 0
- \( \text{red}^0 \) = round 1
- \( \text{red}^{-1} \) = round 2

\( \text{write}(b) \)
An example of round-based register protocol

Increment round

<table>
<thead>
<tr>
<th>:</th>
<th>:</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>(d_0)</td>
</tr>
<tr>
<td>1</td>
<td>a</td>
</tr>
<tr>
<td>0</td>
<td>(d_0)</td>
</tr>
</tbody>
</table>

\(d_0\) = round 0
\(d_0\) = round 1
\(d_0\) = round 2
An example of round-based register protocol

Increment round

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>q₀</td>
<td>A</td>
<td>B</td>
</tr>
<tr>
<td>q₀</td>
<td>A</td>
<td></td>
</tr>
</tbody>
</table>

write(b)

write(a)

$q₀ \leftarrow \text{read}^{-1}(a)$

$q₀ \leftarrow \text{read}^{-1}(d₀)$

$q₀ \leftarrow \text{write}(a)$

$q₀ \leftarrow \text{read}^{-1}(b)$

$q₀ \leftarrow \text{read}^0(d₀)$

$q₀ \leftarrow \text{read}^0(b)$

$q_f$

= round 0

= round 1

= round 2
An example of round-based register protocol

Increment round

\[ \text{write}(b) \]

\[ \text{read}^{-1}(a) \quad \text{read}^{-1}(d_0) \quad \text{write}(a) \quad \text{read}^{-1}(b) \quad \text{read}^{0}(d_0) \quad \text{read}^{0}(b) \quad q_f \]

<table>
<thead>
<tr>
<th>Round</th>
<th>States</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>A, q_0</td>
</tr>
<tr>
<td>1</td>
<td>q_0, A, B, C</td>
</tr>
<tr>
<td>2</td>
<td>q_0, A</td>
</tr>
</tbody>
</table>

- Red = round 0
- Blue = round 1
- Yellow = round 2
An example of round-based register protocol

\[ \text{increment round} \]

\[ \text{write}(b) \]

\[ \text{read}^{-1}(a) \] \[ \text{read}^{-1}(d_0) \]

\[ \text{write}(a) \]

\[ \text{read}^{-1}(b) \]

\[ \text{read}^0(d_0) \] \[ \text{read}^0(b) \]

\[ q_f \]

\[ A \] = round 0

\[ B \] = round 1

\[ C \] = round 2

\( \vdots \)

\( 2 \)

\( 1 \)

\( 0 \)

\( d_0 \)

\( b \)

\( a \)
An example of round-based register protocol

\[
\begin{align*}
C & \xrightarrow{\text{read}^{-1}(a)} B \xleftarrow{\text{read}^{-1}(d_0)} A \xrightarrow{\text{write}(a)} q_0 \xrightarrow{\text{read}^{-1}(b)} D \xleftarrow{\text{read}^0(d_0)} E \xrightarrow{\text{read}^0(b)} q_f
\end{align*}
\]

\[
\begin{array}{c|c|c|c}
\text{Round} & A & B & C \\
0 & q_0 & A & a \\
1 & q_0 & A & B \\
2 & q_0 & & \\
\end{array}
\]

- \(\text{red} = \text{round } 0\)
- \(\text{blue} = \text{round } 1\)
- \(\text{yellow} = \text{round } 2\)

Increment round
An example of round-based register protocol

\[\text{write}(b)\]

\[\text{read}^{-1}(a)\]
\[\text{read}^{-1}(d_0)\]
\[\text{write}(a)\]
\[\text{read}^{-1}(b)\]
\[\text{read}^0(d_0)\]
\[\text{read}^0(b)\]
\[q_f\]

Increment round

<table>
<thead>
<tr>
<th>Round</th>
<th>Symbols</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>(q_0) A</td>
</tr>
<tr>
<td>1</td>
<td>(q_0) A, B, C</td>
</tr>
<tr>
<td>2</td>
<td>(q_0) A, D</td>
</tr>
</tbody>
</table>

- \(=\) round 0
- \(=\) round 1
- \(=\) round 2
An example of round-based register protocol

\[ \text{write}(b) \]

\[ \text{read}^{-1}(a) \quad \text{read}^{-1}(d_0) \quad \text{write}(a) \quad \text{read}^{-1}(b) \]

\[ \text{read}^0(d_0) \quad \text{read}^0(b) \]

\[ q_f \]

Increment round

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>q_0</td>
<td>A</td>
<td>B</td>
</tr>
<tr>
<td>q_0</td>
<td>A</td>
<td>C</td>
</tr>
<tr>
<td>q_0</td>
<td>D</td>
<td>E</td>
</tr>
</tbody>
</table>

\[ \cdot \]

\[ d_0 \]

\[ \cdot \]

\[ a \]

\[ b \]

= round 0

= round 1

= round 2
An example of round-based register protocol

To write $b$ to $\text{reg}[k]$, one must write to $\text{reg}[k]$ while $\text{reg}[k-1]$ still has value $d_0$.

To cover $q_f$ at round $k$, one must have written $b$ to $\text{reg}[k-1]$ while $\text{reg}[k]$ still has value $d_0$.

$q_f$ cannot be covered at any round!
Reachability problems in round-based setting

Round-based COVER: \[ \exists n, \exists \gamma_0, \exists \rho: \gamma_0 \rightarrow^* \gamma, \exists k \gamma(q_f, k) > 0? \]

There exists a round \( k \) such that some process is at round \( k \) and on state \( q_f \)
Reachability problems in round-based setting

Round-based COVER: \( \exists n, \exists \gamma_0, \exists \rho: \gamma_0 \rightarrow^* \gamma, \exists k \gamma(q_f, k) > 0 ? \)

Round-based TARGET: \( \exists n, \exists \gamma_0, \exists \rho: \gamma_0 \rightarrow^* \gamma, \forall k, \forall q \neq q_f, \gamma(q) = 0 ? \)

Every process is on state \( q_f \) regardless of its round
Reachability problems in round-based setting

Round-based COVER: \( \exists n, \exists \gamma_0, \exists \rho: \gamma_0 \rightarrow^* \gamma, \exists k \gamma(q_f, k) > 0 ? \)

Round-based TARGET: \( \exists n, \exists \gamma_0, \exists \rho: \gamma_0 \rightarrow^* \gamma, \forall k, \forall q \neq q_f, \gamma(q) = 0 ? \)

Round-based PRP: \( \exists n, \exists \gamma_0, \exists \rho: \gamma_0 \rightarrow^* \gamma, \gamma \vdash \psi ? \)

with \( \psi \) a first-order formula on rounds with no nested quantifiers, whose terms are in
\( \{\#(q, k + c) = 0, \#(q, k + c) > 0\} \cup \{\text{reg}_i[k + c] = d, \text{reg}_i[k + c] \neq d\} \)

Examples: \( \psi = \exists k (\#(q_1, k + 1) > 0 \land \text{reg}_i[k] = d) \lor \forall k \#(q_0, k) = 0'' \)

At some round, there is a process on state \( q_1 \) while register \( i \) of previous round has value \( d \)

no process is on \( q_0 \)
Complexity results

*Theorem*\textsuperscript{9}: Round-based PRP is PSPACE-hard.

This is already true when:

- the reachability objective is a coverability objective,
- the visibility range is equal to 0 (processes cannot see previous rounds),
- there is only one register per round,
- the registers are uninitialized.

*Parameterized safety verification of round-based shared-memory systems*. ICALP, 2022
Theorem\textsuperscript{9}: Round-based PRP is PSPACE-hard.
This is already true when:
• the reachability objective is a coverability objective,
• the visibility range is equal to 0 (processes cannot see previous rounds),
• there is only one register per round,
• the registers are uninitialized.

Theorem\textsuperscript{9,10}: Round-based PRP is PSPACE-complete.
A challenge: exponential lower bounds

Exponential lower bounds on the number of rounds:
A challenge: exponential lower bounds

Exponential lower bounds on the number *active* rounds:
A non-deterministic polynomial-space algorithm

Witness execution: \( \sigma_0 \xrightarrow{\theta_0} \sigma_1 \xrightarrow{\theta_1} \sigma_2 \xrightarrow{\theta_2} \sigma_3 \xrightarrow{\theta_3} \sigma_4 \xrightarrow{\theta_4} \sigma_5 \xrightarrow{\theta_5} \sigma_6 \xrightarrow{\theta_6} \sigma_7 \models \psi \)
A non-deterministic polynomial-space algorithm

Witness execution: \( \sigma_0 \xrightarrow{\theta_0} \sigma_1 \xrightarrow{\theta_1} \sigma_2 \xrightarrow{\theta_2} \sigma_3 \xrightarrow{\theta_3} \sigma_4 \xrightarrow{\theta_4} \sigma_5 \xrightarrow{\theta_5} \sigma_6 \xrightarrow{\theta_6} \sigma_7 \models \psi \)

Actions: \( \theta_0 \theta_1 \theta_2 \theta_3 \theta_4 \theta_5 \theta_6 \)

Rounds: 1 4 3 2 0 1 4
A non-deterministic polynomial-space algorithm

Witness execution:

\[ \sigma_0 \xrightarrow{\theta_0} \sigma_1 \xrightarrow{\theta_1} \sigma_2 \xrightarrow{\theta_2} \sigma_3 \xrightarrow{\theta_3} \sigma_4 \xrightarrow{\theta_4} \sigma_5 \xrightarrow{\theta_5} \sigma_6 \xrightarrow{\theta_6} \sigma_7 \models \psi \]

Actions: \[ \theta_0, \theta_1, \theta_2, \theta_3, \theta_4, \theta_5, \theta_6 \]

Rounds: \[ 1, 4, 3, 2, 0, 1, 4 \]
A non-deterministic polynomial-space algorithm
A non-deterministic polynomial-space algorithm

The number of relevant rounds may be large…

Storable in polynomial space?
A non-deterministic polynomial-space algorithm

storable in polynomial space using abstract representation

sliding window on \( v + 1 \) rounds

(here \( v = 1 \))
A non-deterministic polynomial-space algorithm
A non-deterministic polynomial-space algorithm
A non-deterministic polynomial-space algorithm

As the execution is guessed, we progressively guess why the configuration reached will satisfy $\psi$. 
From this algorithm, we obtain exponential upper bounds on the number of processes and rounds needed.

As the execution is guessed, we progressively guess why the configuration reached will satisfy $\psi$. 
Conclusion

Summary

• Two models in this talk: roundless register protocols and round-based register protocols.
• Properties studied are reachability properties which do not “count” processes. Two classical such problems are COVER and TARGET; PRP is a general class which encompasses these two problems.
• In the first model, despite its simplicity, PRP is NP-complete, but some restrictions make it PTIME.
• In the second model, PRP is PSPACE-complete, and similar restrictions do not decreases the complexity. The polynomial-space algorithm relies, among others, on a sliding window along the rounds.

Future work

• Almost-sure reachability in round-based register protocols with stochastic schedulers
• Cube reachability (= initial and final constraints may “count” processes), link with almost-sure reachability
• Weak memory
Binary consensus problem:
Make all processes agree on a common value, each process starting an initial preference \( p \).

Validity: If a process decided value \( p \), some process started with value \( p \)
Agreement: Two processes that decide decide of the same value
Termination: All processes eventually decide of a value

Aspnes’ consensus algorithm:\(^3\):
Example of execution of the algorithm

<table>
<thead>
<tr>
<th>rounds</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>3</td>
</tr>
<tr>
<td>4</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>reg_0[k]</th>
<th>reg_1[k]</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>no</td>
<td>no</td>
</tr>
</tbody>
</table>

\( \text{reg}_0[k] \)
Example of execution of the algorithm

A
B
C
reg_0[k]
reg_1[k]

rounds
0
1
2
3
4

writes

no
no
no
no
yes

\( \text{reg}_0[k] \)
\( \text{reg}_1[k] \)
Example of execution of the algorithm

<table>
<thead>
<tr>
<th>rounds</th>
<th>0</th>
<th>1</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>A</td>
<td>B</td>
<td>C</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>writes</th>
<th>0</th>
<th>1</th>
<th>yes</th>
<th>yes</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>reg₀[k]</td>
<td>reg₁[k]</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

no no no no no no yes yes
Example of execution of the algorithm

<table>
<thead>
<tr>
<th>rounds</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>$\text{reg}_0[k]$</th>
<th>$\text{reg}_1[k]$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td>no</td>
<td>no</td>
</tr>
</tbody>
</table>
Example of execution of the algorithm

<table>
<thead>
<tr>
<th>Rounds</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>(\text{reg}_0[k])</th>
<th>(\text{reg}_1[k])</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>no</td>
<td>no</td>
</tr>
</tbody>
</table>
Example of execution of the algorithm

<table>
<thead>
<tr>
<th>rounds</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>reg$_0$[k]</th>
<th>reg$_1$[k]</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td>1</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>2</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>3</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>4</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
Example of execution of the algorithm

<table>
<thead>
<tr>
<th>rounds</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>( \text{reg}_0[k] )</th>
<th>( \text{reg}_1[k] )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td>0</td>
<td>1</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td>no</td>
<td>no</td>
</tr>
</tbody>
</table>

Is ready to write its preference

\[ \text{reads} \]
Example of execution of the algorithm

\[ \text{reads} \]

\begin{align*}
\text{reg}_0[k] & \\
\text{reg}_1[k] & \\
\end{align*}

\begin{tabular}{cccc}
A & B & C & \text{reads} \\
0 & 1 & 1 & yes \\
\end{tabular}

\begin{tabular}{cccc}
0 & 1 & 1 & yes \\
\end{tabular}
Example of execution of the algorithm

\[ \text{reads} \]

<table>
<thead>
<tr>
<th>rounds</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>( \text{reg}_0[k] )</th>
<th>( \text{reg}_1[k] )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td></td>
<td>1</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>1</td>
<td></td>
<td>1</td>
<td></td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td>no</td>
<td>no</td>
</tr>
</tbody>
</table>
Example of execution of the algorithm

<table>
<thead>
<tr>
<th>rounds</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>( \text{reg}_0[k] )</th>
<th>( \text{reg}_1[k] )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td></td>
<td></td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
<td>1</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td>no</td>
<td>no</td>
</tr>
</tbody>
</table>
Example of execution of the algorithm

No winner on this round
Example of execution of the algorithm

<table>
<thead>
<tr>
<th>rounds</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>reg₀[k]</th>
<th>reg₁[k]</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>no</td>
<td>no</td>
</tr>
</tbody>
</table>
Example of execution of the algorithm

reads

<table>
<thead>
<tr>
<th>rounds</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>...</td>
</tr>
</tbody>
</table>

A | B | C | \( \text{reg}_0[k] \) | \( \text{reg}_1[k] \)
Example of execution of the algorithm

<table>
<thead>
<tr>
<th>rounds</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>( \text{reg}_0[k] )</th>
<th>( \text{reg}_1[k] )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td></td>
<td>1</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td>1</td>
<td></td>
<td>1</td>
<td></td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td>no</td>
<td>no</td>
</tr>
</tbody>
</table>

\[ \text{writes} \]
Example of execution of the algorithm

<table>
<thead>
<tr>
<th>rounds</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>reg_0[k]</th>
<th>reg_1[k]</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td></td>
<td></td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td>1</td>
<td>yes</td>
<td></td>
<td></td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td>2</td>
<td>no</td>
<td></td>
<td></td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td>yes</td>
<td>yes</td>
</tr>
</tbody>
</table>

reads

\[
\text{reg}_0[k] = \begin{cases} 
    \text{true} & \text{if reads at round } k \text{ is yes} \\
    \text{false} & \text{otherwise}
\end{cases}
\]

\[
\text{reg}_1[k] = \begin{cases} 
    \text{true} & \text{if reads at round } k \text{ is yes} \\
    \text{false} & \text{otherwise}
\end{cases}
\]
Example of execution of the algorithm

<table>
<thead>
<tr>
<th>Rounds</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>reg₀[k]</th>
<th>reg₁[k]</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td></td>
<td></td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td></td>
<td></td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>2</td>
<td></td>
<td>1</td>
<td></td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td>1</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td>no</td>
<td>no</td>
</tr>
</tbody>
</table>

\[ \text{r} \left[ k \right] \quad \text{g} \]
Example of execution of the algorithm

<table>
<thead>
<tr>
<th>rounds</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>\text{reads}</th>
<th>\text{reg}_0[k]</th>
<th>\text{reg}_1[k]</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td>no</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td>2</td>
<td>no</td>
<td></td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
</tbody>
</table>
Example of execution of the algorithm

reads

<table>
<thead>
<tr>
<th>rounds</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>( \text{reg}_0[k] )</th>
<th>( \text{reg}_1[k] )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>no</td>
<td>yes</td>
</tr>
</tbody>
</table>
Example of execution of the algorithm

A  B  C  \text{reg}_0[k]  \text{reg}_1[k]

\begin{array}{c|c|c}
\text{rounds} & \text{writes} & \text{...}
\hline
0 & \text{no} & \text{no}
\hline
1 & \text{yes} & \text{yes}
\hline
2 & \text{no} & \text{yes}
\hline
3 & \text{no} & \text{yes}
\hline
4 & \text{no} & \text{yes}
\end{array}
Example of execution of the algorithm

<table>
<thead>
<tr>
<th>rounds</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>reg₀[k]</th>
<th>reg₁[k]</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Example of execution of the algorithm

A process getting to this round will convert to preference 1