Checking Presence Reachability Properties on Parameterized Shared-Memory Systems

Nicolas Waldburger

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...

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

Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

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, 16]. 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, most of the proofs can be found in the appendix.

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 $\gamma_0 \xrightarrow{*} \gamma_l$ for the existence of such an execution. γ' is *reachable from* γ when $\gamma \xrightarrow{*} \gamma'$. Given a set C of configurations, we write $\mathsf{Reach}_\mathsf{c}(C) := \{\gamma' \mid \exists \gamma \in C, \gamma \xrightarrow{*} \gamma'\}$. A configuration is *reachable* when it is in $\mathsf{Reach}_\mathsf{c}(\mathsf{Init}_\mathsf{c})$.

▶ **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.

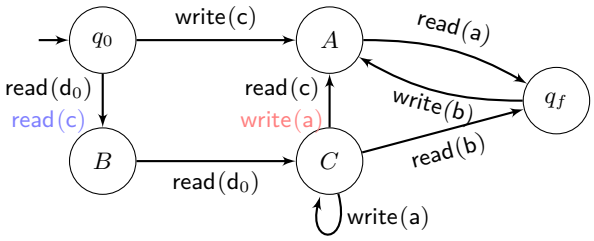

Figure 1 An example of a protocol

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 $\gamma \in \mathsf{Reach_c}(\mathsf{Init_c})$ 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 $\gamma \in \mathsf{Reach_c}(\mathsf{Init_c})$ such that $\gamma \models \phi$ if and only if there exists $\sigma \in \mathsf{Reach_a}(\mathsf{Init_a})$ 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 now 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.

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 $\sigma_0 \in \mathsf{Init_a}$ 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

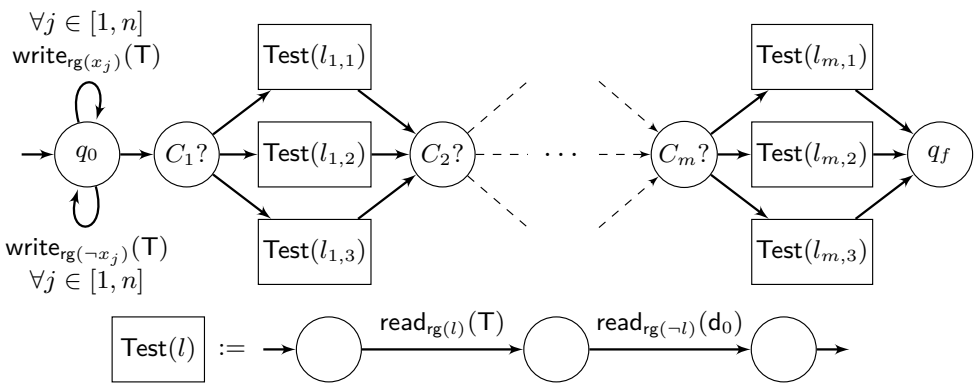

Figure 2 The protocol $\mathcal{P}_{\text{SAT}}(\phi)$ for NP-hardness of COVER.

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. See Appendix A.4 for the proof and pseudocode for DNF-PRP. 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.

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 $\forall (q, k) \notin Q_0 \times \{0\}, \mathsf{loc}(\gamma)(q, k) = 0\}$.

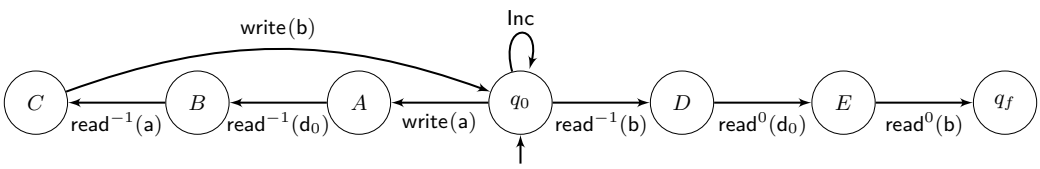

Figure 4 An example of round-based register protocol

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. See Appendix B.1 for the full definition.

▶ **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.

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` and postpone the details to Appendix B.3. 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

Input: A PRP instance $(\mathcal{P}, \psi)$
$E, U, C \leftarrow \emptyset$;
$\tau \leftarrow \epsilon$; `// dummy footprint on rounds` $[-\mathsf{v}, -1]$
Guess the initial set $I \subseteq Q_0$ of populated states at round 0 ;
`NDInit(`E, U, C`)` ;
for k *from* 0 *to* $+\infty$ **do**
  Guess T a footprint on $[k-\mathsf{v}, k]$ such that $\mathsf{footprint}[k-\mathsf{v}, k-1](T) = \tau$;
  Check that T is consistent with the initial configuration ;
  $\lambda \leftarrow$ last configuration in T ;
  `NDComputeIteration(`E, U, C, λ`)` ;
  if `TestPresenceConstraint(`E, U, C, λ`)` **then** Accept ;
  $\tau \leftarrow \mathsf{footprint}[k-\mathsf{v}+1, k](T)$;

Algorithm 1 Non-deterministic algorithm for round-based PRP

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. 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.

Acknowledgements

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

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. 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). 2022. https://arxiv.org/abs/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. 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 Richard E. Ladner. The circuit value problem is log space complete for P. *SIGACT News*, 7(1):18–20, 1975. doi:10.1145/990518.990519.

16 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.

Technical appendix

A Roundless Register Protocols

Recall that the abstract semantics is non-deterministic due to the choice between $\mathsf{st}(\gamma') = (\mathsf{st}(\gamma) \setminus \{q\}) \cup \{q'\}$ and $\mathsf{st}(\gamma') = \mathsf{st}(\gamma) \cup \{q'\}$. The first case is called *deserting*, and the latter *non-deserting*. A deserting step corresponds in the concrete semantics to all processes in q taking the transition at once.

Moreover, given a transition (q, a, q'), state q is called the *source* and state q' is called the *destination* of the transition, and similarly for a step.

A.1 Proof of Proposition 6

▶ **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. $\langle \mathsf{st}(\gamma_1) \oplus q_1, \mathsf{data}(\gamma_1)\rangle \xrightarrow{*} \langle \mathsf{st}(\gamma_2) \oplus q_2, \mathsf{data}(\gamma_2)\rangle$.*

Proof. We prove the result by induction on the length of the execution. If the execution is of length 0 then one simply considers $q_1 := q_2$. Let $\pi : \gamma_1 \xrightarrow{*} \gamma_2$ and suppose that the property is true for all executions of length $|\pi| - 1$. Decompose π into $\gamma_1 \xrightarrow{*} \gamma_3 \xrightarrow{\delta} \gamma_2$. If q_2 is not the destination of δ, then $q_2 \in \mathsf{supp}(\gamma_3)$ and we directly apply the induction hypothesis on $\gamma_1 \xrightarrow{*} \gamma_3$ and q_2 then conclude by taking δ to get to $\langle \mathsf{st}(\gamma_2) \oplus q_2, \mathsf{data}(\gamma_2)\rangle$. Assume that q_2 is the destination of δ; let q_3 the source of δ. We have $q_3 \in \mathsf{supp}(\gamma_3)$, so we apply the induction hypothesis on $\gamma_1 \xrightarrow{*} \gamma_3$ and q_3: we obtain that there exists $q_1 \in \mathsf{supp}(\gamma_1)$ such that $\langle \mathsf{st}(\gamma_1) \oplus q_1, \mathsf{data}(\gamma_3)\rangle \xrightarrow{*} \langle \mathsf{st}(\gamma_3) \oplus q_3, \mathsf{data}(\gamma_2)\rangle$ Moreover, we have $\langle \mathsf{st}(\gamma_3) \oplus q_3, \mathsf{data}(\gamma_2)\rangle \xrightarrow{\delta} \langle \mathsf{st}(\gamma_2) \oplus q_3, \mathsf{data}(\gamma_2)\rangle \xrightarrow{\delta} \langle \mathsf{st}(\gamma_2) \oplus q_2, \mathsf{data}(\gamma_2)\rangle$. Indeed, if δ is a read transition then the symbol is still in the register in γ_2 and may be read again, and if δ is write transition, then writing again a symbol to a register does not change its content. ◀

A.2 Proof of Proposition 7

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

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

First, the following lemma states that any concrete execution can easily be transformed into an abstract one.

▶ **Lemma A.1.** *Let $\gamma, \gamma' \in \Gamma$ and $\pi : \gamma \xrightarrow{*} \gamma'$. There exists $\rho : \mathsf{abst}(\gamma) \xrightarrow{*} \mathsf{abst}(\gamma')$.*

Proof. By induction, it suffices to prove it for one step; suppose that $\gamma \xrightarrow{\delta} \gamma'$. Let $(q, a, q') := \delta$; if $q \in \mathsf{st}(\gamma')$, then we consider the non-deserting abstract step with transition δ, otherwise we consider the deserting step with transition δ. Either way, we have $\mathsf{abst}(\gamma) \xrightarrow{\delta} \mathsf{abst}(\gamma')$. ◀

Conversely, from an abstract execution, for a large enough number of processes, using the copycat property one can build a concrete execution with the same final states and data.

▶ **Lemma A.2.** *Let $\sigma, \sigma' \in \Sigma$ and $\rho : \sigma \xrightarrow{*} \sigma'$. There exist γ, γ' such that $\mathsf{abst}(\gamma) = \sigma$, $\mathsf{abst}(\gamma') = \sigma'$ and $\pi : \gamma \xrightarrow{*} \gamma'$.*

Proof. The proof is by induction on the number of steps in π. If π has 0 steps, then $\sigma = \sigma'$ and suffices to consider $\gamma = \gamma' := \langle \bigoplus_{q \in \mathsf{st}(\sigma)} q, \mathsf{data}(\sigma)\rangle$.

Assume that $\rho : \sigma_1 \xrightarrow{*} \sigma_2 \xrightarrow{\delta} \sigma_3$ and that the property is true for executions shorter than ρ. By induction hypothesis, there exists $\pi : \gamma_1 \xrightarrow{*} \gamma_2$ such that $\mathsf{abst}(\gamma_1) = \sigma_1$, $\mathsf{abst}(\gamma_2) = \sigma_2$. There exists γ_3 such that $\gamma_2 \xrightarrow{\delta} \gamma_3$; however, it could be that γ_3 does not have a process on the source state of δ while σ_3 does. In that case, we modify π to put an additional process on the source state of δ by using the copycat property and increasing the number of processes by one. ◀

Lemmas A.1 and A.2 together prove Proposition 7.

A.3 Proof of Proposition 11

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

A.3.1 PTIME when the Registers are Uninitialized

The algorithm computes the set of coverable states using a fixpoint technique called *saturation*. The algorithm starts with $S := \{Q_0\}$, then iteratively adds to S all states q_2 for which there exist $q_1 \in S$ and an action $a \in \mathcal{A}$ such that $(q_1, a, q_2) \in \Delta$ and:

- either $a = \mathsf{write}_\alpha(\mathsf{d})$ with $\mathsf{d} \in \mathrm{D}$,
- or $a = \mathsf{read}_\alpha(\mathsf{d})$ with α, d s.t. there exist $q_3, q_4 \in S$, $(q_3, \mathsf{write}_\alpha(\mathsf{d}), q_4) \in \Delta$.

We prove that, when a fixpoint is reached, S is exactly the set of coverable states. First, any coverable state is added to S by the algorithm, by induction in the number of steps of the execution. Conversely, we show by induction in the number of iterations of the algorithm that any state added to S is coverable. For the first case, if $(q_1, \mathsf{write}_\alpha(\mathsf{d}), q_2) \in \Delta$ and q_1 is coverable then clearly q_2 is coverable. For the second case, suppose that we have an execution ρ covering $S \subseteq Q$, and that there exist $q_2 \in Q, q_1, q_3, q_4 \in S$, $\mathsf{d} \in \mathrm{D} \setminus \{\mathsf{d}_0\}$, $\alpha \in [1, \mathsf{dim}]$ such that $(q_1, \mathsf{read}_\alpha(\mathsf{d}), q_2), (q_3, \mathsf{write}_\alpha(\mathsf{d}), q_4) \in \Delta$. Because we have an unlimited supply of processes, we use the copycat property to put an extra process on q_3 then make that process write d to $\mathsf{rg}[\alpha]$ again, so that a process in state q_1 may read d from register $\mathsf{rg}[\alpha]$ and get to q_2, which is therefore coverable.

A.3.2 PTIME when the Number of Registers dim is Fixed

The first write to a register is an irreversible action, as d_0 cannot be written again. For that reason, we cannot work with a single saturation phase like in the uninitialized case. We iterate over all possible orders in which registers are first written to (see *first-write orders* in [6, Definition 15]).

For a given such order $\mathsf{r}_1, \ldots, \mathsf{r}_m$ ($m \leq \mathsf{dim}$), we proceed using $m+1$ successive saturation phases, numbered from $i = 0$ to m. The algorithm starts with $S = \{Q_0\}$. During saturation phase i, the algorithm saturates S by iteratively adding all states q_2 such that:

- there exists $q_1 \in S$ with $(q_1, \mathsf{read}_\mathsf{r}(\mathsf{d}_0), q_2)$ and $\mathsf{r} \notin \{\mathsf{r}_1, \ldots, \mathsf{r}_i\}$,
- there exists $q_1 \in S$ with $(q_1, \mathsf{write}_{\mathsf{r}_j}(\mathsf{d}), q_2)$, $j \leq i$,
- there exists $q_1 \in S$ with $(q_1, \mathsf{read}_{\mathsf{r}_j}(\mathsf{d}), q_2)$, $j \leq i$, and d may be written to r_j using a transition whose source is in S.

First, if there exists an execution covering q_f, then it writes to registers for the first time in some order $\mathsf{r}_1, \ldots, \mathsf{r}_m$. When the algorithm considers this first-write order, the set of states

computed includes q_f. Conversely, suppose that the algorithm finds that q_f is covered for some first-write order $\mathsf{r}_1, \ldots, \mathsf{r}_m$. Observe that, if two executions share a first-write order $\mathsf{r}_1, \ldots, \mathsf{r}_m$ then they may be merged into a common execution [6, Lemma 17]. Therefore, all the states computed by the algorithm may be covered in a single, big execution and q_f is coverable.

A.3.3 PTIME-hardness

The proof is similar to the one presented in [9, Proposition 1] for broadcast protocols. It uses a LOGSPACE-reduction for the Circuit Value Problem, which is PTIME-complete for LOGSPACE reductions [15]. This problem consists in determining the output value of an acyclic Boolean circuit with given input values and Boolean gates that can be negations $\neg$, disjunctions $\vee$ and conjunctions $\wedge$.

Consider an instance of the Circuit Value Problem, we write V for the set of input, intermediate and output values of the circuit. A gate is represented as a tuple of the form $(\neg, i, o)$, $(\vee, i_1, i_2, o)$ or $(\wedge, i_1, i_2, o)$ where $i, i_1, i_2 \in V$ denote input(s) and $o \in V$ the output of the gate. We construct an instance $(\mathcal{P}_{\mathrm{CVP}}, q_f)$ of COVER with $\mathsf{dim} = 1$. In D, we have d_0 (which is never read) along with, for every $v \in V$, symbols $v = \mathsf{T}$ and $v = \mathsf{F}$, denoting that v is respectively true and false. First, $\mathcal{P}_{\mathrm{CVP}}$ has a part containing a state in Q_0 from which one may write the symbols corresponding to the assignment of the input values of the circuit. Moreover, for every gate of the circuit, there is a part of the protocol corresponding to this gate, which has a state in Q_0 from which a process may read the values of the inputs and write the corresponding value of the output. A depiction for gate $(\wedge, i_1, i_2, o)$ may be found in Figure 5. Lastly, state q_f is the destination of the transition writing the symbol corresponding to the output variable of the circuit having the desired value, so that q_f is coverable if and only if this desired value is indeed the output value of the circuit.

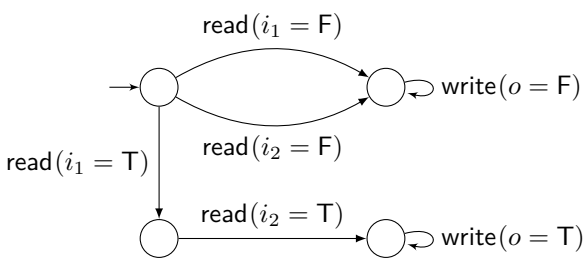

Figure 5 Part of the protocol $\mathcal{P}_{\mathrm{CVP}}$ that corresponds to gate $(\wedge, i_1, i_2, o)$

A.4 Proof of Proposition 12

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

We here prove the result in the general case of Proposition 12. We first prove that it suffices to prove the result for uninitialized protocols.

▶ **Lemma A.3.** *There exists a polynomial-time reduction from initialized DNF-PRP with* $\mathsf{dim} = 1$ *to uninitialized DNF-PRP with* $\mathsf{dim} = 1$.

Proof. Let $\mathcal{P} = \langle Q, Q_0, 1, \mathrm{D}, \mathsf{d}_0, \Delta \rangle$ a roundless register protocol with a single register. Any execution will be composed of two phases: the phase where the register has value d_0 and no

write transition is taken and the phase where the register no longer has value d_0 and write transitions may be taken. The reduction relies on the observation that, in the first phase, only transitions labeled by $\mathsf{read}(\mathsf{d}_0)$ may be taken, and processes do not interact during this phase. Therefore, we can consider as initial any state that may be covered from Q_0 with a path of transitions all labeled by $\mathsf{read}(\mathsf{d}_0)$.

Consider the graph $G = (Q, E)$ whose vertices are the states of the system and whose edges are the transitions labeled by $\mathsf{read}(\mathsf{d}_0)$: $(q_1, q_2) \in E$ if and only if $(q_1, \mathsf{read}(\mathsf{d}_0), q_2) \in \Delta$. Let $Q'_0 := \{q \in Q \mid \exists m \geq 0, \exists q_0 \in Q_0, \exists q_1, \ldots, q_{m-1}, q_m = q \in Q, \forall i \in [0, m-1], (q_i, q_{i+1}) \in E\}$ the set of states reachable from Q_0 in G. Q'_0 can trivially be computed in polynomial time. Additionally, let $\Delta' := \Delta \setminus \{(q, \mathsf{read}(\mathsf{d}_0), q') \mid q, q' \in Q\}$. The reduction maps $\mathcal{P}$ to the protocol $\mathcal{P}' = \langle Q, Q'_0, 1, \mathrm{D}, \mathsf{d}_0, \Delta' \rangle$.

We now prove that $(\mathcal{P}, \psi)$ is a positive instance of DNF-PRP if and only if $(\mathcal{P}', \psi)$ is. First, suppose that, in $\mathcal{P}$, there exists $\rho : \sigma_0 \xrightarrow{*} \sigma$ such that $\sigma \models \psi$. We decompose ρ into $\rho_p : \sigma_0 \xrightarrow{*} \sigma_1$ and $\rho_s : \sigma_1 \xrightarrow{*} \sigma$ where $\mathsf{data}(\sigma_1)(\mathsf{r}) = \mathsf{d}_0$ and ρ_s either is the empty execution or starts with a write transition. ρ_p only uses transitions labeled by $\mathsf{read}(\mathsf{d}_0)$ therefore, for every $q \in \mathsf{st}(\sigma_1)$, there exists a path in G from Q_0 to q; this proves that $\mathsf{st}(\sigma_1) \subseteq Q'_0$ and therefore that σ_1 is initial for $\mathcal{P}'$, moreover ρ_s does not use transitions labeled by $\mathsf{read}(\mathsf{d}_0)$ hence ρ_s is a witness that $(\mathcal{P}', \psi)$ is positive.

Suppose now that $(\mathcal{P}', \psi)$ is positive. There exists $\rho : \sigma_0 \xrightarrow{*} \sigma$ with $\sigma \models \psi$. For every $q \in \mathsf{st}(\sigma_0) \setminus Q_0$, there exists $f(q) \in Q_0$ such that q is reachable from $f(q)$ in G. Let $S := (\mathsf{st}(\sigma_0) \cap Q_0) \cup f(\mathsf{st}(\sigma_0) \setminus Q_0)$, we have $S \subseteq Q_0$. We have that $\langle S, \mathsf{d}_0 \rangle \xrightarrow{*} \sigma_0$ in $\mathcal{P}$ by only taking transitions appearing in G. Therefore $\langle S, \mathsf{d}_0 \rangle \xrightarrow{*} \sigma$ with $\sigma \models \psi$ and $\langle S, \mathsf{d}_0 \rangle$ is initial for $\mathcal{P}$, which proves that $(\mathcal{P}, \psi)$ is positive. ◀

Thanks to the previous lemma, we prove Proposition 12 in the uninitialized case. Consider a instance $(\mathcal{P}, \psi)$ of DNF-PRP where $\mathcal{P}$ is uninitialized and $\mathsf{dim} = 1$. $(\mathcal{P}, \psi)$ is positive if and only if $(\mathcal{P}, C)$ is positive for some clause C of ψ. Our algorithm hence iterates over all clauses in ψ.

Let C a clause in ψ. C is a conjunction of literals, hence it may be seen as a set of atomic propositions that the configuration reached has to satisfy. Let $Q_+(C)$ be the set of states that need to be populated in the final configuration, $Q_-(C)$ the states that need to not be populated, and $\mathrm{D}_{\mathsf{ok}}(C)$ the symbols that are allowed in the final configuration. Formally, $Q_+(C) := \{q \mid \text{“}q \text{ populated”} \in C\}$, $Q_-(C) := \{q \mid \text{“}\neg(q \text{ populated})\text{”} \in C\}$ and $\mathrm{D}_{\mathsf{ok}}(C) := \{\mathsf{d} \in \mathrm{D} \mid \text{“}\neg(\mathsf{r} \text{ contains } \mathsf{d})\text{”} \notin C \text{ and } \forall \mathsf{d}' \neq \mathsf{d}, \text{“}\mathsf{r} \text{ contains } \mathsf{d}'\text{”} \notin C\}$ where r denotes the register. For all $\langle S, \mathsf{d} \rangle \in \Sigma$, $\langle S, \mathsf{d} \rangle \models C$ if and only if $Q_+(C) \subseteq S \subseteq Q \setminus Q_-(C)$ and $\mathsf{d} \in \mathrm{D}_{\mathsf{ok}}(C)$. Let

$$\mathcal{F}(C) := \{S \subseteq Q \mid Q_+(C) \subseteq S \subseteq Q \setminus Q_-(C)\}$$

denote the collection of all sets of states allowed in the final configuration.

▶ **Lemma A.4.** *Any execution $\rho : \sigma_0 \xrightarrow{*} \sigma$ with $\sigma_0 \in \mathsf{Init_a}$ may be decomposed in the following form: $\sigma_0 \xrightarrow{*} \langle S, \mathsf{d_f} \rangle \xrightarrow{*} \sigma$ with S containing all states appearing in ρ.*

Proof. Let $\rho : \sigma_0 \xrightarrow{*} \sigma$; let $\langle S_f, \mathsf{d_f} \rangle := \sigma$. The execution $\sigma_0 \xrightarrow{*} \langle S, \mathsf{d_f} \rangle$ is obtained by turning into non-deserting all deserting steps in ρ, so that all states covered in ρ appear in S. For the second execution, we claim that there exists $\rho' : \langle S, \mathsf{d_f} \rangle \xrightarrow{*} \langle S'_f, \mathsf{d_f} \rangle$ that is obtained by mimicking steps of ρ starting from $\langle S, \mathsf{d_f} \rangle$. First, $\mathcal{P}$ is uninitialized therefore ρ starts with a write and the register value at the beginning of ρ is irrelevant. Moreover, $\mathsf{st}(\sigma_0) \subseteq S$ by definition of S, so that ρ' starts from a configuration with more states that ρ. By induction,

for all $n \geq 1$, the n-th configuration in ρ' has the same register value as and more states than the n-th configuration in ρ. This in fact proves that $S_f \subseteq S'_f$. Moreover, for every $q \in S \setminus S_f$, since $q \notin S_f$ the last step in ρ about step q has q as source and is deserting, hence q is also deserted in ρ' which shows that $S'_f \subseteq S_f$. In the end, ρ' goes from $\langle S, \mathsf{d_f}\rangle$ to $\langle S_f, \mathsf{d_f}\rangle$ which concludes the proof. ◀

We define the following two sets:

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

where the max is for inclusion of sets. Note that for $S \subseteq Q$, it is equivalent that S satisfies the condition $\forall \mathsf{d} \in \mathrm{D}, \ldots$ in the second set and that there exists a witness execution that starts with a write and therefore is applicable from any $\langle S, \mathsf{d}\rangle$ with $\mathsf{d} \in \mathrm{D}$. By convention, we consider that $\max(\emptyset) = \emptyset$, *i.e.*, if $Q_0 = \emptyset$ then $\mathcal{C}(\mathcal{P}, C) = \emptyset$ and if $Q_-(\mathcal{P}, C) = Q$ then $\mathcal{BC}(\mathcal{P}, C) = \emptyset$.

We first prove that both maxima are well-defined because the sets considered are stable under union. Let $\sigma_0, \sigma'_0 \in \mathsf{Init_a}$, $\rho : \sigma_0 \xrightarrow{*} \langle S, \mathsf{d}\rangle$, $\rho' : \sigma'_0 \xrightarrow{*} \langle S', \mathsf{d}'\rangle$. We show that we can merge ρ and ρ' into a single execution $\langle \mathsf{st}(\sigma_0) \cup \mathsf{st}(\sigma'_0), \mathsf{d}_0\rangle \xrightarrow{*} \langle S \cup S', \mathsf{d}'\rangle$. By mimicking ρ (without deserting states from $\mathsf{st}(\sigma'_0)$), we obtain an execution $\langle \mathsf{st}(\sigma_0) \cup \mathsf{st}(\sigma'_0), \mathsf{d}_0\rangle \xrightarrow{*} \langle S \cup \mathsf{st}(\sigma'_0), \mathsf{d}\rangle$. By mimicking ρ', we obtain an execution $\langle S \cup \mathsf{st}(\sigma'_0), \mathsf{d}\rangle \xrightarrow{*} \langle S \cup S', \mathsf{d}'\rangle$ (because $\mathcal{P}$ is uninitialized, ρ' starts with a write). Therefore $S \cup S'$ is in the set $\{S \subseteq Q \mid \exists \sigma_0 \in \mathsf{Init_a}, \exists \mathsf{d} \in \mathrm{D},\ \sigma_0 \xrightarrow{*} \langle S, \mathsf{d}\rangle\}$, proving that it is closed under union.

For $\mathcal{BC}(\mathcal{P}, C)$, we suppose that we have $S, S' \subseteq Q$ that satisfy the condition of the set, and we prove that $S \cup S'$ does as well. Let $\mathsf{d} \in \mathrm{D}$. By hypothesis on S applied with d, there exist S_f, $\mathsf{d_f}$ such that $\langle S, \mathsf{d}\rangle \xrightarrow{*} \langle S_f, \mathsf{d_f}\rangle$, and therefore $\langle S \cup S', \mathsf{d}\rangle \xrightarrow{*} \langle S_f \cup S', \mathsf{d_f}\rangle$. By hypothesis on S' applied with $\mathsf{d_f}$, there exist S'_f, $\mathsf{d_f}'$ such that $\langle S', \mathsf{d_f}\rangle \xrightarrow{*} \langle S'_f, \mathsf{d_f}'\rangle$. Therefore we also have $\langle S_f \cup S', \mathsf{d_f}\rangle \xrightarrow{*} \langle S_f \cup S'_f, \mathsf{d_f}'\rangle$, which combined with the previous execution provides a witness that $S \cup S'$ is in the set.

Algorithm 2 provides functions computing $\mathcal{C}(\mathcal{P}, C)$ and $\mathcal{BC}(C, \mathcal{P})$ along with the function solving DNF-PRP when the protocol is uninitialized and $\mathsf{dim} = 1$.

First, we prove that `Compute_`$\mathcal{C}(\mathcal{P})$ returns $\mathcal{C}(\mathcal{P})$. By induction, any state added in S in `Compute_`$\mathcal{C}(\mathcal{P})$ are in $\mathcal{C}(\mathcal{P})$. Indeed, any state that can be covered from a state in $\mathcal{C}(\mathcal{P})$ using a write transition is in $\mathcal{C}(\mathcal{P})$. Similarly, any state that can be covered from a state in $\mathcal{C}(\mathcal{P})$ using a read transition which symbol may be written from $\mathcal{C}(\mathcal{P})$ is in $\mathcal{C}(\mathcal{P})$: first write the corresponding value then read it (all states of $\mathcal{C}(\mathcal{P})$ can be covered in a single, common execution). Conversely, for any execution $\rho : \sigma_0 \xrightarrow{*} \sigma$, every state appearing in ρ is added to S. Observe that any execution may be split into phases, where a phase starts with a step writing a symbol then performs some number (possibly zero) of steps reading the symbol. We therefore process by induction on the number of such phases. The initialization comes from $\mathsf{st}(\sigma_0) \subseteq Q_0$. Let d the symbol of the last phase in ρ, and suppose that all states appearing before this last phase are added to S. The write transition of the phase is detected at line 11 of the iteration and the corresponding destination state is added to S. This write transition is now a witness for d at line 12, allowing every read transition appearing in the phase to be detected in this iteration.

We now claim that `Compute_`$\mathcal{BC}(\mathcal{P}, C)$ returns $\mathcal{BC}(\mathcal{P}, C)$. If S satisfies the condition in $\mathcal{BC}(\mathcal{P}, C)$ then one can go from $\langle S, *\rangle$ to a configuration satisfies the clause with an execution starting with a write. Again, this execution may be split into phases, each phase being composed of a write of a symbol followed by some reads of this symbol. The symbol of the last phase must be in $\mathrm{D_{ok}}$ as the last configuration satisfies C. Therefore, by induction,

```
Function DNFPRP_Oneregister_Uninit(𝒫):
    for C clause of ψ do
        𝒫_C ← 𝒫 ;                                   // copy of 𝒫 that will be modified
        Until Q(𝒫_C) reaches a fixpoint do
            Q(𝒫_C) ← Q(𝒫_C) ∩ 𝒞(𝒫_C) ∩ ℬ𝒞(𝒫_C, C) ;                    // modifies 𝒫_C
        if Q_+(C) ⊆ Q(𝒫_C) ≠ ∅ then Accept;
    Reject ;
Function Compute_𝒞(𝒫):
    S ← Q_0 ;
    Until S reaches a fixpoint do
        S ← S ∪ {q' | ∃q ∈ S, ∃d ∈ D, (q, write(d), q') ∈ Δ} ;
        S ← S ∪ {q' | ∃q, q_1, q_2 ∈ S, ∃d, (q, read(d), q') ∈ Δ, (q_1, write(d), q_2) ∈ Δ} ;
    return S ;
Function Compute_ℬ𝒞(𝒫, C):
    if PreviousSymbol(Q \ Q_−(C), D_ok) ≠ "Not found" then
      S ←PreviousSymbol(Q \ Q_−(C), D_ok);
    else return ∅ ;
    Until S reaches a fixpoint do
        if PreviousSymbol(S, D \ {d_0}) ≠ "Not found" then
            S ←PreviousSymbol(S, D \ {d_0}) ;
    return S;
Function PreviousSymbol(S, Symbols):
    Found ← False ;
    for d ∈ Symbols do
        T ← S;
        Until T reaches a fixpoint do
            T ← T ∪ {q ∈ Q | ∃q' ∈ T, (q, read(d), q') ∈ Δ} ;
        if there exist q ∈ Q, q' ∈ T s.t. (q, write(d), q') ∈ Δ then
            S ← T ∪ {q} ;
            Found ← True ;
    if Found then return S else return "Not found" ;
```

Algorithm 2 A polynomial-time algorithm for DNF-PRP with $\mathsf{dim} = 1$

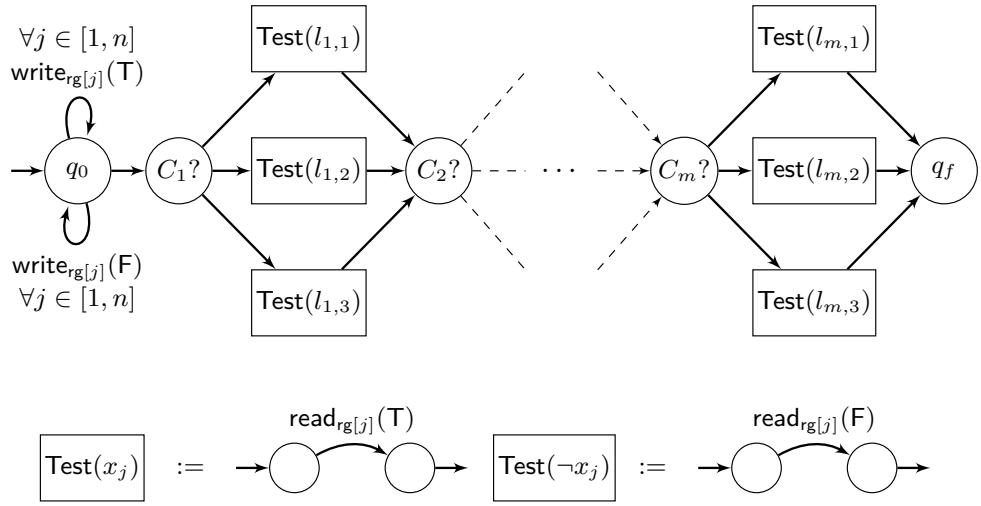

Figure 6 The protocol $\mathcal{P}_{\text{SAT}}(\phi)$ for NP-hardness of uninitialized TARGET.

all states appearing in such an execution appear in some `PreviousSymbol` computation in `Compute_`$\mathcal{BC}(\mathcal{P}, C)$, and the states returned include all states of the execution. Conversely, given a computation of `Compute_`$\mathcal{BC}(\mathcal{P}, C)$ that returns S, one may by reversed induction build an execution that covers every state in S and ends on a configuration satisfying C. All in all, we have proven that `Compute_`$\mathcal{BC}(\mathcal{P}, C)$ computes $\mathcal{BC}(\mathcal{P}, C)$.

We will now prove that `DNFPRP_Oneregister_Uninit` of Algorithm 2 solves DNF-PRP for uninitialized protocols with $\mathsf{dim} = 1$. First, suppose that the algorithm accepts during the iteration corresponding to clause C. It ends with a protocol $\mathcal{P}_C$ such that $Q_+(C) \subseteq Q(\mathcal{P}_C) = \mathcal{C}(\mathcal{P}_C) \cap \mathcal{BC}(\mathcal{P}_C, C)$. In this protocol, there exist $\sigma_0 \in \mathsf{Init_a}$ and $\mathsf{d} \in \mathrm{D}$ such that $\sigma_0 \xrightarrow{*} \langle \mathcal{C}(\mathcal{P}_C), \mathsf{d} \rangle$; since $\mathcal{C}(\mathcal{P}_C) = \mathcal{BC}(\mathcal{P}_C, C)$ we also have $\langle \mathcal{C}(\mathcal{P}_C), \mathsf{d} \rangle \xrightarrow{*} \langle S_f, \mathsf{d_f} \rangle \models C$ and the instance is positive.

Suppose now that the instance is positive. There exist a clause C in ψ and a witness execution $\rho : \sigma_0 \xrightarrow{*} \langle S_f, \mathsf{d_f} \rangle$ with $\mathsf{d_f} \in \mathrm{D}_{\mathsf{ok}}(C)$ and $S_f \in \mathcal{F}(C)$. Let S the set of states appearing in ρ. By induction, we have that $S \subseteq \mathcal{C}(\mathcal{P}_C) \cap \mathcal{BC}(\mathcal{P}_C, C)$ at every iteration of `DNFPRP_Oneregister_Uninit`, because ρ remains a witness of both inclusions at every iteration. Moreover, $Q_+(C) \subseteq S$ therefore the algorithm accepts.

A.5 Proof of Proposition 13

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

Once again, we provide 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 an instance of the uninitialized TARGET $(\mathcal{P}_{\text{SAT}}(\phi), q_f)$ which is positive if and only if ϕ is satisfiable. Let $\mathsf{dim} := n$, *i.e.*, the protocol has a register $\mathsf{rg}[i]$ for each $i \in [1,n]$. Each register can have values T and F (along with $\mathsf{d_0}$ which cannot be read nor written). A depiction of the protocol can be found in Figure 6.

Suppose first that ϕ is satisfiable by an assignment ν. For all $i \in [1,m]$, there exists $k(i) \in [1,3]$ such that $\nu(l_{i,k(i)}) = \mathsf{true}$. Consider the execution that writes symbols according to ν, then deserts q_0 to go to $C_1?$, and one by one deserts all $C_i?$-s through states $\mathsf{Test}(l_{i,k(i)})$. This execution goes from $\langle q_0, \mathsf{d_0^{Reg}} \rangle$ to $\langle q_f, \vec{d_f} \rangle$ hence the instance of TARGET is positive.

Conversely, suppose that there exists such an execution $\rho : \sigma_0 \xrightarrow{*} \langle q_f, \vec{d} \rangle$. Let ν be the valuation corresponding to the register values when ρ deserts q_0 for the last time. From this point onwards, ρ successively deserts all $C_i?$, hence for all $i \in [1, m]$, there exists $k(i) \in [1, 3]$ such that $\nu(l_{i,k(i)}) = \mathsf{true}$, proving that ϕ is satisfied by ν.

B Round-based Register Protocols

B.1 Formal Definition of Atomic Presence Constraints

We first define some more precise notions to refer to parts of presence constraints. 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. A *presence constraint* is a Boolean combination of atomic presence constraints.

B.2 Proof of Lemma 21

We prove the following, more general statement.

▶ **Lemma B.5.** *Let $K \in \mathbb{N}$, $w \geq \mathsf{v}{-}1$, $(\tau_k)_{k \leq K}$ and $(T_k)_{k \leq K-1}$ such that:*

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

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

We start by proving the following lemma:

▶ **Lemma B.6.** *Let $w \geq \mathsf{v}$, $k \in \mathbb{N}$, τ_- a footprint on $[k{-}w, k]$ and τ_+ a footprint on $[k{-}w{+}1, k{+}1]$ such that $\mathsf{footprint}[k{-}w{+}1, k](\tau_-) = \mathsf{footprint}[k{-}w{+}1, k](\tau_+)$. There exists T a footprint on $[k{-}w, k{+}1]$ such that $\mathsf{footprint}[k{-}w, k](T) = \tau_-$ and $\mathsf{footprint}[k{-}w{+}1, k{+}1](T) = \tau_+$.*

Proof. Let $\tau_{com} := \mathsf{footprint}[k{-}w{+}1, k](\tau_-) = \mathsf{footprint}[k{-}w{+}1, k](\tau_+)$. We proceed by induction on the number of steps in τ_{com}.

First if τ_{com} is the dummy footprint with no steps, then all steps in τ_- are at round $k{-}w$ and steps in τ_+ are at round $k{+}1$. It suffices to consider T that first copies the behavior of τ_- and then the behavior of τ_+: steps at round $k{-}w$ cannot depend on the information of rounds $> k{-}w$, and steps at round $k{+}1$ cannot depend on the information of rounds $< k{-}w$ because $w \geq \mathsf{v}$.

Assume that the property is true if τ_{com} has m steps, and suppose that τ_{com} has $m{+}1$ steps. We decompose $\tau_- = t_-, \theta, s_-$ and $\tau_+ = t_+, \theta, s_+$ where t_- and t_+ coincide on rounds $k{-}w{+}1$ to k and their projection on these rounds has exactly m steps, θ is the move of the $m{+}1$-th step of τ_{com}, and s_- and s_+ have no step at rounds $k{-}w{+}1$ to k. By induction hypothesis, there exists t such that $\mathsf{footprint}[k{-}w, k](t) = t_-$ and $\mathsf{footprint}[k{-}w{+}1, k{+}1](t) = t_+$. By applying the property for $m = 0$, there exists s such that $\mathsf{footprint}[k{-}w, k](s) = s_-$ and $\mathsf{footprint}[k{-}w{+}1, k{+}1](s) = s_+$. Letting $T := t, \theta, s$ (with θ deserting if and only if it was deserting in τ_{com}) concludes the proof. ◀

We now prove Lemma B.5. We proceed by induction on K. First, if $K = 0$, footprint τ_0 only has moves at round 0 and may be seen as an execution. Suppose that the property is true for K, and consider $(\tau_k)_{k \leq K+1}$, $(T_k)_{k \leq K}$ satisfying the hypothesis. For all $k \leq K-1$, T_k and T_{k+1} both have projection τ_{k+1} on rounds $[k-w+1, k+1]$, hence thanks to Lemma B.6 applied with $w' := w+1$ and $k' := k+1$, there exists U_k on rounds $[k-w, k+2]$ that projects to T_k and T_{k+1} on $[k-w, k+1]$ and $[k-w+1, k+2]$ respectively. By applying the induction hypothesis on (T_k) and (U_k) with $K' := K-1$, there exists an execution ρ such that, for all $k \leq K$, $\mathsf{footprint}[k-w, k+1](\rho) = T_k$; this implies that, for all $k \leq K+1$, $\mathsf{footprint}[k-w, k](\rho) = \tau_k$, concluding the proof of Lemma B.5. Applying Lemma B.5 with $w := \mathsf{v}-1$ gives Lemma 21.

B.3 Technical Details about Algorithm 1

Here, we describe in full details how Algorithm 1 handles the presence constraint. The pseudocode of the three functions used in Algorithm 1 can be found in Algorithm 3.

For ψ a presence constraint, we write $\mathsf{APC}(\psi)$ for the set containing all *atomic presence constraints* in ψ as well as their negations. For ϕ a closed proposition, we write $\mathsf{AP}(\phi)$ for the set of *atomic propositions* in ϕ. Given a set S of propositions or presence constraints, we write $\mathsf{PosOrNeg}(S) := S \cup \{\neg P \mid P \in S\}$ for the set containing all elements in S and the negations of all elements in S.

B.3.1 Function `NDInit` (Line 1):

At **Line 2**, we guess a partial assignment over atomic presence constraints that makes ψ true. Recall that atomic presence constraints either are closed propositions or of the form "$\forall l\, \phi$" or "$\exists l\, \phi$" with ϕ a proposition that has l as free variable. We see this assignment as a set of atomic presence constraints which, when set to true, make ψ true. Note that negations of atomic presence constraints are atomic presence constraints. All closed atomic propositions refer to constant rounds; guess which ones are true (**Line 5**). This simplifies all closed propositions in X to either true of false: if any of them is false, we reject (**Line 8**). We put universally quantified element of X in U (**Line 9**) and existentially quantified ones in E (**Line 10**).

B.3.2 Function `NDComputeIteration` (Line 11):

The universal atomic presence constraints are checked at every round (**Lines 12** to **14**), while for each existential atomic presence constraints is checked at a round chosen non-deterministically (**Lines 15** to **18**). When checking a proposition, we guess which literals make them true, and put these literals in C to be checked later. Moreover, we check at round k all literals in C that are about round k (at **Lines 19** and **20**). Note that all literals in C are closed formulas hence their terms are constant integers.

B.3.3 Function `TestPresenceConstraint` (Line 21):

In this functon, we check whether we can stop the execution at round k, leaving all rounds $\geq k+1$ untouched. First, we check that E is empty. This means that a round has been guessed for every existential formula that has been in E. Moreover, we check that remaining formulas in C and U would be satisfied at rounds $\geq k+1$ if these rounds are left untouched by the execution, which is done in **Lines 23** to **25**. The test is expressed under the condition $\langle \emptyset, \mathsf{d}_0^{\mathsf{Reg}} \rangle \models \phi$ (although $\langle \emptyset, \mathsf{d}_0^{\mathsf{Reg}} \rangle$ is technically not a configuration as it has zero processes), and is implemented as follows. Any formula ϕ that is in C at the end of iteration k is about

```
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          */
2      Guess X ⊆ PosOrNeg(APC(ψ)) s.t. ψ is true when all APCs in X are true ;
3      for P in X do
4          for φ closed atomic proposition in P do
               /* φ refers to constant rounds only                    */
5              if φ guessed to be true then Add φ to C ; Replace φ by true in P ;
6              else Add ¬φ to C ; Replace φ by false in P ;
7          if P is a closed proposition then
8              Check that P is true with guessed values of atomic propositions ;
9          if P universal then Add P to U ;
10         if P existential then Add P to E ;
11 Function NDComputeIteration(E, U, C, λ) :
12     for "∀l φ" in U do
13         Guess ℒ ⊆ PosOrNeg(AP(φ[l ← k])) s.t. φ[l ← k] is true when all literals in ℒ
            are true ;
14         Add all literals in ℒ to C ;
15     for "∃l φ" in E do
16         if φ[l ← k] guessed to be true then
17             Guess ℒ ⊆ PosOrNeg(AP(φ[l → k])) s.t. φ[l ← k] is true when all literals in
                ℒ are true ;
18             Add all literals in ℒ to C ; Remove "∃l φ" from E ;
19     for φ in C about round k do
           // φ is of the form (negation of) "(q, k) populated", or (negation
              of) "rg_k[α] contains d"
20         Check that φ is satisfied in λ ; Remove φ from C ;
21 Function TestPresenceConstraint(E, U, C, λ) :
22     if E ≠ ∅ then return false ;
23     for φ ∈ C or "∀l φ" in U do
24         if ⟨∅, d_0^Reg⟩ ⊭ φ then
25             return false ;                  // Execution cannot stop at round k
26     return true ;
```

Algorithm 3 The functions at **Line 5**, **Line 10** and **Line 11** of Algorithm 1

round $l \geq k{+}1$ at this stage, and we check that ϕ is either of the form "$\neg((q,l)\ \mathsf{populated})$" or of the form "$\mathsf{rg}_l[\alpha]\ \mathsf{contains}\ \mathsf{d}_0$". A universal presence constraint $\forall l\, \phi$ must be satisfied on arbitrarily large rounds $\geq k{+}1$, and we check that we obtain true by setting in ϕ all "$(q,t)\ \mathsf{populated}$" to false, "$\mathsf{rg}_t[\alpha]\ \mathsf{contains}\ \mathsf{d}_0$" to true and "$\mathsf{rg}_t[\alpha]\ \mathsf{contains}\ \mathsf{d}$" to false for $\mathsf{d} \neq \mathsf{d}_0$.

▶ **Example B.7.** Consider $\phi_1 := \forall l\, ((q,l)\ \mathsf{populated}) \vee (\mathsf{rg}_l[\alpha]\ \mathsf{contains}\ \mathsf{d}_0)$ and $\phi_2 := \forall l\, (\mathsf{rg}_l[\alpha]\ \mathsf{contains}\ \mathsf{d})$ with $\mathsf{d} \neq \mathsf{d}_0$. One has $\langle \emptyset, \mathsf{d}_0^{\mathsf{Reg}} \rangle \models phi_1$, but $\langle \emptyset, \mathsf{d}_0^{\mathsf{Reg}} \rangle \not\models \phi_2$. There is no hope of finding a $\sigma \in \mathsf{Reach_a}(\mathsf{Init_a})$ such that $\sigma \models \phi_2$. ◀

B.4 Proof of Correctness of the Algorithm

▶ **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 $(\mathcal{P}, \psi)$.*

First, consider a computation of the algorithm that accepts at round $K \in \mathbb{N}$. For all $k \in [0, K]$, let τ_k denote the footprint on $[k{-}\mathsf{v}{+}1, k]$ guessed by the algorithm during iteration k. By applying Lemma 21, there exist $\sigma_0 \in \mathsf{Init_a}$ and an execution $\rho : \sigma_0 \xrightarrow{*} \sigma$ such that, for all $k \leq K$, $\mathsf{footprint}[k{-}\mathsf{v}{+}1, k](\rho) = \tau_k$. Moreover, ρ leaves rounds $\geq K$ untouched.

▶ **Lemma B.8.** *For every formula P that was in U, E or C at any point throughout the computation, one has $\sigma \models P$.*

Proof. Let L be a literal that has been in C at some point. If it was removed from C at **Line 20**, then C is satisfied by λ hence by σ. If it has remained in C until the end, then it is about round $l \geq K{+}1$ and $\langle \emptyset, \mathsf{d}_0^{\mathsf{Reg}} \rangle \models L$, hence $\sigma \models L$.

Consider "$\exists l\, \phi$" that has appeared in E at some point; it was added to E at **Line 10**. At some iteration k, "$\exists l\, \phi$" is removed from E at **Line 18**. All literals guessed at **Line 17** are added to C at **Line 18** hence are satisfied by σ, thus $\sigma \models \phi[l \leftarrow k]$ and $\sigma \models \exists l\, \phi$.

Similarly, consider "$\forall l\, \phi$" that has appeared in U at some point. By the same argument, for all $k \leq K$, $\sigma \models \phi[l \leftarrow k]$. Also, thanks to the verification at **Lines 23** to **25**, for all $k \geq K{+}1$, $\sigma \models \phi[l \leftarrow k]$, which proves that $\sigma \models \exists l\, \phi$. ◀

The previous lemma proves that all APCs guessed at **Line 2** are satisfied by σ. Note that the simplification at **Lines 5** and **6** does not change the truth value of APC P. Finally, we have $\sigma \models \psi$.

We now prove the converse implication: suppose that there exists $\rho : \sigma_0 \xrightarrow{*} \sigma$ with $\sigma \models \psi$. Since ρ is a finite execution, there exists K such that σ has no move with effect on rounds $> K$. We build an accepting computation of the algorithm as follows. First, the computation of the algorithm guesses σ_0 as initial configuration. At **Line 2**, it guesses APCs P such that $\sigma \models P$. At **Line 5**, it guesses the truth value of closed APs in σ, so that all formulas added to C, E and U are satisfied by σ. In the loop on k, it guesses ρ footprint by footprint. At execution k, the local configuration λ obtained is equal to $\mathsf{footprint}[k{-}\mathsf{v}, k](\sigma)$. Formulas in E and U do not have closed terms, and since quantified terms are of the form $l{+}m$ with l a free variable, literals added to C at iteration k refer to rounds $\geq k$; thanks to **Lines 19** to **20**, at the end of iteration k, all literals in C are about rounds $\geq k{+}1$. At the end of iteration K (or an earlier iteration), all formulas in C and U are satisfied by σ (which is blank after round K) hence `TestPresenceConstraint`(E, U, C, λ) succeeds and the computation accepts. This concluded the proof of Proposition 22.

B.5 Proof of Proposition 23

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

We first prove that footprints may be stored in polynomial space.

▶ **Lemma B.9.** *For all* $\sigma \in \Sigma, \sigma' \in \mathsf{Reach_a}(\sigma)$*, there exists* $\rho : \sigma \xrightarrow{*} \sigma'$ *s.t., for all* k*,* $\mathsf{footprint}[k{-}\mathsf{v}, k](\rho)$ *is storable in space* $O((|Q|^2\,(\mathsf{v}{+}1)^2\,\log(\mathsf{dim}\,|\mathrm{D}|)))$.

More specifically, we will prove that $\mathsf{footprint}[k{-}\mathsf{v}, k](\rho)$ is storable in space $O((|Q|^2\,(\mathsf{v}{+}1)^2 + |Q|\,(\mathsf{v}{+}1)^2\,\log(\mathsf{dim}\,|\mathrm{D}|)))$. Similarly to the roundless case, we introduce a notion of normal form. An execution ρ is in *normal form* if for every step in ρ, one the following conditions is satisfied:

- the step writes a symbol to a register, and this symbol is later read by another step, or
- it deserts the source location, or
- its destination location was not populated before the step and has never been populated before in the execution.

Note that the last two conditions combined imply that a given location is deserted at most once, as it cannot be deserted and then populated again.

▶ **Lemma B.10.** *For all execution* $\rho : \sigma \xrightarrow{*} \sigma'$*, there exists a execution* $\tilde{\rho} : \sigma \xrightarrow{*} \sigma'$ *that is in normal form.*

Proof. It suffices to iteratively:

- remove any read or increment that is non-deserting and does not cover a new location,
- remove any write that is non-deserting, does not populate a new location and whose written symbol is never read,
- turn into non-deserting any deserting step that deserts a location which is later populated again.

◀

▶ **Lemma B.11.** *An execution in normal form has at most* $|Q|(2\mathsf{v}{+}5)$ *steps on a given round* k.

Proof. First, any read or increment step at round k either deserts its source location which is never populated again, or populates its destination (*i.e.*, its destination was not populated before the step). However, each location has at most one step populating the location and one deserting the location. Since steps at round k may only desert locations of round k and populate locations at rounds k and $k{+}1$, at most $3|Q|$ steps at round k either desert or populate a location, among which at most $2|Q|$ read steps as they may only desert and populate locations of round k. Moreover, any write step at round k that does not populate or desert must be read later, and that has to be by a read step on a round between k and $k{+}\mathsf{v}$. Since there are at most $2|Q|(\mathsf{v}{+}1)$ read steps on these rounds, there are at most $2|Q|(\mathsf{v}{+}1)$ writes at round k that do not populate nor desert, hence in total at most $|Q|(2\mathsf{v}{+}5)$ steps at round k. ◀

▶ **Lemma B.12.** *If* ρ *is in normal form and* $k \in \mathbb{N}$*, then* $\mathsf{footprint}[k{-}\mathsf{v}, k](\rho)$ *is storable in polynomial space* $O((|Q|^2\,(\mathsf{v}{+}1)^2 + |Q|\,(\mathsf{v}{+}1)^2\,\log(\mathsf{dim}\,|\mathrm{D}|)))$.

Proof. This footprint only has steps at rounds $k-\mathsf{v}$ to k, hence in total at most $(\mathsf{v}+1)|Q|(2\mathsf{v}+5)$ steps. Since a move can be stored in $O(\log(|Q|) + \log(\mathrm{D}) + \log(\mathsf{v}) + \log(\mathsf{dim}))$ and a local configuration in $O((|Q| + \mathsf{dim}\ \log(|\mathrm{D}|))(\mathsf{v}+1))$ (storing the relative round instead of the absolute one), a footprint on $[k-\mathsf{v}, k]$ can be stored in polynomial space $O((|Q|^2\,(\mathsf{v}+1)^2 + |Q|\,(\mathsf{v}+1)^2\ \log(\mathsf{dim}\,|\mathrm{D}|)))$. ◀

Combining Lemmas B.10, B.11 and B.12 proves Lemma B.9. Observe that Lemma B.9 is only true under the assumption that we do not store the rounds of a footprint in absolute value but in relative value with respect to k; otherwise the space used would depend on k.

We now prove Proposition 23.

Thanks to Proposition B.9, one may store τ and T in polynomial space. U and E are storable in $O(|\psi|)$, as for every atomic presence constraint ϕ in U and E, either ϕ is present in ψ or its negation $\neg\phi$ is. Let M be the value of the greatest integer constant in ψ, which is in $O(|\psi|)$ thanks to unary encoding of the terms. A literal can get to C in two different ways: during the initialization (**Lines 5** and **6**) or while processing a presence constraint from U or E (**Lines 14** and **18**). There are at most $O(|\psi|)$ literals added to C in the initialization. Consider L a literal that is added to C at **Line 14** or **Line 18** during the computation of round k. Let r be the round appearing in L. Either r is a constant from ψ, or it was added at iteration $k' \leq k$, hence r is of the form $k' + m$ with $m \leq M$. In that case, note that $r \geq k$ because otherwise the literal would have been removed from C at iteration r. Either way, one has $0 \leq r - k \leq M$, hence a given element in C is storable in $O(|\psi|)$. Also, elements in C at round k were added to C either at the initialization or at a round in $[k-M, k]$, which bounds the total number of elements in C by $O(M|\psi|) = O(|\psi|^2)$ at any point in the computation, and C is storable in $O(|\psi|^3)$.