Automatic Analysis of Expected Termination Time for Population Protocols

Michael Blondin[1]
TU Munich, Germany
blondin@in.tum.de
https://orcid.org/0000-0003-2914-2734

Javier Esparza[2]
TU Munich, Germany
esparza@in.tum.de
https://orcid.org/0000-0001-9862-4919

Antonín Kučera[3]
Masaryk University, Brno, Czech Republic
kucera@fi.muni.cz
https://orcid.org/0000-0002-6602-8028

Abstract

Population protocols are a formal model of sensor networks consisting of identical mobile devices. Two devices can interact and thereby change their states. Computations are infinite sequences of interactions in which the interacting devices are chosen uniformly at random.

In well designed population protocols, for every initial configuration of devices, and for every computation starting at this configuration, all devices eventually agree on a consensus value. We address the problem of automatically computing a parametric bound on the expected time the protocol needs to reach this consensus. We present the first algorithm that, when successful, outputs a function $f(n)$ such that the expected time to consensus is bound by $\mathcal{O}(f(n))$, where n is the number of devices executing the protocol. We experimentally show that our algorithm terminates and provides good bounds for many of the protocols found in the literature.

2012 ACM Subject Classification Theory of computation → Distributed computing models, Theory of computation → Probabilistic computation, Theory of computation → Logic and verification

Keywords and phrases population protocols, performance analysis, expected termination time

Digital Object Identifier 10.4230/LIPIcs.CONCUR.2018.33

Related Version Extended version of the paper: [9], `https://arxiv.org/abs/1807.00331`.

Supplement Material Source code: `https://github.com/blondimi/pp-time-analysis`.

[1] Supported by the Fonds de recherche du Québec – Nature et technologies (FRQNT).
[2] Supported by ERC Advanced Grant (787367: PaVeS).
[3] Supported by the Czech Science Foundation, grant No. P202/12/G061. The presented results were achieved during the author's stay at TU München supported by the Friedrich Wilhelm Bessel Research Award (Alexander von Humboldt Foundation).

1 Introduction

Population protocols are a model of distributed computation in which agents with very limited computational resources randomly interact in pairs to perform computational tasks [3, 4]. They have been used as an abstract model of wireless networks, chemical reactions, and gene regulatory networks, and it has been shown that they can be implemented at molecular level (see, e.g., [23, 21, 11, 20]).

Population protocols compute by reaching a stable consensus in which all agents agree on a common output (typically a Boolean value). The output depends on the distribution of the initial states of the agents, called the initial *configuration*, and so a protocol computes a predicate that assigns a Boolean value to each initial configuration. For example, a protocol in which all agents start in the same state computes the predicate $x \geq c$ if the agents agree to output 1 when there are at least c of them, and otherwise agree to output 0. A protocol with two initial states computes the majority predicate $x \geq y$ if the agents agree to output 1 exactly when the initial number of agents in the first state is greater than or equal to the initial number of agents in the second state.

In previous work, some authors have studied the automatic verification of population protocols. Since a protocol has a finite state space for each initial configuration, model checking algorithms can be used to verify that the protocol behaves correctly for *a finite number* of initial configurations. However, this technique cannot prove that the protocol is correct for *every* configuration. In [16] it was shown that the problem of deciding whether a protocol computes some predicate, and the problem of deciding whether it computes a given predicate, are both decidable and at least as hard as the reachability problem for Petri nets.

In practice, protocols should not only correctly compute a predicate, but also do it fast. The most studied quantitative measure is the expected number of pairwise interactions needed to reach a stable consensus. The measure is defined for the stoichiometric model in which the pair of agents of the next interaction are picked uniformly at random. A derived measure is the *parallel time*, defined as the number of interactions divided by the number of agents. The first paper on population protocols already showed that every predicate can be computed by a protocol with expected total number of interactions $\mathcal{O}(n^2 \log n)$, where n is the number of agents [3, 4]. Since then, there has been considerable interest in obtaining upper and lower bounds on the number of interactions for some fundamental tasks, like *leader election* and *majority*, and there is also much work on finding trade-offs between the speed of a protocol and its number of states (see, e.g., [14, 1, 6] and the references therein). However, none of these works addresses the verification [10] problem: given a protocol, determine its expected number of interactions.

As in the qualitative case, probabilistic model checkers can be used to compute the expected number of interactions for a given configuration. Indeed, in this case the behaviour of the protocol is captured by a finite-state Markov chain, and the expected number of interactions can be computed as the expected number of steps until a bottom strongly connected component of the chain is reached. This was the path followed in [12], using the PRISM probabilistic model checker. However, as in the functional case, this technique cannot give a bound valid for *every* configuration.

This paper presents the first algorithm for the automatic computation of an upper bound on the expected number of interactions. The algorithm takes advantage of the hierarchical structure of population protocols where an initial configuration reaches a stable consensus by passing through finitely many "stages". Entering a next stage corresponds to entering a configuration where some behavioral restrictions become permanent (for example, some interactions become permanently disabled, certain states will never be populated again, etc.). The algorithm automatically identifies such stages and computes a finite acyclic *stage graph* representing the protocol evolution. If all bottom stages of the graph correspond to stabilized configurations, the algorithm proceeds by deriving bounds for the expected number of interactions to move from one stage to the next, and computes a bound for the expected number of interactions by taking an "asymptotic maximum" of these bounds. In unsuitable cases, the resulting upper bound can be higher than the actual expected number of interactions. We report on an implementation of the algorithm and its application to case studies.

Related work. To the best of our knowledge, we present the first algorithm for the automatic quantitative verification of population protocols. In fact, even for sequential randomized programs, the automatic computation of the expected time is little studied. After the seminal work of Flajolet *et al.* in [17], there is recent work by Kaminski *et al.* [19] on the computation of expected runtimes using weakest preconditions, by Chatterje *et al.* on the automated analysis of recurrence relations for expected time [10], by Van Chan Ngo *et al.* [22] on the automated computation of bounded expectations using amortized resource analysis, and by Batz *et al.* [5, 22] on the computation of sampling times for Bayesian networks. These works are either not targeted to distributed systems like population protocols, or do not provide the same degree of automation as ours.

Structure of the paper. In Section 2, we introduce population protocols and a simple modal logic to reason about their behaviours. In Section 3, we introduce stage graphs and explain how they allow to prove upper bounds on the expected number of interactions of population protocols. We then give a dedicated algorithm for the computation of stage graphs in Section 4, analyze the bounds derived by this algorithm in Section 5, and report on experimental results in Section 6. Finally, we conclude in Section 7. Due to space constraints, we defer proofs of some statements to a full version of this paper [9].

2 Population protocols

In this section, we introduce population protocols and their semantics. We assume familiarity with basic notions of probability theory, such as probability space, random variables, expected value, etc. When we say that some event happens *almost surely*, we mean that the probability of the event is equal to one. We use $\mathbb{N}$ to denote the set of non-negative integers.

A *population* consists of n agents with states from a finite set $Q = \{A, B, \ldots\}$ interacting according to a directed *interaction graph* $\mathcal{G}$ (without self-loops) over the agents. The interaction proceeds in a sequence of steps, where in each step an edge of the interaction graph is selected uniformly at random, and the states (A, B) of the two chosen agents are updated according to a transition function containing rules of the form $(C, D) \mapsto (E, F)$. We assume that for each pair of states (C, D), there is at least one rule $(C, D) \mapsto (E, F)$. If there are several rules with the same left-hand side, then one is selected uniformly at random. The unique agent identifiers are not known to the agents and not used by the protocol.

Usually, $\mathcal{G}$ is considered as a *complete* graph, and this assumption is adopted also in this work. Since the agent identifiers are hidden and $\mathcal{G}$ is complete, a population is fully determined by the number of agents in each state. Formally, a *configuration* is a vector $\mathcal{C} \in \mathbb{N}^Q$, where $\mathcal{C}(A)$ is the number of agents in state A. For every $A \in Q$, we use $\mathbf{1}_A$ to denote the vector satisfying $\mathbf{1}_A(A) = 1$ and $\mathbf{1}_A(B) = 0$ for all $B \neq A$. Note that there is no difference between transitions $(A, B) \mapsto (C, D)$ and $(A, B) \mapsto (D, C)$, because both of them update a given configuration in the same way.

Most of the population protocols studied for complete interaction graphs have a symmetric transition function where pairs (A, B) and (B, A) are updated in the same way. For the sake of simplicity, we restrict our attention to symmetric protocols.[4] Then, the transitions can be written simply as $AB \mapsto CD$, because the ordering of states before/after the $\mapsto$ symbol is irrelevant. Formally, AB and CD are understood as elements of $Q^{\langle 2\rangle}$, i.e., multisets over Q with precisely two elements.

▶ **Definition 1.** A *population protocol* is a tuple $\mathcal{P} = (Q, T, \Sigma, I, O)$ where
- Q is a non-empty finite set of *states*;
- $T : Q^{\langle 2\rangle} \times Q^{\langle 2\rangle}$ is a total *transition relation*;
- Σ is a non-empty finite *input alphabet*,
- $I : \Sigma \to Q$ is the *input function* mapping input symbols to states,
- $O : Q \to \{0, 1\}$ is the *output function*.

We write $AB \mapsto CD$ to indicate that $(AB, CD) \in T$. When defining the set T, we usually specify the outgoing transitions only for some subset of $Q^{\langle 2\rangle}$. For the other pairs AB, there (implicitly) exists a single *idle* transition $AB \mapsto AB$. We also write $I(\Sigma)$ to denote the set $\{q \in Q \mid q = I(\sigma) \text{ for some } \sigma \in \Sigma\}$.

2.1 Executing population protocols

A transition $AB \mapsto CD$ is *enabled* in a configuration $\mathcal{C}$ if $\mathcal{C} - \mathbf{1}_A - \mathbf{1}_B \geq \mathbf{0}$. A transition $AB \mapsto CD$ enabled in $\mathcal{C}$ can *fire* and thus produce a configuration $\mathcal{C}' = \mathcal{C} - \mathbf{1}_A - \mathbf{1}_B + \mathbf{1}_C + \mathbf{1}_D$. The *probability* of executing a transition $AB \mapsto CD$ enabled in $\mathcal{C}$ is defined by

$$\mathbb{P}[\mathcal{C}, AB \mapsto CD] = \begin{cases} \frac{\mathcal{C}(A)\cdot(\mathcal{C}(A)-1)}{(n^2-n)\cdot|\{EF\in Q^{\langle 2\rangle}: AA\mapsto EF\}|} & \text{if } A = B\,, \\ \frac{2\cdot\mathcal{C}(A)\cdot\mathcal{C}(B)}{(n^2-n)\cdot|\{EF\in Q^{\langle 2\rangle}: AB\mapsto EF\}|} & \text{if } A \neq B\,. \end{cases}$$

where n is the size of $\mathcal{C}$. Note that $2 \cdot \mathcal{C}(A) \cdot \mathcal{C}(B)$ is the number of directed edges connecting agents in states A and B (when $A \neq B$), and $n^2 - n$ is the total number of directed edges in a complete directed graph without self-loops with n vertices. If a pair of agents in states A and B is selected, one of the outgoing transitions of AB is chosen uniformly at random.

We write $\mathcal{C} \to \mathcal{C}'$ to indicate that $\mathcal{C}'$ is obtained from $\mathcal{C}$ by firing some transition, and we use $\mathbb{P}[\mathcal{C} \to \mathcal{C}']$ to denote the probability of executing a transition enabled in $\mathcal{C}$ producing $\mathcal{C}'$. Note that there can be several transitions enabled in $\mathcal{C}$ producing $\mathcal{C}'$, and $\mathbb{P}[\mathcal{C} \to \mathcal{C}']$ is the total probability of executing some of them.

An *execution* initiated in a given configuration $\mathcal{C}$ is a finite sequence $\mathcal{C}_0, \ldots, \mathcal{C}_\ell$ of configurations such that $\ell \in \mathbb{N}$, $\mathcal{C}_0 = \mathcal{C}$, and $\mathcal{C}_i \to \mathcal{C}_{i+1}$ for all $i < \ell$. A configuration $\mathcal{C}'$ is *reachable* from a configuration $\mathcal{C}$ if there is an execution initiated in $\mathcal{C}$ ending in $\mathcal{C}'$. A *run* is an infinite sequence of configurations $\omega = \mathcal{C}_0, \mathcal{C}_1, \ldots$ such that every finite prefix of ω is an execution. The configuration $\mathcal{C}_i$ of a run ω is also denoted by ω_i. For a given execution $\mathcal{C}_0, \ldots, \mathcal{C}_\ell$, we use $\mathit{Run}(\mathcal{C}_0, \ldots, \mathcal{C}_\ell)$ to denote the set of all runs starting with $\mathcal{C}_0, \ldots, \mathcal{C}_\ell$.

[4] All of the presented results can easily be extended to non-symmetric population protocols. The only technical difference is the way of evaluating/estimating the probability of executing a given transition in a given configuration.

For every configuration $\mathcal{C}$, we define the probability space $(Run(\mathcal{C}), \mathcal{F}, \mathbb{P}_{\mathcal{C}})$, where $\mathcal{F}$ is the σ-algebra generated by all $Run(\mathcal{C}_0, \ldots, \mathcal{C}_\ell)$ such that $\mathcal{C}_0, \ldots, \mathcal{C}_\ell$ is an execution initiated in $\mathcal{C}$, and $\mathbb{P}_{\mathcal{C}}$ is the unique probability measure satisfying $\mathbb{P}_{\mathcal{C}}(Run(C_0, \ldots, C_\ell)) = \prod_{i=0}^{\ell-1} \mathbb{P}[\mathcal{C}_i \to \mathcal{C}_{i+1}]$.

2.2 A simple modal logic for population protocols

To specify properties of configurations, we use a qualitative variant of the branching-time logic EF. Let $AP_{\mathcal{P}} = Q \cup \{A! \mid A \in Q$ such that there is a non-idle transition $AA \mapsto BC\}$. The formulae of our qualitative logic are constructed in the following way, where a ranges over $AP_{\mathcal{P}} \cup \{Out_0, Out_1\}$:

$$\varphi \quad ::= \quad a \;\mid\; \neg\varphi \;\mid\; \varphi_0 \wedge \varphi_1 \;\mid\; \Box\varphi \;\mid\; \Diamond\varphi.$$

The semantics is defined inductively:

$$\begin{array}{lll}
\mathcal{C} \models A & \text{iff} & \mathcal{C}(A) > 0, \\
\mathcal{C} \models A! & \text{iff} & \mathcal{C}(A) = 1, \\
\mathcal{C} \models Out_0 & \text{iff} & O(A) = 0 \text{ for all } A \in Q \text{ such that } \mathcal{C}(A) > 0, \\
\mathcal{C} \models Out_1 & \text{iff} & O(A) = 1 \text{ for all } A \in Q \text{ such that } \mathcal{C}(A) > 0, \\
\mathcal{C} \models \neg\varphi & \text{iff} & \mathcal{C} \not\models \varphi, \\
\mathcal{C} \models \varphi_0 \wedge \varphi_1 & \text{iff} & \mathcal{C} \models \varphi_0 \text{ and } \mathcal{C} \models \varphi_1, \\
\mathcal{C} \models \Box\varphi & \text{iff} & \mathbb{P}_{\mathcal{C}}(\{\omega \in Run(\mathcal{C}) \mid \omega_i \models \varphi \text{ for all } i \in \mathbb{N}\}) = 1, \\
\mathcal{C} \models \Diamond\varphi & \text{iff} & \mathbb{P}_{\mathcal{C}}(\{\omega \in Run(\mathcal{C}) \mid \omega_i \models \varphi \text{ for some } i \in \mathbb{N}\}) = 1.
\end{array}$$

Note that $\mathcal{C} \models \Box\varphi$ iff all configurations reachable from $\mathcal{C}$ satisfy φ, and $\mathcal{C} \models \Diamond\varphi$ iff a run initiated in $\mathcal{C}$ visits a configuration satisfying φ almost surely (i.e., with probability one). We also use `tt`, `ff`, and other propositional connectives whose semantics is defined in the standard way. Furthermore, we occasionally interpret a given set of configurations $\mathcal{B}$ as a formula where $\mathcal{C} \models \mathcal{B}$ iff $\mathcal{C} \in \mathcal{B}$.

For every formula φ, we define a random variable $Steps_\varphi$ assigning to every run $\mathcal{C}_0, \mathcal{C}_1, \ldots$ either the least $\ell \in \mathbb{N}$ such that $\mathcal{C}_\ell \models \varphi$, or ∞ if there is no such ℓ. For a given configuration $\mathcal{C}$, we use $\mathbb{E}_{\mathcal{C}}[Steps_\varphi]$ to denote the expected value of $Steps_\varphi$ in the probability space $(Run(\mathcal{C}), \mathcal{F}, \mathbb{P}_{\mathcal{C}})$.

2.3 Computable predicates, interaction complexity

Every *input* $\mathcal{X} \in \mathbb{N}^\Sigma$ is mapped to the configuration $\mathcal{C}_{\mathcal{X}}$ such that

$$\mathcal{C}_{\mathcal{X}}(q) = \sum_{\substack{\sigma \in \Sigma \\ I(\sigma) = q}} \mathcal{X}(\sigma) \qquad \text{for every } q \in Q.$$

An *initial* configuration is a configuration of the form $\mathcal{C}_{\mathcal{X}}$ where $\mathcal{X}$ is an input. A configuration $\mathcal{C}$ is *stable* if $\mathcal{C} \models Stable$, where $Stable \equiv (\Box Out_0) \vee (\Box Out_1)$. We say that a protocol $\mathcal{P}$ *terminates* if $\mathcal{C} \models \Diamond Stable$ for every initial configuration $\mathcal{C}$. A protocol $\mathcal{P}$ *computes* a unary predicate Λ on inputs if it terminates and every stable configuration $\mathcal{C}'$ reachable from an initial configuration $\mathcal{C}_{\mathcal{X}}$ satisfies $\mathcal{C}' \models Out_x$, where x is either 1 or 0 depending on whether $\mathcal{X}$ satisfies Λ or not, respectively.

The *interaction complexity* of $\mathcal{P}$ is a function $InterComplexity_{\mathcal{P}}$ assigning to every $n \geq 1$ the *maximal* $\mathbb{E}_{\mathcal{C}}[Steps_{Stable}]$, where $\mathcal{C}$ ranges over all initial configurations of size n. Since several interactions may be running in parallel, the *time complexity* of $\mathcal{P}$ is defined as

$\mathit{InterComplexity}_{\mathcal{P}}(n)$ divided by n. Hence, asymptotic bounds on interaction complexity immediately induce the corresponding bounds on time complexity.

2.4 Running examples

A well-studied predicate for population protocols is *majority*. Here, $\Sigma = \{A, B\}$, $I(A) = A$, $I(B) = B$, and the protocol computes whether there are at least as many agents in state B as there are in state A. As running examples, we use two different protocols for computing majority, taken from [15] and [18].

▶ **Example 2** (majority protocol of [15]). We have that $Q = \{A, B, a, b\}$, $O(A) = O(a) = 0$, $O(B) = O(b) = 1$, and the transitions are the following: $AB \mapsto ab$, $Ab \mapsto Aa$, $Ba \mapsto Bb$ and $ba \mapsto bb$.

▶ **Example 3** (majority protocol of [18]). Here, $Q = \{A, B, C, a, b\}$, $O(A) = O(a) = 0$, $O(B) = O(b) = O(C) = 1$, and the transitions are the following: $AB \mapsto bC$, $AC \mapsto Aa$, $BC \mapsto Bb$, $Ba \mapsto Bb$, $Ab \mapsto Aa$ and $Ca \mapsto Cb$.

3 Stages of population protocols

Most of the existing population protocols are designed so that each initial configuration passes through finitely many "stages" before reaching a stable configuration. Entering a next stage corresponds to performing some additional non-reversible changes in the structure of configurations. Hence, the transition relation between stages is acyclic, and each configuration in a non-terminal stage eventually enters one of the successor stages with probability one. This intuition is formalized in our next definition.

▶ **Definition 4.** Let $\mathcal{P} = (Q, T, \Sigma, I, O)$ be a population protocol. A *stage graph* for $\mathcal{P}$ is a triple $\mathcal{G} = (\mathbb{S}, \hookrightarrow, [\![\cdot]\!])$ where $\mathbb{S}$ is a finite set of *stages*, $\hookrightarrow \subseteq \mathbb{S} \times \mathbb{S}$ is an acyclic transition relation, and $[\![\cdot]\!]$ is a function assigning to each $S \in \mathbb{S}$ a set of configurations $[\![S]\!]$ such that the following conditions are satisfied:

(a) For every initial configuration $\mathcal{C}$ there is some $S \in \mathbb{S}$ such that $\mathcal{C} \in [\![S]\!]$.

(b) For every $S \in \mathbb{S}$ with at least one successor under $\hookrightarrow$, and for every $\mathcal{C} \in [\![S]\!]$, we have that[5] $\mathcal{C} \models \Diamond \mathit{Succ}(S)$, where $\mathit{Succ}(S) \equiv \bigvee_{S \hookrightarrow S'} [\![S']\!]$.

Note that a stage graph for $\mathcal{P}$ is not determined uniquely. Even a trivial graph with one stage S and no transitions such that $[\![S]\!]$ is the set of all configurations is a valid stage graph by Definition 4. To analyze the interaction complexity of $\mathcal{P}$, we need to construct a stage graph so that the expected number of transitions needed to move from stage to stage can be determined easily, and all terminal stages consist only of stable configurations (see Lemma 6 below).

Formally, a stage S is *terminal* if it does not have any successors, i.e., there is no S' satisfying $S \hookrightarrow S'$. Let $\mathcal{T}$ be the set of all terminal stages, and let $\mathit{Term} \equiv \bigvee_{S \in \mathcal{T}} [\![S]\!]$. It follows directly from Definition 4(b) that $\mathcal{C} \models \Diamond \mathit{Term}$ for every initial configuration $\mathcal{C}$. Let $\mathit{ReachTerminal}_{\mathcal{G}}$ be a function assigning to every $n \geq 1$ the maximal $\mathbb{E}_{\mathcal{C}}[\mathit{Steps}_{\mathit{Term}}]$, where $\mathcal{C}$ ranges over all initial configurations of size n. Furthermore, for every $S \in \mathbb{S}$, we define a

[5] Recall that sets of configurations can be interpreted as formulae of the modal logic introduced in Section 2.2.

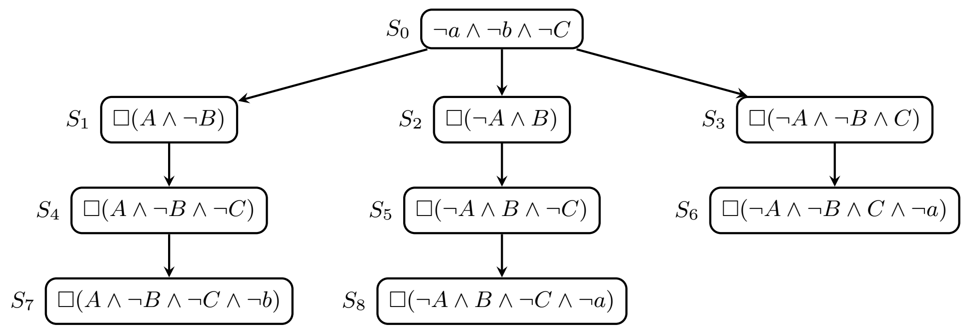
S_0 $\neg a \wedge \neg b \wedge \neg C$
S_1 $\Box(A \wedge \neg B)$
S_2 $\Box(\neg A \wedge B)$
S_3 $\Box(\neg A \wedge \neg B \wedge C)$
S_4 $\Box(A \wedge \neg B \wedge \neg C)$
S_5 $\Box(\neg A \wedge B \wedge \neg C)$
S_6 $\Box(\neg A \wedge \neg B \wedge C \wedge \neg a)$
S_7 $\Box(A \wedge \neg B \wedge \neg C \wedge \neg b)$
S_8 $\Box(\neg A \wedge B \wedge \neg C \wedge \neg a)$

Figure 1 A stage graph for the majority protocol of Example 3.

function $\mathit{ReachNext}_S$ assigning to every $n \geq 1$ the maximal $\mathbb{E}_{\mathcal{C}}[\mathit{Steps}_{\mathit{Succ}(S)}]$, where $\mathcal{C}$ ranges over all configurations of $[\![S]\!]$ of size n (if $[\![S]\!]$ does not contain any configuration of size n, we put $\mathit{ReachNext}_S(n) = 0$).

An asymptotic upper bound for $\mathit{ReachTerminal}_{\mathcal{G}}$ can be obtained by developing an asymptotic upper bound for all $\mathit{ReachNext}_S$, where $S \in \mathbb{S}$. Even though such a bound on $\mathit{ReachTerminal}_{\mathcal{G}}$ depends on $|\mathbb{S}|$, the latter is a constant since it is independent from the number of agents. Therefore, the following holds:

▶ **Lemma 5.** *Let $\mathcal{P} = (Q, T, \Sigma, I, O)$ be a population protocol and $\mathcal{G} = (\mathbb{S}, \hookrightarrow, [\![\cdot]\!])$ a stage graph for $\mathcal{P}$. Let $f : \mathbb{N} \to \mathbb{N}$ be a function such that $\mathit{ReachNext}_S \in \mathcal{O}(f)$ for all $S \in \mathbb{S}$. Then $\mathit{ReachTerminal}_{\mathcal{G}} \in \mathcal{O}(f)$.*

Observe that if every terminal stage S satisfies $[\![S]\!] \subseteq \mathit{Stable}$, then $\mathit{InterComplexity}_{\mathcal{P}} \leq \mathit{ReachTerminal}_{\mathcal{G}}$ (pointwise). Thus, we obtain the following:

▶ **Lemma 6.** *Let $\mathcal{P} = (Q, T, \Sigma, I, O)$ be a population protocol and $\mathcal{G} = (\mathbb{S}, \hookrightarrow, [\![\cdot]\!])$ a stage graph for $\mathcal{P}$ such that $[\![S]\!] \subseteq \mathit{Stable}$ for every terminal stage S. Let $f : \mathbb{N} \to \mathbb{N}$ be a function such that $\mathit{ReachNext}_S \in \mathcal{O}(f)$ for all $S \in \mathbb{S}$. Then $\mathit{InterComplexity}_{\mathcal{P}} \in \mathcal{O}(f)$.*

3.1 An example of a stage graph

In this section, we give an example of a stage graph $\mathcal{G}$ for the majority protocol $\mathcal{P}$ of Example 3, and we show how to analyze the interaction complexity of $\mathcal{P}$ using $\mathcal{G}$.

The stage graph $\mathcal{G}$ of Fig. 1 is a simplified version of the stage graph computed by the algorithm of the forthcoming Section 4. Intuitively, the hierarchy of stages corresponds to "disabling more and more states" along runs initiated in initial configurations. For each stage S_i of $\mathcal{G}$, the set $[\![S_i]\!]$ consists of all configurations satisfying the associated formula shown in Fig. 1. Since $[\![S_0]\!]$ is precisely the set of all initial configurations, Condition (a) of Definition 4 is satisfied. For every $\mathcal{C}_0 \in [\![S_0]\!]$, transition $AB \mapsto bC$ can be executed in all configurations reachable from $\mathcal{C}_0$ until A or B disappears. Furthermore, the number of A's and B's can only decrease along every run initiated in $\mathcal{C}_0$. Hence, $\mathcal{C}_0$ almost surely reaches a configuration $\mathcal{C}$ where A or B (or both of them) disappear. Note that if, e.g., $\mathcal{C}(A) = 0$ and $\mathcal{C}(B) > 0$, then this property is "permanent", i.e., every successor $\mathcal{C}'$ of $\mathcal{C}$ also satisfies $\mathcal{C}'(A) = 0$ and $\mathcal{C}'(B) > 0$. Thus, we obtain the stages S_1, S_2, and S_3. Observe that if A and B disappear simultaneously (which happens iff the initial configuration $\mathcal{C}_0$ satisfies $\mathcal{C}_0(A) = \mathcal{C}_0(B)$), then the configuration $\mathcal{C}$ will contain at least one copy of C which cannot be removed.

In all configurations of $[\![S_1]\!]$, the only potentially executable transitions are the following: $AC \mapsto Aa$, $Ab \mapsto Aa$, $Ca \mapsto Cb$. Since A appears in all configurations reachable from configurations of $[\![S_1]\!]$, the transition $AC \mapsto Aa$ stays enabled in all of these configurations until C disappears. Hence, every configuration of $[\![S_1]\!]$ almost surely reaches a configuration of $[\![S_4]\!]$. Similarly, we can argue that all configurations of $[\![S_4]\!]$ almost surely reach a configuration of $[\![S_7]\!]$, etc. Hence, Condition (b) of Definition 4 is also satisfied.

Let $\mathcal{C}_0 \in [\![S_0]\!]$ be an initial configuration of size n, and let $\mathcal{C}$ be a configuration reachable from $\mathcal{C}_0$ such that $m = \min\{\mathcal{C}(A), \mathcal{C}(B)\} > 0$. The probability of firing $AB \mapsto bC$ stays larger than m^2/n^2 in all configurations reached from $\mathcal{C}$ by executing a finite sequence of transitions *different* from $AB \mapsto bC$. This means that $AB \mapsto bC$ is fired after at most n^2/m^2 trials on average. Since $\min\{\mathcal{C}_0(A), \mathcal{C}_0(B)\} \leq n/2$, we obtain

$$\mathit{ReachNext}_{S_0}(n) \quad\leq\quad \sum_{i=1}^{n/2} \frac{n^2}{i^2} \quad\leq\quad n^2 \cdot \sum_{i=1}^{n} \frac{1}{i^2} \quad\leq\quad n^2 \cdot \mathcal{H}_{n,2} \quad\in\quad \mathcal{O}(n^2)\,.$$

Here, $\mathcal{H}_{n,2}$ is the n-th Harmonic number of order 2. As $\lim_{n\to\infty} \mathcal{H}_{n,2} = c < \infty$, we have that $n^2 \cdot \mathcal{H}_{n,2} \in \mathcal{O}(n^2)$.

Now, let us analyze $\mathit{ReachNext}_{S_1}(n)$. Let $\mathcal{C} \in [\![S_1]\!]$ be a configuration of size n. We need to fire the transition $AC \mapsto Aa$ repeatedly until all C's disappear. Let $\mathcal{C}'$ be a configuration reachable from $\mathcal{C}$ such that $\mathcal{C}'(C) = m$. Since $\mathcal{C} \models \Box(A \wedge \neg B)$, we have that $\mathcal{C}'(A) > 0$, and hence the probability of firing $AC \mapsto Aa$ in $\mathcal{C}'$ is at least m/n^2. Thus, we obtain

$$\mathit{ReachNext}_{S_1}(n) \quad\leq\quad \sum_{i=1}^{n} \frac{n^2}{i} \quad\leq\quad n^2 \cdot \sum_{i=1}^{n} \frac{1}{i} \quad\leq\quad n^2 \cdot \mathcal{H}_{n} \quad\in\quad \mathcal{O}(n^2 \log(n))\,.$$

Here $\mathcal{H}_n$ denotes the n-th Harmonic number (of order 1). Since $\lim_{n\to\infty} \mathcal{H}_n = c \cdot \log(n)$ where c is a constant, we get $n^2 \cdot \mathcal{H}_n \in \mathcal{O}(n^2 \log(n))$.

Similarly, we can show that $\mathit{ReachNext}_{S_i}(n) \in \mathcal{O}(n^2 \log(n))$ for every stage S_i of the considered stage graph. Since all configurations associated to terminal stages are stable, we can apply Lemma 6 and conclude that $\mathit{InterComplexity}_{\mathcal{P}} \in \mathcal{O}(n^2 \log(n))$. Let us note that the algorithm of the forthcoming Section 4 can derive this result fully automatically in less than a second.

4 Computing a stage graph

In this section, we give an algorithm computing a stage graph for a given population protocol. Intuitively, the algorithm tries to identify a subset of transitions which will be simultaneously and permanently disabled in the future with probability one, and also performs a kind of "case analysis" how this can happen. The resulting stage graph admits computing an upper asymptotic bounds on $\mathit{ReachNext}_S$ for every stage S, which allows to compute an asymptotic upper bound on the interaction complexity of the protocol by applying Lemma 6.

For the rest of this section, we fix a population protocol $\mathcal{P} = (Q, T, \Sigma, I, O)$. A *valuation* is a *partial* function $\nu : AP_{\mathcal{P}} \to \{\mathtt{tt}, \mathtt{ff}\}$ such that $\nu(A!) = \mathtt{tt}$ implies $\nu(A) = \mathtt{tt}$ whenever $A!, A \in \mathit{Dom}(\nu)$, where $\mathit{Dom}(\nu)$ is the domain of ν. Slightly abusing our notation, we also denote by ν the propositional formula

$$\bigwedge_{\substack{p \in \mathit{Dom}(\nu)\\ \nu(p)=\mathtt{tt}}} p \quad\wedge\quad \bigwedge_{\substack{p \in \mathit{Dom}(\nu)\\ \nu(p)=\mathtt{ff}}} \neg p$$

Hence, by writing $\mathcal{C} \models \nu$ we mean that $\mathcal{C}$ satisfies the above formula.

For every *transition head* $AB \in Q^{\langle 2\rangle}$, let ξ_{AB} be either the formula $\neg A \vee \neg B$ or the formula $\neg A \vee A!$, depending on whether $A \neq B$ or $A = B$, respectively. Hence, the formulae ξ_{AB} and $\neg\xi_{AB}$ say that all transitions of the form $AB \mapsto CD$ are disabled and enabled, respectively. For a given set $\mathcal{T} \subseteq Q^{\langle 2 \rangle}$, consider the propositional formula $\Psi_{\mathcal{T}} \equiv \bigwedge_{AB \in \mathcal{T}} \xi_{AB}$. To simplify our notation, we write just $\mathcal{T}$ instead of $\Psi_{\mathcal{T}}$, i.e., $\mathcal{C} \models \mathcal{T}$ iff all transitions specified by $\mathcal{T}$ are disabled in $\mathcal{C}$.

▶ **Definition 7.** Let $\mathcal{P} = (Q, T, \Sigma, I, O)$ be a population protocol. A $\mathcal{P}$*-stage* is a triple $S = (\Phi, \pi, \mathcal{T})$ where

- Φ is a propositional formula over $AP_{\mathcal{P}}$,
- π is a valuation, called the *persistent valuation*,
- $\mathcal{T} \subseteq Q^{\langle 2 \rangle}$ is a set of transition heads, called the *permanently disabled transition heads*.

For every $\mathcal{P}$-stage $S = (\Phi, \pi, \mathcal{T})$, we put $[\![S]\!] = \{\mathcal{C} \mid \mathcal{C} \models \Phi \wedge \Box\pi \wedge \Box\mathcal{T}\}$.

Our algorithm computes a stage graph for $\mathcal{P}$ gradually by adding more and more $\mathcal{P}$-stages. It starts by inserting the *initial* $\mathcal{P}$-stage $S_0 = (\Phi, \emptyset, \emptyset)$, where

$$\Phi \equiv \left(\bigvee_{A \in I(\Sigma)} A \right) \wedge \bigwedge_{A \in Q \smallsetminus I(\Sigma)} \neg A \,.$$

Note that $[\![S_0]\!]$ is precisely the set of all initial configurations (the empty conjunction is interpreted as *true*). Then, the algorithm picks an unprocessed $\mathcal{P}$-stage in the part of the stage graph constructed so far, and computes its immediate successors. This goes on until all $\mathcal{P}$-stages become either internal or terminal. Since the total number of constructed $\mathcal{P}$-stages can be exponential in the size of $\mathcal{P}$, the worst-case complexity of our algorithm is exponential. However, as we shall see in Section 6, protocols with hundreds of states and transitions can be successfully analyzed even by our prototype implementation.

Let $S = (\Phi, \pi, \mathcal{T})$ be a non-terminal $\mathcal{P}$-stage, and let $AP_S \subseteq AP_{\mathcal{P}}$ be the set of all atomic propositions appearing in the formula Φ. The successor $\mathcal{P}$-stages of S are constructed as follows. First, the algorithm computes the set Val_S consisting of all valuations ν with domain AP_S such that ν satisfies Φ when the latter is interpreted over AP_S. Intuitively, this corresponds to dividing $[\![S]\!]$ into disjoint "subcases" determined by different ν's (as we shall see, Φ always implies the formula $\pi \wedge \mathcal{T}$, so ν cannot be in conflict with the information represented by π and $\mathcal{T}$; furthermore, we have $\mathit{Dom}(\pi) \subseteq \mathit{Dom}(\nu)$). Then, for each $\nu \in \mathit{Val}_S$, a $\mathcal{P}$-stage S_ν is constructed, and S_ν may or may not become a successor of S. If none of these S_ν becomes a successor of S, then S is declared as terminal.

Let us fix some $\nu \in \mathit{Val}_S$. In the rest of this section, we show how to compute the $\mathcal{P}$-stage $S_\nu = (\Phi_\nu, \pi_\nu, \mathcal{T}_\nu)$, and how to determine whether or not S_ν becomes a successor of S. An explicit pseudocode for constructing S_ν is given in [9].

4.1 Computing the valuation π_ν

The valuation π_ν is obtained by extending π with the "permanent part" of ν. Intuitively, we try to identify $A \in Q$ such that $\nu(A) = \mathtt{tt}$ (or $\nu(A) = \mathtt{ff}$) and all transitions containing A on the left-hand (or the right-hand) side are permanently disabled. Furthermore, we also try to identify $A \in Q$ such that $\nu(A!) = \mathtt{tt}$ and the number of A's cannot change by firing transitions which are not permanently disabled. Technically, this is achieved by a simple fixed-point computation guaranteed to terminate quickly. The details are given in [9].

4.2 Computing the set $\mathcal{T}_\nu$ and the formula Φ_ν

In some cases, the constructed persistent valuation π_ν already guarantees that a configuration satisfying $\pi_\nu \wedge \mathcal{T}$ is stable or cannot evolve (fire non-idle transitions) any further. Then, we in

$$\begin{tikzcd}
A \arrow[d] \arrow[dr] & B \arrow[d] \arrow[dl] \\
a \arrow[r] & b \arrow[l, bend left]
\end{tikzcd}
\qquad\qquad
\begin{tikzcd}
A \arrow[d] \arrow[ddrr] & & B \arrow[dd] \arrow[dll] \\
C \arrow[d] \arrow[drr] & & \\
a \arrow[rr] & & b \arrow[ll, bend left]
\end{tikzcd}$$

Figure 2 Transformation graphs of Example 8.

fact identified a subset of configurations belonging to $[\![S]\!]$ which does not require any further analysis. Hence, we put $\mathcal{T}_\nu = \mathcal{T}$, $\Phi_\nu = \pi_\nu$, and the configuration S_ν becomes a successor $\mathcal{P}$-stage of S declared as terminal.

Formally, we say that $(\pi_\nu, \mathcal{T})$ is *stable* if there is $x \in \{0,1\}$ such that for all states $A \in Q$ where $\pi_\nu(A) = \texttt{tt}$ or $A \notin Dom(\pi_\nu)$ we have that $Out(A) = x$, and for every transition $CD \mapsto EF$ where $Out(E) \neq x$ or $Out(F) \neq x$, the formula $(\pi_\nu \wedge \mathcal{T}) \Rightarrow \xi_{CD}$ is a propositional tautology. Furthermore, we say that $(\pi_\nu, \mathcal{T})$ is *dead* if it is not stable and for every non-idle transition $CD \mapsto EF$ we have that the formula $(\pi_\nu \wedge \mathcal{T}) \Rightarrow \xi_{CD}$ is a propositional tautology.

If S_ν is *not* stable or dead, we use π_ν and $\mathcal{T}$ to compute the *transformation graph* G_ν, and then analyze G_ν to determine $\mathcal{T}_\nu$ and Φ_ν.

4.2.1 The transformation graph

The vertices of the transformation graph G_ν are the states which have *not* yet been permanently disabled according to π_ν, and the edges are determined by a set of transitions whose heads have not yet been permanently disabled according to π_ν and $\mathcal{T}$. Formally, we put $G_\nu = (V, \rightarrow)$ where the set of vertices V consists of all $A \in Q$ such that either $A \notin Dom(\pi_\nu)$ or $\pi_\nu(A) = \texttt{tt}$, and the set of edges is determined as follows: Let $AB \mapsto CD$ be a non-idle transition such that $(\pi_\nu \wedge \mathcal{T}) \Rightarrow \xi_{AB}$ is *not* a tautology.

- If the sets $\{A, B\}$ and $\{C, D\}$ are disjoint, then the transition generates the edges $A \rightarrow C$, $A \rightarrow D$, $B \rightarrow C$, $B \rightarrow D$. Intuitively, both A and B can be "transformed" into C or D.
- Otherwise, the transition has the form $AB \mapsto AD$ for $B \neq D$. In this case it generates the edge $B \rightarrow D$. Intuitively, B can be "transformed" into D in the context of A.

▶ **Example 8.** Consider the protocol of Example 2 and its initial stage $S = (\Phi, \pi, \mathcal{T})$ where $\Phi = (A \vee B) \wedge \neg a \wedge \neg b$ and $\pi = \mathcal{T} = \emptyset$. Three valuations satisfy Φ; in particular the valuation ν which sets to $\texttt{tt}$ precisely the variables A and B. Since both A and B can disappear in the future, and both a and b can become populated, the "permanent part" of ν, i.e., the valuation π_ν, has the empty domain. The transformation graph G_ν is shown in Fig. 2 (left).

Consider now the majority protocol of Example 3 with initial stage $(\Phi, \emptyset, \emptyset)$ (where Φ says there are only A's and B's), and a valuation ν which sets to $\texttt{tt}$ precisely the variables A and B. The domain of π_ν is again the empty set, and the transformation graph G_ν is shown in Fig. 2 (right).

A key observation about transformation graphs is that all transitions generating edges connecting two *different* strongly connected components (SCCs) of G_ν become simultaneously disabled in the future almost surely. More precisely, let Exp_ν be the set of all $AB \in Q^{\langle 2 \rangle}$ such that there exists a transition $AB \mapsto CD$ generating an edge of G_ν connecting two different SCCs of G_ν. We have the following:

▶ **Lemma 9.** *Let G_ν be a transformation graph, and let $\mathcal{C}$ be a configuration such that $\mathcal{C} \models \Box\pi_\nu \wedge \Box\mathcal{T}$. Then $\mathcal{C} \models \Diamond Exp_\nu$. Furthermore, $\mathcal{C} \models \Diamond\Box Exp_\nu$.*

However, there is a subtle problem. When the transitions specified by Exp_ν become simultaneously disabled *for the first time*, they may be disabled only *temporarily*, i.e., $\mathcal{C}$ does *not* have to satisfy the formula $\Box(\mathit{Exp}_\nu \Rightarrow \Box\mathit{Exp}_\nu)$. As we shall see in Section 5, it is relatively easy to obtain an upper bound on the expected number of transitions needed to visit a configuration satisfying Exp_ν. However, it is harder to give an upper bound on the expected number of transitions needed to reach a configuration satisfying $\Box\mathit{Exp}_\nu$ (i.e., entering the next stage) unless $\mathcal{C} \models \Box(\mathit{Exp}_\nu \Rightarrow \Box\mathit{Exp}_\nu)$. This difficulty is addressed in the next section.

▶ **Example 10.** We continue with Example 8. For the transformation graph of Fig. 2 (left), we have $\mathit{Exp}_\nu = \{AB\}$. For the transformation graph of Fig. 2 (right), we have $\mathit{Exp}_\nu = \{AB, AC, BC\}$. Hence, according to Lemma 9, every initial configuration of the majority protocol of Example 2 almost surely reaches a configuration satisfying $\neg A \vee \neg B$, and every initial configuration of the majority protocol of Example 3 almost surely reaches a configuration satisfying $(\neg A \vee \neg B) \wedge (\neg A \vee \neg C) \wedge (\neg B \vee \neg C)$. Furthermore, in both cases $\mathcal{C} \models \Box(\mathit{Exp}_\nu \Rightarrow \Box\mathit{Exp}_\nu)$ for every initial configuration $\mathcal{C}$.

4.2.2 Computing $\mathcal{T}_\nu$ and Φ_ν: Case $\mathit{Exp}_\nu \neq \emptyset$

Let $\Gamma_\nu \equiv \nu \wedge \Box\pi_\nu \wedge \Box\mathcal{T}$, and let $\mathcal{C}$ be a configuration satisfying Γ_ν. A natural idea to construct $\mathcal{T}_\nu$ is to enrich $\mathcal{T}$ by Exp_ν. However, Exp_ν can be empty, i.e., the transformation graph G_ν may consist just of disconnected SCCs. For this reason we first consider the case where Exp_ν is nonempty.

Computing $\mathcal{T}_\nu$. As discussed in Section 4.2.1, the fact that $\mathcal{C} \models \Diamond\Box\mathit{Exp}_\nu$ does not necessarily imply $\mathcal{C} \models \Box(\mathit{Exp}_\nu \Rightarrow \Box\mathit{Exp}_\nu)$ complicates the interaction complexity analysis. Therefore, after computing Exp_ν we try to compute a non-empty subset $\mathcal{J}_\nu \subseteq \mathit{Exp}_\nu$ such that $\mathcal{C} \models \Box(\mathcal{J}_\nu \Rightarrow \Box\mathcal{J}_\nu)$ for all configurations $\mathcal{C}$ satisfying Γ_ν. If we succeed, we put $\mathcal{T}_\nu = \mathcal{T} \cup \mathcal{J}_\nu$. Otherwise, $\mathcal{T}_\nu = \mathcal{T} \cup \mathit{Exp}_\nu$. Intuitively, the set $\mathcal{J}_\nu$ is the largest subset M of Exp_ν such that every element of M can be re-enabled only by firing a transition which has been identified as permanently disabled. This again leads to a simple fixed-point computation, which is detailed in [9].

A proof of the next lemma is straightforward.

▶ **Lemma 11.** *For every configuration $\mathcal{C}$ such that $\mathcal{C} \models \Gamma_\nu$ we have that*
(a) $\mathcal{C} \models \Diamond\Box(\pi_\nu \wedge \mathcal{T} \wedge \mathit{Exp}_\nu)$
(b) $\mathcal{C} \models \Box(\mathcal{J}_\nu \Rightarrow \Box\mathcal{J}_\nu)$
If $\mathcal{J}_\nu \neq \emptyset$, we put $\mathcal{T}_\nu = \mathcal{T} \cup \mathcal{J}_\nu$. Otherwise, we put $\mathcal{T}_\nu = \mathcal{T} \cup \mathit{Exp}_\nu$.

Computing Φ_ν. We say that a configuration $\mathcal{C}$ is S_ν*-entering* if $\mathcal{C} \models \Box\pi_\nu \wedge \Box\mathcal{T}_\nu$ and there is an execution $\mathcal{C}_0, \ldots, \mathcal{C}_\ell$ such that $\mathcal{C}_0 \models \Gamma_\nu$, $\mathcal{C}_\ell = \mathcal{C}$, and $\mathcal{C}_j \not\models \Box\pi_\nu \wedge \Box\mathcal{T}_\nu$ for all $j < \ell$. An immediate consequence of Lemma 11 is the following:

▶ **Lemma 12.** *Almost all runs initiated in a configuration satisfying Γ_ν visit an S_ν-entering configuration.*

The formula Φ_ν specifies the properties of S_ν-entering configurations. The formula Φ_ν always implies $\pi_\nu \wedge \mathcal{T}_\nu$, but it can also be more detailed if $\mathcal{J}_\nu \neq \emptyset$. More precisely, we say that $\mathcal{J}_\nu$ is ν*-disabled* if $\mathcal{J}_\nu \neq \emptyset$ and for all $AB \in \mathcal{J}_\nu$ we have that $\nu \Rightarrow \xi_{AB}$ is a propositional tautology (i.e., all transitions specified by $\mathcal{J}_\nu$ are disabled in all configurations satisfying ν). Similarly, $\mathcal{J}_\nu$ is ν*-enabled* if $\mathcal{J}_\nu \neq \emptyset$ and there exists $AB \in \mathcal{J}_\nu$ such that $\nu \Rightarrow \neg\xi_{AB}$ is a tautology (i.e., some transition specified by $\mathcal{J}_\nu$ is enabled in all configurations satisfying ν).

Observe that if $\mathcal{J}_\nu$ is ν-disabled, then all transitions specified by $\mathcal{J}_\nu$ are simultaneously disabled in every configuration $\mathcal{C}$ satisfying Γ_ν. Hence, all S_ν-entering configurations satisfy Γ_ν (see Lemma 11 (b)). Now suppose that $\mathcal{J}_\nu$ is ν-enabled, and let Q_ν be the set of all $A \in Q$ such that $AB \in \mathcal{J}_\nu$ for some $B \in Q$. Since for every configuration $\mathcal{C}$ satisfying Γ_ν there is a transition specified by $\mathcal{J}_\nu$ enabled in $\mathcal{C}$, the last transition executed before visiting an S_ν-entering configuration must be a transition "transforming" some $A \in Q_\nu$, i.e., a transition of the form $AB \mapsto CD$ generating an edge $A \to C$ of G_ν. Let $\mathcal{K}_\nu$ be the set of all right-hand sides of all such transitions. The formula Φ_ν is defined as follows:

$$\Phi_\nu \quad \equiv \quad \begin{cases} \pi_\nu \wedge \mathcal{T}_\nu \ \wedge \ \nu & \text{if } \mathcal{J}_\nu \text{ is } \nu\text{-disabled,} \\ \pi_\nu \wedge \mathcal{T}_\nu \ \wedge \ \left(\bigvee_{CD \in \mathcal{K}_\nu} \neg \xi_{CD} \right) & \text{if } \mathcal{J}_\nu \text{ is } \nu\text{-enabled,} \\ \pi_\nu \wedge \mathcal{T}_\nu & \text{otherwise.} \end{cases}$$

It is easy to check that every S_ν-entering configuration satisfies the formula Φ_ν. The constructed $\mathcal{P}$-stage $S_\nu = (\Phi_\nu, \pi_\nu, \mathcal{T}_\nu)$ becomes a successor of the $\mathcal{P}$-stage S.

4.2.3 Computing $\mathcal{T}_\nu$ and Φ_ν: Case $Exp_\nu = \emptyset$

In this case G_ν is a collection of disconnected SCCs. We put $\mathcal{T}_\nu = \mathcal{T}$. In the rest of the section we show how to construct the formula Φ_ν.

We say that an edge $A \to B$ of G_ν is *stable* if there is a transition $AC \mapsto BD$ generating $A \to B$ such that $\pi_\nu(C) = \texttt{tt}$. Let $\mathcal{I}_\nu$ be the union of all non-bottom SCCs of the directed graph obtained from G_ν by considering only the stable edges of G_ν.

▶ **Lemma 13.** *For every configuration $\mathcal{C}$ such that $\mathcal{C} \models \Gamma_\nu$ we have that $\mathcal{C} \models \Diamond\big(\bigwedge_{A \in \mathcal{I}_\nu} \neg A\big)$.*

Similarly as above, we say that $\mathcal{C}$ is *S_ν-entering* if $\mathcal{C} \ \models \ \Box\pi_\nu \ \wedge \ \Box\mathcal{T}_\nu \ \wedge \ \bigwedge_{A \in \mathcal{I}_\nu} \neg A$ and there is an execution $\mathcal{C}_0, \ldots, \mathcal{C}_\ell$ such that $\mathcal{C}_0 \models \Gamma_\nu$, $\mathcal{C}_\ell = \mathcal{C}$, and $\mathcal{C}_j$ does *not* satisfy the above formula for all $j < \ell$.

Observe that if $\nu(A) = \texttt{ff}$ for all $A \in \mathcal{I}_\nu$, then ν implies $\bigwedge_{A \in \mathcal{I}_\nu} \neg A$ and hence every configuration $\mathcal{C}$ satisfying Γ_ν is S_ν-entering. Further, if $\nu(A) = \texttt{tt}$ for some $A \in \mathcal{I}_\nu$, then the last transition executed before visiting an S_ν-entering configuration is a transition $EF \mapsto CD$ generating a stable edge $E \to C$ of G_ν where $E \in \mathcal{I}_\nu$ and $C \notin \mathcal{I}_\nu$. Let $\mathcal{L}_\nu$ be the set of all right-hand sides of all such transitions. We put

$$\Phi_\nu \equiv \begin{cases} \pi_\nu \wedge \mathcal{T}_\nu \wedge \left(\bigwedge_{A \in \mathcal{I}_\nu} \neg A \right) \wedge \left(\bigvee_{CD \in \mathcal{L}_\nu} \neg \xi_{CD} \right) & \text{if } \nu(A) = \texttt{tt} \text{ for some } A \in \mathcal{I}_\nu, \\ \pi_\nu \wedge \mathcal{T}_\nu \wedge \ \nu & \text{if } \nu(A) = \texttt{ff} \text{ for all } A \in \mathcal{I}_\nu, \\ \pi_\nu \ \wedge \ \mathcal{T}_\nu \ \wedge \left(\bigwedge_{A \in \mathcal{I}_\nu} \neg A \right) & \text{otherwise.} \end{cases}$$

We say that the constructed $\mathcal{P}$-stage $S_\nu = (\Phi_\nu, \pi_\nu, \mathcal{T}_\nu)$ is *redundant* if there is a $\mathcal{P}$-stage $S' = (\Phi', \pi', \mathcal{T}')$ on the path from the initial stage S_0 to S such that $\pi_\nu = \pi'$, $\mathcal{T}_\nu = \mathcal{T}'$, and Φ' implies Φ_ν. The $\mathcal{P}$-stage S_ν becomes a successor of S iff S_ν is not redundant. This ensures termination of the algorithm even for poorly designed population protocols.

5 Computing the interaction complexity

We show how to compute an upper asymptotic bounds on $\mathit{ReachNext}_S$ for every stage S in the stage graph constructed in Section 4.

For the rest of this section, we fix a population protocol $\mathcal{P} = (Q, T, \Sigma, I, O)$, a $\mathcal{P}$-stage $S = (\Phi, \pi, \mathcal{T})$, and its successor $S_\nu = (\Phi_\nu, \pi_\nu, \mathcal{T}_\nu)$. Recall the formula Γ_ν, the graph $G_\nu = (V, \rightarrow)$, and the sets Exp_ν, $\mathcal{J}_\nu$ defined in Section 4. We show how to compute an asymptotic upper bound on the function Reach_{S,S_ν} that assigns to every $n \geq 1$ the maximal $\mathbb{E}_\mathcal{C}[\mathit{Steps}_{\mathit{Enter}(S_\nu)}]$, where $\mathit{Enter}(S_\nu)$ is a fresh atomic proposition satisfied precisely by all S_ν-entering configurations, and $\mathcal{C}$ ranges over all configurations of size n satisfying Γ_ν (if there is no such configuration of size n, we put $\mathit{Reach}_{S,S_\nu}(n) = 0$). Observe that $\max_{S_\nu}\{\mathit{Reach}_{S,S_\nu}\}$, where S_ν ranges over all successor stages of S, is then an asymptotic upper bound on $\mathit{ReachNext}_S$.

Let us note that if $\mathcal{P}$ terminates, then $\mathit{InterComplexity}_\mathcal{P} \in 2^{2^{\mathcal{O}(n)}}$. This trivial bound follows by observing that the number of all configurations of size n is $2^{\mathcal{O}(n)}$, and the probability of reaching a stable configuration in $2^{\mathcal{O}(n)}$ transitions is $2^{-2^{o(n)}}$; this immediately implies the mentioned upper bound on $\mathit{InterComplexity}_\mathcal{P}$. As we shall see, the worst asymptotic bound on Reach_{S,S_ν} is $2^{\mathcal{O}(n)}$, and in many cases, our results allow to derive even a polynomial upper bound on Reach_{S,S_ν}.

Recall that if $(\pi_\nu, \mathcal{T})$ is stable or dead, we have that $\mathit{Reach}_{S,S_\nu}(n) = 0$ for all $n \in \mathbb{N}$ (in this case, we define S_ν-entering configurations are the configurations satisfying $\Box(\pi_\nu \wedge \mathcal{T})$). Now suppose $(\pi_\nu, \mathcal{T})$ is not stable or dead. Furthermore, let us first assume $\mathit{Exp}_\nu = \emptyset$. Then, the upper bound on Reach_{S,S_ν} is singly exponential in n.

▶ **Theorem 14.** *If $\mathit{Exp}_\nu = \emptyset$, then $\mathit{Reach}_{S,S_\nu} \in 2^{\mathcal{O}(n)}$.*

Now assume $\mathit{Exp}_\nu \neq \emptyset$. Let $\mathcal{U} \subseteq Q$ be the set of all states appearing in some non-bottom SCC of G_ν. We start with some auxiliary definitions.

▶ **Definition 15.** For every $A \in \mathcal{U}$, let $\mathit{Exp}_\nu[A]$ be the set of all $B \in Q$ such that $AB \in \mathit{Exp}_\nu$. We say that S_ν is *fast* if, for every $A \in \mathcal{U}$, the formula $\big(\pi_\nu \wedge \mathcal{T} \wedge \neg\mathit{Exp}_\nu \wedge A\big) \Rightarrow \big(\bigvee_{B \in \mathit{Exp}_\nu[A]} \neg\xi_{AB}\big)$ is a propositional tautology.

▶ **Definition 16.** For every $A \in V$, let $[A]$ be the SCC of G_ν containing A. We say that S_ν is *very fast* if every transition $AB \mapsto CD$ such that $AB, CD \in V^{\langle 2 \rangle}$ and $\{A, B, C, D\} \cap \mathcal{U} \neq \emptyset$ satisfies one of the following conditions:

- The formula $\big(\pi_\nu \wedge \mathcal{T}\big) \Rightarrow \xi_{AB}$ is a propositional tautology.
- $[C] \neq [A] \neq [D]$ and $[C] \neq [B] \neq [D]$.

▶ **Theorem 17.** *If $\mathit{Exp}_\nu \neq \emptyset$ and $\mathcal{J}_\nu \neq \emptyset$, then*

- *$\mathit{Reach}_{S,S_\nu} \in \mathcal{O}(n^3)$.*
- *If S_ν is fast, then $\mathit{Reach}_{S,S_\nu} \in \mathcal{O}(n^2 \cdot \log(n))$.*
- *If S_ν is very fast, then $\mathit{Reach}_{S,S_\nu} \in \mathcal{O}(n^2)$.*

Computing an asymptotic upper bound on Reach_{S,S_ν} when $\mathit{Exp}_\nu \neq \emptyset$ and $\mathcal{J}_\nu = \emptyset$ is more complicated. We show that a *polynomial* upper bound always exists, and that the degree of the polynomial is computable. However, our proof does not yield an efficient algorithm for computing/estimating the degree.

▶ **Theorem 18.** *If $\mathit{Exp}_\nu \neq \emptyset$ and $\mathcal{J}_\nu = \emptyset$, then $\mathit{Reach}_{S,S_\nu} \in \mathcal{O}(n^c)$ for some computable constant c.*

Table 1 Results of the experimental evaluation where $|Q|$, $|T|$ and $|\mathbb{S}|$ correspond respectively to the number of states and transitions of the protocol, and the number of nodes of its stage graph.

Protocol			$\lvert\mathbb{S}\rvert$	**Bound**	**Time**
predicate / params.	$\lvert Q\rvert$	$\lvert T\rvert$			
$x_1 \vee \ldots \vee x_n$ [12]	2	1	5	$n^2 \cdot \log n$	0.1
$x \geq y$ (Example 3)	5	6	13	$n^2 \cdot \log n$	0.4
$x \geq y$ [7]	4	3	9	$n^2 \cdot \log n$	0.2
$x \geq y$ (Example 2)	4	4	11	$\exp(n)$	0.3
Flocks-of-bird protocol [4]: $x \geq c$					
$c = 5$	6	21	26	n^3	0.8
$c = 10$	11	66	46	n^3	4.0
$c = 15$	16	136	66	n^3	12.1
$c = 20$	21	231	86	n^3	28.9
$c = 25$	26	351	106	n^3	58.0
$c = 30$	31	496	126	n^3	118.9
$c = 35$	36	666	146	n^3	222.3
$c = 40$	41	861	166	n^3	366.2
$c = 45$	46	1081	186	n^3	495.3
$c = 50$	51	1326	206	n^3	952.8
$c = 55$	56	1596	—	—	T/O
Logarithmic flock-of-birds protocol[8]: $x \geq c$					
$c = 15$	8	23	66	n^3	2.6
$c = 31$	10	34	130	n^3	6.1
$c = 63$	12	47	258	n^3	13.9
$c = 127$	14	62	514	n^3	39.4
$c = 255$	16	79	1026	n^3	81.0
$c = 1023$	20	119	4098	n^3	395.7
$c = 2047$	22	142	8194	n^3	851.9
$c = 4095$	24	167	—	—	T/O

Protocol			$\lvert\mathbb{S}\rvert$	**Bound**	**Time**
predicate / params.	$\lvert Q\rvert$	$\lvert T\rvert$			
Flocks-of-bird protocol [12]: $x \geq c$					
$c = 5$	6	9	54	n^3	2.5
$c = 7$	8	13	198	n^3	11.3
$c = 10$	11	19	1542	n^3	83.9
$c = 13$	14	25	12294	n^3	816.4
$c = 15$	16	29	—	—	T/O
Average-and-conquer protocol[9] [2]: $x \geq y$ with params. m and d					
$m = 3, d = 1$	6	21	41	$n^2 \cdot \log n$	2.0
$m = 3, d = 2$	8	36	1948	$n^2 \cdot \log n$	98.7
$m = 5, d = 1$	8	36	1870	n^3	80.1
$m = 5, d = 2$	10	55	—	—	T/O
$m = 7, d = 1$	10	55	—	—	T/O
Remainder protocol [8]: $\sum_{1 \leq i < m} i \cdot x_i \equiv 0 \pmod{m}$					
$m = 3$	5	12	27	$n^2 \cdot \log n$	0.8
$m = 5$	7	25	225	$n^2 \cdot \log n$	12.5
$m = 7$	9	42	1351	$n^2 \cdot \log n$	88.9
$m = 9$	11	63	7035	$n^2 \cdot \log n$	544.0
$m = 10$	12	75	—	—	T/O
Threshold protocol [4]: $\sum_{1 \leq i \leq k} a_i \cdot x_i < c$					
$-x_1 + x_2 < 0$	12	57	21	n^3	3.0
$-x_1 + x_2 < 1$	20	155	131	n^3	30.3
$-x_1 + x_2 < 2$	28	301	—	—	T/O
$-2x_1 - x_2 + x_3 + 2x_4 < 0$	20	155	1049	n^3	166.3
$-2x_1 - x_2 + x_3 + 2x_4 < 1$	20	155	1049	n^3	155.2
$-2x_1 - x_2 + x_3 + 2x_4 < 2$	28	301	—	—	T/O

6 Experimental results

We have implemented our approach as a tool[6] that takes a population protocol as input and follows the procedure of Section 4 to construct a stage graph together with an upper bound on $\mathit{InterComplexity}_{\mathcal{P}}$. Our tool is implemented in Python 3, and uses the SMT solver Microsoft Z3 [13] to test for tautologies and to obtain valid valuations.

We tested our implementation on multiple protocols drawn from the literature: a simple broadcast protocol [12], the majority protocols of Example 2, Example 3 and [2], various flock-of-birds protocols [4, 12, 7], a remainder protocol [8] and a threshold protocol [4]. Most of these protocols are parametric, i.e. they are a family of protocols depending on some parameters. For these protocols, we increased their parameters until reaching a timeout. In particular, for the logarithmic flock-of-birds protocol computing $x \geq c$, we used thresholds of the form $c = 2^i - 1$ as they essentially consist the most complicated case of the protocol.

All tests were performed on the same computer equipped with eight Intel® Core™ i5-8250U 1.60 GHz CPUs, 8 GB of memory and Ubuntu Linux 17.10 (64 bits). Each test had a timeout of 1000 seconds (∼16.67 minutes). The duration of each test was evaluated as the sum of the `user` time and `sys` time reported by the Python time library.

The results of the benchmarks are depicted in Table 1, where the *bound* column refers to the derived upper bound on $\mathit{InterComplexity}_{\mathcal{P}}$. In particular, the tool derived exponential and $n^2 \cdot \log n$ bounds for the protocols of Example 2 and Example 3 respectively. The generated trees across all instances grow in width but not much in height: the maximum height between the roots and the leaves varies between 2 and 5, and most nodes are leaves.

[6] The tool and its benchmarks are available at `https://github.com/blondimi/pp-time-analysis`.
[7] Protocol of Example 2 without the tie-breaking rule $ba \mapsto bb$ (only correct if $x \neq y$).
[8] An adapted version of the protocol of [7, Sect. 3] without so-called k-way transitions.
[9] The protocol is only correct assuming $x \neq y$.

It is worth noting that the $n^2 \log n$ bounds obtained in Table 1 for the *average-and-conquer* and *remainder* protocols are tight with respect to the best known bounds [2, 4]. However, some of the obtained bounds are not tight, e.g. we report n^3 for the *threshold protocol* but an $n^2 \log n$ upper bound was shown in [4]. Moreover, it seems possible to decrease the n^3 bound to n^2 for the *flocks-of-bird protocol* of [4]. We are unsure of the precise bounds for the remaining protocols.

7 Conclusion

We have presented the first algorithm for quantitative verification of population protocols able to provide asymptotic bounds valid for any number of agents. The algorithm is able to compute good bounds for many of the protocols described in the literature.

The algorithm is based on the notion of stage graph, a concept that can be of independent value. In particular, we think that stage graphs can be valuable for fault localization and perhaps even automatic repair of ill designed protocols.

An interesting question is whether our algorithm is “weakly complete”, meaning that for every predicate there exists a protocol for which our algorithm can compute the exact time bound. We know that this is the case for protocols with leaders, and conjecture that the result extends to all protocols, but currently we do not have a proof.

Another venue for future research is the automatic computation of lower bounds. Here, while stage graphs will certainly be useful, they do not seem to be enough, and will have to be complemented with other techniques.

References

1 Dan Alistarh, James Aspnes, David Eisenstat, Rati Gelashvili, and Ronald L. Rivest. Time-space trade-offs in population protocols. In *Proc. 28th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 2560–2579, 2017. `doi:10.1137/1.9781611974782.169`.

2 Dan Alistarh, Rati Gelashvili, and Milan Vojnović. Fast and exact majority in population protocols. In *Proc. ACM Symposium on Principles of Distributed Computing (PODC)*, pages 47–56, 2015. `doi:10.1145/2767386.2767429`.

3 Dana Angluin, James Aspnes, Zoë Diamadi, Michael J. Fischer, and René Peralta. Computation in networks of passively mobile finite-state sensors. In *Proc. 23rd Annual ACM Symposium on Principles of Distributed Computing (PODC)*, pages 290–299, 2004. `doi:10.1145/1011767.1011810`.

4 Dana Angluin, James Aspnes, Zoë Diamadi, Michael J. Fischer, and René Peralta. Computation in networks of passively mobile finite-state sensors. *Distributed Computing*, 18(4):235–253, jan 2006. `doi:10.1007/s00446-005-0138-3`.

5 Kevin Batz, Benjamin Lucien Kaminski, Joost-Pieter Katoen, and Christoph Matheja. How long, O bayesian network, will I sample thee? - A program analysis perspective on expected sampling times. In *Proc. 27th European Symposium on Programming (ESOP)*, pages 186–213. Springer, 2018. `doi:10.1007/978-3-319-89884-1_7`.

6 Amanda Belleville, David Doty, and David Soloveichik. Hardness of computing and approximating predicates and functions with leaderless population protocols. In *Proc. 44th International Colloquium on Automata, Languages, and Programming (ICALP)*, pages 141:1–141:14, 2017. `doi:10.4230/LIPIcs.ICALP.2017.141`.

7 Michael Blondin, Javier Esparza, and Stefan Jaax. Large flocks of small birds: on the minimal size of population protocols. In *Proc. 35th Symposium on Theoretical Aspects of*

Computer Science (STACS), pages 16:1–16:14, 2018. doi:10.4230/LIPIcs.STACS.2018.16.

8 Michael Blondin, Javier Esparza, Stefan Jaax, and Philipp J. Meyer. Towards efficient verification of population protocols. In *Proc. 36th ACM Symposium on Principles of Distributed Computing (PODC)*, pages 423–430, 2017. doi:10.1145/3087801.3087816.

9 Michael Blondin, Javier Esparza, and Antonín Kučera. Automatic analysis of expected termination time for population protocols. *ArXiv e-prints*, 2018. arXiv:1807.00331.

10 Krishnendu Chatterjee, Hongfei Fu, and Aniket Murhekar. Automated recurrence analysis for almost-linear expected-runtime bounds. In *Proc. 29th International Conference on Computer Aided Verification (CAV)*, pages 118–139, 2017. doi:10.1007/978-3-319-63387-9_6.

11 Ho-Lin Chen, Rachel Cummings, David Doty, and David Soloveichik. Speed faults in computation by chemical reaction networks. *Distributed Computing*, 30(5):373–390, 2017. doi:10.1007/s00446-015-0255-6.

12 Julien Clément, Carole Delporte-Gallet, Hugues Fauconnier, and Mihaela Sighireanu. Guidelines for the verification of population protocols. In *Proc. 31st International Conference on Distributed Computing Systems (ICDCS)*, pages 215–224, 2011. doi:10.1109/ICDCS.2011.36.

13 Leonardo Mendonça de Moura and Nikolaj Bjørner. Z3: an efficient SMT solver. In *Proc. 14th International Conference Tools and Algorithms for the Construction and Analysis of Systems (TACAS)*, pages 337–340, 2008. doi:10.1007/978-3-540-78800-3_24.

14 David Doty and David Soloveichik. Stable leader election in population protocols requires linear time. In *Proc. 29th International Symposium on Distributed Computing (DISC)*, pages 602–616, 2015. doi:10.1007/978-3-662-48653-5_40.

15 Moez Draief and Milan Vojnović. Convergence speed of binary interval consensus. In *Proc. 29th IEEE International Conference on Computer Communications (INFOCOM)*, pages 1792–1800, 2010. doi:10.1109/INFCOM.2010.5461999.

16 Javier Esparza, Pierre Ganty, Jérôme Leroux, and Rupak Majumdar. Verification of population protocols. *Acta Informatica*, 54(2):191–215, 2017. doi:10.1007/s00236-016-0272-3.

17 Philippe Flajolet, Bruno Salvy, and Paul Zimmermann. Automatic average-case analysis of algorithm. *Theoretical Computer Science*, 79(1):37–109, 1991. doi:10.1016/0304-3975(91)90145-R.

18 Stefan Jaax. Personal communication, April 2018.

19 Benjamin Lucien Kaminski, Joost-Pieter Katoen, Christoph Matheja, and Federico Olmedo. Weakest precondition reasoning for expected run-times of probabilistic programs. In *Proc. 25th European Symposium on Programming (ESOP)*, pages 364–389. Springer, 2016. doi:10.1007/978-3-662-49498-1_15.

20 Othon Michail and Paul G. Spirakis. Elements of the theory of dynamic networks. *Communications of the ACM*, 61(2):72, 2018. doi:10.1145/3156693.

21 Saket Navlakha and Ziv Bar-Joseph. Distributed information processing in biological and computational systems. *Communications of the ACM*, 58(1):94–102, 2015. doi:10.1145/2678280.

22 Van Chan Ngo, Quentin Carbonneaux, and Jan Hoffmann. Bounded expectations: resource analysis for probabilistic programs. In *Proc. 39th ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI)*, pages 496–512, 2018. doi:10.1145/3192366.3192394.

23 Etienne Perron, Dinkar Vasudevan, and Milan Vojnović. Using three states for binary consensus on complete graphs. In *Proc. 28th IEEE International Conference on Computer Communications (INFOCOM)*, pages 2527–2535, 2009. doi:10.1109/INFCOM.2009.5062181.