Analysis of Message Passing Programs using SMT-Solvers

Parosh Aziz Abdulla, Mohamed Faouzi Atig, and Jonathan Cederberg

Uppsala University, Sweden

Abstract. We consider message passing programs where processes communicate asynchronously over unbounded channels. The reachability problem for such systems are either undecidable or have very high complexity. In order to achieve efficiency, we consider the *phase-bounded* reachability problem, where each process is allowed to perform a bounded number of *phases* during a run of the system. In a given phase, the process is allowed to perform *send* or *receive* transitions (but not both). We present a uniform framework where the channels are assigned different types of semantics such as *lossy*, *stuttering*, or *unordered*. We show that the framework allows a uniform translation of bounded-phase reachability for each of the above mentioned semantics to the satisfiability of quantifier-free Presburger formulas. This means that we can use the full power of modern SMT-solvers for efficient analysis of our systems. Furthermore, we show that the translation implies that bounded-phase reachability is NP-COMPLETE. Finally, we prove that the problem becomes undecidable if we allow *perfect* channels or push-down processes communicating through (stuttering) lossy channels. We report on the result of applying the prototype on a number of non-trivial examples.

1 Introduction

Programs modeled as message passing processes have a wide range of applications including communication protocols [13, 5], programs operating on weak memory models [3, 7], WEB service protocols [27], and as semantic models for modern languages such as ERLANG [31] and SCALA [32]. Typically, the processes exchange information asynchronously through a shared unbounded data structure, e.g., counters, multisets, and channels. Despite the increasing popularity of such program models, precise algorithmic analysis is still a major challenge. This is perhaps not without a good reason: it is well known that basic analysis problems (e.g., state reachability) are undecidable for processes communicating via *perfect* FIFO channels [13], even under the assumption that each process is finite-state. Although, checking state reachability becomes decidable for (important) special cases such as *lossy* FIFO channels [1], or *unordered* channels [26], the algorithms have very high complexity (non-primitive recursive for lossy channels [29] and EXPSPACE-HARD for unordered channels [20]).

Given the importance of concurrent software, much research has been devoted in recent years to developing practically useful algorithms. The undecidability and high complexity obstacles are usually addressed by considering different types of over- or under-approximations of system behavior (e.g., [16, 4, 9, 12, 27, 8, 11, 10, 33, 17, 15]).

One useful approach that has recently been proposed is *context-bounding* [25]. The idea is to only consider computations performing at most some fixed number of context switches between processes. This provides a trade-off between computational complexity and verification coverage: on the one hand, context-bounded verification can be more efficient than unbounded verification; and on the other hand, many concurrency errors, such as data races and atomicity violations, are manifested in executions with few context switches [22].

In this paper, we present a new approach to model checking of concurrent processes that communicate through channels. We introduce a new bounding parameter in the behavior of such systems, namely the number of alternations between *send* operations and *receive* operations performed by each processes. We consider the *bounded-phase* reachability problem, where each process is restricted to performing at most k phases (for some natural number k). A *phase* is a run where the process performs either send or receive operations (but not both). Notice that the bounded-phase restriction does not limit the *number* of sends or receives, and in particular it does not put any restriction on the length of the run. Also, the number of context switches is not limited. We will present a framework and instantiate it for several variants of channel semantics, such as *lossy*, *stuttering*, and *multiset* that allow the messages inside the channels to be lost, duplicated, and re-ordered respectively. One main contribution of this paper is to show that our framework allows to translate (in polynomial time) the bounded-phase reachability problem to the satisfiability of quantifier-free Presburger formulas. This opens the way to leveraging the full power of state-of-the-art SMT-solvers for obtaining a very efficient solution to the bounded-phase reachability problem for all above mentioned models. We perform the translation in two steps. First, we show that bounded-phase reachability can be reduced to (general) reachability under a new restriction, namely that we only consider simple computations. A computation is *simple* if any (local) state of a process appears at most once along the computation. In the second step, we show that simple reachability can be captured by satisfiability of a quantifier-free Presburger formula (that we can then feed to an SMT-solver).

In order to simplify the presentation, we first describe our framework for lossy channel systems LCS. Then, we describe how the method can be modified (in a straightforward manner) to the other channel semantics. Also, as consequence of our translation, we show that bounded-phase reachability for LCS (and the other models) is NP-COMPLETE. This is to be contrasted with the fact that the general reachability problem is not primitive recursive.

Finally, we show undecidability of bounded-phase reachability for several cases, e.g., under the *perfect* channel semantics, or under the *(stuttering) lossy* semantics when one the processes is allowed to have a (single) stack.

We have implemented our method in a prototype that we have applied on a number of examples with promising results. The examples span several application areas, such as WEB service protocols, communication protocols, and multithreaded programs counters. The prototype and the details of the examples and experimentation are available online (see Section 11).

Related Work. Our work can be seen as a non-trivial extension of bounded-context switches for concurrent shared-memory programs [25] and reversal-bounded analysis for programs manipulating counters [18, 16] to the class of message-passing programs.

La Torre et al. [33] propose context-bounded analysis for pushdown processes communicating through perfect channels, where in each context a process is allowed to receive from only one channel (but is allowed to send to all other channels). This implies that, in a context, a process can have an unbounded number of alternations between send and receive modes (which is not allowed in our bounded-phase analysis). However, our bounded-phase analysis allows multiple processes to be active at the same time and each one of them can send or receive to/from all the channels (which is not permitted by context-bounded analysis of [33]).

In [9, 4, 12] symbolic representations of the contents of the channels have been proposed for analysis of message-passing programs. Our technique does not restrict the content of channels to a class of representable descriptions. Moreover, our reduction to the satisfiability of quantifier-free Presburger formulas allows us to use highly-developed and optimized SMT-solvers.

In [10], the authors consider a different model where the communication is done via Perfect FIFO channels and where "messages/tasks" can be consumed only when the process stack is empty. Although, their proposed bounding scheme is more general than bounding the number of alternations between receive and send operations, their notion leads to undecidability. To obtain decidability, they restrict the number of processor interleavings. We do not do this since we would like to allow any possible shuffle between two processes.

2 Preliminaries

We let $\mathbb{N}$ denote the set of natural numbers. For a natural number n, we define $[n] := \{1, 2, \ldots, n\}$. For a set A, we use $|A|$ to denote its cardinality. For a function $f : A \mapsto B$ from a set A to a set B we use $f[a \leftarrow b]$ to denote the function f' such that $f(a) = b$ and $f'(a') = f(a')$ if $a' \neq a$. We use $[A \mapsto B]$ to denote the set of (total) functions from A to B. For a set A, we let A^* denote the set of finite words over A. We let $|w|$ denote the length of w. We use $w[i]$ to denote the i^{th} element of w, and write $a \in w$ to denote that $w[i] = a$ for some i. For words $w_1 = a_1 a_2 \cdots a_m$ and $w_2 = b_1 b_2 \cdots b_n$, we write $w_1 \preceq w_2$ to denote that there is an injection $h : [m] \mapsto [n]$ such that $i < j$ implies $h(i) < h(j)$ and $a_i = b_{h(i)}$, i.e., w_1 is a (not necessarily contiguous) subword of w_2. We use $w_1 \cdot w_2$ to denote the concatenation of w_1 and w_2, and ε to denote the empty word. For a word $w = a_1 a_2 \cdots a_m$, we use $\mathit{Stuttering}(w)$ to denote the set of words defined as $\{a_1^{i_1} a_2^{i_2} \cdots a_m^{i_m} \mid 1 \leq i_1, i_2, \ldots, i_m\}$.

3 Communicating Finite-State Processes

In this section, we introduce finite-state processes communicating through lossy channels. We introduce the notion of processes and the transition system induced by communicating processes, and then consider bounded-phase computations. In the rest of the section, we fix a finite set M of *messages* and a finite set C of *channels*.

Processes. A *process* p is a tuple $\left\langle Q_p, q_p^{init}, \Delta_p \right\rangle$ where Q_p is a finite set of *states*, $q_p^{init} \in Q_p$ is the *initial state*, and Δ_p is a finite set of *transitions*. A transition $t \in \Delta_p$ is a triple $\langle q_1, op, q_2 \rangle$ where $q_1, q_2 \in Q_p$ are states, and op is an *operation* of one of the following three forms: (i) $c!m$ sends the message $m \in M$ to channel $c \in C$ (m is appended to the tail of c), (ii) $c?m$ receives the message $m \in M$ from channel $c \in C$ (only enabled if m is at the head of c, and if performed, m is removed from the head of c), (iii) nop is the dummy operation (it does not affect the contents of the channels). We define $source(t) := q_1$, $target(t) := q_2$, and $operation(t) := op$. For a state $q \in Q_p$, we define $source^{-1}(q) := \{t | source(t) = q\}$ and define $target^{-1}(q)$ and $operation^{-1}(op)$ similarly. We define Δ_p^{snd} to be the set of transitions in Δ_p whose operations are *send*. We define Δ_p^{rcv} and Δ^{nop} similarly. A sequence $\delta = t_1 t_2 \cdots t_n$ is said to be a *cycle* if (i) $target(t_i) = source(t_{i+1})$ for $i : 1 \leq i < n$, (ii) $target(t_n) = source(t_1)$, and (iii) $t_i \neq t_j$ if $i \neq j$. We say that δ is a q-loop if $source(t_1) = q$.

Transition System. We define the transition system induced by processes communicating through lossy channels. A *Lossy Channel System* (LCS for short) consists of a set P of processes. Let process $p \in P$ be of the form $\left\langle Q_p, q_p^{init}, \Delta_p \right\rangle$. Define $Q := \cup_{p \in P} Q_p$, $\Delta := \cup_{p \in P} \Delta_p$, $\Delta^{snd} := \cup_{p \in P} \Delta_p^{snd}$, and define Δ^{rcv}, Δ^{nop} similarly. A *state map* is a function $s : P \mapsto Q$ such that $s(p) \in Q_p$, and a *channel map* is a function $\omega : C \mapsto M^*$. We extend the subword ordering $\preceq$ to *channel maps* as follows: Given two *channel maps* $\omega_1 : C \mapsto M^*$ and $\omega_2 : C \mapsto M^*$, we write $\omega_1 \preceq \omega_2$ if and only if $\omega_1(c) \preceq \omega_2(c)$ for all $c \in C$. A *configuration* γ is of the form $\langle s, \omega \rangle$ where s is a state map and ω is a channel map. Intuitively, s defines the states of the processes, while ω defines the contents of the channels. We define a transition relation $\longrightarrow$ on the set of configurations as follows. Consider configurations $\gamma_1 = \langle s_1, \omega_1 \rangle$, $\gamma_2 = \langle s_2, \omega_2 \rangle$, and a transition $t = \langle q_1, op, q_2 \rangle \in \Delta_p$ for some $p \in P$. We write $\gamma_1 \xrightarrow{t} \gamma_2$ to denote that $s_1(p) = q_1$, $s_2 = s_1[p \leftarrow q_2]$, and one of the following three properties is satisfied: (i) $op = c!m$ and $\omega_2 \preceq \omega_1[c \leftarrow m \cdot \omega_1(c)]$, (ii) $op = c?m$ and $\omega_2 \preceq \omega_1[c \leftarrow w]$ where $\omega_1(c) = w \cdot m$, and (iii) $op = nop$ and $\omega_2 \preceq \omega_1$. A computation π (from a configuration γ to a configuration γ') is a sequence $\gamma_0 \xrightarrow{t_1} \gamma_1 \cdots \xrightarrow{t_n} \gamma_n$ such that $\gamma_0 = \gamma$ and $\gamma_n = \gamma'$. In such a case we say that γ' is *reachable* from γ by π. We define the *initial configuration* $\gamma^{init} := \left\langle s^{init}, \omega^{init} \right\rangle$, where $s^{init}(p) = q_p^{init}$ for all $p \in P$, and $\omega^{init}(c) = \varepsilon$ for $c \in C$. In other words, the system starts from a configuration where all the processes are in their initial states and where all the channels are empty. A configuration γ is said to be *reachable* if it is reachable from γ^{init}. A state map $s \in [P \mapsto Q]$ is *reachable*, if there is a channel map ω such that $\langle s, \omega \rangle$ is reachable. In the *reachability problem* for the LCS P, we are given a state map $s^{target} \in [P \mapsto Q]$, and we are asked whether s^{target} is reachable.

Bounded-Phase Computations. We introduce bounded-phase computations. From the point of view of any process p, the computation consists of a number of phases where, during a given phase, process p either only performs *send* operations, or only performs *receive* operations (in addition to the dummy operation). Consider a computation $\pi = \gamma_0 \xrightarrow{t_1} \gamma_1 \xrightarrow{t_2} \cdots \xrightarrow{t_n} \gamma_n$. We define $\pi\uparrow := t_1 t_2 \cdots t_n$, i.e., it is the sequence of transitions that occur in π. For a process $p \in P$, we define $\pi\uparrow p$ to be the maximal subword $t'_1 t'_2 \cdots t'_m$ of $\pi\uparrow$ such that $t'_i \in \Delta_p$ for $i : 1 \leq i \leq m$, i.e., it is the sequence of transitions performed by p in π. Given a sequence of transitions $\delta = t_1 t_2 \cdots t_n \in \Delta_p^*$, we say that δ is a *phase* if either $t_i \in \Delta_p^{snd} \cup \Delta_p^{nop}$ for all $i : 1 \leq i \leq n$, or $t_i \in \Delta_p^{rcv} \cup \Delta_p^{nop}$ for all $i : 1 \leq i \leq n$. We define $\sim snd := rcv$, and $\sim rcv := snd$.

A computation π is said to be *k-bounded* with respect to a process p if $\pi \uparrow p = \delta_1 \cdot \delta_2 \cdots \delta_j$ where $j \leq k$ and δ_i is a phase for all $i : 1 \leq i \leq j$. In other words, the transitions performed by p in π form at most k phases. We say that π is *k-bounded* if it is *k-bounded* with respect to all process $p \in P$. For configurations γ and γ', we say that γ' is k-reachable from γ if γ' is reachable from γ by a k-bounded computation. (State map) k-reachability is defined in a similar manner to state map reachability (see above). In the *bounded-phase reachability problem*, we are also given a natural number $k \in \mathbb{N}$, and we are asked whether s^{target} is k-reachable. The following theorem follows from the definitions. It shows that k-reachability is an under-approximation of reachability.

Theorem 1. *A state map $s : P \mapsto Q$ is reachable iff s is k-reachable for some $k \in \mathbb{N}$.*

4 Simple Reachability

In this section, we introduce *simple reachability*, i.e., reachability by computations in which a state may occur at most once along the computation. We show that k-reachability is polynomially reducible to simple reachability. We do that in two steps. First, we define *pure* LCS and show that the k-reachability problem for general LCS can be reduced to the reachability problem for pure LCS. Second, for a pure LCS, we derive a new LCS and show that the reachability problem for the former coincides with the simple reachability problem for the latter.

Simple Computations. Consider a set of processes P. Let $p = \langle Q_p, q_p^{init}, \Delta_p \rangle$ for $p \in P$. A sequence of transitions $\delta = t_1 t_2 \cdots t_n \in \Delta_p^*$ is said to be simple if there are no $\delta_1, \delta_2, \delta_3$ such that $\delta = \delta_1 \cdot \delta_2 \cdot \delta_3$ and δ_2 is a q-cycle for some state $q \in Q_p$. In other words, the states appearing along the sequence are all different. A computation π is *simple* if $\pi \uparrow p$ is simple for all $p \in P$. A simple computation then does not visit any state more than once. For configurations γ, γ', we say that γ' is *simply reachable* from γ if γ' is reachable from γ by a simple computation. The *simple reachability problem* is defined in a similar manner to the reachability problem (see Section 3) except that we replace *computations* in the definition by *simple computations*.

Pure LCS. Consider a process $p = \langle Q_p, q_p^{init}, \Delta_p \rangle$. We say that p is *pure* if there is no cycle $t_1 t_1 \cdots t_n$ such that $t_i \in \Delta_p^{snd}$ and $t_j \in \Delta_p^{rcv}$ for some $i \neq j$. In other words, p is pure if there is no cycle (equivalently there is no strongly connected component) in the graph of p that contains both a *send* and a *receive* transition. Notice that this is a syntactic property of the process and it does not depend on the operational semantics. An LCS consisting of a set P of processes is *pure* if all processes $p \in P$ are pure. We will now reduce the k-reachability problem for general LCS to the reachability problem for pure LCS. Suppose that we are given an instance of the k-reachability problem, defined by a set P of processes and a target state map s^{target}. We will derive an equivalent instance of the reachability problem where the given LCS is pure. We do this by transforming each process to a pure one. The idea of the transformation is to make a number of copies of (parts of) the graph of p where each copy contains either transitions in $\Delta_p^{snd} \cup \Delta_p^{nop}$ or transitions in $\Delta_p^{rcv} \cup \Delta_p^{nop}$. Each copy will represent one phase of the computation from the point of view of p. If the next transition of p is consistent with the current "mode" of the phase (i.e., *send* or *receive*) then p will continue in states belonging to the current copy; otherwise it moves to the next one. Let $p = \left\langle Q_p, q_p^{init}, \Delta_p \right\rangle$. Define $pure(p) := \left\langle R_p, r_p^{init}, \Delta_p^R \right\rangle$, where:

- $R_p := \left\{ r_p^{init}, r_p^{target} \right\} \cup \{ \langle q, \mathtt{m}, i \rangle | q \in Q_p \wedge \mathtt{m} \in \{snd, rcv\} \wedge 1 \le i \le k \}$. In other words, $pure(p)$ has an initial state r_p^{init}, a target state r_p^{target}, together with a set of states each of which is triple. A triple consists of a state q of p, a mode $\mathtt{m}$, and a natural number $i \le k$. Intuitively, triples containing i are used to simulate the i^{th} phase performed by the process, and the mode $\mathtt{m}$ describes whether the process is sending or receiving during the current phase.
- Δ_p^R contains the following transitions:
  - $\left\langle r_p^{init}, nop, \left\langle q_p^{init}, \mathtt{m}, 1 \right\rangle \right\rangle$ for $\mathtt{m} \in \{snd, rcv\}$. This corresponds to a transition from the initial state of $pure(p)$ to the initial state of p in its first phase. In the first mode, the process may be either sending or receiving.
  - $\langle \langle q_1, \mathtt{m}, i \rangle, op, \langle q_2, \mathtt{m}, i \rangle \rangle$ if $t = \langle q_1, op, q_2 \rangle \in \Delta_p^{\mathtt{m}} \cup \Delta_p^{nop}$ for $\mathtt{m} \in \{snd, rcv\}$ and $1 \le i \le k$. The process performs another transition in the same mode $\mathtt{m}$ and therefore it stays in the same phase i.
  - $\langle \langle q_1, \mathtt{m}, i \rangle, op, \langle q_2, \sim\mathtt{m}, i+1 \rangle \rangle$ if $t = \langle q_1, op, q_2 \rangle \in \Delta_p^{\sim\mathtt{m}}$ for $\mathtt{m} \in \{snd, rcv\}$ and $1 \le i < k$. The process performs a transition that violates the current mode $\mathtt{m}$, and hence it moves to the next phase $i+1$.
  - $\left\langle \langle s^{target}(p), \mathtt{m}, i \rangle, nop, r_p^{target} \right\rangle$ for $\mathtt{m} \in \{snd, rcv\}$ and $i : 1 \le i \le k$. In its final phase, the process moves from the target state of p to the target state of $pure(p)$. The mode of the final phase may be *snd* or *rcv*.

Define $pure(P) := \{pure(p) | p \in P\}$, and define the state map $s_R^{target}(p) := r_p^{target}$ for all $p \in P$. It follows that s_R^{target} is reachable in (the pure LCS) $pure(P)$ iff s^{target} is k-reachable in P, which leads to the following lemma.

Lemma 2. *k-Reachability for* LCS *is polynomially reducible to reachability for pure* LCS.

Saturation. Consider an LCS consisting of a set P of processes. Let $p = \left\langle Q_p, q_p^{init}, \Delta_p \right\rangle$ and $q \in Q_p$. We define q^{snd} to be the set of operations of the form $c!m$ such that there is a q-cycle δ and a transition $t \in \delta$ with $operation(t) = c!m$. In other words, it is the set of all *send* operations that appear in cycles visiting q. For a given $k \in \mathbb{N}$, we derive a new LCS through "k-saturating" P as follows. For a process $p \in P$, we derive $sat(p,k)$ from p by adding a number of states and transitions. For each transition $\langle q, op, q' \rangle \in \Delta_p$, with $q^{snd} \ne \emptyset$, we add $k+1$ new states $q_0^{tmp}, q_1^{tmp}, \ldots, q_k^{tmp}$. Furthermore, for each operation $c!m \in q^{snd}$ and each $i : 0 \le i < k$ we add the transition $\left\langle q_i^{tmp}, c!m, q_{i+1}^{tmp} \right\rangle$. Finally, we add the transitions $\left\langle q, nop, q_0^{tmp} \right\rangle$ and $\left\langle q_k^{tmp}, op, q' \right\rangle$. We define $sat(P,k) := \{sat(p,k) | p \in P\}$, i.e., we k-saturate all the processes in the set P. From the definitions, we notice that $sat(P,k)$ satisfies the following properties. (i) The size $|sat(P,k)|$ of $sat(P,k)$ is polynomial in k and in the size $|P|$ of P (for any appropriate definition of the size $|P|$). This holds since we add at most $k+1$ new states per state of p (more precisely, either $k+1$ states if $q^{snd} \neq \emptyset$, or no states if $q^{snd} = \emptyset$). Also, for each new state the number of added transitions is bounded by $\sum_{p \in P} |\Delta_p|$. (ii) If P is pure then $sat(P,k)$ is pure. This follows from the fact that we only add *send* transitions and we add such transitions only from states that are on cycles not containing *receive* transitions. This implies that we will not create any cycles involving both *send* and *receive* transitions. (iii) If P is pure then, for any configurations γ, γ', we have that γ' is reachable from γ in P iff γ' is reachable from γ in $sat(P,k)$. This follows from the fact that for any added sequence of transitions, the *send* operations are already present in existing cycles. Therefore, the effect of the added cycles can be simulated by iterating the existing cycles (possibly) combined with the loss of messages. (iv) For any state mapping s^{target}, we have that s^{target} is reachable in P iff s^{target} is simply reachable in $sat\left(P, \sum_{p \in P} |Q_p|\right)$. The reason is that effect of performing all the *receive* transitions between the two occurrences of a state q can be simulated by losing messages (by purity of P, none of these transitions can perform a *send* operation). This implies that we need only to consider computations where the number of *receive* transitions is bounded by $\sum_{p \in P} |Q_p|$. In turn, this implies that each cycle involving *send* transitions need not be iterated more than $\sum_{p \in P} |Q_p|$ times. The result follows from the fact that we add the $(\sum_{p \in P} |Q_p|)$-unfolding of all such cycles in the construction of $sat\left(P, \sum_{p \in P} |Q_p|\right)$. This gives the following lemma.

Lemma 3. *Reachability for pure* LCS *is polynomially reducible to simple reachability for (pure)* LCS.

5 Translation

In this section, we reduce the simple reachability problem for LCS to the problem of checking satisfiability of existential Presburger formulas. Suppose that we are given an instance of the simple reachability problem defined by an LCS consisting of a set P of processes, and a state map s^{target}. We will derive a quantifier-free Presburger formula ϕ such that ϕ is satisfiable iff s^{target} is reachable. For each state and transition in P, we introduce a number of variables that we use to build ϕ. We define ϕ as a conjunction where the conjuncts are divided into four sets, called *indexing*, *traversal*, *simplicity*, and *matching* respectively. Each set of conjuncts is used to describe one aspect of a potential computation reaching s^{target}. For $p \in P$, let $p = \left\langle Q_p, q_p^{init}, \Delta_p \right\rangle$. Define Q and Δ as in Section 3.

Indexing. For a state $q \in Q$, we use an "index variable" $\mathtt{index}(q)$. For all pair of states $q, q' \in Q$, ϕ contains $\mathtt{index}(q) \neq \mathtt{index}(q')$, i.e., we assign to each state a unique index.

Traversal. This set of conjuncts ensures that each computation corresponds to a traversal of the graphs of the processes. This is inspired by the construction of an existential Presburger formula for the Parikh image of the language of finite-state automata [30]. To define this group we use the following variables. For each $t \in \Delta$ we use an "occurrence variable" $\mathtt{occ}(t)$ that encodes whether the transition t is executed during the computation or not (1 if *yes* and 0 if *no*). For each state $q \in Q$, we use an "in-flow" variable $\mathtt{in}(q)$ and an "out-flow" variable $\mathtt{out}(q)$ that encode whether state q is entered resp. left during the computation (1 if *yes* and 0 if *no*). The formula ϕ contains the following conjuncts: (i) For each $q \in Q$, ϕ contains $\mathtt{in}(q) = \sum_{t \in target^{-1}(q)} \mathtt{occ}(t)$, i.e., q is entered iff exactly one of its ingoing transitions is executed. (ii) For each $q \in Q$, ϕ contains $\mathtt{out}(q) = \sum_{t \in source^{-1}(q)} \mathtt{occ}(t)$, i.e., q is left iff exactly one of its outgoing transitions is executed. (iii) For each process $p \in P$, ϕ contains $\mathtt{out}\left(q_p^{init}\right) = \mathtt{in}\left(q_p^{init}\right) + 1$, i.e., the initial state of a process is left once but not entered. (iv) For each process $p \in P$, ϕ contains $\mathtt{in}(s^{target}(p)) = \mathtt{out}(s^{target}(p)) + 1$, i.e., the target state in a process is entered once but not left. (v) For each process $p \in P$ and each state $q \in Q_p - \left\{q_p^{init}, s^{target}(p)\right\}$, ϕ contains $\mathtt{in}(q) = \mathtt{out}(q)$, i.e., all other states are either not visited or both entered once and left once. (vi) For each $q \in Q$, ϕ contains $(\mathtt{in}(q) = 1) \implies \left(\bigvee_{t \in target^{-1}(q)} (\mathtt{occ}(t) = 1) \wedge \mathtt{index}(source(t)) < \mathtt{index}(q)\right)$. The indexing on the states guarantees that the computation corresponds to executing successive edges in the graph of each process p. Each visited state is indexed higher than its (unique) predecessor in the computation. Notice that this means that the order in which states occur in the computation is consistent with the indexing (if q appears before q' then $\mathtt{index}(q) < \mathtt{index}(q')$).

Simplicity. For each $q \in Q$, ϕ contains $\mathtt{in}(q) \leq 1$. Since the computation is simple, each state is visited at most once.

Matching. This set of conjuncts ensures that each *receive* transition is matched by a preceding *send* transitions. More precisely, we will match the occurrence of a *receive* transition by the target state of a corresponding *send* transition as follows. For each transition $t \in \Delta^{rcv}$, we use a "matching" variable $\mathtt{match}(t)$. For each transition $t = \langle q_1, c?m, q_2 \rangle \in \Delta^{rcv}$, ϕ contains

$$(\mathtt{occ}(t) = 1) \Longrightarrow \left(\bigvee_{t' \in operation^{-1}(c!m)} \left(\begin{array}{c} \mathtt{match}(t) = \mathtt{index}(target(t')) \\ \wedge \\ \mathtt{occ}(t') = 1 \\ \wedge \\ \mathtt{index}(target(t')) < \mathtt{index}(q_2) \end{array} \right) \right)$$

Intuitively, if the *receive* transition t occurs in the computation (i.e., $\mathtt{occ}(t) = 1$), then a matching transition t' occurs (i.e., $\mathtt{occ}(t') = 1$). The matching of t with t' is achieved by requiring that the "matching" variable of t is equal to the index of the target state of t'. Furthermore, t' should occur before t. The latter condition requires that the index of the target state of t (state q_2) is larger than the index of the target state of t'.

Finally, for any pair of *receive* transitions $t, t' \in \Delta^{rcv}$ the formula ϕ contains $(\mathtt{occ}(t) = 1) \wedge (\mathtt{occ}(t') = 1) \wedge (\mathtt{index}(target(t)) < \mathtt{index}(target(t'))) \Longrightarrow (\mathtt{match}(t) < \mathtt{match}(t'))$. This means that if both t and t' occur (i.e., $\mathtt{occ}(t) = 1$ and $\mathtt{occ}(t') = 1$) and t occurs before t' (the index of the target state of t occurs before the one of t') then the matching *send* transition of t occurs before the matching *send* transition of t'.

The above construction gives the following lemma.

Lemma 4. *Simple reachability for* LCS *is polynomially reducible to the satisfiability of quantifier-free Presburger formulas.*

6 Bounded-Phase Reachability

In this section, we collect the results of the previous sections to prove that k-reachability for LCS is polynomially reducible to the satisfiability of quantifier-free Presburger formulas. The main consequence of this is that it allows the use of advanced tools for SMT-solving for efficient analysis of LCS (see Section 11). Furthermore, we use this result to show an upper bound on the complexity of the k-reachability problem for LCS, namely inclusion in NP. We complete the picture by giving a lower bound which shows that the problem is NP-COMPLETE.

Upper Bound. From Lemma 2, Lemma 3, Lemma 4, we get the following theorem.

Theorem 5. *k-reachability for* LCS *is polynomially reducible to the satisfiability of quantifier-free Presburger formulas.*

Since the latter problem is known to be NP-COMPLETE, we get the following corollary.

Corollary 6. *k-reachability for* LCS *is in* NP.

Lower Bound. We show NP-hardness by a reduction from the Boolean Satisfiability Problem (SAT) (which is known to be NP-COMPLETE [14]). Consider a propositional formula ϕ in conjunctive normal form. We will construct an LCS consisting of a set of processes P. The set P contains (i) one process p_x for each variable x appearing in ϕ, and (ii) one process p_ℓ for each clause ℓ in ϕ. Furthermore, for each variable x, we associate a channel c_x, and two messages $m_x, m_{\bar{x}}$. The finite-state automaton describing the behavior of the process p_x generates traces in the language $(c_x!m_x)^* \cup (c_x!m_{\bar{x}})^*$ from its initial state $q_{p_x}^{init}$ to its unique target state $q_{p_x}^{target}$. Intuitively, the process p_x guesses the assigned value to the variable x by sending a number of copies of the message m_x to the channel c_x if the value assigned to x is *true*, and sending $m_{\bar{x}}$ otherwise. For a clause ℓ, the process p_ℓ contains two states, namely an initial state $q_{p_\ell}^{init}$ and a target state $q_{p_\ell}^{target}$. For any positive (resp. negative) literal x (resp. $\bar{x}$) in ℓ, the process p_ℓ has a transition of the form $\langle q_{p_\ell}^{init}, c_x?m_x, q_{p_\ell}^{target} \rangle$ (resp. $\langle q_{p_\ell}^{init}, c_x?m_{\bar{x}}, q_{p_\ell}^{target} \rangle$). The transition checks if the assigned value to x is *true* (resp. *false*). Define the state map s^{target} such that $s^{target}(p) := q_p^{target}$ for each $p \in P$. It is easy to see that ϕ is satisfiable if and only if s^{target} is 2-reachable. This shows that the k-reachability problem for LCS is NP-HARD for $k \geq 2$. From this and Corollary 6 we get the following theorem.

Theorem 7. *k-reachability for* LCS *is* NP-COMPLETE.

7 Communicating Pushdown Processes

In this section, we define pushdown processes communicating through lossy channels and we show the undecidability of its k-reachability problem. Let M be a finite set of *messages* and C be a finite set of *channels*.

Pushdown Processes. A *pushdown process* p is a tuple $\langle Q_p, q_p^{init}, \Gamma_p, \Delta_p \rangle$ where Q_p is finite set of *states*, $q_p^{init} \in Q_p$ is the *initial state*, Γ_p is the *stack alphabet*, and Δ_p is the set of *pushdown transitions*. A transition $t \in \Delta_p$ is a tuple $\langle q_1, a, op, a', q_2 \rangle$ where $q_1, q_2 \in Q_p$ are states, $a, a' \in \Gamma_p \cup \{\varepsilon\}$ are stack symbols, and op is an *operation* of the form $c!m$, $c?m$, or nop with $m \in M$ and $c \in C$.

Transition System. We define the transition system induced by pushdown processes communicating through lossy channels. A *Lossy Channel Pushdown System* (LCPS for short) consists of a set P of pushdown processes. Let process $p \in P$ be of the form $\langle Q_p, q_p^{init}, \Gamma_p, \Delta_p \rangle$. Define $Q := \cup_{p \in P} Q_p$ and $\Gamma := \cup_{p \in P} \Gamma_p$. A configuration γ is of the form $\langle s, \alpha, \omega \rangle$, where $s : P \mapsto Q$ is a state map such that $s(p) \in Q_p$, $\alpha : P \mapsto \Gamma^*$ is a stack map such that $\alpha(p) \in {\Gamma_p}^*$, and $\omega : C \mapsto M^*$ is a channel map. Intuitively, α defines the contents of the stacks of the processes, while s and ω have the same meaning as for the case of LCS. We define the initial configuration $\gamma^{init} := \langle s^{init}, \alpha^{init}, \omega^{init} \rangle$ where $s^{init}(p) = q_p^{init}$ and $\alpha^{init}(p) = \varepsilon$ for all $p \in P$, and $\omega^{init}(c) = \varepsilon$ for all $c \in C$ (i.e., the system starts from a configuration where all the processes are in their initial states and where all the stacks and channels are empty).

We define a transition relation $\longrightarrow$ on the set of configurations as follows. Consider configurations $\gamma_1 = \langle s_1, \alpha_1, \omega_1 \rangle$, $\gamma_2 = \langle s_2, \alpha_2, \omega_2 \rangle$, and a transition $t = \langle q_1, a_1, op, a_2, q_2 \rangle \in \Delta_p$ for some $p \in P$. We write $\gamma_1 \xrightarrow{t} \gamma_2$ to denote that $s_1(p) = q_1$, $s_2 = s_1[p \leftarrow q_2]$, $\alpha_1(p) = a_1 \cdot u$ for some $u \in {\Gamma_p}^*$, $\alpha_2 = \alpha_1[p \leftarrow a_2 \cdot u]$ and one of the following properties is satisfied: (i) $op = c!m$ and $\omega_2 \preceq \omega_1[c \leftarrow m \cdot \omega_1(c)]$, (ii) $op = c?m$ and $\omega_2 \preceq \omega_1[c \leftarrow w]$ where $\omega_1(c) = w \cdot m$, (iii) $op = nop$ and $\omega_2 \preceq \omega_1$. The notions of *computations* and *bounded phase computations* are defined in the similar way as for the case of LCS.

Bounded-Phase Reachability Problem. In the following, we show that the (bounded-phase) reachability problem for LCPS is undecidable. The undecidability holds even for the 2-reachability problem for an LCPS that contains only one pushdown process with two lossy channels (the proof is in Appendix A).

Theorem 8. *k-reachability for* LCPS *is undecidable.*

8 Stuttering Lossy Channels

In this section, we consider processes communicating through stuttering lossy channels where messages can be both lost and duplicated.

Communicating Finite-State Processes. In the following, we give the model definition for finite-state processes communicating through stuttering lossy channels and show that the bounded-phase reachability problem is NP-COMPLETE. The syntax of the considered system is exactly the same as the one of LCS (described in Section 3). Next, we define the induced transition system. A *Stuttering Lossy Channel System* (SLCS for short) consists of a set P of finite-state processes. Let process $p \in P$ be of the form $\langle Q_p, q_p^{init}, \Delta_p \rangle$. Configurations are defined as for the case of LCS. We define a transition relation $\longrightarrow$ on the set of configurations as follows. Consider configurations $\gamma_1 = \langle s_1, \omega_1 \rangle$, $\gamma_2 = \langle s_2, \omega_2 \rangle$, and a transition $t = \langle q_1, op, q_2 \rangle \in \Delta_p$ for some $p \in P$. We write $\gamma_1 \xrightarrow{t} \gamma_2$ to denote that $s_1(p) = q_1$, $s_2 = s_1[p \leftarrow q_2]$, and that there is $\omega : C \mapsto M^*$ such that $\omega_2(c') \in \mathit{Stuttering}(\omega(c'))$ for all $c' \in C$ and one of the following properties is satisfied: (i) $op = c!m$ and $\omega \preceq \omega_1[c \leftarrow m \cdot \omega_1(c)]$, (ii) $op = c?m$ and $\omega \preceq \omega_1[c \leftarrow w]$ where $\omega_1(c) = w \cdot m$, (iii) $op = nop$ and $\omega \preceq \omega_1$. The notions of *computations* and *bounded phase computations* are defined in the similar way as for the case of LCS. Then, we can show the following theorem (the proof is in Appendix B).

Theorem 9. *k-reachability for* SLCS *is* NP-COMPLETE.

Communicating Pushdown Processes. We can extend the definition of SLCS to the case where each process is a pushdown as for the case of LCPS (Section 7). This leads to the class of *Stuttering Lossy Channel Pushdown System* (SLCPS). The proof of the following theorem is in Appendix C.

Theorem 10. *k-reachability problem for* SLCPS *is undecidable.*

9 Unordered Channels

In this section, we consider finite-state processes communicating through unordered lossy channels where messages can be re-ordered.

Communicating Finite-State Processes. In the following, we give the model when the processes are finite-state. finite-state processes. The syntax of the system is the same as the one of LCS (Section 3). Next, we define the induced transition system. An *Unordered Channel System* (UCS for short) consists of a set P of finite-state processes. Let process $p \in P$ be of the form $\langle Q_p, q_p^{init}, \Delta_p \rangle$. Define $Q := \cup_{p \in P} Q_p$. A configuration γ is of the form $\langle s, \omega \rangle$, where $s : P \mapsto Q$ is a state map such that $s(p) \in Q_p$, and $\omega : C \times M \mapsto \mathbb{N}$ is a channel map. Intuitively, ω defines the contents of the channels (i.e., we associate to each message its number of occurrences in each channel). We define the initial configuration $\gamma^{init} := \langle s^{init}, \omega^{init} \rangle$ where $s^{init}(p) = q_p^{init}$ for all $p \in P$, and $\omega^{init}(c, m) = 0$ for all $c \in C$ and $m \in M$ (i.e., the system starts from a configuration where all the processes are in their initial states and where all the channels are empty).

We define a transition relation $\longrightarrow$ on the set of configurations as follows. Consider configurations $\gamma_1 = \langle s_1, \omega_1 \rangle$, $\gamma_2 = \langle s_2, \omega_2 \rangle$, and a transition $t = \langle q_1, op, q_2 \rangle \in \Delta_p$ for some $p \in P$. We write $\gamma_1 \xrightarrow{t} \gamma_2$ to denote that $s_1(p) = q_1$, $s_2 = s_1[p \leftarrow q_2]$, and one of the following three properties is satisfied: (i) $op = c!m$ and $\omega_2 = \omega_1[(c,m) \leftarrow (\omega_1(c,m) + 1)]$, (ii) $op = c?m$, $\omega_1(c,m) \geq 1$ and $\omega_2 = \omega_1[(c,m) \leftarrow (\omega_1(c,m) - 1)]$, or (iii) $op = nop$ and $\omega_2 = \omega_1$. The notions of *computations* and *bounded phase computations* are defined in the similar way as in the case of LCS. Then, we can show the following theorem (the proof is in Appendix D).

Theorem 11. *k-reachability for* UCS *is* NP-COMPLETE.

Communicating Pushdown Processes. We can extend the definition of UCS to the case where each process is a pushdown as for the case of LCPS (Section 7). This leads to the class of *Unordered Channel Pushdown System* (UCPS for short).

Theorem 12. *k-reachability problem for* UCPS *is* NP-COMPLETE.

10 Perfect Channels

In this section, we consider finite-state processes communicating through perfect channels, and show that the bounded-phase reachability problem is undecidable.

The syntax of the considered system is the same as the one of LCS (described in Section 3). We define the induced transition system. A *Perfect Channel System* (PCS for short) consists of a set P of finite-state processes. Let process $p \in P$ be of the form $\langle Q_p, q_p^{init}, \Delta_p \rangle$. Configurations are defined as for the case of LCS. We define a transition relation $\longrightarrow$ on the set of configurations as follows. Consider configurations $\gamma_1 = \langle s_1, \omega_1 \rangle$, $\gamma_2 = \langle s_2, \omega_2 \rangle$, and a transition $t = \langle q_1, op, q_2 \rangle \in \Delta_p$ for some $p \in P$. We write $\gamma_1 \xrightarrow{t} \gamma_2$ to denote that $s_1(p) = q_1$, $s_2 = s_1[p \leftarrow q_2]$, and one of the following properties is satisfied: (i) $op = c!m$ and $\omega_2 = \omega_1[c \leftarrow m \cdot \omega_1(c)]$, (ii) $op = c?m$ and $\omega_2 = \omega_1[c \leftarrow w]$ where $\omega_1(c) = w \cdot m$, (iii) $op = nop$ and $\omega_2 = \omega_1$. The notions of *computations* and *bounded phase computations* are defined in the similar way as for the case of LCS. Then, we can show the following theorem (the proof is in Appendix E).

Theorem 13. *k-reachability problem for* PCS *is undecidable.*

11 Experimental Data

We have implemented our technique in a prototype tool called Alternator. The tool is available on GitHub [2], where we also supply the source of all experiments listed below. The tool uses the frontend of the implementation provided by Marques et al in [19], to get XML representations of the protocols from spreadsheets. We have implemented a Python application that, given such an XML representation, builds an SMT-LIB [6] formula as described in Section 5. This SMT-LIB formula can then be given to any SMT solver supporting the SMT-LIB version 2 standard. In our case we use the Z3 solver [21].

We have applied our prototype to a number of different protocols. The results demonstrate the efficiency of our framework. We analyze the web service protocols Subservice Termination Protocol (STP) and Business Agreement with Coordinator Completion (CC). The purpose of these protocols is to ensure that two (or three in the case of STP) processes agree on the global state of the system, as is commonly needed in SOA (Service-Oriented Architecture) frameworks. For more information on these protocols, see [23, 19]. By CCv2, we mean the augmented version of the CC protocols that can be found in [23]. Furthermore, we have applied our tool to modified versions of the well-known Alternating Bit Protocol (ABP_f) and Sliding Window protocol (SW_f) where we have intentionally introduced some errors. The SYNC protocol is a simple protocol requiring perfect channels. The Jingle example [28] is a multimedia session establishment protocol that is used by applications such as Google Talk, Coccinella and Miranda IM.

P	Sem	Gen. Time	SMT	Ph.	Res	P	Sem	Gen. Time	SMT	Ph.	Res
STP	UCS	0.1	0.1	12	U	CC	UCS	0.8	0.2	6	U
STP	SLCS	2.8	38.4	8	S	CC	SLCS	70.8	10.7	2	S
STP	LCS	2.8	13.0	8	S	CC	LCS	70.2	10.1	2	S
CCv2	UCS	1.8	0.8	8	U	ABP$_f$	SLCS	0.5	3.7	4	U
CCv2	SLCS	163.8	26.2	2	S	ABP$_f$	LCS	0.5	5.9	4	U
CCv2	LCS	159.3	24.3	2	S	ABP$_f$	UCS	0.1	0.0	4	U
SW$_f$	SLCS	0.4	0.6	2	U	SYNC	SLCS	0.2	1.3	14	U
SW$_f$	LCS	0.4	0.4	2	U	SYNC	LCS	0.2	2.1	14	U
SW$_f$	UCS	0.0	0.0	2	U	SYNC	UCS	0.2	0.1	14	U
JINGLE	SLCS	18.4	10.8	8	U	JINGLE	LCS	21.2	21.1	8	U

Table 1. Experimental Results

The results of our analyses can be seen in Table 1. The column "Gen. Time" gives the time that our tool takes to build an SMT-LIB formula. The column "SMT" shows the time that the SMT-solver takes to decide the satisfiability of the generated formula. All times are in seconds. The column "Sem" shows under which channel semantics we have run the examples. Finally the columns "Ph." and "Res" show the number of phases and the result of our analysis. If the result of the analysis is "U" (Unsafe), the number in the "Ph." column is the bound required to prove the result. If the result is "S" (Safe), meaning we did not reach the bad state within the given bound, the number in the "Ph." column is the greatest number of phases that we are able to use without the SMT-solver needing more than 30 s to return an answer. The sizes of the generated automata and the number of assertions fed to the SMT-solver are reported in [2]. More examples and results are also available in [2]. All experiments were performed on a 3.1 GHz Intel Core i5 with 4 GB of RAM.

12 Conclusions and Future Work

Semantics	Finite-state process	Pushdown process
Lossy	NP-COMPLETE	undecidable
Stuttering Lossy	NP-COMPLETE	undecidable
Unordered	NP-COMPLETE	NP-COMPLETE
Perfect	undecidable	undecidable

Table 2. Decidability/Complexity Results for the Bounded-Reachability

We have introduced a new concept for under-approximating the behavior of communicating processes, namely *phase-bounded* computations. We have shown that phase-bounded reachability can be reduced to the satisfiability of logical formulas whose satisfiability can be checked by SMT-solvers, thus yielding an efficient analysis of system behavior. The framework can be instantiated to several classes of channel semantics such as *lossy*, *stuttering*, and *unordered*. The strength of the method is confirmed by results form the application of our prototype on examples from several different application areas. Using the translation, we have also established complexity results for checking bounded reachability on the above classes of systems. Finally, we give undecidability results for the case where the channels are *perfect* and for the case where the processes are not finite-state with (stuttering) lossy channels. A summary of these results is given in Table 2. While our prototype is already efficient on the considered examples, there is room for several improvements such as minimizing the graphs of processes and the size of the unfolding of processes obtained in the purification step (Section 4) in order to reduce the time that our prototype takes to build an SMT-LIB formula. Also, we are planning to consider systems where the message alphabet is infinite, e.g., ranging over numerical domains.

References

1. Abdulla, P., Jonsson, B.: Undecidable verification problems for programs with unreliable channels. In: ICALP. pp. 316–327. LNCS 820, Springer (1994)
2. Abdulla, P.A., Atig, M.F., Cederberg, J.: Alternator - Verifier of programs by bounding mode alternations `https://github.com/it-apv/alternator`
3. Abdulla, P.A., Atig, M.F., Chen, Y.F., Leonardsson, C., Rezine, A.: Counter-example guided fence insertion under tso. In: TACAS. LNCS, vol. 7214, pp. 204–219. Springer (2012)
4. Abdulla, P.A., Collomb-Annichini, A., Bouajjani, A., Jonsson, B.: Using forward reachability analysis for verification of lossy channel systems. Formal Methods in System Design 25(1), 39–65 (2004)
5. Abdulla, P.A., Jonsson, B.: Verifying programs with unreliable channels. In: Proc. LICS '93, 8th IEEE Int. Symp. on Logic in Computer Science. pp. 160–170 (1993)
6. et al, C.B.: C.: The smt-lib standard: Version 2.0. Tech. rep. (2010)
7. Atig, M.F., Bouajjani, A., Burckhardt, S., Musuvathi, M.: On the verification problem for weak memory models. In: POPL. pp. 7–18. ACM (2010)
8. Atig, M.F., Bouajjani, A., Touili, T.: On the reachability analysis of acyclic networks of pushdown systems. In: CONCUR. LNCS, vol. 5201, pp. 356–371. Springer (2008)
9. Boigelot, B., Godefroid, P.: Symbolic verification of communication protocols with infinite state spaces using qdds. FMSD 14(3), 237–255 (1999)
10. Bouajjani, A., Emmi, M.: Bounded phase analysis of message-passing programs. In: TACAS. LNCS, vol. 7214, pp. 451–465. Springer (2012)
11. Bouajjani, A., Esparza, J., Touili, T.: A generic approach to the static analysis of concurrent programs with procedures. In: POPL. pp. 62–73. ACM (2003)
12. Bouajjani, A., Habermehl, P.: Symbolic reachability analysis of fifo-channel systems with nonregular sets of configurations. Theor. Comput. Sci. 221(1-2), 211–250 (1999)
13. Brand, D., Zafiropulo, P.: On communicating finite-state machines. J. ACM 30(2), 323–342 (1983)

14. Cook, S.A.: The complexity of theorem-proving procedures. In: STOC. pp. 151–158. ACM (1971)
15. Geeraerts, G., Raskin, J.F., Begin, L.V.: Expand, enlarge and check: New algorithms for the coverability problem of wsts. J. Comput. Syst. Sci. 72(1), 180–203 (2006)
16. Hague, M., Lin, A.W.: Synchronisation- and reversal-bounded analysis of multithreaded programs with counters. In: CAV. LNCS, vol. 7358, pp. 260–276. Springer (2012)
17. Heußner, A., Leroux, J., Muscholl, A., Sutre, G.: Reachability analysis of communicating pushdown systems. Logical Methods in Computer Science 8(3) (2012)
18. Ibarra, O.H.: Reversal-bounded multicounter machines and their decision problems. J. ACM 25(1), 116–133 (1978)
19. Jr., A.P.M., Ravn, A., Srba, J., Vighio., S.: csv2uppaal `https://github.com/csv2uppaal`
20. Lipton, R.: The reachability problem requires exponential time. Technical Report TR 66 (1976)
21. de Moura, L.M., Bjørner, N.: Z3: An efficient SMT solver. In: TACAS. LNCS, vol. 4963, pp. 337–340. Springer (2008)
22. Musuvathi, M., Qadeer, S.: Iterative context bounding for systematic testing of multithreaded programs. In: PLDI. pp. 446–455. ACM (2007)
23. Newcomer, E., Robinson, I.: (chairs). Web Services Business Activity Version 1.2 (2009), `http://docs.oasis-open.org/ws-tx/wstx-wsba-1.2-spec-os.pdf`
24. Post, E.L.: A variant of a recursively unsolvable problem. Bull. of the American Mathematical Society 52, 264–268 (1946)
25. Qadeer, S., Rehof, J.: Context-bounded model checking of concurrent software. In: TACAS'05. LNCS 3440 (2005)
26. Rackoff, C.: The covering and boundedness problems for vector addition systems. Theor. Comput. Sci. 6, 223–231 (1978)
27. Ravn, A.P., Srba, J., Vighio, S.: Modelling and verification of web services business activity protocol. In: TACAS. LNCS, vol. 6605, pp. 357–371. Springer (2011)
28. Saint-Andre, P.: Jingle: Jabber does multimedia. IEEE MultiMedia 14(1), 90–94 (Jan 2007)
29. Schnoebelen, P.: Verifying lossy channel systems has nonprimitive recursive complexity. Information Processing Letters 83(5), 251–261 (Sep 2002)
30. Seidl, H., Schwentick, T., Muscholl, A., Habermehl, P.: Counting in trees for free. In: ICALP. LNCS, vol. 3142, pp. 1136–1149. Springer (2004)
31. The Erlang Programming Language: `http://erlang.org`
32. The Scala Programming Language: `http://scala-lang.org`
33. Torre, S.L., Madhusudan, P., Parlato, G.: Context-bounded analysis of concurrent queue systems. In: TACAS. LNCS, vol. 4963, pp. 299–314. Springer (2008)

A Proof of Theorem 8

We prove Theorem 8 through a reduction from PCP (Post's Correspondence Problem), which is known to be undecidable [24]. We recall that PCP consists of two finite lists $\{u_1,\ldots,u_n\}$ and $\{v_1,\ldots,v_n\}$ of nonempty words over some alphabet Σ, and the task is to check whether there is a sequence of indices $i_1,\ldots,i_k \in [n]$ such that $u_{i_1}\cdots u_{i_k} = v_{i_1}\cdots v_{i_k}$.

We construct a pushdown process p communicating through two lossy channels $C=\{c_1,c_2\}$ such that, a specific state of p is 2-reachable from the initial configuration if and only if PCP has a solution for the considered instance. The pushdown process p proceeds as follows. First, it pushes twice the stack symbol $\bot$. Then, p guesses the solution of PCP as a sequence of indices $i_k,\ldots,i_1$ and performs iteratively a sequence of operations: It (1) sends the symbols of u_{i_j} to the channel c_1 while pushing $|u_{i_j}|$-times the stack symbol a into its stack, and (2) sends the symbols of v_{i_j} to the channel c_2 and pushes $|v_{i_j}|$-times the stack symbol a into its stack. At the end of this step, p sends the special message $\sharp$ to the channels c_1 and c_2 (to mark the end of the sent sequences). Thus, the channels c_1 and c_2 contain respectively subwords of $u := \sharp\cdot u_{i_1}\cdots u_{i_k}$ and $v := \sharp\cdot v_{i_1}\cdots v_{i_k}$ (due to the loss of some messages) while the stack of p contains the word $a^{(|u|+|v|-2)}\cdot\bot\cdot\bot$. Then, p checks that the two channels have the same contents and that no message has been lost. This is done by successively receiving a message in Σ from c_1 and c_2 while popping twice the symbol a from its stack. Finally, p checks whether it can receive the message $\sharp$ from c_1 and c_2 while popping twice the symbol $\bot$.

We observe that p can reach a configuration where its stack is empty (which can be reduced to state reachability problem) iff PCP has a solution (i.e., no message in the lossy channels has been lost). The formal definition of p can be found below.

Let $\Gamma_p = \{a,\bot\}$ be the stack alphabet of p and $M = \Sigma\cup\{\sharp\}$ be the set of messages manipulated by p.

To simplify the presentation, we will introduce some notations. Let $c \in C$ be a channel, $u \in \Gamma_p^+$ be a stack content, and $w \in \Sigma^+$ be a sequence of messages such that $|u| = |w| = i$. We use the macro transition $\langle q,\varepsilon,c!w,u,q'\rangle$ to denote the sequence of consecutive transitions $\langle q,\varepsilon,nop,\varepsilon,q_1^{\mathsf{tmp}}\rangle$, $\langle q_l^{\mathsf{tmp}},\varepsilon,c!w[i-l+1],u[l],q_{l+1}^{\mathsf{tmp}}\rangle$ for all $l\in[i]$, and $\langle q_{i+1}^{\mathsf{tmp}},\varepsilon,nop,\varepsilon,q'\rangle$ where $q_1^{\mathsf{tmp}},\ldots,q_{i+1}^{\mathsf{tmp}}$ are extra intermediate states of p that are not used anywhere else (and that we may omit from the set of states of p). Then, the set of transitions of p is defined as follows:

1. Push twice the symbol $\bot$: $\langle q_p^{init},\varepsilon,nop,\bot,q_\bot\rangle$ and $\langle q_\bot,\varepsilon,nop,\bot,q\rangle$
2. Choose a number $l\in[1..n]$: $\langle q,\varepsilon,nop,\varepsilon,q_l\rangle$ and $\langle q',\varepsilon,nop,\varepsilon,q_l\rangle$
3. Send the sequence of u_l to c_1: $\langle q_l,\varepsilon,c_1!u_l,a^{|u_l|},q_l'\rangle$
4. Send the sequence of v_l to c_2: $\langle q_l',\varepsilon,c_2!v_l,a^{|v_l|},q'\rangle$
5. Send the end marker $\sharp$: $\langle q',\varepsilon,c_1!\sharp,\varepsilon,q_\sharp\rangle$ and $\langle q_\sharp,\varepsilon,c_2!\sharp,\varepsilon,r\rangle$
6. Choose to receive a symbol $m\in\Sigma$ from c_1 and c_2: $\langle r,\varepsilon,nop,\varepsilon,r_m\rangle$
7. Receive m from c_1 and c_2: $\langle r_m,a,c_1?m,\varepsilon,r_m'\rangle$ and $\langle r_m',a,c_2?m,\varepsilon,r\rangle$
8. Receive the end marker $\sharp$: $\langle r,\bot,c_1?\sharp,\varepsilon,r_\sharp\rangle$ and $\langle r_\sharp,\bot,c_2?\sharp,\varepsilon,r'\rangle$

Let P be the LCPS consisting in the single pushdown processes p. Let s^{target} be the target state map of P such that $s^{target}(p) = r'$. Then the relation between the 2-

reachability problem for the LCPS $P := \{p\}$ and the existence of a solution for the PCP is given by the following lemma:

Lemma 14. *There is $i_1, \ldots, i_k \in [n]$ such that $u_{i_1} \cdots u_{i_k} = v_{i_1} \cdots v_{i_k}$ and $k \geq 1$ iff s^{target} is 2-reachable in P.*

B Proof of Theorem 9

The proof of Theorem 9 is similar to the case of LCS. The NP-hardness can be obtained using the same reduction from the Boolean Satisfiability Problem to the bounded-phase reachability problem for LCS (see Section 6).

The upper-bound is obtained by a polynomial reduction to the satisfiability of quantifier-free Presburger formulas. The construction of the formula follows the same steps as for the case of LCS. The only difference is in the *matching* conjunct that ensures that there is one-to-one matching between each *receive* transition and *send* transition (see Section 5). In the case of SLCS, this condition is weakened in order to allow that a *send* transition can be matched to several *receive* transitions (i.e., taking into account that any message in the channel can be duplicated). More precisely, we replace the conjunct $(\texttt{occ}(t) = 1) \wedge (\texttt{occ}(t') = 1) \wedge (\texttt{index}(\mathit{target}(t)) < \texttt{index}(\mathit{target}(t'))) \implies (\texttt{match}(t) < \texttt{match}(t'))$ by the following one: $(\texttt{occ}(t) = 1) \wedge (\texttt{occ}(t') = 1) \wedge (\texttt{index}(\mathit{target}(t)) < \texttt{index}(\mathit{target}(t'))) \implies (\texttt{match}(t) \leq \texttt{match}(t'))$.

C Proof of Theorem 10

We will prove the undecidability of the k-reachability problem for SLCPS by showing the every LCPS can be simulated by an SLCPS. More precisely, for any LCPS P over a finite set of channels C and a finite set of messages M, there is an SLCPS P' over the set of channels C and the set of messages $M \cup \{\sharp\}$, where $\sharp \notin M$ is a fresh message, such that a given target state map s^{target} is k-reachable in P if and only if it is k-reachable in P'. In the following we will sketch the main details of the reductions. Corresponding to each pushdown process p in P, P' has a pushdown process p'. For each state q of p, p' has two states q and q^{tmp}. Moreover, for each *send* transition $t = \langle q_1, a_1, c!m, a_2, q_2 \rangle \in \Delta_p$, p' has two consecutive transitions $t_1 = \langle q_1, a_1, c!m, a_2, q_2^{tmp} \rangle \in \Delta_{p'}$ and $t_2 = \langle q_2^{tmp}, \varepsilon, c!\sharp, \varepsilon, q_2 \rangle \in \Delta_{p'}$. Intuitively, if p sends a message $m \in M$ to the channel $c \in C$, then p' sends m to c followed by a *send* operation of the message $\sharp$ to c. The insertion of the special symbol $\sharp$ allows to ensure that, any sent message in M, just one copy of it can be received (i.e., eliminating the stuttering behavior). A *receive* transition $t = \langle q_1, a_1, c?m, a_2, q_2 \rangle \in \Delta_p$ is simulated in p' by two consecutive transitions $t_1 = \langle q_1, a_1, c?m, a_2, q_2^{tmp} \rangle \in \Delta_{p'}$ and $t_2 = \langle q_2^{tmp}, \varepsilon, c?\sharp, \varepsilon, q_2 \rangle \in \Delta_{p'}$. This means that if p receives a message $m \in M$ from the channel $c \in C$, then p' receives m from c followed by a *receive* operation of $\sharp$ from c. Finally, any dummy transition of p is a transition of p'. Then, it is easy to see that: s^{target} is 2-reachable in P iff s^{target} is 2-reachable in P'.

D Proof of Theorem 11

The NP-hardness can be shown using the same reduction from the Boolean Satisfiability Problem to the bounded-phase reachability problem for LCS (see Section 6). The upper-bound is obtained by a polynomial reduction the satisfiability of quantifier-free Presburger formulas. The construction of the formula follows the same steps as in the case of LCS. The only difference is in the *matching* conjunct that ensures that if a *receive* transition t occurs before another one t' then the matching *send* transition of t occurs before the matching *send* transition of t' (see Section 5). For the case of UCS, this condition is weakened in order to allow that the matching *send* transition of t' can occur before the matching *send* transition of t. More precisely, we replace the conjunct $(\mathtt{occ}\,(t) = 1) \wedge (\mathtt{occ}\,(t') = 1) \wedge (\mathtt{index}\,(\mathit{target}\,(t)) < \mathtt{index}\,(\mathit{target}\,(t'))) \implies (\mathtt{match}\,(t) < \mathtt{match}\,(t'))$ by the following one: $(\mathtt{occ}\,(t) = 1) \wedge (\mathtt{occ}\,(t') = 1) \wedge (\mathtt{index}\,(\mathit{target}\,(t)) \neq \mathtt{index}\,(\mathit{target}\,(t'))) \implies (\mathtt{match}\,(t) \neq \mathtt{match}\,(t'))$.

E Proof of Theorem 13

We prove Theorem 13 through a reduction from the PCP (Post's Correspondence Problem). Let $\{u_1, \ldots, u_n\}$ and $\{v_1, \ldots, v_n\}$ be an instance of PCP over some alphabet Σ.

We construct a process p communicating through two perfect channels $C = \{c_1, c_2\}$ such that, a specific state of p is 2-reachable from the initial configuration if and only if PCP has a solution for the considered instance. The process p proceeds as follows. First, it guesses the solution of PCP as a sequence of indices $i_k, \ldots, i_1$ and performs iteratively a sequence of operations: It (1) sends the symbols of u_{i_j} to the channel c_1, and (2) sends the symbols of v_{i_j} to the channel c_2. At the end of this step, p sends the special message $\sharp$ to the channels c_1 and c_2 (to mark the end of the sent sequences). Thus, the channels c_1 and c_2 contain respectively $u := \sharp \cdot u_{i_1} \cdots u_{i_k}$ and $v := \sharp \cdot v_{i_1} \cdots v_{i_k}$. Then, p checks that the two channels have the same contents. This is done by successively receiving a message in Σ from c_1 and c_2. Finally, p checks if it can receive the message $\sharp$ from c_1 and c_2.

We observe that p can reach a configuration where its channels are empty (which can be reduced to state reachability problem) iff PCP has a solution.

Let us define more formally the reduction. Let $M = \Sigma \cup \{\sharp\}$ be the set of messages manipulated by p.

To simplify the presentation, we will introduce some notations. Let $c \in C$ be a channel, and $w \in \Sigma^+$ be a sequence of messages such that $|w| = i$. We use the macro transition $\langle q, c!w, q' \rangle$ to denote the sequence of consecutive transitions $\langle q, , nop, q_1^{\mathsf{tmp}} \rangle$, $\langle q_l^{\mathsf{tmp}}, c!w[i-l+1], q_{l+1}^{\mathsf{tmp}} \rangle$ for all $l \in [i]$, and $\langle q_{i+1}^{\mathsf{tmp}}, nop, q' \rangle$ where $q_1^{\mathsf{tmp}}, \ldots, q_{i+1}^{\mathsf{tmp}}$ are extra intermediate states of p that are not used anywhere else (and that we may omit from the set of states of p). Then, the set of transitions of p is defined as follows:

1. Choose a number $l \in [1..n]$: $\langle q^{\mathsf{init}}, nop, q_l \rangle$ and $\langle q, nop, q_l \rangle$
2. Send the sequence of u_l to c_1: $\langle q_l, c_1!u_l, q_l' \rangle$
3. Send the sequence of v_l to c_2: $\langle q_l', c_2!v_l, q \rangle$

4. Send the end marker $\sharp$: $\langle q, c_1!\sharp, q_\sharp\rangle$ and $\langle q_\sharp, c_2!\sharp, r\rangle$
5. Choose to receive a symbol $m \in \Sigma$ from c_1 and c_2: $\langle r, nop, r_m\rangle$
6. Receive m from c_1 and c_2: $\langle r_m, c_1?m, r'_m\rangle$ and $\langle r'_m, c_2?m, r\rangle$
7. Receive the end marker $\sharp$: $\langle r, c_1?\sharp, r_\sharp\rangle$ and $\langle r_\sharp, c_2?\sharp, r'\rangle$

Let P be the PCS consisting in the single processes p. Let s^{target} be the target state map of P such that $s^{target}(p) = r'$. Then the relation between the 2-reachability problem for the PCS $P := \{p\}$ and the existence of a solution for the PCP is given by the following lemma:

Lemma 15. *There is $i_1, \ldots, i_k \in [n]$ such that $u_{i_1} \cdots u_{i_k} = v_{i_1} \cdots v_{i_k}$ and $k \geq 1$ iff s^{target} is 2-reachable in P.*

F Proof of Theorem 12

Globally bounded-phase reachability problem. First, we introduce the notion of globally bounded-phase computations for UCPS. Given a bound $k \in \mathbb{N}$, a computation is k-globally bounded if it can be seen as a concatenation of a at most k computations where all the processes are either in a receive or send mode. Formally, let C be a finite set of channels, M be a finite set of messages, and P be a UCPS such that for every $p \in P$, we have $\langle Q_p, q_p^{init}, \Gamma_p, \Delta_p\rangle$. A computation π is said to be k-globally-bounded, for a given $k \in \mathbb{N}$, if $\pi{\uparrow} = \delta_1 \cdot \delta_2 \cdot \cdots \cdot \delta_j$ where $j \leq k$ and $\delta_i \in (\bigcup_{p \in P}(\Delta_p^{snd} \cup \Delta_p^{nop}))^*$ or $\delta_i \in (\bigcup_{p \in P}(\Delta_p^{rcv} \cup \Delta_p^{nop}))^*$ for all $i : 1 \leq i \leq j$. The k-global reachability problem is defined in the similar way as for the k-reachability problem.

We can show that:

Theorem 16. *k-globally reachability problem for* UCPS *is* NP-COMPLETE.

The NP-hardness can be obtained using the same reduction from the Boolean Satisfiability Problem to the bounded-phase reachability problem for LCS (see Section 6).

The upper-bound can be obtained by an easy reduction from the k-globally reachability problem for a given UCPS P, with C as a finite set of channels and M as a finite set of messages, to the 0-synchronisation reachability problem for communicating pushdown automata $\mathbb{C}$ with $(|C| \times |M|)$ k-reversal bounded counters (which is in NP [16]) . We use a similar notation as in [16]. The idea is to associate a counter $x_{(m,c)}$ for each message $m \in M$ and channel $c \in C$. The current value of $x_{(m,c)}$ will denote the number of pending messages m in the channel c. For each process $p = \langle Q_p, q_p^{init}, \Gamma_p, \Delta_p\rangle$, we associate a pushdown automaton $\mathcal{P}_p$ that has the same set of states as p. The stack alphabet of $\mathcal{P}_p$ consists of the stack alphabet Γ_p of p and the bottom stack symbol $\perp \notin \Gamma_p$.

A send operation of $t = \langle q_1, a_1, c!m, a_2, q_2\rangle \in \Delta_p$ of p, with $a_1 \in \Gamma_p$, will be simulated in $\mathcal{P}_p$ by an increment transition of the form $((q_1, a_1, true), \varepsilon, (q_2, a_2, \mathbf{u}))$ where $\mathbf{u}$ is defined as follows $\mathbf{u}(x_{(m,c)}) = 1$ and $\mathbf{u}(x_{(m',c')}) = 0$ for all $(m,c) \neq (m',c')$ (i.e., incrementing the counter $x_{(m,c)}$). A send operation of the form $t = \langle q_1, \varepsilon, c!m, a_2, q_2\rangle \in \Delta_p$ of p will be simulated in $\mathcal{P}_p$ by an increment transition of the form $((q_1, a, true), \varepsilon, (q_2, a_2 \cdot a, \mathbf{u}))$ where $a \in \Gamma_p \cup \{\perp\}$ and $\mathbf{u}$ is defined as in the previous case.

A receive operation of $t = \langle q_1, a_1, c?m, a_2, q_2\rangle \in \Delta_p$ of p, with $a_1 \in \Gamma_p$, will be simulated in $\mathcal{P}_p$ by a decrement transition of the form $((q_1, a_1, true), \varepsilon, (q_2, a_2, \mathbf{u}))$ where $\mathbf{u}$ is defined as follows $\mathbf{u}(x_{(m,c)}) = -1$ and $\mathbf{u}(x_{(m',c')}) = 0$ for all $(m,c) \neq (m',c')$ (i.e., decrementing the counter $x_{(m,c)}$). A receive operation of the form $t = \langle q_1, \varepsilon, c?m, a_2, q_2\rangle \in \Delta_p$ of p will be simulated in $\mathcal{P}_p$ by a decrement transition of the form $((q_1, a, true), \varepsilon, (q_2, a_2 \cdot a, \mathbf{u}))$ where $a \in \Gamma_p \cup \{\perp\}$ and $\mathbf{u}$ is defined as in the previous case.

A *nop* operation of $t = \langle q_1, a_1, nop, a_2, q_2\rangle \in \Delta_p$ of p, with $a_1 \in \Gamma_p$, will be simulated in $\mathcal{P}_p$ by a transition of the form $((q_1, a_1, true), \varepsilon, (q_2, a_2, \mathbf{u}))$ where $\mathbf{u}$ is defined as follows $\mathbf{u}(x_{(m,c)}) = 0$ for all $(m,c) \in M \times C$. A *nop* operation of the form $t = \langle q_1, \varepsilon, nop, a_2, q_2\rangle \in \Delta_p$ of p will be simulated in $\mathcal{P}_p$ by a transition of the form $((q_1, a, true), \varepsilon, (q_2, a_2 \cdot a, \mathbf{u}))$ where $a \in \Gamma_p \cup \{\perp\}$ and $\mathbf{u}$ is defined as in the previous case.

Moreover, for every state $q \in Q_p$, $c \in C$, $a \in \Gamma_p \cup \{\perp\}$ and $m \in M$, the pushdown automaton $\mathcal{P}_p$ has a decrement transition of the form $((q, a, true), \varepsilon, (q, a, \mathbf{u}))$ where $\mathbf{u}$ is defined as follows $\mathbf{u}(x_{(m,c)}) = -1$ and $\mathbf{u}(x_{(m',c')}) = 0$ for all $(m,c) \neq (m',c')$ (i.e., decrementing the counter $x_{(m,c)}$). This transition is added in order to reduce the state reachability for UCPS problem to the configuration reachability problem for communicating pushdown automata with reversal bounded counters as considered in [16].

Then, it is easy to see that a target state s^{target} is k-globally reachability in P if and only if the configuration $((s^{target}(p), \perp)_{p\in P}, \mathbf{v})$ is 0-synchronisation-bounded reachability in $\mathbb{C}$ from the initial configuration $((q_p^{init}, \perp)_{p\in P}, \mathbf{v})$ where $\mathbf{v}(x_{(m,c)}) = 0$ for all $(m,c) \in M \times C$.

Bounded-phase reachability problem. In the following we show that the bounded-phase reachability problem for UCPS is reducible to the globally bounded-phase reachability problem for UCPS.

Let us fix an UCPS P with a finite set of messages M and a finite set of channels C. Let $p = \langle Q_p, q_p^{init}, \Gamma_p, \Delta_p\rangle$ for all $p \in P$. Let s^{target} be a target state.

We define $pure(P) := \{pure(p) | p \in P\}$, and define the state map $s_R^{target}(p) := r_p^{target}$ for all $p \in P$ in similar way as for the case of LCS (see Section 4). It follows that s_R^{target} is reachable in (the pure UCPS) $pure(P)$ iff s^{target} is k-reachable in P, which leads to the following lemma.

We recall that the set of states of $pure(p)$ by $\{r_p^{init}, r_p^{target}\} \cup \{\langle q, \mathtt{m}, i\rangle | q \in Q_p \wedge \mathtt{m} \in \{snd, rcv\} \wedge 1 \leq i \leq k\}$.

Lemma 17. *s^{target} is k-reachable in P iff s_R^{target} is reachable in $pure(P)$.*

Now, we borrow the idea of mode vectors from [16] (called here mode map). A mode map is a mapping from P to $\{snd, rcv\}$. Given a computation π of the form $\gamma_0 \xrightarrow{t_1} \gamma_1 \xrightarrow{t_2} \cdots \xrightarrow{t_n} \gamma_n$ in $pure(P)$, we associate to it a sequence $map(\pi)$ of mode maps as follows: $\mathbf{m_0 m_1} \cdots \mathbf{m_n}$ such that $\mathbf{m_0}(p) = snd$ for all $p \in P$ (we can also choose to set $\mathbf{m_0}(p) = rcv$ for all $p \in P$), and $\mathbf{m_j} = \mathbf{m_{j+1}}[p \leftarrow rcv]$ (resp. $\mathbf{m_j} = \mathbf{m_{j+1}}[p \leftarrow snd]$) if t_{j+1} is a receive (resp. send) transition of the process p, and $\mathbf{m_j} = \mathbf{m_{j+1}}$ otherwise.

Since each process in $pure(P)$ can only perform at most k phases in π. There are at most $n(k+1)$ changes in the mode map where a change is defined when $\mathbf{m_j} \neq \mathbf{m_{j+1}}$.

This implies that there is q sequence of indices $i_1,\ldots,i_h \in [n]$ such that: (1) $h \leq n(k+1)$, (2) $\mathbf{m_{i_{\ell}-1}} \neq \mathbf{m_{i_\ell}}$ for all $\ell \in [h]$, and (3) $\mathbf{m_{j-1}} = \mathbf{m_j}$ for all $j \in (\{1\ldots,n\} \setminus \{i_1,i_2,\ldots,i_h\})$.

Let $i_0 = 0$ and $i_{h+1} = n+1$. For every $\ell \in \{0,\ldots,h+1\}$, let π_ℓ be a sub-computation of π defined by the following sequence $\gamma_{i_\ell} \xrightarrow{t_{i_\ell+1}} \gamma_{i_\ell+1} \xrightarrow{t_{i_\ell+2}} \cdots \xrightarrow{t_{i_{\ell+1}-1}} \gamma_{i_{\ell+1}-1}$. We know from the mode vector that each process is either performing non-receive or non-send transitions (i.e., π_ℓ is an 1-bounded computation). Let P_ℓ^{snd} be the set of processes in a non-receive phase in π_ℓ (i.e., $\mathbf{m_{i_\ell}}(p) = snd$ for all $p \in P_\ell^{snd}$) and P_ℓ^{rcv} be the set of processes in a non-send phase in π_ℓ (i.e., $\mathbf{m_{i_\ell}}(p) = rcv$ for all $p \in P_\ell^{rcv}$).

Since channels have unordered semantics, we can construct a run π'_ℓ of the form $\gamma_{i_\ell} \xrightarrow{t'_{i_\ell+1}} \gamma'_{i_\ell+1} \xrightarrow{t'_{i_\ell+2}} \cdots \xrightarrow{t'_{i_{\ell+1}-1}} \gamma_{i_{\ell+1}-1}$ such that $\pi'{}_\ell \uparrow p = \pi_\ell \uparrow p$ for all $p \in P$ and there is an index $e \in \{i_\ell, i_\ell+2,\ldots,i_{\ell+1}-1\}$ such that for every $j \in \{i_\ell+1, i_\ell+2,\ldots,i_{\ell+1}-1\}$ with $j \leq e$, we have t_j is a transition of a process belonging to P_ℓ^{snd} and for every $j' \in \{i_\ell+1, i_\ell+2,\ldots,i_{\ell+1}-1\}$ with $j' > e$, we have $t_{j'}$ is a transition of a process belonging to P_ℓ^{rcv}. Observe that the order in which the non-send (resp. non-receive) operation of different processes in a non-receive (resp. non-send) phase is not important. Moreover, π'_ℓ is at most 2-globally bounded computation.

Since π'_ℓ and π_ℓ have the same source and target configurations, we can construct a π' having the same source and target configurations as π'. This is done by concatenating the computations π'_ℓ and $\gamma_{i_\ell-1} \xrightarrow{t'_{i_\ell}} \gamma'_{i_\ell}$. Thus, the computation π' is at most $2n(k+1)$-globally bounded. Hence, we have:

Lemma 18. *s_R^{target} is reachable in pure(P) iff s_R^{target} $2n(k+1)$-globally bounded reachable in pure(P).*

As corollary of Theorem 16, Lemma 17, and Lemma 18, we obtain Theorem 12.