Checking Presence Reachability Properties on Parameterized Shared-Memory Systems

Nicolas Waldburger
Univ Rennes, Inria, CNRS, IRISA, France

Abstract

We consider the verification of distributed systems composed of an arbitrary number of asynchronous processes. Processes are identical finite-state machines that communicate by reading from and writing to a shared memory. Beyond the standard model with finitely many registers, we tackle round-based shared-memory systems with fresh registers at each round. In the latter model, both the number of processes and the number of registers are unbounded, making verification particularly challenging. The properties studied are generic presence reachability objectives, which subsume classical questions such as safety or synchronization by expressing the presence or absence of processes in some states. In the more general round-based setting, we establish that the parameterized verification of presence reachability properties is PSPACE-complete. Moreover, for the roundless model with finitely many registers, we prove that the complexity drops down to NP-complete and we provide several natural restrictions that make the problem solvable in polynomial time.

2012 ACM Subject Classification Theory of computation → Verification by model checking; Theory of computation → Distributed algorithms

Keywords and phrases Verification, Parameterized models, Distributed algorithms

Digital Object Identifier 10.4230/LIPIcs.MFCS.2023.88

Related Version *Full Version*: https://arxiv.org/abs/2306.17476

Acknowledgements Many thanks to Nathalie Bertrand, Nicolas Markey and Ocan Sankur for their invaluable advice.

1 Introduction

Parameterized verification. Distributed systems consist of multiple processes running in parallel. Verification of such systems is a major topic of modern verification, because of how common these systems are and how difficult their verification has proven to be. Indeed, when multiple processes run asynchronously, the number of relevant interleavings to consider quickly becomes large. An intuitive approach for their verification is to fix the number of processes involved and try to apply classical verification techniques. Another approach is that of parameterized verification, where one aims to prove the more general statement that the property of interest holds for any number of participants. The interest of this approach is threefold. First, it allows to prove that the system is correct regardless of the number of processes. Second, the efficiency of parameterized techniques does not depend on the number of participants, which makes them more suitable for large systems for which classical techniques scale poorly. Third, parameterized verification often yields decidability or better computational complexity for problems that are hard to solve with classical techniques; see for example [14] for a problem that becomes decidable in the parameterized case. In their seminal work [13], German and Sistla consider systems consisting of a leader and arbitrarily many contributors, all of which are finite-state machines communicating via *rendez-vous*. In this setting, the safety verification problem is EXPSPACE-complete and the complexity drops down to polynomial time when the leader is removed. Since then, many similar models have been studied, with variations on the expressiveness of the processes and the means of communication in order to capture the large variety of existing distributed algorithms [10, 7].

Contributions. We study parameterized verification of systems where all processes are identical and anonymous finite-state machines that communicate via reading from and writing to a shared memory. The read and write actions are performed non-atomically, meaning that no process may perform a read-write combination while preventing all other processes from acting. Our registers are *initialized* with a special symbol; this assumption is common in parameterized verification of shared-memory systems [8, 1], since some algorithms require initialized registers, *e.g.* [2]. First, we study a model with finitely many registers. This model is inspired by [11] where registers were uninitialized and the verification is restricted to safety properties. In contrast, we study the more general *presence reachability problems*, in which one asks whether one may reach a configuration that satisfies a property. This property takes the form of a Boolean combination of constraints expressing whether there is at least one process in a given state of the finite-state machine. We prove that this problem is NP-complete and we provide several natural restrictions on the process description and on the property that make the problem solvable in polynomial time. We then work on the more general setting of *round-based* shared-memory systems [6], which are designed to model round-based shared-memory algorithms present in the literature, see *e.g.* [2, 15]. In this model, the processes proceed in asynchronous rounds, each round having its own fresh set of registers. The source of infinity is twofold, as the number of processes and the number of registers are both unbounded, making round-based systems particularly challenging to verify. The safety problem was proved to be PSPACE-complete in round-based shared-memory systems [6]. In this article, we go beyond safety by considering a round-based, richer version of the presence reachability problem where the property may quantify existentially and universally over the rounds. Nonetheless, we establish that the round-based presence reachability problem is PSPACE-complete.

Related work. Similar models and problems have been studied in the literature. In the shared-memory model (without rounds and without register initialization), the safety problem has been studied extensively with variations on the expressiveness given to the leader and the contributors [11]; in particular, when processes are finite-state machines, the safety problem is shown to be coNP-complete and to decrease to PTIME when the leader is removed. However, this result does not hold when registers are initialized or when the property is more general than safety. A model that has perhaps been more studied is that of *reconfigurable broadcast networks* (RBN), where processes communicate via broadcasting messages that can be received by any of the other processes. This model has similarities with shared-memory systems , although broadcast tends to be simpler (messages disappear after being sent, while written values remain in the registers). A source of inspiration for the first part of our article is the study of reachability problems in RBN [9], where it is shown that the cardinality reachability problem, where one wants to reach a configuration that satisfies cardinality constraints, is PSPACE-complete. When the constraints cannot count processes, this problem is analogous to our presence reachability problem; for RBN, it is shown to be NP-complete, a complexity that we also obtain in our setting. Finally, this complexity drops down to PTIME in RBN when considering the special case of safety. This tractability result no longer holds in the shared-memory world unless we make further assumptions about the number of registers or their initialization. The cube reachability problem is a generalization of the cardinality constraint problem where the initial configuration is also subject to cardinality constraints; this problem is PSPACE-complete both in RBN and in (roundless) asynchronous shared-memory systems [9, 5, 4], although it is unknown whether this remains true when allowing the Pre^* and Post^* operators in the description of the cubes [4, 3]. While it is

interesting to compare results on RBN with our results on shared-memory systems without rounds, such a comparison is not possible with the more expressive model of round-based shared-memory systems, in particular because the unboundedness in the number of registers has no equivalent in broadcast networks.

Due to space constraints, some details are omitted. These details can be found in the full version of this paper.

2 Roundless Register Protocols

In this section, we introduce *register protocols*, a model inspired by [11]. We call these systems *roundless* to distinguish them from *round-based* systems introduced later in this article.

2.1 Definitions

▶ **Definition 1** (Roundless register protocols). *A* roundless register protocol *is a tuple* $\mathcal{P} = \langle Q, Q_0, \mathsf{dim}, \mathrm{D}, \mathsf{d}_0, \Delta \rangle$ *where*

- Q *is a finite set of* states *with a distinguished subset of* initial states $Q_0 \subseteq Q$*;*
- $\mathsf{dim} \in \mathbb{N}$ *is the number of shared* registers*;*
- D *is a finite* data alphabet *containing the initial symbol* d_0*;*
- $\Delta \subseteq Q \times \mathcal{A} \times Q$ *is the set of* transitions*, where* $\mathcal{A} := \{\mathsf{read}_\alpha(\mathsf{d}) \mid \alpha \in [1, \mathsf{dim}], \mathsf{d} \in \mathrm{D}\} \cup \{\mathsf{write}_\alpha(\mathsf{d}) \mid \alpha \in [1, \mathsf{dim}], \mathsf{d} \in \mathrm{D} \setminus \{\mathsf{d}_0\}\}$ *is the set of* actions*.*

Roundless register protocols are executed on multiple processes that behave asynchronously and can only communicate via reading from and writing to the shared registers. The behavior of a process is described by a finite-state machine. The possible actions of the transitions are reading a symbol from and writing a symbol to one of the dim shared registers; $\mathsf{d} \in \mathrm{D}$ denotes the symbol and α indicates the register on which the action is performed. Each register stores one *symbol* from the finite set D at a time. Read-write combinations are performed non-atomically, i.e., no process can perform a read-write combination while excluding all other processes. The *size* of the protocol $\mathcal{P}$ is defined as $|\mathcal{P}| := |Q| + |\mathrm{D}| + |\Delta| + \mathsf{dim}$. For all $\alpha \in [1, \mathsf{dim}]$, we write $\mathsf{rg}[\alpha]$ for the register of index α. We also write Reg for the set $\{\mathsf{rg}[\alpha] \mid \alpha \in [1, \mathsf{dim}]\}$ of all registers.

Processes are assumed to have no identifiers so they are identical anonymous agents. Therefore, a *configuration* is a pair $\gamma = \langle \mu, \vec{d} \rangle \in \mathbb{N}^Q \times \mathrm{D}^{\mathsf{Reg}}$ such that $0 < \sum_{q \in Q} \mu(q) < \infty$. Let $\mathsf{st}(\gamma) := \mu$ which indicates the number of processes in each state, and $\mathsf{data}(\gamma) := \vec{d}$ mapping to each register its symbol: for all $\mathsf{r} \in \mathsf{Reg}$, $\mathsf{data}(\gamma)(\mathsf{r})$ is the symbol contained in register r in γ. Let $\Gamma := \mathbb{N}^Q \times \mathrm{D}^{\mathsf{Reg}}$ denote the set of all configurations. Let $\mathsf{supp}(\gamma) := \{q \in Q \mid \mathsf{st}(\gamma)(q) > 0\}$ denote the support of the multiset $\mathsf{st}(\gamma)$. We write $\oplus$ and $\ominus$ the operations on multisets that add and remove elements, respectively. A configuration is *initial* if all processes are in states from Q_0 while all registers have value d_0. We denote by Init_c the set of initial configurations (the letter c stands for "concrete" as opposed to "abstract" configurations defined later). Formally, $\mathsf{Init}_\mathsf{c} := \{\gamma \mid \mathsf{st}(\gamma) \subseteq Q_0, \mathsf{data}(\gamma) = \mathsf{d}_0^{\mathsf{Reg}}\}$.

Given $\gamma, \gamma' \in \Gamma$, γ' is a *successor* of γ when there exists $\delta = (q, a, q') \in \Delta$ such that $\mathsf{st}(\gamma)(q) > 0$, $\mathsf{st}(\gamma') = (\mathsf{st}(\gamma) \ominus \{q\}) \oplus \{q'\}$ and:

- if $a = \mathsf{read}_\alpha(\mathsf{d})$ then $\mathsf{data}(\gamma)(\mathsf{rg}[\alpha]) = \mathsf{d}$ and $\mathsf{data}(\gamma') = \mathsf{data}(\gamma)$,
- if $a = \mathsf{write}_\alpha(\mathsf{d})$ then $\mathsf{data}(\gamma')(\mathsf{rg}[\alpha]) = \mathsf{d}$ and $\forall \alpha' \neq \alpha$, $\mathsf{data}(\gamma')(\mathsf{rg}[\alpha']) = \mathsf{data}(\gamma)(\mathsf{rg}[\alpha'])$.

In that case, we write $\gamma \xrightarrow{\delta} \gamma'$ or simply $\gamma \to \gamma'$, which is called a *step*. . A *concrete execution* is a sequence $\pi = \gamma_0, \delta_1, \gamma_1, \ldots, \gamma_{l-1}, \delta_l, \gamma_l$ such that for all i, $\gamma_i \xrightarrow{\delta_{i+1}} \gamma_{i+1}$. We write $\gamma_0 \xrightarrow{*} \gamma_l$

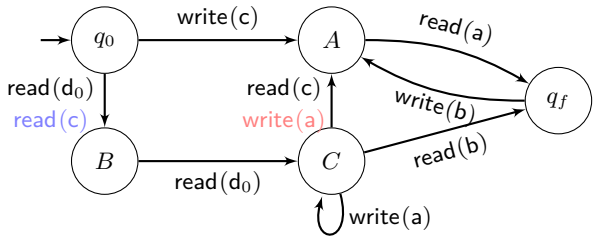

Figure 1 An example of a protocol.

for the existence of such an execution. γ' is *reachable from* γ when $\gamma \xrightarrow{*} \gamma'$. Given a set C of configurations, we write $\mathsf{Reach_c}(C) := \{\gamma' \mid \exists \gamma \in C, \gamma \xrightarrow{*} \gamma'\}$. A configuration is *reachable* when it is in $\mathsf{Reach_c}(\mathsf{Init_c})$.

▶ **Example 2.** Figure 1 provides an example of a roundless register protocol $\mathcal{P}$ with $\mathrm{D} = \{\mathsf{d}_0, \mathsf{a}, \mathsf{b}, \mathsf{c}\}$, $Q_0 = \{q_0\}$ and $\mathsf{dim} = 1$, hence read and write actions are implicitly on register $\alpha = 1$. The red and blue labels are to be ignored for now.

The set of initial configurations is $\mathsf{Init_c} := \{\langle q_0^n, \mathsf{d}_0\rangle \mid n \geq 1\}$. The following execution with two processes witnesses that $\langle q_f \oplus C, \mathsf{a}\rangle \in \mathsf{Reach_c}(\mathsf{Init_c})$:

$\langle q_0^2, \mathsf{d}_0\rangle \xrightarrow{(q_0,\mathsf{read}(\mathsf{d}_0),B)} \langle q_0 \oplus B, \mathsf{d}_0\rangle \xrightarrow{(B,\mathsf{read}(\mathsf{d}_0),C)} \langle q_0 \oplus C, \mathsf{d}_0\rangle \xrightarrow{(q_0,\mathsf{write}(\mathsf{c}),A)}$

$\langle A \oplus C, \mathsf{c}\rangle \xrightarrow{(C,\mathsf{write}(\mathsf{a}),C)} \langle A \oplus C, \mathsf{a}\rangle \xrightarrow{(A,\mathsf{read}(\mathsf{a}),q_f)} \langle q_f \oplus C, \mathsf{a}\rangle.$ ⌟

2.2 Reachability Problems

Our first problem of interest is the *coverability problem* (COVER):

COVER FOR ROUNDLESS REGISTER PROTOCOLS
Input: A roundless register protocol $\mathcal{P}$, $q_f \in Q$
Question: Does there exist $\gamma \in \mathsf{Reach_c}(\mathsf{Init_c})$ such that $\mathsf{st}(\gamma)(q_f) > 0$?

Note that, because the model is parameterized, a witness execution of COVER may have an arbitrarily large number of processes. The dual is the *safety problem*, the answer to which is yes when an error state cannot be covered regardless of the number of processes. A similar problem is the *target problem* (TARGET) where processes must synchronize at q_f :

TARGET FOR ROUNDLESS REGISTER PROTOCOLS
Input: A roundless register protocol $\mathcal{P}$, $q_f \in Q$
Question: Does there exist $\gamma \in \mathsf{Reach_c}(\mathsf{Init_c})$ s.t. for all $q \neq q_f$, $\mathsf{st}(\gamma)(q) = 0$?

▶ Remark 3. TARGET is harder than COVER: consider the reduction in which one adds a loop on q_f writing a joker symbol which, from any state, may be read to reach q_f.

Presence constraints are Boolean combinations (with $\wedge$, $\vee$ and $\neg$) of atomic propositions of the form "q populated" with $q \in Q$, or of the form "r contains d" with $\mathsf{r} \in \mathsf{Reg}$ and $\mathsf{d} \in \mathrm{D}$. A presence constraint is interpreted over a configuration γ by interpreting "q populated" as true if and only if $\mathsf{st}(\gamma)(q) > 0$ and "r contains d" as true if and only if $\mathsf{data}(\gamma)(\mathsf{r}) = \mathsf{d}$. Note that presence constraints cannot refer to how many processes are on a given state. We write $\gamma \models \phi$ when configuration γ satisfies presence constraint ϕ.

▶ **Example 4.** If $Q = \{q_1, q_2, q_3\}$, $\mathsf{dim} = 2$, $\mathrm{D} = \{\mathsf{d}_0, a, b\}$ and $\phi := (q_1 \text{ populated}) \vee ((q_2 \text{ populated}) \wedge (\mathsf{rg}[1] \text{ contains } a))$ then $\langle q_1 \oplus q_3, \mathsf{d}_0^2\rangle \models \phi$, $\langle q_2^2, (a, b)\rangle \models \phi$ but $\langle q_2^2, b^2\rangle \not\models \phi$. ⌟

The *Presence Reachability Problem* (PRP) generalizes both COVER and TARGET. It corresponds to the cardinality reachability problem for cardinality constraints restricted to $\mathrm{CC}[\geq 1, = 0]$ studied for broadcast protocols [9].

PRP FOR ROUNDLESS REGISTER PROTOCOLS
Input: A roundless register protocol $\mathcal{P}$, a presence constraint ϕ
Question: Does there exist $\gamma \in \mathsf{Reach}_{\mathsf{c}}(\mathsf{Init}_{\mathsf{c}})$ such that $\gamma \models \phi$?

The formula ϕ automatically makes PRP NP-hard, since one can encode the SAT problem. Therefore, we also consider the *DNF Presence Reachability Problem* (DNF-PRP), in which ϕ is in *disjunctive normal form*. COVER and TARGET are special cases of DNF-PRP, with $\phi = (q_f \text{ populated})$ for COVER and $\phi = \bigwedge_{q \neq q_f} \neg(q \text{ populated})$ for TARGET.

▶ **Example 5.** Consider again the protocol $\mathcal{P}$ defined in Figure 1. $(\mathcal{P}, q_f)$ is a positive instance of COVER, as proved in Example 2. Let $\mathcal{P}_{\mathsf{blue}}$ be the protocol obtained from $\mathcal{P}$ by changing to $\mathsf{read}(\mathsf{c})$ the label of the transition from q_0 to B (in blue in Figure 1). $(\mathcal{P}_{\mathsf{blue}}, q_f)$ is a negative instance of COVER. In fact, a process can only get to B if c has been written to the register, and then d_0 can no longer be read so no process may go to state C, a cannot be written and no process may go from A to q_f.

$(\mathcal{P}, q_f)$ is a negative instance of TARGET: to leave A, one needs to read a, hence must have a process on state C, and to leave C, one must read b which would force us to send a process to A. Let $\mathcal{P}_{\mathsf{red}}$ be the protocol obtained from $\mathcal{P}$ by changing to $\mathsf{write}(\mathsf{a})$ the label of the transition from C to A (in red in Figure 1). $(\mathcal{P}_{\mathsf{red}}, q_f)$ is a positive instance of TARGET:
$\langle q_0^2, \mathsf{d}_0\rangle \xrightarrow{(q_0, \mathsf{read}(\mathsf{d}_0), B)} \langle q_0 \oplus B, \mathsf{d}_0\rangle \xrightarrow{(B, \mathsf{read}(\mathsf{d}_0), C)} \langle q_0 \oplus C, \mathsf{d}_0\rangle \xrightarrow{(q_0, \mathsf{write}(\mathsf{c}), A)}$
$\langle A \oplus C, \mathsf{c}\rangle \xrightarrow{(C, \mathsf{write}(\mathsf{a}), A)} \langle A^2, \mathsf{a}\rangle \xrightarrow{(A, \mathsf{read}(\mathsf{a}), q_f)} \langle A \oplus q_f, \mathsf{a}\rangle \xrightarrow{(A, \mathsf{read}(\mathsf{a}), q_f)} \langle q_f^2, \mathsf{a}\rangle$.
Let $\phi := \neg(C \text{ populated}) \wedge ((\mathsf{rg} \text{ contains } \mathsf{a}) \vee [(\mathsf{rg} \text{ contains } \mathsf{b}) \wedge \neg(A \text{ populated})])$. ϕ is a presence constraint and $(\mathcal{P}, \phi)$ is a negative instance of PRP. Indeed, if a is in the register, then C must be populated and if b is in the register, then A must be populated. ⌟

2.3 Abstract Semantics

In this subsection, we define an abstraction of the semantics that is sound and complete with respect to PRP. The intuition of this abstraction is that the exact number of processes in a given state is not relevant. Indeed, register protocols, thanks to non-atomicity, enjoy a classical monotonicity property named copycat property.

▶ **Lemma 6** (Copycat). *Consider γ_1, γ_2, q_2 such that $\gamma_1 \xrightarrow{*} \gamma_2$, $q_2 \in \mathsf{supp}(\gamma_2)$. There exists $q_1 \in \mathsf{supp}(\gamma_1)$ s.t. $\langle \mathsf{st}(\gamma_1) \oplus q_1, \mathsf{data}(\gamma_1)\rangle \xrightarrow{*} \langle \mathsf{st}(\gamma_2) \oplus q_2, \mathsf{data}(\gamma_2)\rangle$.*

An *abstract configuration* is a pair $\sigma = \langle \mathsf{st}(\sigma), \mathsf{data}(\sigma)\rangle \in 2^Q \times \mathrm{D}^{\mathsf{Reg}}$ such that $\mathsf{st}(\sigma) \neq \emptyset$. The set of *initial configurations* is $\mathsf{Init}_{\mathsf{a}} := \{\langle S, \mathsf{d}_0^{\mathsf{dim}}\rangle \mid S \subseteq Q_0\}$. Given a concrete configuration γ, the projection $\mathsf{abst}(\gamma)$ is the abstract configuration $\langle \mathsf{supp}(\gamma), \mathsf{data}(\gamma)\rangle$. Let $\Sigma := 2^Q \times \mathrm{D}^{\mathsf{Reg}}$ denote the set of abstract configurations. For $\sigma, \sigma' \in \Sigma$, σ' is the *successor* of σ when there exists $\delta = (q, a, q') \in \Delta$ such that $q \in \mathsf{st}(\gamma)$, either $\mathsf{st}(\gamma') = \mathsf{st}(\gamma) \cup \{q'\}$ or $\mathsf{st}(\gamma') = (\mathsf{st}(\gamma) \setminus \{q\}) \cup \{q'\}$, and: if $a = \mathsf{read}_\alpha(\mathsf{d})$ then $\mathsf{data}(\gamma)(\mathsf{rg}[\alpha]) = \mathsf{d}$ and $\mathsf{data}(\sigma) = \mathsf{data}(\sigma')$, and if $a = \mathsf{write}_\alpha(\mathsf{d})$ then $\mathsf{data}(\sigma')(\mathsf{rg}[\alpha]) = \mathsf{d}$ and for all $\alpha' \neq \alpha$, $\mathsf{data}(\sigma')(\mathsf{rg}[\alpha']) = \mathsf{data}(\sigma)(\mathsf{rg}[\alpha'])$. Again, we denote such a step by $\sigma \xrightarrow{\delta} \sigma'$ or $\sigma \to \sigma'$. Note that one could equivalently define $\sigma \xrightarrow{\delta} \sigma'$ by: $\sigma \xrightarrow{\delta} \sigma' \iff \exists \gamma, \gamma' \in \Gamma$, $\gamma \xrightarrow{\delta} \gamma'$ and $\mathsf{abst}(\gamma) = \sigma, \mathsf{abst}(\gamma') = \sigma'$. This notion of abstraction is classical in parameterized verification of systems with identical anonymous agents that enjoy monotonicity properties. Note, however, that this semantics is

non-deterministic: one could have $\sigma'' \neq \sigma'$ such that $\sigma \xrightarrow{\delta} \sigma'$ and $\sigma \xrightarrow{\delta} \sigma''$. This alternative corresponds to whether all processes in q take the transition ($\mathsf{st}(\gamma') = (\mathsf{st}(\gamma) \setminus \{q\}) \cup \{q'\}$) or only some ($\mathsf{st}(\gamma') = \mathsf{st}(\gamma) \cup \{q'\}$). We define *abstract executions* similarly to concrete ones, and denote them using ρ. We also define the *reachability set* $\mathsf{Reach}_{\mathsf{a}}(A)$ and the notion of *coverability* as in the concrete case. This abstraction is sound and complete for PRP:

▶ **Proposition 7** (Soundness and completeness of the abstraction). *For all $S \subseteq Q$, $\vec{d} \in \mathrm{D}^{\mathsf{Reg}}$:*

$$(\exists \gamma \in \mathsf{Reach}_{\mathsf{c}}(\mathsf{Init}_{\mathsf{c}}) : \mathsf{supp}(\gamma){=}S,\ \mathsf{data}(\gamma){=}\vec{d}) \iff (\exists \sigma \in \mathsf{Reach}_{\mathsf{a}}(\mathsf{Init}_{\mathsf{a}}) :\ \mathsf{st}(\sigma){=}S,\ \mathsf{data}(\sigma){=}\vec{d}).$$

The intuition of the proof is the following: any concrete configuration can easily be lifted into an abstract one. Conversely, any abstract execution may be simulated in the concrete semantics for a sufficiently large number of processes by using the copycat property.

Given a presence constraint ϕ and $\sigma \in \Sigma$, we define whether σ satisfies ϕ, written $\sigma \models \phi$, in a natural way. Given a concrete configuration γ, one has $\gamma \models \phi$ if and only if $\mathsf{abst}(\gamma) \models \phi$. Indeed, γ and $\mathsf{abst}(\gamma)$ have the same populated states and register values. Therefore, there exists $\gamma \in \mathsf{Reach}_{\mathsf{c}}(\mathsf{Init}_{\mathsf{c}})$ such that $\gamma \models \phi$ if and only if there exists $\sigma \in \mathsf{Reach}_{\mathsf{a}}(\mathsf{Init}_{\mathsf{a}})$ such that $\sigma \models \phi$: one can consider PRP directly in the abstract semantics.

3 Complexity Results for Roundless Register Protocols

In this section, we provide complexity results for the presence reachability problems defined above in the general case and in some restricted cases. Throughout the rest of the section, all configurations and executions are implicitly abstract.

3.1 NP-Completeness of the General Case

First, all problems defined in the previous section are NP-complete.

▶ **Proposition 8.** *COVER, TARGET, DNF-PRP and PRP for roundless register protocols are all NP-complete.*

Proof. First, we prove that all four problems are in NP. It suffices to prove it for PRP, as the three other problems reduce to it.

Let $\rho : \sigma_0 \xrightarrow{*} \sigma$ an abstract execution, we simply prove the existence of $\rho' : \sigma_0 \xrightarrow{*} \sigma$ of length at most $4|Q|$. To obtain ρ' from ρ, we iteratively:

- remove any read step that is non-deserting and does not cover a new location,
- remove any write step that is non-deserting, does not populate a new state and whose written symbol is never read,
- make non-deserting any deserting step whose source state is populated again later in ρ.

In ρ', at most $|Q|$ steps populate a new state and at most $|Q|$ steps are deserting. This implies that there are at most $2|Q|$ read steps, therefore, at most $2|Q|$ write steps whose written value is actually read. In total, this bounds the number of steps by $4|Q|$. In particular, for PRP, we can look for an execution of length less than $4|Q|$ which can be guessed in polynomial time.

We prove NP-hardness of COVER, as it reduces to the three other problems.

The proof is by a reduction from 3-SAT. Consider a 3-CNF formula $\phi = \bigwedge_{i=1}^{m} l_{i,1} \vee l_{i,2} \vee l_{i,3}$ over n variables $x_1, \ldots, x_n$ where, for all $i \in [1, m]$, for all $k \in [1, 3]$, $l_{i,k} \in \{x_j, \neg x_j \mid j \in [1, n]\}$. We define a roundless register protocol $\mathcal{P}_{\mathrm{SAT}}(\phi)$ with a distinguished state q_f which is coverable if and only if ϕ is satisfiable. In $\mathcal{P}_{\mathrm{SAT}}(\phi)$, one has $\mathrm{D} = \{\mathsf{d}_0, \mathsf{T}\}$ and $\mathsf{dim} = 2n$, there are two registers for each variable x_i, $\mathsf{rg}(x_i)$ and $\mathsf{rg}(\neg x_i)$. The protocol is represented on Figure 2.

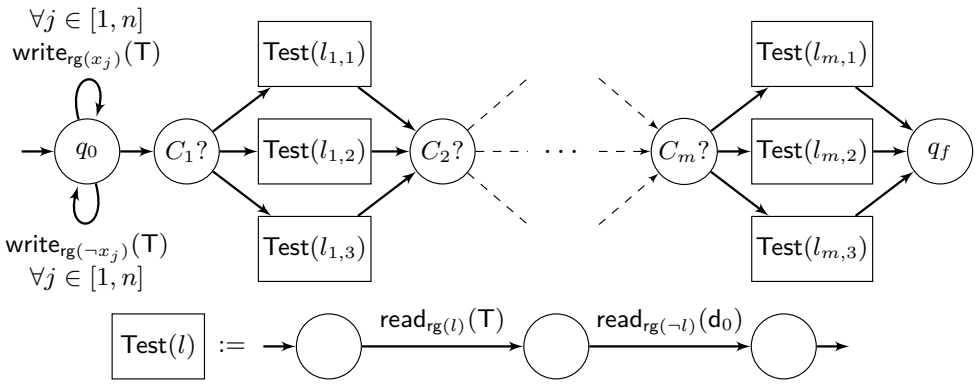

Figure 2 The protocol $\mathcal{P}_{\mathrm{SAT}}(\phi)$ for NP-hardness of COVER.

While any register may be set to T thanks to the loops on q_0, a register set to T can never be set back to d_0. l is considered true if $\mathsf{rg}(l)$ is set to T while $\mathsf{rg}(\neg l)$ still has value d_0. Suppose that the instance of 3-SAT is positive, *i.e.*, ϕ is satisfiable by some assignment ν. Consider an execution that writes T exactly to all $\mathsf{rg}(l)$ with l true in ν. For each clause, one of the three literals is true in ν. Therefore the execution may cover $C_i?$ for all i so it may cover q_f and the instance of COVER is positive. Conversely, if the instance COVER is positive, there exists an execution $\rho : \sigma_0 \xrightarrow{*} \sigma$ with $\sigma_0 \in \mathsf{Init}_{\mathsf{a}}$ and $q_f \in \mathsf{st}(\sigma)$. Consider ν that assigns to each variable x value true if $\mathsf{rg}(x)$ is written before $\mathsf{rg}(\neg x)$ in ρ and false otherwise. Given a litteral l, ρ may only go through $\mathsf{Test}(l)$ if $\nu(l)$ is true; because ρ covers q_f, this proves that $\nu \models \phi$. ◀

▶ Remark 9. In [11], the authors prove NP-completeness of COVER in a similar model, but with a leader: in the NP-hardness reduction, the leader make non-determinstic decisions about the values of the variable. This argument does not hold in the leaderless case.

3.2 Interesting Restrictions

Although all the problems defined above are NP-complete, they are sometimes tractable under appropriate restrictions on the protocols. We will consider two restrictions on the protocols. The first one is having $\mathsf{dim} = 1$, *i.e.*, a single register. The second restriction is the *uninitialized case* where processes are not allowed to read the initial value d_0 from the registers. Formally, a protocol $\mathcal{P}$ is *uninitialized* if its set of transitions Δ does not contain an action reading symbol d_0: in uninitialized protocols, it is structurally impossible to read from an unwritten register. One might object that forbidding transitions that read d_0 contradicts the intuition that, when a process reads from a register, it does not know whether the value is initial or not; one could settle the issue by considering that reading d_0 sends processes to a sink state. The uninitialized setting tends to yield better complexity than the general, initialized case, see for example [1, Section 7].

Of course, for PRP, the formula itself always makes the problem NP-hard.

▶ **Proposition 10.** *PRP for roundless register protocols is NP-hard even with* $\mathsf{dim} = 1$ *and the register uninitialized.*

3.3 Tractability of COVER and DNF-PRP under Restrictions

In this subsection, we prove that COVER is solvable in PTIME when the protocol is uninitialized or when dim is fixed and that DNF-PRP is solvable is PTIME when $\mathsf{dim} = 1$.

In [11, Theorem 9.2], uninitialized COVER is proved to be PTIME-complete; their approach, based on languages, is quite different from the one presented here. Our approach, similar to the one presented in [9, Algorithm 1] in the setting of reconfigurable broadcast networks, is to compute the set of coverable states using a simple *saturation* technique, a fixed-point computation over the set of states.

When registers are initialized, the saturation technique breaks down as it may be that some states are coverable but not in the same execution, as they require registers to lose their initial value in different orders (see the notion of first-write order developed in [6] for more development on this in a round-based setting). However, in the initialized case with a fixed number of registers, one can iterate over every such order and COVER is tractable as well.

▶ **Proposition 11.** *COVER for roundless register protocols is PTIME-complete either when the registers are uninitialized or when* dim *is fixed.*

For DNF-PRP, we provide a PTIME algorithm in the more restrictive case of $\mathsf{dim} = 1$.

▶ **Proposition 12.** *DNF-PRP for roundless register protocols with* $\mathsf{dim} = 1$ *is in PTIME.*

Proof sketch. We give here the proof for TARGET. Our algorithm shares similarities with [12, page 41] for broadcast protocols, although it is more complex because of the persistence of symbols in the register.

First, we have a polynomial reduction from initialized TARGET with $\mathsf{dim} = 1$ to uninitialized TARGET with $\mathsf{dim} = 1$. It proceeds as follows. Consider the graph $G = (Q, E)$ when $(q_1, q_2) \in E$ when there exists $(q_1, \mathsf{read}(\mathsf{d}_0), q_2) \in \Delta$. Let $I \subseteq Q$ the set of states that are reachable in G from Q_0. The reduction simply replaces Q_0 by I as set of initial states.

Any (abstract) execution $\rho : \sigma_0 \xrightarrow{*} \langle q_f, \mathsf{d_f}\rangle$, called *synchronizing execution*, can be rearranged into $\rho_+ : \sigma_0 \xrightarrow{*} \langle S, \mathsf{d}\rangle$ and $\rho_- : \langle S, \mathsf{d}\rangle \xrightarrow{*} \langle q_f, \mathsf{d_f}\rangle$ where S contains all states that appear in ρ. Additionally, we can make ρ_- start with a write action (there is a transition in ρ that writes d). To obtain the decomposition, ρ_+ mimics ρ but does not empty any state, and ρ_- mimics ρ but from a configuration with more states. We compute the maximum such set S by iteratively deleting states that cannot appear in any synchronizing execution. Let

$$\mathcal{C}(\mathcal{P}) := \max\{S \subseteq Q \mid \exists \mathsf{d} \in \mathrm{D},\ \exists \sigma_0 \in \mathsf{Init_a},\ \sigma_0 \xrightarrow{*} \langle S, \mathsf{d}\rangle\}$$
$$\mathcal{BC}(\mathcal{P}) := \max\{S \subseteq Q \mid \forall \mathsf{d} \in \mathrm{D},\ \exists \mathsf{d_f} \in \mathrm{D},\ \langle S, \mathsf{d}\rangle \xrightarrow{*} \langle q_f, \mathsf{d_f}\rangle\}$$

Both maxima exist as the sets are non-empty (Q_0 is included in the first set and q_f is in the second set) and they are stable by union (concatenate the corresponding executions). Intuitively, $\mathcal{C}(\mathcal{P})$ corresponds to the set of coverable sets, and $\mathcal{BC}(\mathcal{P})$ to the set of backward coverable states. In the decomposition $\rho_+ : \sigma_0 \xrightarrow{*} \langle S, \mathsf{d}\rangle$, $\rho_- : \langle S, \mathsf{d}\rangle \xrightarrow{*} \langle q_f, \mathsf{d_f}\rangle$, ρ_+ is a witness that $S \subseteq \mathcal{C}(\mathcal{P})$ and ρ_- that $S \subseteq \mathcal{BC}(\mathcal{P})$ (because ρ_- starts with a write action, for every $\mathsf{d}' \in \mathrm{D}$ one has $\langle S, \mathsf{d}'\rangle \xrightarrow{*} \langle q_f, \mathsf{d_f}\rangle$).

$\mathcal{C}(\mathcal{P})$ and $\mathcal{BC}(\mathcal{P})$ can be computed in polynomial time. For $\mathcal{C}(\mathcal{P})$, we use a saturation technique. For $\mathcal{BC}(\mathcal{P})$, we work backwards: a symbol is read before it is written. We start with $S := \{q_f\}$. Until a fixpoint for S is reached, we do the following. We iterate on D, trying to pick the symbol that was in the register before S could be reached. For each $\mathsf{d} \in \mathrm{D}$, we saturate S with backward transitions reading d, then check if d can be written by a transition ending in S. If not, we backtrack by removing states that were just added.

The algorithm iteratively removes from $\mathcal{P}$ states that are not in $\mathcal{C}(\mathcal{P}) \cap \mathcal{BC}(\mathcal{P})$. Indeed, states that are not in $\mathcal{C}(\mathcal{P}) \cap \mathcal{BC}(\mathcal{P})$ cannot appear in any synchronizing execution. If it ends up with $Q(\mathcal{P}) = \emptyset$, then there is no synchronizing execution and the algorithm rejects. If it ends up with $\mathcal{C}(\mathcal{P}) = \mathcal{BC}(\mathcal{P}) = Q(\mathcal{P}) \neq \emptyset$, then applying the definitions of $\mathcal{C}(\mathcal{P})$ and $\mathcal{BC}(\mathcal{P})$ gives a synchronizing execution, and the algorithm accepts. ◀

It is unknown whether the previous result still holds when dim is fixed to a value greater than 1. The case $\mathsf{dim} = 1$ is particularly easy because writing to the register completely erases its content.

Unlike COVER, TARGET and therefore DNF-PRP are not tractable under the uninitialized hypothesis. For TARGET, one cannot add fresh processes at no cost, since the fresh processes would eventually have to get to q_f. For example, if a register r can only be written from a given state q, the last process to leave q will fix the value in register r.

▶ **Proposition 13.** *TARGET for uninitialized roundless register protocols is NP-hard.*

	COVER	TARGET	DNF-PRP	PRP
General case	NP-complete (Prop. 8)	NP-complete (Prop. 8)	NP-complete (Prop. 8)	NP-complete (Prop. 8)
Uninitialized	PTIME-complete (Prop. 11)	NP-complete (Prop. 8 & 13)	NP-complete (Prop. 8 & 13)	NP-complete (Prop. 8 & 10)
$\mathsf{dim} = 1$ (one register)	PTIME-complete (Prop. 11)	PTIME-complete (Prop. 12 & 11)	PTIME-complete (Prop. 12 & 11)	NP-complete (Prop. 8 & 10)

Figure 3 Summary of complexity results for roundless register protocols.

4 Round-based Register Protocols

We now extend the previous model to a round-based setting. The model and semantics are the same as in [6], however we consider a more general problem than COVER. Thus, the abstract semantics developed here differs from [6].

4.1 Definitions

In round-based settings, there is a fresh set of dim registers at each round, and each process has its own private round value that starts at 0 and never decreases. Processes may only read from and write to registers of nearby rounds.

▶ **Definition 14** (Round-based register protocols). *A* round-based register protocol *is a tuple* $\mathcal{P} = \langle Q, Q_0, \mathsf{dim}, \mathrm{D}, \mathsf{d}_0, \mathsf{v}, \Delta \rangle$ *where*

- Q *is a finite set of states with a distinguished subset of initial states* $Q_0 \subseteq Q$*;*
- $\mathsf{dim} \in \mathbb{N}$ *is the number of shared registers per round;*
- D *is a finite data alphabet with an initial symbol* d_0*;*
- v *is the visibility range;*
- $\Delta \subseteq Q \times \mathcal{A} \times Q$ *is the set of transitions, where* $\mathcal{A} = \{\mathsf{read}_\alpha^{-i}(\mathsf{d}) \mid i \in [0, \mathsf{v}], \alpha \in [1, \mathsf{dim}], \mathsf{d} \in \mathrm{D}\} \cup \{\mathsf{write}_\alpha(\mathsf{d}) \mid \alpha \in [1, \mathsf{dim}], \mathsf{d} \in \mathrm{D} \setminus \{\mathsf{d}_0\}\} \cup \{\mathsf{Inc}\}$ *is the set of actions.*

Read actions specify the round of the register: $\mathsf{read}_\alpha^{-i}(\mathsf{d})$ means, for a process at round k, "read d from register α of round $k-i$". A process at round k may only write to the registers of round k. The Inc action increments the round of a process.

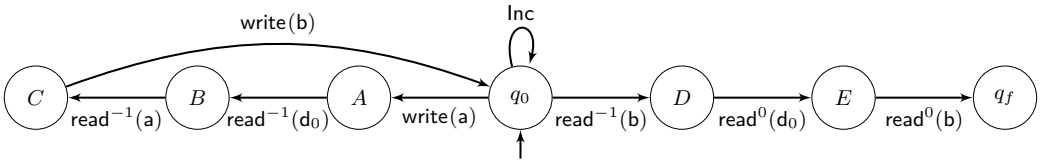

Figure 4 An example of round-based register protocol.

Let $\mathsf{rg}_k[\alpha]$ denote the register α of round k. The set of registers of round k is written Reg_k, and we let $\mathsf{Reg} = \bigcup_{k\in\mathbb{N}} \mathsf{Reg}_k$. The size of a protocol is $|\mathcal{P}| = |Q| + |\mathrm{D}| + |\Delta| + \mathsf{v} + \mathsf{dim}$. A given process is described by its state and round, formalized by a pair $(q,k) \in Q \times \mathbb{N}$ called *location*. Let $\mathsf{Loc} := Q \times \mathbb{N}$ denote the set of locations. A *concrete configuration* describes the number of processes in each location along with the value of each register. Formally, a *concrete configuration* is a pair $\langle \mu, \vec{d} \rangle$ with $\mu \in \mathbb{N}^{\mathsf{Loc}}$ such that $0 < \sum_{(q,k)\in\mathsf{Loc}} \mu(q,k) < \infty$ and $\vec{d} \in \mathrm{D}^{\mathsf{Reg}}$. For $\gamma = \langle \mu, \vec{d} \rangle$, we write $\mathsf{loc}(\gamma) := \mu$ and $\mathsf{data}(\gamma) := \vec{d}$. Again, we write Γ for the set of concrete configurations. The set of initial configurations is $\mathsf{Init_c} := \{\gamma \in \Gamma \mid \mathsf{data}(\gamma) = \mathsf{d}_0^{\mathsf{Reg}}$ and $\forall (q,k) \notin Q_0 \times \{0\}, \mathsf{loc}(\gamma)(q,k) = 0\}$.

A *move* is a pair $\theta \in \Delta \times \mathbb{N}$: move (δ, k) expresses that transition δ is taken by a process at round k; we write $\mathsf{Moves} := \Delta \times \mathbb{N}$ for the set of all moves. A move θ has *effect* on round k when θ is at round k or θ is an increment at round $k-1$. We define a step as follows: for $\theta = ((q,a,q'),k) \in \mathsf{Moves}$, $\gamma \xrightarrow{\theta} \gamma'$ when $(q,k) \in \mathsf{loc}(\gamma)$ and:

- if $a = \mathsf{read}_\alpha^{-i}(\mathsf{d})$, $\mathsf{loc}(\gamma') = (\mathsf{loc}(\gamma) \ominus \{(q,k)\}) \oplus \{(q',k)\}$, $\mathsf{data}(\gamma)(\mathsf{rg}_{k-i}[\alpha]) = \mathsf{d}$ and $\mathsf{data}(\gamma') = \mathsf{data}(\gamma)$;
- if $a = \mathsf{write}_\alpha(\mathsf{d})$, $\mathsf{loc}(\gamma') = (\mathsf{loc}(\gamma) \ominus \{(q,k)\}) \oplus \{(q',k)\}$, $\mathsf{data}(\gamma')(\mathsf{rg}_k[\alpha]) = \mathsf{d}$ and for all $\mathsf{r} \neq \mathsf{rg}_k[\alpha]$, $\mathsf{data}(\gamma')(\mathsf{r}) = \mathsf{data}(\gamma)(\mathsf{r})$;
- if $a = \mathsf{Inc}$, $\mathsf{loc}(\gamma') = (\mathsf{loc}(\gamma) \ominus \{(q,k)\}) \oplus \{(q',k{+}1)\}$ and $\mathsf{data}(\gamma') = \mathsf{data}(\gamma)$.

A step is *at round* k when the corresponding move is of the form (δ, k). Note that action $\mathsf{read}_\alpha^{-i}(\mathsf{d})$ is only possible for processes at rounds $k \geq i$. The notions of execution, of reachability and of coverability are defined as in the roundless case.

▶ **Example 15.** Consider the round-based protocol $\mathcal{P}$ from Figure 4, with $\mathsf{dim} = 1$, $\mathsf{v} = 1$, $Q_0 = \{q_0\}$ and $\mathrm{D} = \{\mathsf{d}_0, \mathsf{a}, \mathsf{b}\}$. In this protocol, state q_f cannot be covered. By contradiction, consider an execution $\pi : \gamma_0 \xrightarrow{*} \gamma$ with $\gamma_0 \in \mathsf{Init_c}$ and $\mathsf{loc}(\gamma)(q_f,k) > 0$ fo some $k \in \mathbb{N}$. We have that, at some point in π, (E,k) is populated and b is in $\mathsf{rg}[k]$. Therefore, some process went from (A,k) to (B,k), which implies that $\mathsf{rg}[k]$ lost value d_0 before $\mathsf{rg}[k{-}1]$; this in turn implies that π does not send any process to (E,k) which is a contradiction. ⌟

Since round-based register protocols enjoy the same monotonicity properties as roundless register protocols, we define the same non-counting abstraction. Note that this abstraction differs from the one in [6] which was designed specifically for COVER. The set of *abstract configurations* is $\Sigma := 2^{\mathsf{Loc}} \times \mathrm{D}^{\mathsf{Reg}}$; the abstract semantics are defined as in Subsection 2.3. Again, $\sigma \xrightarrow{\delta} \sigma$ if and only if there exist $\gamma, \gamma' \in \Gamma$, $\gamma \xrightarrow{\delta} \gamma'$ and $\mathsf{abst}(\gamma) = \sigma, \mathsf{abst}(\gamma') = \sigma'$. All the properties of Subsection 2.3 apply to round-based abstract semantics. In particular, we have the soundness and completeness of the abstraction:

▶ **Proposition 16** (Soundness and completeness of the abstraction). *For all* $L \subseteq \mathsf{Loc}$, $\vec{d} \in \mathrm{D}^{\mathsf{Reg}}$*:*

$$(\exists \gamma \in \mathsf{Reach_c}(\mathsf{Init_c}) : \mathsf{supp}(\gamma){=}L,\ \mathsf{data}(\gamma){=}\vec{d}) \iff (\exists \sigma \in \mathsf{Reach_a}(\mathsf{Init_a}) :\ \mathsf{loc}(\sigma){=}L,\ \mathsf{data}(\sigma){=}\vec{d}).$$

4.2 Presence Reachability Problem

COVER is extended to round-based protocols by asking whether some reachable configuration has a process on q_f on some round k, and TARGET by asking whether some reachable configuration has no process on states $q \neq q_f$ on any round k. Formally, one asks whether there exists $\gamma \in \mathsf{Reach_c}(\mathsf{Init_c})$ such that $\gamma \models \psi$ where $\psi =$"$\exists k \in \mathbb{N}, (q,k) \in \mathsf{loc}(\gamma)$" for COVER and $\psi =$"$\forall k \in \mathbb{N}, \forall q \neq q_f, (q,k) \notin \mathsf{loc}(\gamma)$" for TARGET. We will now extend roundless PRP to round-based PRP, where the formula is allowed to have non-nested quantification over rounds.

Presence constraints are first-order formulas (quantifying over the rounds) without any nested quantifiers. Formally:

- a *term* is of the form m or $k{+}m$ with $m \in \mathbb{N}$ and k a free variable;
- an *atomic proposition* is either of the form "(q,t) $\mathsf{populated}$" with t a term and $q \in Q$ or of the form "$\mathsf{rg}_t[\alpha]$ $\mathsf{contains}$ d" with t a term, $\alpha \in [1, \mathsf{dim}]$ and $\mathsf{d} \in \mathrm{D}$;
- a *literal* is either an atomic proposition or the negation of an atomic proposition;
- a *proposition* is a Boolean combination of atomic propositions that has at most one free variable;
- an *atomic presence constraint* is either a *closed* proposition (no free variables), or of the form "$\exists k\, \phi$" or "$\forall k\, \phi$" where ϕ is a proposition with k as a free variable.

Finally, a *presence constraint* is a Boolean combination of atomic presence constraints.

▶ **Example 17.** "$(\exists k\, (q_2,k)$ $\mathsf{populated}) \vee (\forall k\, ((q_0,k{+}2)$ $\mathsf{populated}) \wedge \mathsf{rg}_1[1]$ $\mathsf{contains}$ $\mathsf{a})$" is an example of presence constraint.
Let $\gamma := ((q_0,0) \oplus (q_1,1), \mathsf{d}_0^{\mathsf{Reg}})$ with $q_0 \neq q_1$, $\mathsf{dim} = 1$. One has $\gamma \models (\mathsf{rg}_0[1]$ $\mathsf{contains}$ $\mathsf{d}_0) \wedge (\exists k\, (q_1,k{+}1)$ $\mathsf{populated})$ but $\gamma \not\models \forall k\, (((q_0,k)$ $\mathsf{populated}) \vee \neg((q_1,k)$ $\mathsf{populated}))$. ⌟

We define the *round-based presence reachability problem* (round-based PRP):

ROUND-BASED PRP
Input: A round-based register protocol $\mathcal{P}$, a presence constraint ψ
Question: Does there exist $\gamma \in \mathsf{Reach_c}(\mathsf{Init_c})$ such that $\gamma \models \psi$?

▶ **Example 18.** Consider $\mathcal{P}$ from Example 15. If $\psi := \exists k, (q_f,k)$ $\mathsf{populated}$, then $(\mathcal{P},\psi)$ is a negative instance of round-based PRP. If $\psi' := \exists k, ((E,k)$ $\mathsf{populated}) \wedge ((E,k{+}1)$ $\mathsf{populated})$, then $(\mathcal{P},\psi')$ is also negative. However, if $\psi'' := ((E,2)$ $\mathsf{populated}) \wedge [\forall k, (\mathsf{rg}[k+1]$ $\mathsf{contains}$ $\mathsf{b}) \vee (\mathsf{rg}[k+1]$ $\mathsf{contains}$ $\mathsf{d}_0)]$, then $(\mathcal{P},\psi'')$ is positive: a witness execution sends a process to $(B,1)$, writes a to $\mathsf{rg}[0]$ then b to $\mathsf{rg}[1]$ and finally sends a process from $(q_0,2)$ to $(E,2)$. ⌟

COVER and TARGET for round-based register protocols are special cases of PRP. The following lower bound hence applies to all these problems:

▶ **Proposition 19** ([6, Theorem 23])**.** *COVER for round-based register protocols is PSPACE-hard, even in the uninitialized case with* $\mathsf{v} = 0$ *and* $\mathsf{dim} = 1$.

Note that, in the round-based setting, $\mathsf{dim} = 1$ means one register *per round*, therefore still an unbounded number of registers. $\mathsf{v} = 0$ means that a process can only interact with registers of its current round. The previous proposition implies that all problems considered in Figure 3 are PSPACE-hard when working with round-based protocols. In [6], COVER for round-based register protocols is shown to be PSPACE-complete. In the rest of this paper, we establish that the more general round-based PRP lies in the same complexity class:

▶ **Theorem 20.** *Round-based PRP is PSPACE-complete.*

5 A Polynomial-Space Algorithm for Round-Based PRP

In this section, we provide a polynomial-space algorithm for round-based PRP. Thanks to Savitch's theorem, it suffices to find a non-deterministic polynomial-space algorithm. To do so, one wants to guess an execution that reaches a configuration satisfying the presence constraint. However, as shown in [6, Proposition 13], one may need, at a given point along such an execution, the number of active rounds to be exponential (an active round being informally a round on which something has already happened and something else is yet to happen). Thus, storing the execution step by step in polynomial space seems hard; instead, our algorithm will guess the execution round by round. To do this, we define the notion of footprint, which represents the projection of an execution onto a narrow window of rounds.

Thanks to Proposition 7, round-based PRP can be studied directly in the abstraction. In the rest of the paper, all configurations and executions are implicitly abstract.

5.1 Footprints

Let $j \leq k$. We write $\mathsf{Loc}[j,k]$ for the set of locations at rounds $\max(j,0)$ to k; similarly, we write $\mathsf{Reg}[j,k]$ for the set of registers of rounds $\max(j,0)$ to k. A *local configuration* on (rounds) $[j,k]$ is an element of $2^{\mathsf{Loc}[j,k]} \times \mathrm{D}^{\mathsf{Reg}[j,k]}$. The set of local configurations on $[j,k]$ is written $\Sigma[j,k]$. Given $\sigma \in \Sigma$, the local configuration $\mathsf{local}[j,k](\sigma)$ is obtained by removing from σ all information that is not about rounds j to k. Note that local configurations are local with respect to the rounds, and not with respect to processes.

Given $\lambda, \lambda' \in \Sigma[j,k]$ and a move θ, we write $\lambda \xrightarrow{\theta} \lambda'$ when there exist two configurations σ and σ' such that $\sigma \xrightarrow{\theta} \sigma'$, $\mathsf{local}[j,k](\sigma) = \lambda$ and $\mathsf{local}[j,k](\sigma') = \lambda'$. In practice:

- if θ is a move with no effect on rounds j to k, then $\lambda \xrightarrow{\theta} \lambda'$ if $\lambda = \lambda'$;
- if $\theta = ((q, \mathsf{Inc}, q'), j{-}1)$ then $\lambda \xrightarrow{\theta} \lambda'$ holds with no condition that $(q, j{-}1)$ is populated in λ, since $j{-}1$ is outside of $[j,k]$;
- if $\theta = ((q, \mathsf{read}_\alpha^{-b}(\mathsf{d})), l)$ with $l{-}b < j$ (read from register of round $< j$), there is no condition on the content of the register.

A *footprint* on (rounds) $[j,k]$ corresponds to the projection of an execution on rounds $[j,k]$. Formally, it is an alternating sequence $\lambda_0, \theta_0, \lambda_1, \ldots, \theta_{m-1}, \lambda_m$ where for all $i \in [0,m]$, $\lambda_i \in \Sigma[j,k]$ and for all $i \leq m-1$, $\lambda_i \xrightarrow{\theta_i} \lambda_{i+1}$ and $\lambda_i \neq \lambda_{i+1}$.

Let $\rho = \sigma_0, \theta_0, \sigma_1, \ldots, \theta_{m-1}, \sigma_m$ be an execution. The *footprint of ρ on (rounds)* $[j,k]$, written $\mathsf{footprint}[j,k](\rho)$, is the footprint on $[j,k]$ obtained from ρ by replacing σ_i by $\lambda_i = \mathsf{local}[j,k](\sigma_i)$ and then removing all useless steps $\lambda_i \xrightarrow{\theta} \lambda_{i+1}$ with $\lambda_i = \lambda_{i+1}$ (by merging λ_i and λ_{i+1}, so $\mathsf{footprint}[j,k](\rho)$ can be shorter than ρ). Similarly, for $[j',k'] \supseteq [j,k]$ and τ a footprint on $[j',k']$, define the *projection* $\mathsf{footprint}[j,k](\tau)$ by the footprint obtained by replacing each local configuration in τ by its projection on $[j,k]$ and removing useless steps.

The following result provides a sufficient condition for a sequence of footprints to be seen as projections of a single common execution.

▶ **Lemma 21.** *Let $K \in \mathbb{N}$, $(\tau_k)_{k \leq K}$ and $(T_k)_{k \leq K-1}$ such that:*

- *for all $k \leq K$, τ_k is a footprint on $[k{-}\mathsf{v}{+}1, k]$,*
- *for all $k \leq K{-}1$, T_k is a footprint on $[k{-}\mathsf{v}{+}1, k{+}1]$,*
- *for all $k \leq K{-}1$, $\mathsf{footprint}[k{-}\mathsf{v}{+}1, k](T_k) = \tau_k$,*
- *for all $k \leq K{-}1$, $\mathsf{footprint}[k{-}\mathsf{v}{+}2, k{+}1](T_k) = \tau_{k+1}$.*

There exists an execution ρ such that, for all $k \leq K$, $\mathsf{footprint}[k{-}\mathsf{v}{+}1, k](\rho) = \tau_k$.

5.2 A Polynomial-Space Algorithm for Round-Based PRP

The algorithms guesses the witness execution footprint by footprint, and stops when the presence constraint is satisfied. Algorithm 1 provides the skeleton of this procedure. For the sake of simplicity, we suppose that $\mathsf{v} \geq 1$. If $\mathsf{v} = 0$, we artificially increase v to 1.

Algorithm 1 Non-deterministic algorithm for round-based PRP.

```
1  Input: A PRP instance (𝒫, ψ)
2  E, U, C ← ∅ ;
3  τ ← ϵ ;                                   // dummy footprint on rounds [−v, −1]
4  Guess the initial set I ⊆ Q_0 of populated states at round 0 ;
5  NDInit(E, U, C) ;
6  for k from 0 to +∞ do
7      Guess T a footprint on [k−v, k] such that footprint[k−v, k−1](T) = τ ;
8      Check that T is consistent with the initial configuration ;
9      λ ← last configuration in T ;
10     NDComputeIteration(E, U, C, λ) ;
11     if TestPresenceConstraint(E, U, C, λ) then Accept ;
12     τ ← footprint[k−v+1, k](T) ;
```

For all $k \in \mathbb{N}$, let τ_k be the value of τ at the end of iteration k and T_k the value of T guessed at iteration $k{+}1$. Thanks to Lemma 21, if the algorithm reaches the end of iteration K then there exists an execution ρ whose projection on $[k{-}\mathsf{v}, k-1]$ is τ_k for every $k \leq K$.

Handling the round-based presence constraint is technical, so we hide it in functions `NDInit`, `NDComputeIteration` and `TestPresenceConstraint`, whose pseudocode can be found in Algorithm 2. We guess why ψ is true by guessing satisfied *atomic propositions* of three types: existentially quantified on the round (*i.e.*, of the form "$\exists k\, \phi$" where ϕ has no quantifiers and only k as free variable) which we put in E; universally quantified on the round (*i.e.*, of the form "$\forall k\, \phi$" where ϕ has no quantifiers and only k as free variable) which we put in U; with no quantifier (*i.e.*, of the form "ϕ" where ϕ has no quantifiers and no free variables) which we put in C. Formulas in C refer to constant rounds and are checked at these rounds only. Formulas in U are checked at every round. For formulas in E, the algorithm guesses at which round the formula is true. See Appendix B.3 of the full version for more detailed explainations. Our algorithm is correct with respect to round-based PRP:

▶ **Proposition 22.** *$(\mathcal{P}, \psi)$ is a positive instance of round-based PRP if and only if there exists an accepting computation of Algorithm 1 on $(\mathcal{P}, \psi)$.*

The integer constants in the presence constraint ψ are encoded in unary, like the visibility range v. These two hypotheses are reasonable since practical examples typically use constants of small value (*e.g.*, 1). Under these hypotheses, we obtain a polynomial spatial bound on the size of footprints of a well-chosen witness execution, which in turn gives a polynomial spatial bound for the algorithm:

▶ **Proposition 23.** *Algorithm 1 works in space $O(|\psi|^3 + |Q|^2\,(\mathsf{v}{+}1)^2\,\log(\mathsf{dim}\,|\mathrm{D}|))$.*

Finally, we need to discuss the termination of the algorithm. According to the pigeonhole principle, after an exponential number of iterations, the elements stored in memory repeat from a previous iteration and we can stop the computation. One can thus use a counter, encoded in polynomial space, to count iterations and return a decision when the counter

reaches its largest value. Thanks to the space bounds from Proposition 23, correctness from Proposition 22 and the stopping criterion, our algorithm decides round-based PRP in non-deterministic polynomial space, proving Theorem 20.

Algorithm 2 The functions at **Line 5**, **Line 10** and **Line 11** of Algorithm 1.

Function `NDInit`(E, U, C) **:**
  `/* Sets containing what needs to be checked:` U `and` E `contain respectively universally and existentially quantified atomic presence constraints,` C `contains closed literals */`
  Guess $X \subseteq \mathsf{PosOrNeg}(\mathsf{APC}(\psi))$ s.t. ψ is true when all APCs in X are true ;
  for P *in* X **do**
    for ϕ *closed atomic proposition in* P **do**
      `/*` ϕ `refers to constant rounds only */`
      if ϕ *guessed to be true* **then** Add ϕ to C ; Replace ϕ by true in P ;
      else Add $\neg\phi$ to C ; Replace ϕ by false in P ;
    if P *is a closed proposition* **then**
      Check that P is true with guessed values of atomic propositions ;
    if P *universal* **then** Add P to U ;
    if P *existential* **then** Add P to E ;
Function `NDComputeIteration`(E, U, C, λ) **:**
  for *"*$\forall l\, \phi$*" in* U **do**
    Guess $\mathcal{L} \subseteq \mathsf{PosOrNeg}(\mathsf{AP}(\phi[l \leftarrow k]))$ s.t. $\phi[l \leftarrow k]$ is true when all literals in $\mathcal{L}$ are true ;
    Add all literals in $\mathcal{L}$ to C ;
  for *"*$\exists l\, \phi$*" in* E **do**
    if $\phi[l \leftarrow k]$ *guessed to be true* **then**
      Guess $\mathcal{L} \subseteq \mathsf{PosOrNeg}(\mathsf{AP}(\phi[l \to k]))$ s.t. $\phi[l \leftarrow k]$ is true when all literals in $\mathcal{L}$ are true ;
      Add all literals in $\mathcal{L}$ to C ; Remove "$\exists l\, \phi$" from E ;
  for ϕ *in* C *about round* k **do**
    `//` ϕ `is of the form (negation of) "`(q, k) `populated", or (negation of) "`$\mathsf{rg}_k[\alpha]$ `contains d"`
    Check that ϕ is satisfied in λ ; Remove ϕ from C ;
Function `TestPresenceConstraint`(E, U, C, λ) **:**
  if $E \neq \emptyset$ **then return** false ;
  for $\phi \in C$ *or "*$\forall l\, \phi$*" in* U **do**
    if $\langle \emptyset, \mathsf{d}_0^{\mathsf{Reg}} \rangle \not\models \phi$ **then**
      return false ; `// Execution cannot stop at round` k
  return true ;

References

1 Parosh Aziz Abdulla, Mohamed Faouzi Atig, and Rojin Rezvan. Parameterized verification under TSO is PSPACE-complete. *Proc. ACM Program. Lang.*, 4(POPL):26:1–26:29, 2020. doi:10.1145/3371094.

2 James Aspnes. Fast deterministic consensus in a noisy environment. *Journal of Algorithms*, 45(1):16–39, 2002. doi:10.1016/S0196-6774(02)00220-1.

3 A. R. Balasubramanian, Lucie Guillou, and Chana Weil-Kennedy. Erratum to parameterized analysis of reconfigurable broadcast networks, 2022. URL: https://www.model.in.tum.de/~weilkenn/erratum-fossacs22.pdf.

4 A. R. Balasubramanian, Lucie Guillou, and Chana Weil-Kennedy. Parameterized analysis of reconfigurable broadcast networks (long version). *CoRR*, abs/2201.10432, 2022. arXiv:2201.10432.

5 A. R. Balasubramanian and Chana Weil-Kennedy. Reconfigurable broadcast networks and asynchronous shared-memory systems are equivalent. In *GandALF 2021*, volume 346, pages 18–34, 2021. doi:10.4204/EPTCS.346.2.

6 Nathalie Bertrand, Nicolas Markey, Ocan Sankur, and Nicolas Waldburger. Parameterized safety verification of round-based shared-memory systems. In *ICALP 2022*, pages 113:1–113:20. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPIcs.ICALP.2022.113.

7 Roderick Bloem, Swen Jacobs, Ayrat Khalimov, Igor Konnov, Sasha Rubin, Helmut Veith, and Josef Widder. *Decidability of Parameterized Verification*. Synthesis Lectures on Distributed Computing Theory. Morgan & Claypool Publishers, 2015. doi:10.2200/S00658ED1V01Y201508DCT013.

8 Patricia Bouyer, Nicolas Markey, Mickael Randour, Arnaud Sangnier, and Daniel Stan. Reachability in networks of register protocols under stochastic schedulers. In *ICALP 2016*, volume 55 of *LIPIcs*, pages 106:1–106:14. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2016. doi:10.4230/LIPIcs.ICALP.2016.106.

9 Giorgio Delzanno, Arnaud Sangnier, Riccardo Traverso, and Gianluigi Zavattaro. On the complexity of parameterized reachability in reconfigurable broadcast networks. In *FSTTCS 2012*, volume 18 of *LIPIcs*, pages 289–300. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2012. doi:10.4230/LIPIcs.FSTTCS.2012.289.

10 Javier Esparza. Keeping a crowd safe: On the complexity of parameterized verification (invited talk). In *STACS 2014*, volume 25 of *LIPIcs*, pages 1–10. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2014. doi:10.4230/LIPIcs.STACS.2014.1.

11 Javier Esparza, Pierre Ganty, and Rupak Majumdar. Parameterized verification of asynchronous shared-memory systems. *Journal of the ACM*, 63(1):10:1–10:48, 2016. doi:10.1145/2842603.

12 Paulin Fournier. *Parameterized verification of networks of many identical processes*. PhD thesis, University of Rennes 1, France, 2015. URL: https://tel.archives-ouvertes.fr/tel-01355847.

13 Steven M. German and A. Prasad Sistla. Reasoning about systems with many processes. *Journal of the ACM*, 39(3):675–735, 1992. doi:10.1145/146637.146681.

14 Matthew Hague. Parameterised pushdown systems with non-atomic writes. In *FSTTCS 2011*, volume 13 of *LIPIcs*, pages 457–468. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2011. doi:10.4230/LIPIcs.FSTTCS.2011.457.

15 Michel Raynal and Julien Stainer. A simple asynchronous shared memory consensus algorithm based on omega and closing sets. In *CISIS 2012*, pages 357–364. IEEE Computer Society, 2012. doi:10.1109/CISIS.2012.198.