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 \( O(f(n)) \), 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


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 $O(n^2 \log n)$, 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 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 \( G \) (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 \((C, D) \rightarrow (E, F)\). We assume that for each pair of states \((C, D)\), there is at least one rule \((C, D) \rightarrow (E, F)\). 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, \( G \) is considered as a complete graph, and this assumption is adopted also in this work. Since the agent identifiers are hidden and \( G \) is complete, a population is fully determined by the number of agents in each state. Formally, a configuration is a vector \( C \in \mathbb{N}^Q \), where \( C(A) \) is the number of agents in state \( A \). For every \( A \in Q \), we use \( 1_A \) to
denote the vector satisfying \(1_A(A) = 1\) and \(1_A(B) = 0\) for all \(B \neq A\). Note that there is no difference between transitions \((A, B) \rightarrow (C, D)\) and \((A, B) \rightarrow (D, C)\), 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 \(AB \rightarrow CD\), because the ordering of states before/after the \(\rightarrow\) symbol is irrelevant. Formally, \(AB\) and \(CD\) are understood as elements of \(Q^{(2)}\), 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;
- \(T : Q^{(2)} \times Q^{(2)}\) is a total transition relation;
- \(\Sigma\) is a non-empty finite input alphabet,
- \(I : \Sigma \rightarrow Q\) is the input function mapping input symbols to states,
- \(O : Q \rightarrow \{0, 1\}\) is the output function.

We write \(AB \rightarrow CD\) to indicate that \((AB, CD) \in T\). When defining the set \(T\), we usually specify the outgoing transitions only for some subset of \(Q^{(2)}\). For the other pairs \(AB\), there (implicitly) exists a single idle transition \(AB \rightarrow AB\). 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 \(AB \rightarrow CD\) is enabled in a configuration \(C\) if \(C - 1_A - 1_B \geq 0\). A transition \(AB \rightarrow CD\) enabled in \(C\) can fire and thus produce a configuration \(C' = C - 1_A - 1_B + 1_C + 1_D\). The probability of executing a transition \(AB \rightarrow CD\) enabled in \(C\) is defined by

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

where \(n\) is the size of \(C\). Note that \(2 \cdot C(A) \cdot C(B)\) 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 \(C \rightarrow C'\) to indicate that \(C'\) is obtained from \(C\) by firing some transition, and we use \(P[C \rightarrow C']\) to denote the probability of executing a transition enabled in \(C\) producing \(C'\). Note that there can be several transitions enabled in \(C\) producing \(C'\), and \(P[C \rightarrow C']\) is the total probability of executing some of them.

An execution initiated in a given configuration \(C\) is a finite sequence \(C_0, \ldots, C_\ell\) of configurations such that \(\ell \in \mathbb{N}\), \(C_0 = C\), and \(C_i \rightarrow C_{i+1}\) for all \(i < \ell\). A configuration \(C'\) is reachable from a configuration \(C\) if there is an execution initiated in \(C\) ending in \(C'\). A run is an infinite sequence of configurations \(\omega = C_0, C_1, \ldots\) such that every finite prefix of \(\omega\) is an execution. The configuration \(C_\ell\) of a run \(\omega\) is also denoted by \(\omega_\ell\). For a given execution \(C_0, \ldots, C_\ell\), we use \(\text{Run}(C_0, \ldots, C_\ell)\) to denote the set of all runs starting with \(C_0, \ldots, C_\ell\).

\(^4\) 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 $C$, we define the probability space $(\text{Run}(C), \mathcal{F}, \mathbb{P}_C)$, where $\mathcal{F}$ is the $\sigma$-algebra generated by all $\text{Run}(C_0, \ldots, C_t)$ such that $C_0, \ldots, C_t$ is an execution initiated in $C$, and $\mathbb{P}_C$ is the unique probability measure satisfying $\mathbb{P}_C(\text{Run}(C_0, \ldots, C_t)) = \prod_{i=0}^{t-1} \mathbb{P}[C_i \rightarrow C_{i+1}]$.

### 2.2 A simple modal logic for population protocols

To specify properties of configurations, we use a qualitative variant of the branching-time logic $\text{EF}$. Let $\text{AP} \equiv Q \cup \{ A! | A \in Q \text{ such that there is a non-idle transition } AA \rightarrow BC \}$. The formulae of our qualitative logic are constructed in the following way, where $a$ ranges over $\text{AP} \cup \{ \text{Out}_0, \text{Out}_1 \}$:

$$\varphi ::= a \mid \neg \varphi \mid \varphi_0 \land \varphi_1 \mid \Box \varphi \mid \Diamond \varphi.$$ 

The semantics is defined inductively:

- $C \models A$ iff $C(A) > 0$,
- $C \models A!$ iff $C(A) = 1$,
- $C \models \text{Out}_0$ iff $O(A) = 0$ for all $A \in Q$ such that $C(A) > 0$,
- $C \models \text{Out}_1$ iff $O(A) = 1$ for all $A \in Q$ such that $C(A) > 0$,
- $C \models \neg \varphi$ iff $C \nvdash \varphi$,
- $C \models \varphi_0 \land \varphi_1$ iff $C \models \varphi_0$ and $C \models \varphi_1$,
- $C \models \Box \varphi$ iff $\mathbb{P}_C(\{ \omega \in \text{Run}(C) \mid \omega_i \models \varphi \text{ for all } i \in \mathbb{N} \}) = 1$,
- $C \models \Diamond \varphi$ iff $\mathbb{P}_C(\{ \omega \in \text{Run}(C) \mid \omega_i \models \varphi \text{ for some } i \in \mathbb{N} \}) = 1$.

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

For every formula $\varphi$, we define a random variable $\text{Steps}_\varphi$, assigning to every run $C_0, C_1, \ldots$ either the least $\ell \in \mathbb{N}$ such that $C_\ell \models \varphi$, or $\infty$ if there is no such $\ell$. For a given configuration $C$, we use $E_C[\text{Steps}_\varphi]$ to denote the expected value of $\text{Steps}_\varphi$ in the probability space $(\text{Run}(C), \mathcal{F}, \mathbb{P}_C)$.

### 2.3 Computable predicates, interaction complexity

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

$$C_\mathcal{X}(q) = \sum_{\sigma \in \Sigma, I(\sigma) = q} \mathcal{X}(\sigma) \text{ for every } q \in Q.$$ 

An initial configuration is a configuration of the form $C_\mathcal{X}$ where $\mathcal{X}$ is an input. A configuration $C$ is stable if $C \models \text{Stable}$, where $\text{Stable} \equiv (\Box \text{Out}_0) \lor (\Box \text{Out}_1)$. We say that a protocol $\mathcal{P}$ terminates if $C \models \Diamond \text{Stable}$ for every initial configuration $C$. A protocol $\mathcal{P}$ computes a unary predicate $\Lambda$ on inputs if it terminates and every stable configuration $C'$ reachable from an initial configuration $C_\mathcal{X}$ satisfies $C' \models \text{Out}_x$, where $x$ is either 1 or 0 depending on whether $\mathcal{X}$ satisfies $\Lambda$ or not, respectively.

The interaction complexity of $\mathcal{P}$ is a function $\text{InterComplexity}_\mathcal{P}$, assigning to every $n \geq 1$ the maximal $E_C[\text{Steps}_{\text{Stable}}]$, where $C$ 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
For every initial configuration \( S \), we define a \( \mathcal{G} = (S, \rightarrow, [\cdot]) \) where \( S \) is a finite set of stages, \( \rightarrow \subseteq S \times S \) is an acyclic transition relation, and \([\cdot]\) is a function assigning to each \( S \in S \) a set of configurations \([S]\) such that the following conditions are satisfied:

(a) For every initial configuration \( C \) there is some \( S \in S \) such that \( C \in [S] \).
(b) For every \( S \in S \) with at least one successor under \( \rightarrow \), and for every \( C \in [S] \), we have that \( C \models \Diamond \text{Succ}(S) \), where \( \text{Succ}(S) \equiv \bigvee_{S' \rightarrow S} [S'] \).

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 \( S \rightarrow S' \). Let \( T \) be the set of all terminal stages, and let \( \text{Term} \equiv \bigvee_{S \in T} [S] \). It follows directly from Definition 4(b) that \( C \models \Diamond \text{Term} \) for every initial configuration \( C \). Let \( \text{ReachTerminal}_C \) be a function assigning to every \( n \geq 1 \) the maximal \( \mathbb{E}_C[\text{Steps}_{\text{Term}}] \), where \( C \) ranges over all initial configurations of size \( n \). Furthermore, for every \( S \in S \), we define a

---

5 Recall that sets of configurations can be interpreted as formulae of the modal logic introduced in Section 2.2.
function \( \text{ReachNext}_S \) assigning to every \( n \geq 1 \) the maximal \( \mathbb{E}_C[\text{Steps}_{\text{Succ}(S)}] \), where \( C \) ranges over all configurations of \( [S] \) of size \( n \) (if \( [S] \) does not contain any configuration of size \( n \), we put \( \text{ReachNext}_S(n) = 0 \)).

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

\[ \text{Lemma 5.} \quad \text{Let } \mathbb{P} = (Q,T,\Sigma,I,O) \text{ be a population protocol and } G = (S, \rightarrow, [\cdot]) \text{ a stage graph for } \mathbb{P}. \text{ Let } f : \mathbb{N} \rightarrow \mathbb{N} \text{ be a function such that } \text{ReachNext}_S \in O(f) \text{ for all } S \in S. \text{ Then } \text{ReachTerminal}_G \in O(f). \]

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

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

### 3.1 An example of a stage graph

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

The stage graph \( G \) 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 \( G \), 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 \( C_0 \in [S_0] \), transition \( AB \rightarrow bC \) can be executed in all configurations reachable from \( C_0 \) until \( A \) or \( B \) disappears. Furthermore, the number of \( A \)'s and \( B \)'s can only decrease along every run initiated in \( C_0 \). Hence, \( C_0 \) almost surely reaches a configuration \( C \) where \( A \) or \( B \) (or both of them) disappear. Note that if, e.g., \( C(A) = 0 \) and \( C(B) > 0 \), then this property is “permanent”, i.e., every successor \( C' \) of \( C \) also satisfies \( C'(A) = 0 \) and \( C'(B) > 0 \). 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 \( C_0 \) satisfies \( C_0(A) = C_0(B) \)), then the configuration \( C \) will contain at least one copy of \( C \) which cannot be removed.

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

Let $C_0 \in [S_0]$ be an initial configuration of size $n$, and let $C$ be a configuration reachable from $C_0$ such that $m = \min\{C(A), C(B)\} > 0$. The probability of firing $AB \mapsto bC$ stays larger than $m^2/n^2$ in all configurations reached from $C$ 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 $\min\{C_0(A), C_0(B)\} \leq n/2$, we obtain

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

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

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

$$\text{ReachNext}_{S_1}(n) \leq \sum_{i=1}^{n} \frac{n^2}{i} \leq n^2 \cdot \sum_{i=1}^{n} \frac{1}{i} \leq n^2 \cdot H_n \in O(n^2 \log(n)).$$

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

Similarly, we can show that $\text{ReachNext}_{S_i}(n) \in O(n^2 \log(n))$ 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 $\text{InterComplexity}_P \in O(n^2 \log(n))$. 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 $\text{ReachNext}_S$ 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 $P = (Q, T, \Sigma, I, O)$. A valuation is a partial function $\nu : AP_P \mapsto \{tt, ff\}$ such that $\nu(A!) = tt$ implies $\nu(A) = tt$ whenever $A!, A \in \text{Dom}(\nu)$, where $\text{Dom}(\nu)$ is the domain of $\nu$. Slightly abusing our notation, we also denote by $\nu$ the propositional formula

$$\bigwedge_{p \in \text{Dom}(\nu)} p \land \bigwedge_{p \in \text{Dom}(\nu)} \neg p$$

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

For every transition head $AB \in Q^{(2)}$, let $\xi_{AB}$ be either the formula $\neg A \lor \neg B$ or the formula $\neg A \lor A!$, depending on whether $A \neq B$ or $A = B$, respectively. Hence, the formulae $\xi_{AB}$ and $\neg \xi_{AB}$ say that all transitions of the form $AB \mapsto CD$ are disabled and enabled,
The valuation

To simplify our notation, we write just $\pi$ instead of $\Psi_T$, i.e., $C \vDash T$ iff all transitions specified by $T$ are disabled in $C$.

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

1. $\Phi$ is a propositional formula over $AP_{\mathcal{P}}$,
2. $\pi$ is a valuation, called the persistent valuation,
3. $T \subseteq Q^{(2)}$ is a set of transition heads, called the permanently disabled transition heads.

For every $\mathcal{P}$-stage $S = (\Phi, \pi, T)$, we put $\llbracket S \rrbracket = \{C \mid C \vDash \Phi \wedge \pi \wedge \neg T\}$.

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

$$\Phi = \left( \bigvee_{A \in I(\Sigma)} A \right) \wedge \bigwedge_{A \in Q \setminus I(\Sigma)} \neg A.$$ 

Note that $\llbracket S_0 \rrbracket$ 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 $S = (\Phi, \pi, T)$ 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 $\Phi$. The successor $\mathcal{P}$-stages of $S$ are constructed as follows. First, the algorithm computes the set $Val_S$ consisting of all valuations $\nu$ with domain $AP_{S}$ such that $\nu$ satisfies $\Phi$ when the latter is interpreted over $AP_{S}$. Intuitively, this corresponds to dividing $\llbracket S \rrbracket$ into disjoint “subcases” determined by different $\nu$’s (as we shall see, $\Phi$ always implies the formula $\pi \wedge T$, so $\nu$ cannot be in conflict with the information represented by $\pi$ and $T$; furthermore, we have $Dom(\pi) \subseteq Dom(\nu)$). Then, for each $\nu \in Val_S$, a $\mathcal{P}$-stage $S_\nu$ is constructed, and $S_\nu$ may or may not become a successor of $S$. If none of these $S_\nu$ becomes a successor of $S$, then $S$ is declared as terminal.

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

### 4.1 Computing the valuation $\pi_\nu$

The valuation $\pi_\nu$ is obtained by extending $\pi$ with the “permanent part” of $\nu$. Intuitively, we try to identify $A \in Q$ such that $\nu(A) = \text{tt}$ (or $\nu(A) = \text{ff}$) 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 $\nu(A!) = \text{tt}$ 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 $T_\nu$ and the formula $\Phi_\nu$

In some cases, the constructed persistent valuation $\pi_\nu$ already guarantees that a configuration satisfying $\pi_\nu \wedge T$ is stable or cannot evolve (fire non-idle transitions) any further. Then, we in
fact identified a subset of configurations belonging to \([S]\) which does not require any further analysis. Hence, we put \(T_\nu = T\), \(\Phi_\nu = \pi_\nu\), and the configuration \(S_\nu\) becomes a successor \(P\)-stage of \(S\) declared as terminal.

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

If \(S_\nu\) is not stable or dead, we use \(\pi_\nu\) and \(T\) to compute the transformation graph \(G_\nu\), and then analyze \(G_\nu\) to determine \(T_\nu\) and \(\Phi_\nu\).

### 4.2.1 The transformation graph

The vertices of the transformation graph \(G_\nu\) are the states which have not yet been permanently disabled according to \(\pi_\nu\), and the edges are determined by a set of transitions whose heads have not yet been permanently disabled according to \(\pi_\nu\) and \(T\). Formally, we put \(G_\nu = (V, \rightarrow)\) where the set of vertices \(V\) consists of all \(A \in Q\) such that either \(A \notin \text{Dom}(\pi_\nu)\) or \(\pi_\nu(A) = \texttt{tt}\), and the set of edges is determined as follows: Let \(AB \rightarrow CD\) be a non-idle transition such that \((\pi_\nu \land T) \Rightarrow \xi_{CD}\) 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 \(AB \rightarrow AD\) for \(B \neq D\). In this case it generates the edge \(B \rightarrow D\).

\[\text{Example 8.}\] Consider the protocol of Example 2 and its initial stage \(S = (\Phi, \pi, T)\) where \(\Phi = (A \lor B) \land \neg a \land \neg b\) and \(\pi = T = \emptyset\). Three valuations satisfy \(\Phi\); in particular the valuation \(\nu\) 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 \(\nu\), i.e., the valuation \(\pi_\nu\), has the empty domain. The transformation graph \(G_\nu\) is shown in Fig. 2 (left).

Consider now the majority protocol of Example 3 with initial stage \((\Phi, \emptyset, \emptyset)\) (where \(\Phi\) says there are only \(A\)'s and \(B\)'s), and a valuation \(\nu\) which sets to \(\texttt{tt}\) precisely the variables \(A\) and \(B\). The domain of \(\pi_\nu\) is again the empty set, and the transformation graph \(G_\nu\) 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 \(G_\nu\) become simultaneously disabled in the future almost surely. More precisely, let \(\text{Exp}_\nu\) be the set of all \(AB \in Q^{(2)}\) such that there exists a transition \(AB \rightarrow CD\) generating an edge of \(G_\nu\) connecting two different SCCs of \(G_\nu\). We have the following:

\[\text{Lemma 9.}\] Let \(G_\nu\) be a transformation graph, and let \(\mathcal{C}\) be a configuration such that \(\mathcal{C} \models \Box \pi_\nu \land \Box T\). Then \(\mathcal{C} \models \Diamond \text{Exp}_\nu\). Furthermore, \(\mathcal{C} \models \Diamond \Box \text{Exp}_\nu\).
However, there is a subtle problem. When the transitions specified by $\text{Exp}_\nu$ become simultaneously disabled for the first time, they may be disabled only temporarily, i.e., $C$ does not have to satisfy the formula $\Box(\text{Exp}_\nu \Rightarrow \Box \text{Exp}_\nu)$. 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 $\text{Exp}_\nu$. However, it is harder to give an upper bound on the expected number of transitions needed to reach a configuration satisfying $\Box \text{Exp}_\nu$ (i.e., entering the next stage) unless $C \models \Box(\text{Exp}_\nu \Rightarrow \Box \text{Exp}_\nu)$. 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 $\text{Exp}_\nu = \{AB\}$. For the transformation graph of Fig. 2 (right), we have $\text{Exp}_\nu = \{AB, AC, BC\}$. Hence, according to Lemma 9, every initial configuration of the majority protocol of Example 2 almost surely reaches a configuration satisfying $\neg A \lor \neg B$, and every initial configuration of the majority protocol of Example 3 almost surely reaches a configuration satisfying $(\neg A \lor \neg B) \land (\neg A \lor \neg C) \land (\neg B \lor \neg C)$. Furthermore, in both cases $C \models \Box(\text{Exp}_\nu \Rightarrow \Box \text{Exp}_\nu)$ for every initial configuration $C$.

### 4.2.2 Computing $\mathcal{T}_\nu$ and $\Phi_\nu$: Case $\text{Exp}_\nu \neq \emptyset$

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

**Computing $\mathcal{T}_\nu$.** As discussed in Section 4.2.1, the fact that $C \models \Box \Box \text{Exp}_\nu$ does not necessarily imply $C \models \Box(\text{Exp}_\nu \Rightarrow \Box \text{Exp}_\nu)$ complicates the interaction complexity analysis. Therefore, after computing $\text{Exp}_\nu$ we try to compute a non-empty subset $\mathcal{J}_\nu \subseteq \text{Exp}_\nu$ such that $C \models \Box(\mathcal{J}_\nu \Rightarrow \Box \mathcal{J}_\nu)$ for all configurations $C$ satisfying $\Gamma_\nu$. If we succeed, we put $\mathcal{T}_\nu = \mathcal{T} \cup \mathcal{J}_\nu$. Otherwise, $\mathcal{T}_\nu = \mathcal{T} \cup \text{Exp}_\nu$. Intuitively, the set $\mathcal{J}_\nu$ is the largest subset $M$ of $\text{Exp}_\nu$ 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 $C$ such that $C \models \Gamma_\nu$ we have that

(a) $C \models \Diamond \Box (\pi_\nu \land \mathcal{T} \land \text{Exp}_\nu)$

(b) $C \models \Diamond \Box (\mathcal{J}_\nu \Rightarrow \Box \mathcal{J}_\nu)$

If $\mathcal{J}_\nu \neq \emptyset$, we put $\mathcal{T}_\nu = \mathcal{T} \cup \mathcal{J}_\nu$. Otherwise, we put $\mathcal{T}_\nu = \mathcal{T} \cup \text{Exp}_\nu$.

**Computing $\Phi_\nu$.** We say that a configuration $C$ is $S_\nu$-entering if $C \models \Box \pi_\nu \land \Box \mathcal{T}_\nu$ and there is an execution $c_0, \ldots, c_\ell$ such that $C_0 \models \Gamma_\nu$, $C_\ell = C$, and $C_j \models \Box \pi_\nu \land \Box \mathcal{T}_\nu$ for all $j < \ell$. An immediate consequence of Lemma 11 is the following:

**Lemma 12.** Almost all runs initiated in a configuration satisfying $\Gamma_\nu$ visit an $S_\nu$-entering configuration.

The formula $\Phi_\nu$ specifies the properties of $S_\nu$-entering configurations. The formula $\Phi_\nu$ always implies $\pi_\nu \land \mathcal{T}_\nu$, but it can also be more detailed if $\mathcal{J}_\nu \neq \emptyset$. More precisely, we say that $\mathcal{J}_\nu$ is $\nu$-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 $\nu$). Similarly, $\mathcal{J}_\nu$ is $\nu$-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 $\nu$).
Observe that if $J_\nu$ is $\nu$-disabled, then all transitions specified by $J_\nu$ are simultaneously disabled in every configuration $C$ satisfying $\Gamma_\nu$. Hence, all $S_\nu$-entering configurations satisfy $\Gamma_\nu$ (see Lemma 11 (b)). Now suppose that $J_\nu$ is $\nu$-enabled, and let $Q_\nu$ be the set of all $A \in Q$ such that $AB \in J_\nu$ for some $B \in Q$. Since for every configuration $C$ satisfying $\Gamma_\nu$ there is a transition specified by $J_\nu$ enabled in $C$, the last transition executed before visiting an $S_\nu$-entering configuration must be a transition “transforming” some $A \in Q_\nu$, i.e., a transition of the form $AB \rightarrow CD$ generating an edge $A \rightarrow C$ of $G_\nu$. Let $K_\nu$ be the set of all right-hand sides of all such transitions. The formula $\Phi_\nu$ is defined as follows:

$$
\Phi_\nu \equiv \begin{cases} 
\pi_\nu \land T_\nu \land \nu & \text{if } J_\nu \text{ is } \nu\text{-disabled}, \\
\pi_\nu \land T_\nu \land \left( \bigvee_{CD \in K_\nu} \neg \xi_{CD} \right) & \text{if } J_\nu \text{ is } \nu\text{-enabled}, \\
\pi_\nu \land T_\nu & \text{otherwise}.
\end{cases}
$$

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

### 4.2.3 Computing $T_\nu$ and $\Phi_\nu$: Case $\text{Exp}_\nu = \emptyset$

In this case $G_\nu$ is a collection of disconnected SCCs. We put $T_\nu = T$. In the rest of the section we show how to construct the formula $\Phi_\nu$.

We say that an edge $A \rightarrow B$ of $G_\nu$ is stable if there is a transition $AC \rightarrow BD$ generating $A \rightarrow B$ such that $\pi_\nu(C) = \text{tt}$. Let $L_\nu$ be the union of all non-bottom SCCs of the directed graph obtained from $G_\nu$ by considering only the stable edges of $G_\nu$.

> **Lemma 13.** For every configuration $C$ such that $C \models \Gamma_\nu$ we have that $C \models \Diamond \left( \bigwedge_{A \in L_\nu} \neg A \right)$.

Similarly as above, we say that $C$ is $S_\nu$-entering if $C \models \Box \pi_\nu \land \Box T_\nu \land \bigwedge_{A \in L_\nu} \neg A$ and there is an execution $C_0, \ldots, C_\ell$ such that $C_0 \models \Gamma_\nu$, $C_\ell = C$, and $C_j$ does not satisfy the above formula for all $j < \ell$.

Observe that if $\nu(A) = \text{ff}$ for all $A \in I_\nu$, then $\nu$ implies $\bigwedge_{A \in L_\nu} \neg A$ and hence every configuration $C$ satisfying $\Gamma_\nu$ is $S_\nu$-entering. Further, if $\nu(A) = \text{tt}$ for some $A \in I_\nu$, then the last transition executed before visiting an $S_\nu$-entering configuration is a transition $EF \rightarrow CD$ generating a stable edge $E \rightarrow C$ of $G_\nu$, where $E \in I_\nu$ and $C \notin I_\nu$. Let $L_\nu$ be the set of all right-hand sides of all such transitions. We put

$$
\Phi_\nu \equiv \begin{cases} 
\pi_\nu \land T_\nu \land \left( \bigwedge_{A \in L_\nu} \neg A \right) \land \left( \bigvee_{CD \in E_\nu} \neg \xi_{CD} \right) & \text{if } \nu(A) = \text{tt} \text{ for some } A \in I_\nu, \\
\pi_\nu \land T_\nu \land \nu & \text{if } \nu(A) = \text{ff} \text{ for all } A \in I_\nu, \\
\pi_\nu \land T_\nu \land \left( \bigwedge_{A \in L_\nu} \neg A \right) & \text{otherwise}.
\end{cases}
$$

We say that the constructed $P$-stage $S_\nu = (\Phi_\nu, \pi_\nu, T_\nu)$ is redundant if there is a $P$-stage $S' = (\Phi', \pi', T')$ on the path from the initial stage $S_0$ to $S$ such that $\pi_\nu = \pi'$, $T_\nu = T'$, and $\Phi'$ implies $\Phi_\nu$. The $P$-stage $S_\nu$ becomes a successor of $S$ if $S_\nu$ 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 \( \text{ReachNext}_S \) 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 \( S = (\Phi, \pi, T) \), and its successor \( S_\nu = (\Phi_\nu, \pi_\nu, T_\nu) \). Recall the formula \( \Gamma_\nu \), the graph \( G_\nu = (V, \to) \), and the sets \( \text{Exp}_\nu, J_\nu \) defined in Section 4. We show how to compute an asymptotic upper bound on the function \( \text{Reach}_{S,S_\nu} \) that assigns to every \( n \geq 1 \) the maximal \( \mathbb{E}_C[\text{Steps}_{\text{Enter}}(S_\nu)] \), where \( \text{Enter}(S_\nu) \) is a fresh atomic proposition satisfied precisely by all \( S_\nu \)-entering configurations, and \( C \) ranges over all configurations of size \( n \) satisfying \( \Gamma_\nu \), if there is no such configuration of size \( n \), we put \( \text{Reach}_{S,S_\nu}(n) = 0 \). Observe that \( \max_{S_\nu} \{ \text{Reach}_{S,S_\nu} \} \), where \( S_\nu \) ranges over all successor stages of \( S \), is then an asymptotic upper bound on \( \text{ReachNext}_S \).

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

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

**Theorem 14.** If \( \text{Exp}_\nu = \emptyset \), then \( \text{Reach}_{S,S_\nu} \in 2^\Theta(n) \).

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

**Definition 15.** For every \( A \in \mathcal{U} \), let \( \text{Exp}_\nu[A] \) be the set of all \( B \in Q \) such that \( AB \in \text{Exp}_\nu \). We say that \( S_\nu \) is \textit{fast} if, for every \( A \in \mathcal{U} \), the formula \( (\pi_\nu \land T \land \neg \text{Exp}_\nu \land A) \Rightarrow (\lor_{B \in \text{Exp}_\nu[A]} \neg \xi_{AB}) \) is a propositional tautology.

**Definition 16.** For every \( A \in V \), let \([A]\) be the SCC of \( G_\nu \) containing \( A \). We say that \( S_\nu \) is \textit{very fast} if every transition \( AB \to CD \) such that \( AB, CD \in V^{(2)} \) and \( \{A, B, C, D\} \cap \mathcal{U} \neq \emptyset \) satisfies one of the following conditions:

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

**Theorem 17.** If \( \text{Exp}_\nu \neq \emptyset \) and \( J_\nu \neq \emptyset \), then

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

Computing an asymptotic upper bound on \( \text{Reach}_{S,S_\nu} \) when \( \text{Exp}_\nu \neq \emptyset \) and \( J_\nu = \emptyset \) is more complicated. We show that a \textit{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 \( \text{Exp}_\nu \neq \emptyset \) and \( J_\nu = \emptyset \), then \( \text{Reach}_{S,S_\nu} \in O(n^c) \) for some computable constant \( c \).
Table 1 Results of the experimental evaluation where \(|Q|, |T|\) and \(|S|\) correspond respectively to the number of states and transitions of the protocol, and the number of nodes of its stage graph.

| Protocol | \(|Q|\) | \(|T|\) | \(|S|\) | Bound | Time |
|----------|-------|------|-----|-------|------|
| \(x \geq c\) | \(n^2 \cdot \log n\) | 0.1 | | | |
| \(x \geq y\) (Example 3) | 5 | 6 | 13 | 0.4 | |
| \(x \geq y\) | 4 | 5 | 9 | 0.2 | |
| \(x \geq y\) (Example 2) | 4 | 4 | 11 | 0.3 | |

Table 1: Results of the experimental evaluation

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 \(\text{InterComplexity}_p\). 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 \texttt{user} time and \texttt{sys} time reported by the Python time library.

The results of the benchmarks are depicted in Table 1, where the \texttt{bound} column refers to the derived upper bound on \(\text{InterComplexity}_p\). 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 in much 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 \(ba \rightarrow bb\) (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 \textit{average-and-conquer} and \textit{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 \textit{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 \textit{flocks-of-bird protocol} of [4]. We are unsure of the precise bounds for the remaining protocols.

\section{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.

\section*{References}


7. Michael Blondin, Javier Esparza, and Stefan Jaax. Large flocks of small birds: on the minimal size of population protocols. In \textit{Proc. 35th Symposium on Theoretical Aspects of
Automatic Analysis of Expected Termination Time for Population Protocols


Stefan Jaax. Personal communication, April 2018.


