Data Communicating Processes with Unreliable Channels

Parosh Aziz Abdulla
Uppsala University, Sweden
parosh@it.uu.se

C. Aiswarya
Chennai Mathematical Institute, India
Uppsala University, Sweden
aiswarya@cmi.ac.in

Mohamed Faouzi Atig
Uppsala University, Sweden
mohamed_faouzi.atig@it.uu.se

Abstract

We extend the classical model of lossy channel systems by considering systems that operate on a finite set of variables ranging over an infinite data domain. Furthermore, each message inside a channel is equipped with a data item representing its value. Although we restrict the model by allowing the variables to be only tested for (dis-)equality, we show that the state reachability problem is undecidable. In light of this negative result, we consider bounded-phase reachability, where the processes are restricted to performing either *send* or *receive* operations during each phase. We show decidability of state reachability in this case by computing a symbolic encoding of the set of system configurations that are reachable from a given configuration.

Categories and Subject Descriptors F.3.1 [*Specifying and Verifying and Reasoning about Programs*]

General Terms Theory

Keywords Reachability Problem, Lossy Channel Systems

1. Introduction

Model checking has played an important role in the area of algorithmic program verification. The original applications of model checking were aimed at finite-state systems, such as hardware circuits and protocol skeletons, which have a finite control structure and operate on finite data domains. However, in the last two decades, there has been a large amount of research devoted to the development of techniques for

- models with unbounded (or infinite) control structure, usually owing to auxiliary storage, such as Petri nets, (multi-) pushdown systems, channel machines etc., and
- models handling data from an unbounded domain such as timed automata, (data) register automata, etc.

There is a body of research extending finite state automata to infinite alphabets [9, 20, 21, 26, 29]. The unbounded data domain in these models is uninterpreted and data values may be compared only for (dis-)equality. Of particular interest are register automata where data values may be stored in a finite set of registers (variables) and transition guards involve register comparisons, and enjoy a PSPACE-complete reachability problem [17].

LICS '16, July 05 - 08, 2016, New York, NY, USA

ACM 978-1-4503-4391-6/16/07…$15.00
DOI: http://dx.doi.org/10.1145/http://dx.doi.org/10.1145/2933575.2934535

Recent works have considered systems that are infinite in the control structure as well as the data domain. For example infinite control structure may be due to pushdown storage and the infinite data may be interpreted as timestamps giving rise to timed pushdown systems [3, 14, 15]. In [11], distributed systems with unbounded number of processes are considered where (multi-) pushdown is used to model recursively definable protocols and the infinite data represent the process identifiers. Individual processes in such systems may be described using communicating register automata [10] (see also [6, 12]). The global evolution of such systems may be described using *data nets* (e.g., [24, 28]). In the setting of communicating timed automata [8, 16, 22] the infinite control structure is due to unbounded channels and infinite data is interpreted as time. Such systems are also studied when the channels are assumed lossy [2].

Lossy Channel Systems (LCS) are a variant of communicating finite state machines, where the channel contents are assumed to be lossy. This gives a decidable control state reachability problem, and at the same time is useful to model communication protocols that are designed to work correctly even when the underlying medium is unreliable in the sense that it can lose messages [1, 18, 19]. The model, however, assumes finite domain, which means that program variables and the messages inside the channels are assumed to range over finite domains.

In this paper, we extend lossy channel systems by adding (uninterpreted) data, thus obtaining data lossy channel systems (DLCS). Processes in DLCS are register automata and messages in the channels can carry data values from an infinite domain $\mathcal{D}$.

Our model is unbounded in two dimensions, namely we have an unbounded number of messages inside the channels each with an attribute that is fetched from the infinite domain $\mathcal{D}$. The operations we allow on the channels are the standard *send* and *receive* operations. When sending a message to a channel, its value may be defined to be the value of a program variable. When a message is received from a channel, its value may be copied to a variable. A DLCS allows comparing the values of variables, where two variables may be tested for equality or disequality. Also, a variable may be assigned a new arbitrary value, or the value of another variable.

Thus, our model combines the expressiveness of two orthogonal models, each having a decidable reachability problem. However, our first main result is undecidability of the control-state reachability for DLCS. Our undecidability proof relies on a novel encoding that uses the relation (equality and disequality) among the messages inside the channels in order to encode counters. An important property of the encoding is its "stability" in the presence of messages losses inside the channels. If a message is lost at any point during a run of the system, no subsequent steps of the system (including message losses) will make the channel contents a valid encoding of a counter again. Furthermore, this will be detected by the system which will then halt its run and thereby not reach the target process

state. Thus, the runs that may potentially reach a target state are only those that faithfully mimic a (two-) counter machine.

A useful policy that has recently been used extensively in order to circumvent undecidability (or to increase efficiency), in the verification of multi-threaded programs communicating through shared memory, is that of *context-bounding* [25, 27]. Inspired by the success of bounded context switches for shared memory programs, several definitions of contexts (and phases) have been proposed also for message passing programs (e.g., [4, 7, 13, 23]). In [4], the definition of a *phase* is a run of the system where processes are restricted to perform either *send* or *receive* operations on the channels [4]. In this paper, we show that, when considering only runs with a given bounded number k of phases, the reachability problem becomes decidable for DLCS. In fact, we show a stronger result, namely that the set of configurations reachable under the bounded phase assumption can be represented symbolically, and we provide a procedure for computing it.

Our proof of decidability relies on an intricate construction that exploits several properties of DLCS. For instance, configurations that have identical (dis)equality relations on the set of variables and messages inside the channels, will have similar behaviors, and in particular they are able to perform identical sequences of transitions. We will then provide a symbolic encoding based on a class of regular expressions augmented with data. For a given set of configurations, we will use the symbolic encoding for uniform characterization of the (infinite) set of reachable configurations during any phase of the computation.

Related Work The timed extension of LCS [2] has an entirely different behavior compared to DLCS. For instance, the state reachability problem is decidable for the former. In particular, the model of [2] does not allow the assignment of data (clock) values carried by messages to the process variables.

In [4], we consider the bounded-phase reachability problem for LCS (without data). The construction in [4] is different from the one presented here, and is based on a translation to a quantifier-free Presburger formula. In particular, it is shown in [4] that the problem is NP-COMPLETE for LCS, while there is trivially a PSPACE lower bound on the problem for DLCS since DLCS embed register automata [17]. Also note that with the definition of phase from [4], phase bounded reachability is undecidable for perfect channel system even when a finite message alphabet is considered. Lossiness of the channel was necessary for obtaining decidability in [4]. In this paper we show that the decidability continues even when the lossy channel system is enhanced to operate on unbounded data.

Multi-pushdown systems were extended with data in [11], and a bound on the number of phases were imposed to obtain decidability of model checking against monadic second-order logic. The definition of phase used in [11] as well as the proof technique is different from ours.

2. Lossy Channel system with Data

In this section, we introduce our model, by defining its syntax and operational semantics. We assume an arbitrary infinite data domain $\mathcal{D}$ (e.g., the set of natural numbers $\mathbb{N}$). First we need to fix some notations for the rest of this paper.

For sets A and B, we use $f : A \mapsto B$ to denote that f is a (possibly partial) function that maps A to B. We write $f(a) = \bot$ to indicate that f is undefined for a. For $a \in A$ and $b \in B$, we use $f[a \leftarrow b]$ to denote the function f' where $f'(a) = b$ and $f'(a') = f(a')$ for all $a' \neq a$. For $A' \subseteq A$, we use $f\big|_{A'}$ to denote the restriction of f to A'. For a relation $\mathcal{R} \subseteq A \times A$, we use $\mathcal{R}\big|_{A'}$ to denote the restriction of $\mathcal{R}$ to A'. For functions $f_1 : A_1 \mapsto B$ and $f_2 : A_2 \mapsto B$ with $A_1 \subseteq A_2$, we write $f_1 \sqsubseteq f_2$ to denote that $f_1(a) = f_2(a)$ for all $a \in A_1$. We use A^* to denote the set of words over A; and use ϵ to denote the empty word. For words $w, w' \in A^*$, we use $w_1 \bullet w_2$ to denote the concatenation of w_1 and w_2. We write $w \preceq w'$ to denote that w is a (not necessarily contiguous) subword of w', and define $w\downarrow := \{w' \mid w' \preceq w\}$ to be the downward closure of w. We use $|w|$ to denote the length of w. For $i : 1 \leqslant i \leqslant |w|$ we use $w[i]$ to denote the i^{th} element of w.

Syntax. A *Lossy Channel System with Data* (or simply DLCS) is a tuple $\mathcal{L} = \langle Q, \mathcal{X}, \mathcal{C}, \Sigma, \Delta \rangle$ where Q is a finite set of (local) states, $\mathcal{X}$ is a finite set of variables that range over $\mathcal{D}$, $\mathcal{C}$ is a finite set of channels, Σ is a finite (message) alphabet, and Δ is a finite set of transitions. A transition t is a triple $\langle q, op, q' \rangle$ where op is one of the following forms (where $x, y \in \mathcal{X}$ are variables, $a \in \Sigma$ is a symbol in the alphabet, and $\mathsf{ch} \in \mathcal{C}$ is a channel):

(a) $x \leftarrow y$, assigns the value of y to x.

(b) $x \leftarrow \circledast$, assigns a locally fresh value to x. That is x is nondeterministically assigned a value that is different from the current values of other variables and itself.

(c) $x = y$ tests whether the values of x and y are equal.

(d) $x \neq y$ tests whether the values of x and y are different.

(e) $\mathsf{ch}!\langle a, x \rangle$ sends a together with the value of the variable x to the channel ch.

(f) $x := \mathsf{ch}?a$ receives a from ch, and stores the associated value in the variable x.

We define $\texttt{SourceOf}(t) := q_1$, $\texttt{OpOf}(t) := op$, and $\texttt{TargetOf}(t) := q_2$. We define Δ^{snd} to be the set of transitions in Δ that perform operations of the form (a)–(e). That is, it is the set of transitions that do *not* perform *receive* operations. We define Δ^{rcv} analogously for the *receive* transitions, to be those performing operations of the form (a)–(d) or (f).

Semantics. A *variable state* is a function $V : \mathcal{X} \mapsto \mathcal{D}$ that defines the values of the variables. A *channel state* is a function $\omega : \mathcal{C} \mapsto (\Sigma \times \mathcal{D})^*$, which gives the content of each channel. This content is a word of pairs each consisting of an alphabet symbol together with a value. A *configuration* is a triple $\gamma = \langle q, V, \omega \rangle$ where $q \in Q$ is a state, V is a variable state, and ω is a channel state. We use $\texttt{StateOf}(\gamma)$, $\texttt{VarStateOf}(\gamma)$, $\texttt{ChannelStateOf}(\gamma)$ to denote q, V, and ω respectively. We say γ is *plain* if $\omega(\mathsf{ch}) = \epsilon$ for all channels $\mathsf{ch} \in \mathcal{C}$. We use $\texttt{ConfsOf}(\mathcal{L})$ to denote the set of configurations of $\mathcal{L}$.

We extend the ordering $\preceq$ to channel states, such that $\omega \preceq \omega'$ if $\omega(\mathsf{ch}) \preceq \omega'(\mathsf{ch})$ for all channels $\mathsf{ch} \in \mathcal{C}$. We further extend the ordering to configurations such that $\gamma = \langle q, V, \omega \rangle \preceq \langle q', V', \omega' \rangle = \gamma'$ if (i) $q' = q$, (ii) $V' = V$, and (iii) $\omega \preceq \omega'$. In other words, the states and values of variables in γ and γ' coincide, while the content of each channel in γ is a subword of the content of the same channel in γ'

We define the transition relation $\longrightarrow_{\mathcal{L}}$ on the set of configurations in two steps as follows. In the first step, we define the relation $\hookrightarrow_{\mathcal{L}}$. For configurations $\gamma = \langle q, V, \omega \rangle$ and $\gamma' = \langle q', V', \omega' \rangle$, and a transition $t = \langle q, op, q' \rangle \in \Delta$, we write $\gamma \overset{t}{\hookrightarrow}_{\mathcal{L}} \gamma'$ if one of the following conditions is satisfied:

- op is of the form $x \leftarrow y$, $V' = V[x \leftarrow V(y)]$ and $\omega' = \omega$. The variable x is assigned the value of y; the values of other variables and the contents of channels are not changed.
- op is of the form $x \leftarrow \circledast$, $V' = V[x \leftarrow d]$ for some $d \in \mathcal{D} \backslash \{V(x) \mid x \in \mathcal{X}\}$, and $\omega' = \omega$. The variable x is nondeterministically assigned an arbitrary value different from the values of other variables; the other variable values and channel contents remain the same.

- op is of the form $x = y$, $V(x) = V(y)$, $V' = V$, and $\omega' = \omega$. The transition is enabled only if the values of x and y are identical; only the control state is modified.
- op is of the form $x \neq y$, $V(x) \neq V(y)$, $V' = V$, and $\omega' = \omega$. The transition is enabled only if the values of x and y are different; only the control state is modified.
- op is of the form $\mathsf{ch}!\langle a, x\rangle$, $V' = V$, and $\omega' = \omega[\mathsf{ch} \leftarrow \langle a, V(x)\rangle \bullet \omega(\mathsf{ch})]$. The transition adds a message consisting of the symbol a and the value of x to the tail of channel ch.
- op is of the form $x := \mathsf{ch}?a$, $V' = V[x \leftarrow d]$, and $\omega = \omega'[\mathsf{ch} \leftarrow \omega'(\mathsf{ch}) \bullet \langle a, d\rangle]$ for some $d \in \mathcal{D}$. The transition receives the message at the head of channel ch if the alphabet symbol in that message is a. The value stored inside the message is assigned to x.

In the second step, we define the relation $\longrightarrow_{\mathcal{L}}$. More precisely, we let $\gamma \xrightarrow{t}_{\mathcal{L}} \gamma'$ denote that there are configurations γ_1, γ_2 such that $\gamma_1 \preceq \gamma$, $\gamma' \preceq \gamma_2$, and $\gamma_1 \xrightarrow{t}_{\mathcal{L}} \gamma_2$. We write $\gamma \longrightarrow_{\mathcal{L}} \gamma'$ to denote that $\gamma \xrightarrow{t}_{\mathcal{L}} \gamma'$ for some $t \in \Delta$. The reflexive transitive closure of the relation $\longrightarrow_{\mathcal{L}}$ is denoted by $\xrightarrow{*}_{\mathcal{L}}$. A *computation* of $\mathcal{L}$ is a sequence $\gamma_0 \xrightarrow{t_1}_{\mathcal{L}} \gamma_1 \xrightarrow{t_2}_{\mathcal{L}} \cdots \xrightarrow{t_n}_{\mathcal{L}} \gamma_n$.

Reachability For a configuration γ, we define $\mathtt{Reach}\,(\mathcal{L})\,(\gamma) := \left\{\gamma' \mid \gamma \xrightarrow{*}_{\mathcal{L}} \gamma'\right\}$, i.e., it is the set of configuration reachable from γ. For a set $\Gamma \subseteq \mathtt{ConfsOf}\,(\mathcal{L})$ of configurations, we define $\mathtt{Reach}\,(\mathcal{L})\,(\Gamma) := \cup_{\gamma\in\Gamma}\mathtt{Reach}\,(\mathcal{L})\,(\gamma)$. For a state $q \in Q$, we use $\gamma \xrightarrow{*}_{\mathcal{L}} q$ to denote that $\gamma \xrightarrow{*}_{\mathcal{L}} \gamma'$ for some γ' with $\mathtt{StateOf}\,(\gamma') = q$, i.e., from γ, we can reach a configuration whose state is q. In such a case we say that q is *reachable* from γ. An instance of the *reachability problem* is defined by a plain configuration γ and a state $\mathsf{Target} \in Q$. The question is whether Target is reachable from γ.

Bounded-Phase Reachability We introduce bounded-phase computations [4]. We view a computation as consisting of a number of phases where, during a given phase, the system either only performs *send* operations, or only performs *receive* operations (in addition to the operations on variables). Consider a computation $\pi = \gamma_0 \xrightarrow{t_1}_{\mathcal{L}} \gamma_1 \xrightarrow{t_2}_{\mathcal{L}} \cdots \xrightarrow{t_n}_{\mathcal{L}} \gamma_n$. We define $\pi^\bullet := t_1 t_2 \cdots t_n$, i.e., it is the sequence of transitions that occur in π. Given a sequence of transitions $\delta = t_1 t_2 \cdots t_n \in \Delta^*$, we say that δ is a *phase* if either $t_i \in \Delta^{snd}$ for all $i : 1 \leqslant i \leqslant n$, or $t_i \in \Delta^{rcv}$ for all $i : 1 \leqslant i \leqslant n$. A computation π is said to be *k-phase bounded* if $\pi^\bullet = \delta_1 \bullet \delta_2 \bullet \cdots \bullet \delta_j$ where $j \leqslant k$ and δ_i is a phase for all $i : 1 \leqslant i \leqslant j$. In other words, the transitions in π form at most k phases. For configurations γ and γ', we say that γ' is *k-phase reachable* from γ if γ' is reachable from γ by a k-phase bounded computation. For a configuration γ, we define $\mathtt{Reach}\,(k)\,(\mathcal{L})\,(\gamma) := \left\{\gamma' \mid \gamma' \text{ is } k\text{-phase reachable from } \gamma\right\}$, i.e., it is the set of configurations that are k-phase reachable from γ'. Bounded phase reachability is defined in a similar manner to state reachability. In the *bounded-phase reachability problem*, we are also given a natural number $k \in \mathbb{N}$, and we are asked whether Target is k-phase reachable from γ.

3. Undecidability of Reachability

In this section, we reduce the reachability problem for 2-counter machines to the reachability problem for DLCS.

The main idea of the reduction is to encode the value of each counter using a family of sets $\mathcal{F}_n$ (described below). These sets have an important feature (captured by Lemma 1), namely that we cannot obtain one set from another through the deletion of messages. This makes them good candidates for the exact representation of counter values (in the presence of losses) in the channels.

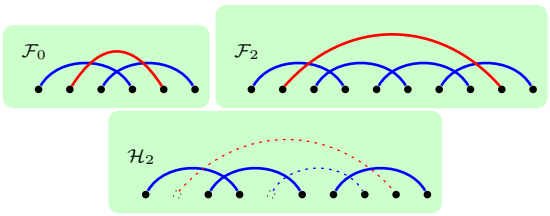

Figure 1: Illustrations of $\mathcal{F}_n$ and $\mathcal{H}_n$.

A counter value cannot be "accidentally" reduced through the loss of messages. This would give an element of a different set $\mathcal{H}_n$. As we shall see below, our simulation can recognize that this has happened. In such a case, we make sure that the simulation will never reach the target state. After presenting the encoding, we describe how to translate an instance of the reachability problem for 2-counter machines to an equivalent instance of the reachability problem for DLCS. We will give three gadgets that simulate the operations of decrementing, incrementing, and testing the value of a counter respectively. Using this, we describe how a run of the counter machine is simulated by a run of the DLCS.

3.1 Counter Machines

We recall the standard model of 2-counter machines operating on two counters c_1 and c_2. Such a machine is a triple $\mathcal{M} = \langle Q, q_{init}, \Delta\rangle$, where Q is a finite set of (local) states, $q_{init} \in Q$ is the initial state, and Δ is a finite set of transitions. A transition is a triple $\langle q, op, q'\rangle$ where $q, q' \in Q$ are states, and op is an operation of one of the three forms (where $\mathsf{c} \in \{\mathsf{c}_1, \mathsf{c}_2\}$): (i) $\mathtt{inc}(\mathsf{c})$ increases the value of c by one; (ii) $\mathtt{dec}(\mathsf{c})$ decreases the value of c by one; (iii) $\mathsf{c} = 0?$ checks whether the value of c is equal to zero. The machine $\mathcal{M}$ induces a transition system as follows. A configuration γ is a triple $\langle q, n_1, n_2\rangle$ where $q \in Q$ gives the state of $\mathcal{M}$, and $n_1, n_2 \in \mathbb{N}$ are the values of the counters. The transition relation $\longrightarrow_{\mathcal{M}}$ is the standard one for Minsky machines. An instance of the reachability problem consists of a state $\mathsf{Target} \in Q$, and the task is to check whether $\langle q_{init}, 0, 0\rangle \xrightarrow{*}_{\mathcal{M}} \langle \mathsf{Target}, n_1, n_2\rangle$ for some $n_1, n_2 \in \mathbb{N}$. It is well-known that this problem is undecidable.

3.2 Encoding

We present a family of sets that we use to encode counter values. Define the set $\mathcal{E} := \mathcal{D}^*$, i.e., $\mathcal{E}$ is the set of words over $\mathcal{D}$. We define a family $\mathcal{F}_0, \mathcal{F}_1, \mathcal{F}_2, \ldots \subset \mathcal{E}$ such that $w \in \mathcal{F}_n$ if the following two conditions are satisfied: (i) $|w| = 2n + 6$, and (ii) $w[i_1] = w[i_2]$ if and only if either (2a) $i_1 = 2$ and $i_2 = |w| - 1$, or (2b) $1 \leqslant i_1 \leqslant |w| - 3$, $i_1 \mod 2 = 1$, and $i_2 = i_1 + 3$. In other words, two elements are equal if either (i) the first element is in an odd position and the second element is three positions away, or (ii) they are the second and next last elements respectively. As an example, if we take $\mathcal{D}$ to be $\mathbb{N}$, then an element of $\mathcal{F}_2$ would be $4\ 2\ 5\ 4\ 3\ 5\ 8\ 3\ 2\ 8$. We can imagine the sets $\mathcal{F}_n$ as words with edges that connect the partner elements. Illustrations of elements in the sets $\mathcal{F}_0$ and $\mathcal{F}_2$ are given in the Fig. 1, where a line connecting two positions indicates the values stored in these positions are equal. We notice that each word in $\mathcal{F}_n$ consists of pairs of elements with identical values. We call each element of a pair, a (*left* or *right*) *partner* of the other. For instance, in a word in $\mathcal{F}_3$, element 5 is the left partner of 8, and 10 is the right partner of 7. For each $\mathcal{F}_n$ we distinguish six elements that we call the *pivot elements*. The three left-most symbols are called the first, second, and third left pivot elements; and the three right-most symbols are called the first, second, and third right pivot elements. We call the rest of the elements in the word *intermediate elements*. For

instance, in a word in $\mathcal{F}_2$, the pivot elements are the first, second, third, 8^{th}, 9^{th}, and 10^{th} elements respectively; while the 4^{th}, 5^{th}, 6^{th} and 7^{th} elements are intermediate. Notice that the second left pivot and the right second pivot are always partners. Also, notice that a word in $\mathcal{F}_0$ only contains pivot elements, and that each left pivot element is the left partner is the partner of the corresponding right pivot element.

For $n \in \mathbb{N}$, we define $\mathcal{G}_n := \mathcal{F}_n \downarrow$ and define $\mathcal{H}_n := \mathcal{G}_n - \mathcal{F}_n$. In other words, we get a member of $\mathcal{H}_n$ by deleting some elements in $\mathcal{F}_n$. Fig. 1 shows a member of $\mathcal{H}_2$ derived by deleting the second and fifth elements. We define $\mathcal{F} := \cup_{n \geqslant 0} \mathcal{F}_n$, $\mathcal{G} := \cup_{n \geqslant 0} \mathcal{G}_n$, and $\mathcal{H} := \cup_{n \geqslant 0} \mathcal{H}_n$. Notice that any data word from the set $\mathcal{F}$ forms a closed inter-linked chain. If some message (or letter data pair) is lost, then a link is broken and the chain opens up. Losing more messages will only result in breaking more links, and hence the chain once open can never be closed again. This observation allows us to derive the following lemma, namely, for $m \neq n$, we cannot derive $\mathcal{F}_m$ from $\mathcal{F}_n$ though the deletion of elements.

LEMMA 1. $\mathcal{F} \cap \mathcal{H} = \varnothing$.

3.3 Translation

Consider an instance of the reachability problem for 2-counter machines defined by a 2-counter machine $\mathcal{M} = \langle Q^{\mathcal{M}}, q^{\mathcal{M}}_{init}, \Delta^{\mathcal{M}} \rangle$ and a state $\mathsf{Target} \in Q^{\mathcal{M}}$. We will construct an equivalent instance of the reachability problem for DLCS. More precisely, we will derive a DLCS $\mathcal{L} = \langle Q^{\mathcal{L}}, \mathcal{X}^{\mathcal{L}}, \mathcal{C}^{\mathcal{L}}, \Sigma^{\mathcal{L}}, \Delta^{\mathcal{L}} \rangle$ with $\mathsf{Target} \in Q^{\mathcal{L}}$, and a plain configuration $\gamma^{\mathcal{L}}_{init} \in \mathtt{ConfsOf}(\mathcal{L})$, such that $\gamma^{\mathcal{L}}_{init} \xrightarrow{*}_{\mathcal{L}} \mathsf{Target}$ iff $\gamma^{\mathcal{M}}_{init} = \langle q^{\mathcal{M}}_{init}, 0, 0 \rangle \xrightarrow{*}_{\mathcal{M}} \langle \mathsf{Target}, n_1, n_2 \rangle$ for some $n_1, n_2 \in \mathbb{N}$. The DLCS $\mathcal{L}$ operates on two channels ch_1 and ch_2 which are used to encode the values of the counters c_1 and c_2 respectively. The message alphabet is defined by $\Sigma^{\mathcal{L}} = \{a, \triangleleft, \triangleright\}$. The occurrences of a together with the associated data values encode some set $\mathcal{F}_n$ corresponding to the value n of a counter. The symbols $\triangleleft$ and $\triangleright$ are end-markers: they mark the left and right ends of the word of messages inside the channel. A word $w \in (\{a, \triangleleft, \triangleright\} \times \mathcal{D})^*$ is said to be an *encoding* of *value* n if w is of the form $\langle \triangleleft, d \rangle \langle a, d_1 \rangle \langle a, d_2 \rangle \cdots \langle a, d_k \rangle \langle \triangleright, d' \rangle$, and $d_1 d_2 \cdots d_k \in \mathcal{F}_n$ (the values d and d' can be chosen arbitrarily and have no effect on the encoding.) We say that w a *semi-encoding* of *value* n if w is of one of the forms (i) $\langle \triangleleft, d \rangle \langle a, d_1 \rangle \langle a, d_2 \rangle \cdots \langle a, d_k \rangle \langle \triangleright, d' \rangle$, (ii) $\langle a, d_1 \rangle \langle a, d_2 \rangle \cdots \langle a, d_k \rangle \langle \triangleright, d' \rangle$, (iii) $\langle \triangleleft, d \rangle \langle a, d_1 \rangle \langle a, d_2 \rangle \cdots \langle a, d_k \rangle$ or (iv) $\langle a, d_1 \rangle \langle a, d_2 \rangle \cdots \langle a, d_k \rangle$ where $d_1 d_2 \cdots d_k \in \mathcal{G}_n$. A channel state ω of $\mathcal{L}$ is said to be an *encoding* (of *values* n_1 and n_2) if $\omega(\mathsf{ch}_1)$ and $\omega(\mathsf{ch}_2)$ are encodings (of values n_1 and n_2 respectively). By Lemma 1 it follows that if ch is a semi-encoding of value n then it cannot be an encoding of value m for any $m \neq n$. The set of states $Q^{\mathcal{L}}$ contains a unique initial state $q^{\mathcal{L}}_{init}$. For each state $q \in Q^{\mathcal{M}}$ there is a copy $q \in Q^{\mathcal{L}}$. Furthermore, $Q^{\mathcal{L}}$ also contains a number of additional "temporary" states that are used to perform "intermediate steps" in the simulation. These states will be given names like tmp, tmp^1_t, tmp^{12}_t, etc. Whenever we introduce such a new state in the simulation we assume that it is unique (different from all other states, whether temporary or not). A configuration $\langle q, n_1, n_2 \rangle$ of $\mathcal{M}$ is encoded by configurations of $\mathcal{L}$ of the form $\langle q, V, \omega \rangle$ where ω is an encoding of values n_1 and n_2. Finally, the set of variables is given by $\mathcal{X}^{\mathcal{L}} = \{x, x_0, x_1, x_2, x_3\}$.

We introduce some syntactic sugar that we will use in the rest of the section. Consider a channel ch and states q_1, q_2. We use $\langle q_1, \mathsf{ch} ? \triangleleft, q_2 \rangle$ to denote the sequence of transitions $\langle q_1, x := \mathsf{ch} ? \triangleleft, \mathsf{tmp} \rangle \langle \mathsf{tmp}, \mathsf{ch}! \langle x, \triangleleft \rangle, q_2 \rangle$. In other words, we rotate the left end-marker by first receiving it, storing its value in x, and then sending it back to the channel (in fact, the value of $\triangleleft$ will not be relevant in our simulation). We define the operation $\mathsf{ch} ? \triangleright$ analogously. For $y \in \{x, x_0, x_1, x_2, x_3\}$, we use $\langle q_1, \mathsf{ch} ? y, q_2 \rangle$ to denote the sequence of transitions $\langle q_1, y := \mathsf{ch} ? a, \mathsf{tmp} \rangle \langle \mathsf{tmp}, \mathsf{ch}! \langle y, a \rangle, q_2 \rangle$. In other words, we rotate the next a-symbol in the channel storing its value in y.

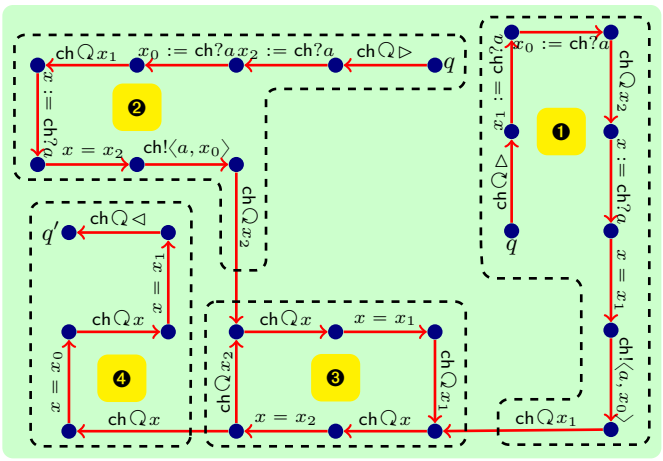

Figure 2: Decrementing the value of a counter

3.4 Decrementing

Consider a transition $\langle q, \mathtt{dec}(\mathsf{c}), q' \rangle \in \Delta^{\mathcal{M}}$. Let ch be the channel used for encoding the value of c (i.e., $\mathsf{ch} = \mathsf{ch}_1$ if $\mathsf{c} = \mathsf{c}_1$ and $\mathsf{ch} = \mathsf{ch}_2$ if $\mathsf{c} = \mathsf{c}_2$). The gadget for decrementing the value of c is shown in Fig. 2. It performs two tasks (i) it checks whether the content of ch is an encoding of value n for some $n > 0$, and (ii) at the same time transforms the content of ch to an arbitrary semi-encoding of value $n - 1$ (in particular it may transform the content of ch to an encoding of value $n - 1$). If the test in task (i) fails at any point of the simulation, the system is immediately blocked. To obtain a (semi-) encoding of value $n - 1$, we do the following: (i) Remove the third right pivot together with its partner. (ii) Keep the role of the second right pivot. (iii) Transform the first right pivot to the third right pivot. (iv) Transform the fifth right-most element into the first right pivot. This is carried out by four segments (sequences) of transitions, called ❶, ❷, ❸, and ❹, described below.

Segments ❶ and ❷ These segments are the initial phase of the gadget. They correspond to the case where n is odd and even respectively. We will consider the case where n is odd (segment ❶). The case where n is even (segment ❷) is similar. In segment ❶, we start from q and perform the following steps: (i) Rotate the right end-marker. (ii) Remove the third right pivot and store its value in x_1. (iii) Receive the second pivot and store its value in x_0. (iv) Rotate the first right pivot, which means that it will now become the third right pivot. (v) Remove the partner of the third right pivot and store its value in x (together with step (ii), this means that we have now removed both the third right pivot and its partner). (vi) check whether the values of the third pivot and its partner are equal. This is done by comparing the values of x and x_1. This step is needed since we want to ensure that the content of the channel is an encoding. (vii) Put back the second right pivot (Recall that the value of the second right pivot was stored in x_0 in step (iii)). (viii) Rotate the new third right pivot. (ix) Enter segment ❸. Observe that if the simulation starts from segment ❶ and an odd value of n, then it will get blocked during the simulation of segment ❸ and therefore will never succeed to enter into segment ❹.

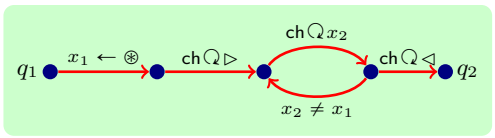

Figure 3: Generating a fresh value.

Segment ❸ In this segment, the intermediate elements are checked and rotated in the channel. We read the values of the right partners and compare them with the values of their left partners. Recall that such partners are three positions apart in the channels, and that their values should be equal. During this segment, the variables x_1 and x_2 hold the values of the two most recently read right partners. The value of the most recently read left partner is stored in the variable x.

Segment ❹ We perform the following steps: (i) Rotate the next message which we guess to be the second left pivot and store its value in x (if the guess fails then the simulation is halted). (ii) Compare the value of this message for equality with the value x_0 (which still holds the value of the second right pivot). (iii) Rotate the next message which we expect to be the first left pivot and store its value in x. (iv) Compare the value of this message for equality with the value x_1 (which holds the value of most recently read right partner, i.e., the supposed partner of the first left pivot). (v) Rotate the left end-marker and reach q'.

3.5 Incrementing

Consider a transition $\langle q, \mathtt{inc}(\mathsf{c}), q' \rangle \in \Delta^{\mathcal{M}}$. Let ch be the channel used for encoding the value of c. The gadget for incrementing the value of c is shown in Fig. 4. In a similar manner to the case of decrementing, the gadget for incrementing the value of c performs two tasks (i) it checks whether the content of ch is an encoding of value n for some $n \in \mathbb{N}$, and (ii) transforms the content of ch to an arbitrary semi-encoding of value $n+1$ (also here, it may transform the content of ch to an encoding of value $n+1$). Even in this case, if the test in step (i) fails at any point, the system will be blocked. To obtain a (semi-) encoding of value $n+1$, we do the following: (i) Keep the right pivots as they are. (ii) Keep the intermediate elements. (iii) Transform the third left pivot to an intermediate element. (iv) Transform the first left pivot into the third left pivot. (v) Keep the second left pivot. (vi) Add a new first left pivot with a fresh value (together with its right partner). The gadget consists of four segments of transitions, namely, ❶, ❷, ❸, and ❹. We first introduce a gadget that produces a value that is *fresh* wrt. a channel, i.e., a value that is different from all values that are currently inside the channel.

Generating Fresh Values To get a value that is fresh wrt. channel ch, we generate an arbitrary value and then rotate the contents of ch (see Fig. 3.) In each rotation step, we check that the generated value is different from the value that is being rotated. Concretely, we perform the following steps: (i) Assign an arbitrary value to x_1. (ii) Rotate the right end-marker. (iii) Rotate each message in the channel storing its value in x_2 and then compare (to check disequality) with the newly generated value (stored in x_1). (iv) Rotate the left end-marker. We use $x \leftarrow \mathit{fresh}\,(\mathsf{ch})$ to denote that we generate a value that is fresh wrt. channel ch, and store the value in the variable x.

Segments ❶ ***and*** ❷ These segments are the initial phase of the gadget, and they handle changes to the right pivots. They correspond to the case where n is odd and even respectively. We will consider the case where n is odd (segment ❶). In segment ❶, we start from q and perform the following steps: (i) Use the freshness

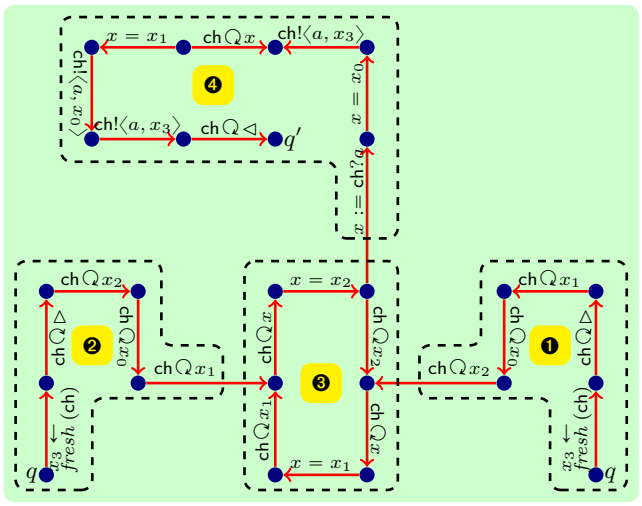

Figure 4: Incrementing the value of a counter

gadget to generate a new value that we store in the variable x_3. This value will be later used to define the new left pivot together with its right partner. (ii) Rotate the right end-marker. (iii) Rotate the third, second, and first right pivots and store them in the variables x_1, x_0, and x_2 respectively. These messages are rotated (and not removed) since they will keep their places in the new channel content.

Segment ❸. This segment is similar to the case of decrementing. The intermediate elements are checked and rotated to the channel in an identical manner.

Segment ❹. We perform the following steps: (i) Remove the next message which we guess to be the old second left pivot and store its value in x (if the guess fails then the simulation is halted). This message will keep its status as the second left pivot, and will therefore be added back to the channel in a later step. (ii) Compare the value of this message for equality with the value x_0 (which still holds the value of the second right pivot). (iii) Insert the right partner of the new first left pivot. The value of this message is still stored in x_3. (iv) Rotate the next message which expect to be the old first left pivot and store its value in x. By this rotation, the message will become the third left pivot element. (v) Compare the value of this message for equality with the value x_1 which holds the value of the current third left pivot (i.e., the value of the old first left pivot). (vi) Insert the new second left pivot (whose value is stored in x_0). (vii) Insert the new first left pivot (whose value is stored in x_3). (viii) Rotate the left end-marker.

3.6 Zero Testing

Consider a transition $\langle q, \mathsf{c} = 0?, q' \rangle \in \Delta^{\mathcal{M}}$. Let ch be the channel used for encoding the value of c. The gadget for checking whether the value of c is equal to zero is shown in Fig. 5. It checks whether the content of ch is an encoding of value 0. As a side effect of the testing the content of the channel may be changed to an arbitrary semi-encoding of value 0. In particular, it may keep the content of ch as an encoding of value 0 (if no messages are lost in the channel during the simulation). If the content of the channel is not an encoding of value 0 then the simulation is blocked. We start from q and perform the following steps: (i) Check that the right end-marker is the last message in the channel, in which case we rotate it. (ii) Rotate the next three messages. These messages are supposed to contain the values of the third, second, and first right pivots. We store these values in the variables x_2, x_1, and x_0 respectively. (iii) Rotate the next message. This message is supposed to contain the

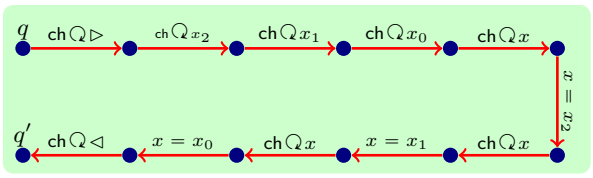

Figure 5: Testing whether the value of a counter is equal to 0.

value of the third left pivot. We store this value in the variable x. Since the values of the third left and right pivots should be equal, we compare the values of x and x_2 for equality. We repeat the procedure for the second and first pivots. (iv) We rotate the left end-marker, and enter state q'

3.7 Simulation

$\mathcal{L}$ initializes the contents of the channels ch_1 and ch_2. More precisely, $\mathcal{L}$ starts from $q^{\mathcal{L}}_{init}$ (with empty channels) and sends two arbitrary sequences of messages to the channels ch_1 and ch_2. Let $\gamma^{\mathcal{L}}_{init} = \langle q^{\mathcal{L}}_{init}, V_{init}, \omega_{init}\rangle$ be the initial configuration of $\mathcal{L}$ with $\omega_{init}(\mathsf{ch}_1) = \omega_{init}(\mathsf{ch}_2) = \epsilon$. Then, it checks whether the sent sequences form encodings of size 0 (using the gadget for zero testing). If the test is successful, $\mathcal{L}$ enters the state $q^{\mathcal{M}}_{init}$ from which it starts the proper simulation. In such a way, for each computation π of $\mathcal{M}$ from $\gamma^{\mathcal{M}}_{init} = \langle q^{\mathcal{M}}_{init}, 0, 0\rangle$ to a configuration of the form $\langle \mathsf{Target}, n_1, n_2\rangle$, there is a computation π' of $\mathcal{L}$ from $\gamma^{\mathcal{L}}_{init}$ to a configuration of the form $\langle \mathsf{Target}, V, \omega\rangle$ where ω is an encoding of values n_1 and n_2. On the other hand, since each gadget represents one transition in $\mathcal{M}$, and the simulation halts immediately if it detects that a semi-encoding is not an encoding (that is, if any messages have been lost inside the channels), it follows that if there is a computation π of $\mathcal{L}$ from $\gamma^{\mathcal{L}}_{init}$ to a configuration of the form $\langle \mathsf{Target}, V, \omega\rangle$ then there is a computation π' in $\mathcal{M}$ from $\gamma^{\mathcal{M}}_{init}$ to a configuration of the form $\langle \mathsf{Target}, n_1, n_2\rangle$. Since we have reduced the reachability problem for 2-counter machines to the reachability problem for DLCS, we get the following theorem.

THEOREM 2. *The reachability problem for* DLCS *is undecidable.*

4. Symbolic Encoding

We introduce our symbolic encoding of (infinite) sets of configurations. We will first recall different concepts related to equivalence relations, and define some operations on such relations.

Equivalence Relations Consider an equivalence relation, i.e., a reflexive, symmetric, and transitive relation $\mathcal{R}$ on a set A. We use $a\mathcal{R}b$ to denote that $\langle a, b\rangle \in \mathcal{R}$, and use $A/\mathcal{R}$ to denote the set of blocks (equivalence classes) of $\mathcal{R}$. For an element $a \in A$, we use $[a]_{\mathcal{R}}$ to denote the block of a. In other words $a\mathcal{R}b$ iff $[a]_{\mathcal{R}} = [b]_{\mathcal{R}}$. We use $\mathtt{EqRel}\,(A)$ to denote the set of equivalence relations on the set A. In our symbolic encoding, we put variables with identical values in the same block. We define $\mathcal{R}\,[a \rightsquigarrow b]$ to be the equivalence relation we get by moving a from its original block to the block of b. We use this operation to encode the effect of assigning the value of one variable to another (we move the second variable to the block of the first variable). We define $\mathcal{R}\,[a \rightsquigarrow \circledast]$ to be the equivalence relation we get by removing a from its original block, and putting it alone in a new block. We use this operation to encode the effect of updating the value of a variable to a value that is different from the values of all the other variables.

For sets A_1, A_2, an equivalence relation $\mathcal{R} \in \mathtt{EqRel}\,(A_1)$, and a function $f : A_1 \mapsto A_2$ from A_1 to A_2, we write $f \models \mathcal{R}$ to denote that, for all $a_1, a_2 \in A_1$, we have that $[a_1]_{\mathcal{R}} = [a_2]_{\mathcal{R}}$ iff $f(a_1) = f(a_2)$, i.e., f assigns identical (and unique) values to elements belonging to the same block in $\mathcal{R}$. We will use this to encode the fact that the process variables have values that respect a given equivalence relation on the set of variables.

We will define a number of operations on equivalence relations. For a set A, and an operation op, we define a partial function $op : \mathtt{EqRel}\,(A) \mapsto \mathtt{EqRel}\,(A)$. For an equivalence relation $\mathcal{R} \in \mathtt{EqRel}\,(A)$, we define $op(\mathcal{R}) := \mathcal{R}' \in \mathtt{EqRel}\,(A)$ as follows. (i) if op is $x \leftarrow y$ and then $\mathcal{R}' = \mathcal{R}\,[x \rightsquigarrow y]$, i.e., if x is assigned y then we move x to the block of y. (ii) op is $x \leftarrow \circledast$ then $\mathcal{R}' = \mathcal{R}\,[x \rightsquigarrow \circledast]$, i.e., x is moved to a newly created block. (iii) op is $x = y$, $[x]_{\mathcal{R}} = [y]_{\mathcal{R}}$, and $\mathcal{R}' = \mathcal{R}$, i.e., if x and y belong to the same block, then they satisfy the condition of the operation and the relation $\mathcal{R}$ is not changed. (iv) op is $x = y$, $[x]_{\mathcal{R}} \neq [y]_{\mathcal{R}}$ then $\mathcal{R}' = \bot$, i.e., since x and y belong to different blocks, they do not satisfy the condition of the operation, and hence the operation is blocked. The cases when op is $x \neq y$ are similar to $x = y$. (v) op is $\mathsf{ch}!\langle a, x\rangle$ and $\mathcal{R}' = \mathcal{R}$, i.e., sending a message to a channel does not affect the relation.

Symbolic Configurations Assume a DLCS $\mathcal{L} = \langle Q, \mathcal{X}, \mathcal{C}, \Sigma, \Delta\rangle$. The symbolic encoding will characterize, in a uniform manner, sets of k-phase reachable configurations in terms of the initial values of the variables. To that end, we will use a finite set I whose elements we call *initiators*. The initiators will be used as place holders for the initial values of the variables. For a set A (typically a subset of $\mathcal{X} \cup I$), an *A-valuation* is a mapping $\theta : A \mapsto \mathcal{D}$.

First, we define a symbolic encoding to represent (infinite) sets of channel states. A *symbolic word* v over I is of the form $a_1(\iota_1)a_2(\iota_2)\cdots a_n(\iota_n)$, where $a, a_1, \ldots, a_n \in \Sigma$ and, for all $i : 1 \leqslant i \leqslant n$, either $\iota_i \in I$ or ι_i is of the form $\neg J$ for some set $J \subseteq I$. We will use symbolic words as encodings of channel states. If the argument is $\iota \in I$, then the value carried by the message is equal to the value of ι. If the argument is $\neg J$, then the value is different from the values of all initiators in J. Formally, for an I-valuation θ, we define the denotation $[\![v]\!]_\theta$ to be the set of all words $w \in (\Sigma \times \mathcal{D})^*$ of the form $a_1(d_1)a_2(d_2)\cdots a_n(d_n)$ satisfying the following two conditions for all $i : 1 \leqslant i \leqslant n$: (i) if $\iota_i \in I$ then $d_i = \theta(\iota_i)$; and (ii) if $\iota_i = \neg J$ then $d_i \neq \theta(\iota)$ for all $\iota \in J$.

We introduce a class of expressions to generate symbolic words as follows. A *plain atomic expression* over I is of the form $a(\iota) + \epsilon$, where $a \in \Sigma$ and either $\iota \in I$ or $\iota = \neg J$ for some $J \subseteq I$. A *star atomic expression* over I is of the form $(a_1(\iota_1) + \cdots + a_n(\iota_n))^*$, where $a_1, \ldots, a_n \in \Sigma$ and, for all $i : 1 \leqslant i \leqslant n$, either $\iota_i \in I$ or $\iota_i = \neg J$ for some $J \subseteq I$. Sometimes, we use a set notation E^* to write the star expression above where $E = \{a_1(\iota_1), \ldots, a_n(\iota_n)\}$. An *atomic expression* e over I is either a plain or a star atomic expression over I. A product p over I is either (i) of the form $e_1 \bullet \cdots \bullet e_n$ where $e_1, \ldots, e_n$ are atomic expressions over I, or (ii) the empty word ϵ. The denotation $[\![\phi]\!]$ of a product over I is a set of symbolic words defined (in the expected manner) as follows. $[\![a(\iota) + \epsilon]\!] := \{a(\iota), \epsilon\}$, $[\![(a_1(\iota_1) + \ldots + a_k(\iota_n))^*]\!] := \{a_{j_1}(\iota_{j_1}) \bullet \cdots \bullet a_{j_k}(\iota_{j_k}) \mid j_1, \ldots, j_k \in \{1, \ldots, n\}\}$, and $[\![e_1 \bullet \cdots \bullet e_n]\!] := \{v_1 \bullet \cdots \bullet v_n \mid \forall i : 1 \leqslant i \leqslant n.\ v_i \in [\![e_i]\!]\}$. For a product p and an I-valuation θ, we define the denotation $[\![p]\!]_\theta := \{w \mid \exists v.\ (v \in [\![p]\!]) \wedge (w \in [\![v]\!]_\theta)\}$. In other words, we first generate the symbolic words in the denotation of p and then instantiate them according to the valuation θ.

A (*$\mathcal{C}$-indexed Data Simple Regular Expression*) (or simply a DSRE) over I is a function ϕ that maps each channel $\mathsf{ch} \in \mathcal{C}$ to a product p over I. For an I-valuation θ, and a DSRE ϕ, we define $[\![\phi]\!]_\theta$ such that $[\![\phi]\!]_\theta(\mathsf{ch}) := [\![\phi\,(\mathsf{ch})]\!]_\theta$ for each $\mathsf{ch} \in \mathcal{C}$. For DSREs ϕ_1, ϕ_2, we define $\phi_1 \bullet \phi_2$ to be the DSRE ϕ where $\phi\,(\mathsf{ch}) = \phi_1(\mathsf{ch}) \bullet \phi_2(\mathsf{ch})$ for all channels $\mathsf{ch} \in \mathcal{C}$.

Next, we define a symbolic encoding for (infinite) sets of configurations. A *symbolic configuration* over I is a triple $\beta = \langle q, \mathbb{V}, \phi\rangle$ where $q \in Q$ is a state, $\mathbb{V} \in \mathtt{EqRel}\,(\mathcal{X} \cup I)$ is an equiva-

lence relation over $\mathcal{X} \cup I$, and ϕ is a DSRE over I. For a set $J \subseteq \mathcal{X} \cup I$, and a J-valuation θ, we define the denotation $[\![\beta]\!]_\theta$ to be the set of all configurations of the form $\langle q, V, \omega\rangle$ such that there is a $\theta' : \mathcal{X} \cup I \mapsto \mathcal{D}$ where (i) $\theta \sqsubseteq \theta'$, (ii) $\theta' \models \mathbb{V}$, (iii) $V = \theta'|_{\mathcal{X}}$, and (iv) $\omega \in [\![\phi]\!]_{\theta'|_I}$. In other words, it is the set all configurations whose states are q, whose variable states respect the equivalence relation $\mathbb{V}$ wrt. the values assigned to the initiators, and where the channel states are defined by the denotation of the given DSRE. Notice that, if $J = \mathcal{X} \cup I$ and $\theta \models \mathbb{V}$ then $[\![\beta]\!]_\theta := \left\{\langle q, V, \omega\rangle \mid \left(V = \theta|_{\mathcal{X}}\right) \wedge \left(\omega \in [\![\phi]\!]_{\theta|_I}\right)\right\}$. For a set $\mathbb{B}$ of symbolic configurations, we define $[\![\mathbb{B}]\!]_\theta := \cup_{\beta\in\mathbb{B}}[\![\beta]\!]_\theta$.

For a symbolic configuration β, and a transition $t \in \Delta$, we define $\texttt{Post}(t)(\beta)$ to be a finite set $\mathbb{B}$ of symbolic configurations such that for every $J \subseteq I$ and $\theta : J \mapsto \mathcal{D}$, we have that $[\![\mathbb{B}]\!]_\theta = \left\{\gamma' \mid \exists\gamma \in [\![\beta]\!]_\theta.\ \gamma \xrightarrow{t}_{\mathcal{L}} \gamma'\right\}$. Define $\texttt{Post}(\beta) := \cup_{t\in\Delta}\texttt{Post}(t)(\beta)$, and $\texttt{Post}(\mathbb{B}) := \cup_{\beta\in\mathbb{B}}\texttt{Post}(\beta)$

LEMMA 3. *The set* $\texttt{Post}(\mathbb{B})$ *is computable.*

5. Bounded-Phase Reachability

In this section, we show, for a given symbolic configuration β, how to characterize the set of bounded-phase reachable configurations from the denotation of β. We do this in a stepwise manner. More precisely, we consider six types of DLCS, ①, ②, . . . , ⑥, where each type is defined by imposing different kinds of restrictions on the set of allowed transitions. We show how to reduce the reachability problem for each type to the reachability problem for a simpler type (one with more restrictions). Furthermore, for the simplest types, we show how to compute the needed sets of symbolic configurations. Finally, we show how to derive the sets of symbolic configurations for the more general types in terms of the sets of symbolic configurations for the simpler types. First, we consider a "graph" view of DLCS that we will use in our construction.

5.1 DLCS as Graphs

Sometimes, we view a DLCS $\mathcal{L} = \langle Q, \mathcal{X}, \mathcal{C}, \Sigma, \Delta\rangle$ as a labeled graph $\texttt{GraphOf}(\mathcal{L})$ where each node in the graph is a state $q \in Q$, and where there is an edge between two nodes q_1 and q_2 labeled with an operation op, denoted $q_1 \overset{op}{\rightsquigarrow}_{\mathcal{L}} q_2$, if there is a transition $t = \langle q_1, op q_2\rangle \in \Delta$. Furthermore, we write $q_1 \rightsquigarrow_{\mathcal{L}} q_2$ to denote that $q_1 \overset{op}{\rightsquigarrow}_{\mathcal{L}} q_2$ for some op, and use $\overset{*}{\rightsquigarrow}_{\mathcal{L}}$ to denote the reflexive transitive closure of $\rightsquigarrow_{\mathcal{L}}$. We define $(q\overset{*}{\rightsquigarrow}_{\mathcal{L}}) := \left\{q' \mid q\overset{*}{\rightsquigarrow}_{\mathcal{L}} q'\right\}$. Using the graph view, we will use graph terms to describe properties of $\mathcal{L}$. For instance, by "$\mathcal{L}$ is a Strongly Connect Component (SCC)" we mean that $\texttt{GraphOf}(\mathcal{L})$ is an SCC, and by "the SCC graph of $\mathcal{L}$" we mean the SCC graph of $\texttt{GraphOf}(\mathcal{L})$, etc. For a state $q \in Q$, we use $\texttt{SCCOf}(\mathcal{L}, q)$ to denote the (set of states in the) SCC to which q belongs. For a set $Q' \subseteq Q$, we define the DLCS $\mathcal{L}|_{Q'} := \left\langle Q', \mathcal{X}, \mathcal{C}, \Sigma, \Delta'\right\rangle$ where $\Delta' := \left\{\left\langle q, op, q'\right\rangle \mid \left(\left\langle q, op, q'\right\rangle \in \Delta\right) \wedge (q \in Q') \wedge (q' \in Q')\right\}$.

5.2 DLCS Types

We introduce different types of DLCS by imposing restrictions of the allowed types of transitions.

Type ① A general DLCS $\mathcal{L}$ is of type ①. In the sequel we will identify syntactic fragments of $\mathcal{L}$ which does not use send operations and receive operations. These two fragments would correspond to type ⑥ and ② respectively. We further identify more refined fragments of type ② DLCS which give rise to types ③ . . . ⑤.

Type ② A DLCS $\mathcal{L}$ is of type ② if $\Delta = \Delta^{snd}$, i.e., $\mathcal{L}$ does not contain any *receive* transitions. Notice that for any DLCS of type ②, reachability and k-phase reachability coincide for all $k \geqslant 1$.

Type ③ Consider a DLCS $\mathcal{L} = \langle Q, \mathcal{X}, \mathcal{C}, \Sigma, \Delta\rangle$. Consider a function $\lambda : Q \mapsto \texttt{EqRel}(\mathcal{X})$ that labels each state of $\mathcal{L}$ with an equivalence relation on the set of variables $\mathcal{X}$. We say that $\mathcal{L}$ is of *type* ③ wrt. λ if the following conditions are satisfied: (i) $\mathcal{L}$ is of type ②. (ii) Each transition $\langle q_1, op, q_2\rangle \in \Delta$, with $\lambda(q_1) = \mathcal{R}_1$ and $\lambda(q_2) = \mathcal{R}_2$, satisfies that $\mathcal{R}_2 = op(\mathcal{R}_1) \neq \bot$. A configuration $\gamma = \langle q, V, \omega\rangle$ is said to be *consistent* wrt. λ if $V \models \lambda(q)$.

We show how to reduce the reachability problem for DLCS $\mathcal{L} = \langle Q, \mathcal{X}, \mathcal{C}, \Sigma, \Delta\rangle$ of type ② to the reachability problem for DLCS of type ③. To that end, we will define three functions that (i) convert a DLCS $\mathcal{L}$ of type ② to a DLCS $\mathcal{K}$ type ③, (ii) convert a configuration of $\mathcal{L}$ to a configuration of $\mathcal{K}$, (iii) convert a set of configurations of $\mathcal{K}$ to a set of configurations of $\mathcal{L}$. We define $\texttt{Conv}^{②.③}(\mathcal{L})$ to be the DLCS $\mathcal{K} := \left\langle Q^{\mathcal{K}}, \mathcal{X}^{\mathcal{K}}, \mathcal{C}^{\mathcal{K}}, \Sigma^{\mathcal{K}}, \Delta^{\mathcal{K}}\right\rangle$, where $\mathcal{X}^{\mathcal{K}} := \mathcal{X}$, $\mathcal{C}^{\mathcal{K}} := \mathcal{C}$, $\Sigma^{\mathcal{K}} := \Sigma$. We define $Q^{\mathcal{K}} := \{\langle q, \mathcal{R}\rangle \mid (q \in Q) \wedge (\mathcal{R} \in \texttt{EqRel}(\mathcal{X}))\}$, i.e., each state in $\mathcal{K}$ is composed of a state of $\mathcal{L}$ together with an encoding of the equivalence relation to be satisfied by the current values of the variables. We define $\Delta^{\mathcal{K}}$ to be the set of tuples $\langle\langle q_1, \mathcal{R}_1\rangle, op, \langle q_2, \mathcal{R}_2\rangle\rangle$ such that (i) $\langle q_1, op, q_2\rangle \in \Delta$, and (ii) $\mathcal{R}_2 = op(\mathcal{R}_1) \neq \bot$. Define the labeling λ where $\lambda(\langle q, \mathcal{R}\rangle) := \mathcal{R}$.

LEMMA 4. $\mathcal{K}$ *is of type* ③ *wrt.* λ.

For a configuration $\gamma = \langle q, V, \omega\rangle$, we define $\texttt{Conv}^{②.③}(\gamma) := \langle\langle q, \mathcal{R}\rangle, V, \omega\rangle$ where $\mathcal{R}$ is the (unique) equivalence relation $\mathcal{R} \in \texttt{EqRel}(\mathcal{X})$ such that $V \models \mathcal{R}$. In other words, $\mathcal{R}$ is an abstraction of V relating elements that are assigned identical values by V.

For a configuration $\gamma = \langle\langle q, \mathcal{R}\rangle, V, \omega\rangle \in \texttt{ConfsOf}(\mathcal{K})$ with $V \models \mathcal{R}$, we define $\texttt{Conv}^{③.②}(\gamma) := \langle q, V, \omega\rangle$, i.e., we abstract away the equivalence relation in the definition of a state in $\mathcal{K}$. For a set of configurations $\Gamma \subseteq \texttt{ConfsOf}(\mathcal{K})$, we define $\texttt{Conv}^{③.②}(\Gamma) := \left\{\texttt{Conv}^{③.②}(\gamma) \mid \gamma \in \Gamma\right\}$.

LEMMA 5. $\texttt{Reach}(\mathcal{L})(\gamma) = \texttt{Conv}^{③.②}\left(\texttt{Reach}\left(\texttt{Conv}^{②.③}(\mathcal{L})\right)\left(\texttt{Conv}^{②.③}(\gamma)\right)\right)$.

Type ④ We say that $\mathcal{L}$ is of type ④ wrt. a labeling function $\lambda : Q \mapsto \texttt{EqRel}(\mathcal{X})$ if (i) $\mathcal{L}$ is of type ③ wrt. λ. (ii) $\mathcal{L}$ is an SCC.

Type ⑤ To define DLCS of type ⑤, we use the following definition. For a DLCS $\mathcal{L} = \langle Q, \mathcal{X}, \mathcal{C}, \Sigma, \Delta\rangle$, and a set of variables $I \subseteq \mathcal{X}$, we say that $\mathcal{L}$ is *stable* wrt. I if each variable $\iota \in I$ is only used in operations of the form $x \leftarrow \iota$, where $x \notin I$. In other words, the value of a variable in I may be assigned to another variable (outside I). However, its value is never modified, or compared to other variables. Furthermore, such a variable is not used in *send* operations.

Consider a DLCS $\mathcal{L} = \langle Q, \mathcal{X} \cup I, \mathcal{C}, \Sigma, \Delta\rangle$, where the set of variables is partitioned in two subsets $\mathcal{X}$ and I. Consider a function $\lambda : Q \mapsto \texttt{EqRel}(\mathcal{X} \cup I)$. We say that $\mathcal{L}$ is of *type* ⑤ wrt. λ and I if the following conditions are satisfied: (i) $\mathcal{L}$ is of type ④ wrt. λ. (ii) $\mathcal{L}$ is stable wrt. I. Intuitively, each variable $\iota \in I$ is used to record the initial value of some variables in $\mathcal{X}$, and hence ι is never used in transitions except when its value is assigned to some other variable. We will define three functions, namely $\texttt{Conv}^{④.⑤}(\mathcal{L}, \lambda, r)$, $\texttt{Conv}^{④.⑤}(\gamma)$, and $\texttt{Conv}^{⑤.④}(\Gamma)$, to convert the reachability problem for type ④ DLCS to the reachability problem for type ⑤ DLCS.

Consider a DLCS $\mathcal{L} = \langle Q, \mathcal{X}, \mathcal{C}, \Sigma, \Delta\rangle$ of type ④ wrt. a labeling $\lambda : Q \mapsto \texttt{EqRel}(\mathcal{X})$, and a state $r \in Q$. We define

$\texttt{Conv}^{④.⑤}(\mathcal{L},\lambda,r)$ to be the DLCS $\mathcal{K} := \langle Q^{\mathcal{K}}, \mathcal{X}^{\mathcal{K}}, \mathcal{C}^{\mathcal{K}}, \Sigma^{\mathcal{K}}, \Delta^{\mathcal{K}}\rangle$ where $\mathcal{C}^{\mathcal{K}} = \mathcal{C}$, $\Sigma^{\mathcal{K}} = \Sigma$, and $\mathcal{X}^{\mathcal{K}} := \mathcal{X} \cup I$ with $I = \{\iota_x \mid x \in \mathcal{X}\}$. In other words, we add a new variable ι_x for each variable $x \in \mathcal{X}$. Define $\mathcal{S} \in \texttt{EqRel}(\mathcal{X} \cup I)$ to be the unique relation such that $\mathcal{S}\big|_{\mathcal{X}} = \lambda(r)$ and $[x]_{\mathcal{S}} = [\iota_x]_{\mathcal{S}}$ for all $x \in \mathcal{X}$. In other words, $\mathcal{S}$ extends each block B in $\lambda(r)$ with the corresponding elements from I. We define the set of states $Q^{\mathcal{K}} \subseteq Q \times \texttt{EqRel}(\mathcal{X}^{\mathcal{K}})$ and define $\Delta^{\mathcal{K}}$ to be the smallest set such that

- $\langle r, \mathcal{S}\rangle \in Q^{\mathcal{K}}$, and
- if $\langle q_1, \mathcal{R}_1\rangle \in Q^{\mathcal{K}}$, $\langle q_1, op, q_2\rangle \in \Delta$ and $\mathcal{R}_2 = op(\mathcal{R}_1) \neq \bot$ then (i) $\langle q_2, \mathcal{R}_2\rangle \in Q^{\mathcal{K}}$, (ii) $\langle\langle q_1, \mathcal{R}_1\rangle, op, \langle q_2, \mathcal{R}_2\rangle\rangle \in \Delta^{\mathcal{K}}$, and (iii) if $op = (x \leftarrow \circledast)$ and $[\iota]_{\mathcal{R}_1} \cap \mathcal{X} = \varnothing$ then $\langle\langle q_1, \mathcal{R}_1\rangle, x \leftarrow \iota, \langle q_2, \mathcal{R}_2\rangle\rangle \in \Delta^{\mathcal{K}}$ where $\mathcal{R}_2 = \mathcal{R}_1[x \rightharpoonup \iota]$.

Intuitively, for each transition in $t \in \Delta$, we add a transition performing an identical operation in $\Delta^{\mathcal{K}}$ such that the latter respects the equivalence relations on the variables (the ones in both $\mathcal{X}$ and I). Notice that t only uses the variables in $\mathcal{X}$. Furthermore, since operations of the form $x \leftarrow \circledast$ will assign to x values that are different from the values of all the variables in I, we also need to add a transition of the form $x \leftarrow \iota$ in case the block of ι does not contain elements from $\mathcal{X}$ (the block in such a case is a singleton.) Define the labeling λ' such that $\lambda'(\langle q, \mathcal{R}\rangle) := \mathcal{R}$.

LEMMA 6. *$\mathcal{K}$ is of type ⑤ wrt. λ' and I.*

Consider a configuration $\gamma = \langle r, V, \omega\rangle \in \texttt{ConfsOf}(\mathcal{L})$ that is consistent wrt. λ. Define $\texttt{Conv}^{④.⑤}(\gamma) := \gamma' = \langle\langle r, \mathcal{S}\rangle, V', \omega\rangle$ where $V' : (\mathcal{X} \cup I) \mapsto \mathcal{D}$ is the unique variable state such that $V'\big|_{\mathcal{X}} = V\big|_{\mathcal{X}}$ and $V'(\iota_x) = V(x)$. In other words, we derive γ' from γ by defining the value of each member of I to be equal to the value of corresponding variable in $\mathcal{X}$. For a configuration $\gamma = \langle\langle q, \mathcal{R}\rangle, V, \omega\rangle \in \texttt{ConfsOf}(\mathcal{K})$, we define $\texttt{Conv}^{⑤.④}(\gamma) := \langle\langle q, \mathcal{R}\big|_{\mathcal{X}}\rangle, V\big|_{\mathcal{X}}, \omega\rangle$, i.e., we abstract away the values of the variables in I. For a set of configurations $\Gamma \subseteq \texttt{ConfsOf}(\mathcal{K})$, we define $\texttt{Conv}^{⑤.④}(\Gamma) := \left\{\texttt{Conv}^{⑤.④}(\gamma) \mid \gamma \in \Gamma\right\}$.

LEMMA 7. $\texttt{Reach}(\mathcal{L})(\gamma) = \texttt{Conv}^{⑤.④}\left(\texttt{Reach}\left(\texttt{Conv}^{④.⑤}(\mathcal{L},\lambda,r)\right)\left(\texttt{Conv}^{④.⑤}(\gamma)\right)\right)$.

Type ⑥ We say that $\mathcal{L}$ is of type ⑥ if $\Delta = \Delta^{rcv}$, i.e., $\mathcal{L}$ does not contain any *send* transitions.

5.3 Computing k-Phase Reachability

In this section, we show how to compute a finite set of symbolic configurations that characterize the reachability set. We start by the simplest types of DLCS, and then derive the set of symbolic configurations for one type from those computed for the simpler types.

Type ⑤. Consider a DLCS $\mathcal{L} = \langle Q, \mathcal{X} \cup I, \mathcal{C}, \Sigma, \Delta\rangle$ of type ⑤ wrt. a labeling λ and I. For a state $q \in Q$, we say that q is *covered* wrt. λ and I, if, for every $x \in \mathcal{X}$, there is an $\iota \in I$ such that $[x]_{\lambda(q)} = [\iota]_{\lambda(q)}$. For a configuration $\gamma \in \texttt{ConfsOf}(\mathcal{L})$, we say that γ is *covered* wrt. λ and I, if $\texttt{StateOf}(\gamma)$ is *covered* wrt. λ and I. We say that γ is *proper* wrt. λ and I, if γ is (i) covered wrt. λ and I, and (ii) consistent wrt. λ. Proper configurations represent "initial configurations" from which we will start the computations of the system. We will define a set of symbolic configurations over the set I that uniformly characterizes the reachable sets from all proper configurations that have a given local state and whose variables satisfy a given equivalence relation. The characterization is based on several properties of DLCS of type ⑤. First, if we start from a proper configuration γ, then for any reachable configuration γ' (i.e., $\gamma \xrightarrow{*}_{\mathcal{L}} \gamma'$), it is the case that γ' is consistent with λ. This implies that all transitions t with $\texttt{SourceOf}(t) = \texttt{StateOf}(\gamma')$ are enabled from γ', and hence, from γ' we can traverse any sequence of transitions corresponding to a path inside $\texttt{GraphOf}(\mathcal{L})$. Consequently all states in Q are actually reachable from γ. Furthermore, for all states, we reach exactly the same set of channel states. The reason is that whenever we have a given word w in a channel ch in a state q_1, we can traverse a sequence of transitions leading from q_1 to another state q_2 while losing all the extra messages we send to ch along the path, and hence obtaining w in ch in q_2. Finally, the set of channel states can be characterized by a star atomic expression. The reason is that different messages can be sent to the channels arbitrary numbers of times, by traversing the graph of $\mathcal{L}$, and the order among messages is irrelevant (we can obtain any order trough re-sending and losing of messages).

We characterize the channel states as a DSRE over I For each channel $\mathsf{ch} \in \mathcal{C}$, we define $\texttt{DSREOf}(\mathsf{ch}, \mathcal{L}, \lambda) := E^*$ where E is the smallest star atomic expression E^* over I, with E containing the following elements: (i) For each $q_1, q_2 \in Q$, $\mathsf{ch} \in \mathcal{C}$, $x \in \mathcal{X}$, $\iota \in I$, $a \in \Sigma$, such that $[x]_{\lambda(q_1)} \cap I \neq \varnothing$ and $\langle q_1, \mathsf{ch}!\langle a, x\rangle, q_2\rangle \in \Delta$, we have that $a(\iota) \in E$ for some ι with $[x]_{\lambda(q_1)} = [\iota]_{\lambda(q_1)}$. Intuitively, the current value of x is equal to the initial value ι, and hence the value of the message sent to the channel is encoded correspondingly. (ii) For each $q_1, q_2 \in Q$, $\mathsf{ch} \in \mathcal{C}$, $x \in \mathcal{X}$, $a \in \Sigma$, such that $[x]_{\lambda(q_1)} \cap I = \varnothing$ and $\langle q_1, \mathsf{ch}!\langle a, x\rangle, q_2\rangle \in \Delta$, we have that $a(\neg I) \in E$. The current value of x is different from the initial values of all the variables, and hence the value of the message is encoded accordingly. Define the DSRE $\texttt{DSREOf}(\mathcal{L}, \lambda) := \phi$ where $\phi(\mathsf{ch}) := \texttt{DSREOf}(\mathsf{ch}, \mathcal{L}, \lambda)$ for all $\mathsf{ch} \in \mathcal{C}$. Define the set of symbolic configurations $\texttt{SymConfOf}(\mathcal{L}, \lambda) := \{\langle q, \lambda(q), \texttt{DSREOf}(\mathcal{L}, \lambda)\rangle \mid q \in Q\}$. Notice that $\texttt{SymConfOf}(\mathcal{L}, \lambda)$ is defined over I. Consider a plain configuration $\gamma \in \texttt{ConfsOf}(\mathcal{L})$ that is proper wrt. λ and I. Define the valuation $\theta : I \mapsto \mathcal{D}$ such that $\theta(\iota_x) := \texttt{VarStateOf}(\gamma)(x)$.

LEMMA 8. $\left\{\langle q, V\big|_{\mathcal{X}}, \omega\rangle \mid \langle q, V, \omega\rangle \in \texttt{Reach}(\mathcal{L})(\gamma)\right\} = [\![\texttt{SymConfOf}(\mathcal{L}, \lambda)]\!]_\theta$.

Type ④ We derive the set of symbolic configurations for DLCS of type ④ from the set of symbolic configurations DLCS of type ⑤. Consider a DLCS $\mathcal{L} = \langle Q, \mathcal{X}, \mathcal{C}, \Sigma, \Delta\rangle$ of type ④ wrt. a labeling $\lambda : Q \mapsto \texttt{EqRel}(\mathcal{X})$, and a state $r \in Q$. Define $\mathcal{K} := \texttt{Conv}^{④.⑤}(\mathcal{L}, \lambda, r)$. Let the labeling λ' be such that $\lambda'(q, \mathcal{R}) = \mathcal{R}$. Define $\texttt{SymConfOf}(\mathcal{L}, \lambda, q)$ to be the set of symbolic configurations of the form $\langle q, \mathbb{V}, \phi\rangle$ such that $\langle\langle q, \mathcal{R}\rangle, \mathbb{V}, \phi\rangle \in \texttt{SymConfOf}(\mathcal{K}, \lambda')$ for some $\mathcal{R} \in \texttt{EqRel}(\mathcal{X}^{\mathcal{K}})$. Consider a plain configuration $\gamma \in \texttt{ConfsOf}(\mathcal{L})$ such that $\texttt{StateOf}(\gamma) = r$ and such that γ is consistent wrt. λ. Define the valuation $\theta : I \mapsto \mathcal{D}$ such that $\theta(\iota_x) := \texttt{VarStateOf}(\gamma)(x)$. Using Lemma 8 and Lemma 7 we can prove the following lemma.

LEMMA 9. $\texttt{Reach}(\mathcal{L})(\gamma) = [\![\texttt{SymConfOf}(\mathcal{L}, \lambda, r)]\!]_\theta$.

Type ③ Consider a DLCS $\mathcal{L} = \langle Q, \mathcal{X}, \mathcal{C}, \Sigma, \Delta\rangle$ that is of type ③ wrt. a labeling $\lambda : Q \mapsto \texttt{EqRel}(\mathcal{X})$. We describe how to compute the symbolic configurations for $\mathcal{L}$. The idea is to consider the SCC graph of $\texttt{GraphOf}(\mathcal{L})$, derive the symbolic configurations for each SCC separately, and then combine the results for the different SCCs. In defining the symbolic configurations, we will use particular sets of initiators. More precisely, for a DLCS with a set of variables $\mathcal{X}$, we will use the set $I_{\mathcal{X}} := \{\iota_x \mid x \in \mathcal{X}\}$, which contains one initiator for each variable in $\mathcal{X}$. Intuitively, we will use ι_x to carry the initial value of x. Furthermore, for each $i, j \in \mathbb{N}$, we consider the initiator set $I^i_{\mathcal{X}} := \{\iota^i_x \mid x \in \mathcal{X}\}$, and define $I^{[i..j]}_{\mathcal{X}} := \cup_{i \leqslant k \leqslant j} I^k_{\mathcal{X}}$. We will use the different sets to characterize initial values of variables in the different SCCs in the graph of a

DLCS. We define $\texttt{Depth}(\mathcal{L})$ to be the length of the longest simple path in the SCC graph of $\mathcal{L}$. Let $d = \texttt{Depth}(\mathcal{L})$. We will use induction on d to compute the set of symbolic configurations. To that end we use the set of $I_{\mathcal{X}}^{[0..d]}$ of initiators. Intuitively, we first define a set of symbolic configurations over $I_{\mathcal{X}}$ for the current SCC C using the construction for type ④ described above. Then, using induction, we derive a set of symbolic configurations over $I_{\mathcal{X}}^{[0..d-1]}$ corresponding to the paths from C in SCC graph of $\texttt{GraphOf}(\mathcal{L})$. Finally, we compose these sets to obtain a set over $I_{\mathcal{X}}^{[0..d]}$. First we define some operations on the sets. For $x \in \mathcal{X}$, and $i \in \mathbb{N}$, we define $x^{++} := x$, $(\iota_x)^{++} := \iota_x^0$, and $\left(\iota_x^i\right)^{++} := \iota_x^{i+1}$. In other words, the operation leaves a variable in $\mathcal{X}$ as it is, transforms an initiator in $I_{\mathcal{X}}$ to the corresponding initiator in $I_{\mathcal{X}}^0$, and increases the index of the initiator otherwise. We will now define operations to compose symbolic configurations for a given SCC C with the ones derived for the SCCs below C. Consider a set $J \subseteq \mathcal{X} \cup I_{\mathcal{X}} \cup I_{\mathcal{X}}^{[i..j]}$, and a relation $\mathcal{R} \in \texttt{EqRel}(J)$. We define the relation $\mathcal{R}^{++} := \left\{\left\langle x^{++}, y^{++}\right\rangle \mid \langle x, y\rangle \in \mathcal{R}\right\}$. Notice that, if $J \subseteq \mathcal{X} \cup I_{\mathcal{X}}^{[i..j]}$ then $\mathcal{R}^{++} \in \texttt{EqRel}\left(\mathcal{X} \cup I_{\mathcal{X}}^{[i+1..j+1]}\right)$. For relations $\mathcal{R}_1 \in \texttt{EqRel}(\mathcal{X} \cup I_{\mathcal{X}})$ and $\mathcal{R}_2 \in \texttt{EqRel}\left(\mathcal{X} \cup I_{\mathcal{X}}^{[0..i]}\right)$, we define $\mathcal{R}_1 \odot \mathcal{R}_2$ to be the set of equivalence relations $\mathcal{R}_3 \in \texttt{EqRel}\left(\mathcal{X} \cup I_{\mathcal{X}}^{[0..i+1]}\right)$ such that $\mathcal{R}_2^{++} = \mathcal{R}_3\big|_{\mathcal{X} \cup I_{\mathcal{X}}^{[1..i+1]}}$, $\left(\mathcal{R}_3 \cap (I_{\mathcal{X}}^0 \times I_{\mathcal{X}}^0)\right) = \left\{\left\langle \iota_x^0, \iota_y^0\right\rangle \mid \langle \iota_x, \iota_y\rangle \in \mathcal{R}_1\right\}$, $\left(\mathcal{R}_3 \cap (I_{\mathcal{X}}^1 \times I_{\mathcal{X}}^1)\right) = \left\{\left\langle \iota_x^1, \iota_y^1\right\rangle \mid \langle x, y\rangle \in \mathcal{R}_1\right\}$, and $\left(\mathcal{R}_3 \cap (I_{\mathcal{X}}^0 \times I_{\mathcal{X}}^1)\right) = \left\{\left\langle \iota_x^0, \iota_y^1\right\rangle \mid \langle \iota_x, y\rangle \in \mathcal{R}_1\right\}$. For a DSRE ϕ over $I_{\mathcal{X}}$ or over $I_{\mathcal{X}}^{[0..i]}$, we define ϕ^{++} to be the DSRE ϕ' we get by replacing each occurrence of an initiator ι in ϕ with ι^{++}. Notice that ϕ' is defined over $I_{\mathcal{X}}^0$ in the first case, and over $I_{\mathcal{X}}^{[1..i+1]}$ in the second case. For a symbolic configuration $\beta = \langle q, \mathbb{V}, \phi\rangle$, we define $\beta^{++} := \left\langle q, \mathbb{V}^{++}, \phi^{++}\right\rangle$. Consider symbolic configurations $\beta_1 = \langle q_1, \mathbb{V}_1, \phi_1\rangle$ over $I_{\mathcal{X}}$, and $\beta_2 = \langle q_2, \mathbb{V}_2, \phi_2\rangle$ over $I_{\mathcal{X}}^{[0..i]}$. We define $\beta_1 \odot \beta_2$ to be the set of symbolic configurations $\beta_3 = \langle q_3, \mathbb{V}_3, \phi_3\rangle$ where $q_3 = q_2$, $\mathbb{V}_3 \in \mathbb{V}_1 \odot \mathbb{V}_2$, and $\phi_3 = \phi_1^{++} \bullet \phi_2^{++}$. Notice that β_3 is defined over $I_{\mathcal{X}}^{[0..i+1]}$. Intuitively, β_1 characterizes a set of reachable configurations, defined over $I_{\mathcal{X}}$, for a given SCC C, while β_2 characterizes a set of reachable configurations, defined over $I_{\mathcal{X}}^{[0..d]}$, for the SCCs below C. For a state $q \in Q$, define $\texttt{OutOf}(q) := \{t \in \Delta \mid (\texttt{SourceOf}(t) = q) \wedge (\texttt{TargetOf}(t) \notin \texttt{SCCOf}(\mathcal{L}, q))\}$.

Consider a state $q \in Q$. We define a set of symbolic configurations $\mathbb{B} := \texttt{SymConfOf}(\mathcal{L}, \lambda, q)$ where $\mathbb{B}$ over $I_{\mathcal{X}}^{[0..d]}$. We define $\mathbb{B}$ using induction on d as follows. Define $\mathcal{L}_1 := \mathcal{L}\big|_{\texttt{SCCOf}(\mathcal{L}, q)}$, and $\lambda_1 := \lambda\big|_{\texttt{SCCOf}(\mathcal{L}, q)}$. Since $\mathcal{L}_1$ is of type ④ wrt. λ_1, we can, as described above, compute the set of symbolic configurations $\mathbb{B}_1 = \texttt{SymConfOf}(\mathcal{L}_1, \lambda_1, q)$ over $I_{\mathcal{X}}$. Define $\mathbb{B}_2 := \bigcup_{t \in \texttt{OutOf}(q)} \texttt{Post}(t)(\mathbb{B}_1)$. Intuitively, the set $\mathbb{B}_2$ characterizes the set of configurations we get after performing transitions that connect the SCC corresponding to $\mathcal{L}_1$ to the next SCCs in $\texttt{GraphOf}(\mathcal{L})$. We define $\mathbb{B}$ to be the smallest set containing both $\mathbb{B}_1^{++}$ and the following elements. Take any $\beta_2 = \langle q_2, \mathbb{V}_2, \phi_2\rangle \in \mathbb{B}_2$. Define the DLCS $\mathcal{L}_2 := \mathcal{L}\big|_{q_2 \overset{*}{\rightsquigarrow}_{\mathcal{L}}}$. Notice that $d_2 := \texttt{Depth}(\mathcal{L}_2) \leqslant d - 1$. Define $\lambda_2 := \lambda\big|_{q_2 \overset{*}{\rightsquigarrow}_{\mathcal{L}}}$. By the induction hypothesis, we can compute the set $\mathbb{B}_3 := \texttt{SymConfOf}(\mathcal{L}_2, \lambda_2, q_2)$ of symbolic configurations over $I_{\mathcal{X}}^{[0..d_2]}$. For each $\beta_3 \in \mathbb{B}_3$, the set $\mathbb{B}$ contains the set $\beta_2 \odot \beta_3$.

Consider a plain configuration $\gamma \in \texttt{ConfsOf}(\mathcal{L})$ such that $\texttt{StateOf}(\gamma) = q$ and such that γ is consistent wrt. λ. Define the valuation $\theta : I_{\mathcal{X}}^0 \mapsto \mathcal{D}$ such that $\theta(\iota_x^0) := \texttt{VarStateOf}(\gamma)(x)$. By Lemma 9 we get the following lemma.

LEMMA 10. $\texttt{Reach}(\mathcal{L})(\gamma) = [\![\texttt{SymConfOf}(\mathcal{L}, \lambda, q)]\!]_\theta$.

Type ② Consider a DLCS $\mathcal{L} = \langle Q, \mathcal{X}, \mathcal{C}, \Sigma, \Delta\rangle$ of type ②, a state $q \in Q$, and a relation $\mathcal{R} \in \texttt{EqRel}(\mathcal{X})$. Define $\mathcal{L}' := \texttt{Conv}^{②,③}(\mathcal{L})$. Define $\texttt{SymConfOf}(\mathcal{L}, q, \mathcal{R}) := \texttt{SymConfOf}\left(\mathcal{L}', \lambda, \langle q, \mathcal{R}\rangle\right)$ where $\lambda(\langle q, \mathcal{R}\rangle) = \mathcal{R}$. Consider a plain configuration $\gamma \in \texttt{ConfsOf}(\mathcal{L})$ such that $\texttt{StateOf}(\gamma) = q$ and $\texttt{VarStateOf}(\gamma) \models \mathcal{R}$. Define the valuation $\theta : I_{\mathcal{X}}^0 \mapsto \mathcal{D}$ such that $\theta(\iota_x^0) := \texttt{VarStateOf}(\gamma)(x)$. By Lemma 10 and Lemma 5 we get the following lemma.

LEMMA 11. $\texttt{Reach}(\mathcal{L})(\gamma) = [\![\texttt{SymConfOf}(\mathcal{L}, q, \mathcal{R})]\!]_\theta$.

Type ⑥ Consider a DLCS $\mathcal{L} = \langle Q, \mathcal{X}, \mathcal{C}, \Sigma, \Delta\rangle$ of type ⑥ Consider a set of symbolic configurations $\mathbb{B}$ defined over an initiator set I. We define $\texttt{measure}(\mathbb{B}) \in \mathbb{N}$ as follows. For a product $p = e_1 \bullet \cdots \bullet e_n$, we define $\texttt{measure}(p) := n$, i.e., it is the length of p. For a DSRE ϕ, we define $\texttt{measure}(\phi) := \max_{\texttt{ch} \in \mathcal{C}} \texttt{measure}(\phi(\texttt{ch}))$. For a symbolic configuration β, we define $\texttt{measure}(\beta) := \texttt{measure}(\texttt{ChannelStateOf}(\beta))$. Finally we define $\texttt{measure}(\mathbb{B}) := \max_{\beta \in \mathbb{B}} \texttt{measure}(\beta)$. Notice that, for symbolic configurations β_1, β_2, and a transition $t \in \Delta^{rcv}$, if $\beta_2 \in \texttt{Post}(t)(\beta_1)$ then $\texttt{measure}(\beta_2) \leqslant \texttt{measure}(\beta_1)$. Notice also that for each $k \in \mathbb{N}$, there are only finitely many symbolic configurations over I with measure k.

Consider a finite set $\mathbb{B}$ of symbolic configurations over I. Let $\mathbb{B}'$ be the smallest set such that (i) $\mathbb{B} \subseteq \mathbb{B}'$, and (ii) $\mathbb{B}' = \texttt{Post}(\mathbb{B}')$. By the two properties of the function `measure` stated above, it follows that $\mathbb{B}'$ is finite. Consider a valuation $\theta : I_0^{\mathcal{X}} \mapsto \mathcal{D}$. By the definition of `Post` it follows that:

LEMMA 12. $\texttt{Reach}(\mathcal{L})\left([\![\mathbb{B}]\!]_\theta\right) = [\![\mathbb{B}']\!]_\theta$.

Type ① Consider a DLCS $\mathcal{L} = \langle Q, \mathcal{X}, \mathcal{C}, \Sigma, \Delta\rangle$ of type ①, a state $q \in Q$, and a relation $\mathcal{R} \in \texttt{EqRel}(\mathcal{X})$. We show by induction that, for each k, we can derive an $\ell \in \mathbb{N}$ and construct a set of symbolic configurations $\texttt{SymConfOf}(\mathcal{L}, q, \mathcal{R}) := \mathbb{B}$ over $I_{\mathcal{X}}^{[0..\ell]}$ such that, for each plain configuration γ, with $\texttt{StateOf}(\gamma) = q$ and $\texttt{VarStateOf}(\gamma) \models \mathcal{R}$, the following holds. Define the valuation $\theta : I_{\mathcal{X}}^0 \mapsto \mathcal{D}$ such that $\theta(\iota_x^0) := \texttt{VarStateOf}(\gamma)(x)$.

LEMMA 13. $\texttt{Reach}(k)(\mathcal{L})(\gamma) = [\![\texttt{SymConfOf}(\mathcal{L}, q, \mathcal{R})]\!]_\theta$.

Suppose that we have already derived $\mathbb{B}$ for k phases. We will show by induction that we can, for the next phase, derive a new set $\mathbb{B}'$ and $\ell' \in \mathbb{N}$ such that the statement of Lemma 13 is satisfied. If the next phase is a *receive* then we apply the construction of Lemma 12 to derive $\mathbb{B}'$ and define $\ell' := \ell$ (notice that the we do not change the set of initiators in this case.)

Now, we consider the case where the next phase is a *send*. The next phase will then correspond to a DLCS $\mathcal{L}' = \left\langle Q', \mathcal{X}', \mathcal{C}', \Sigma', \Delta'\right\rangle$ of type ②. We define the set $\mathbb{B}'$ to contain all symbolic configurations of the form of β' derived below. Let $\beta_1 = \langle q_1, \mathbb{V}_1, \phi_1\rangle \in \mathbb{B}$. Let $\mathcal{R}_1 := \mathbb{V}_1\big|_{\mathcal{X}}$. Use the construction of Lemma 11 to derive $\mathbb{B}_2 := \texttt{SymConfOf}(\mathcal{L}, q_1, \mathcal{R}_1)$. Suppose that $\mathbb{B}_2$ is defined over the set $I_{\mathcal{X}}^{[0..\ell_2]}$. Select any symbolic configuration $\beta_3 = \langle q_3, \mathbb{V}_3, \phi_3\rangle \in \mathbb{B}_2$. Let $\beta_4 = \langle q_4, \mathbb{V}_4, \phi_4\rangle$ be the symbolic configuration we get from β_3 by replacing each initiator ι_x^i by the initiator $\iota_x^{i+\ell+1}$. Notice that $q_4 = q_3$ and that β_4 is defined over $I_{\mathcal{X}}^{[\ell+1..\ell+\ell_2+1]}$. Define $\beta' := \langle q_5, \mathbb{V}_5, \phi_5\rangle$, where $q_5 = q_4$ and $\phi_5 = \phi_4 \bullet \phi_1$. Furthermore, we select $\mathbb{V}_5 \in \texttt{EqRel}\left(\mathcal{X} \cup I_{\mathcal{X}}^{[0..+\ell+\ell_2+1]}\right)$ to be a relation such that $\mathbb{V}_5\big|_{I_{\mathcal{X}}^{[0..\ell]}} = \mathbb{V}_1$, $\mathbb{V}_5\big|_{\mathcal{X} \cup I_{\mathcal{X}}^{[\ell+1..\ell+\ell_2+1]}} = \mathbb{V}_4$, and $\mathbb{V}_5 \cap (I_{\mathcal{X}}^j \times I_{\mathcal{X}}^{[\ell+1]}) = \left\{\left\langle \iota_x^j, \iota_y^{\ell+1}\right\rangle \mid \left\langle \iota_x^j, y\right\rangle \in \mathbb{V}_1\right\}$, for all $j : 0 \leqslant j \leqslant \ell$. This gives the following theorem.

THEOREM 14. *Consider a* DLCS $\mathcal{L} = \langle Q, \mathcal{X}, \mathcal{C}, \Sigma, \Delta \rangle$*, a state* $q \in Q$*, and a relation* $\mathcal{R} \in \texttt{EqRel}(\mathcal{X})$*. For any* $k \in \mathbb{N}$*, we can derive an* $\ell \in \mathbb{N}$ *and a finite set* $\mathbb{B}$ *of symbolic configurations over* $I_{\mathcal{X}}^{[0..\ell]}$ *such that for any plain configuration* $\gamma \in \texttt{ConfsOf}(\mathcal{L})$ *such that* $\texttt{StateOf}(\gamma) = q$ *and* $\texttt{VarStateOf}(\gamma) \models \mathcal{R}$*, we have* $\texttt{Reach}(k)(\gamma)(q) = [\![\mathbb{B}]\!]_\theta$*, where* $\theta : I_{\mathcal{X}}^0 \mapsto \mathcal{D}$ *is such that* $\theta(\iota_x^0) = V(x)$ *for all* $x \in \mathcal{X}$*.*

From Theorem 14 we get decidability of the k-bounded-phase reachability problem. More precisely, given a DLCS $\mathcal{L} = \langle Q, \mathcal{X}, \mathcal{C}, \Sigma, \Delta \rangle$, a plain configuration γ, and a state $\mathsf{Target} \in Q$ we proceed as follows. Let $q = \texttt{StateOf}(\gamma)$ and let $\mathcal{R} \in \texttt{EqRel}(\mathcal{X})$ be the unique relation such that $\texttt{VarStateOf}(\gamma) \models \mathcal{R}$. Derive a set $\mathbb{B}$ of symbolic configurations as described in Theorem 14, and check whether the state of any member of $\mathbb{B}$ is equal to Target.

THEOREM 15. *The bounded-phase reachability problem is decidable.*

6. Conclusions and Future Work

We have presented a model, called DLCS, that generalizes lossy channel systems by augmenting the model with a finite set of variables and channel messages that carry values from an infinite data domain. We have shown undecidability of the reachability problem, and decidability of the bounded phase reachability problem. The undecidability result can be strengthened in several ways. For instance, we can show undecidability for DLCS that are restricted to have a single channel, by concatenating the encoding of the two counters in the given channel. We can also remove the endmarkers used in the simulation thus obtaining undecidability for the case where the channel alphabet is a singleton. We use five variables in our undecidability proof, and leave open the case where we restrict the DLCS to have a fewer number of variables (the case of a single variable can be easily reduced to the case of LCS without data.) Another direction for future work is to consider bounded-phase reachability under a richer set of relations on the data domain, e.g., by allowing gap-order constraints [5] on the set of variables instead of only equalities and disequalities. Finally we intend to study context-bounded analysis [23] as another way to achieve decidability results for DLCS.

References

[1] P. A. Abdulla and B. Jonsson. Verifying programs with unreliable channels. *Inf. Comput.*, 127(2):91–101, 1996.

[2] P. A. Abdulla, M. F. Atig, and J. Cederberg. Timed lossy channel systems. In *Proc. FSTTCS '12, 32nd Conf. on Foundations of Software Technology and Theoretical Computer Science*, 2012.

[3] P. A. Abdulla, M. F. Atig, and J. Stenman. Dense-timed pushdown automata. In *LICS*. IEEE Computer Society, 2012.

[4] P. A. Abdulla, M. F. Atig, and J. Cederberg. Analysis of message passing programs using SMT-solvers. In *ATVA 2013*, volume 8172 of *LNCS*, pages 272–286, 2013.

[5] P. A. Abdulla, M. F. Atig, G. Delzanno, and A. Podelski. Push-down automata with gap-order constraints. In *FSEN 2013*, volume 8161 of *LNCS*, pages 199–216, 2013.

[6] P. A. Abdulla, M. F. Atig, A. Kara, and O. Rezine. Verification of dynamic register automata. In *FSTTCS*, volume 29 of *LIPIcs*, pages 653–665. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2014.

[7] C. Aiswarya, P. Gastin, and K. Narayan Kumar. Controllers for the verification of communicating multi-pushdown systems. In *CONCUR*, volume 8704 of *LNCS*, pages 297–311, 2014.

[8] S. Akshay, B. Bollig, and P. Gastin. Automata and logics for timed message sequence charts. In *FSTTCS 2007*, volume 4855 of *LNCS*, pages 290–302. Springer, 2007.

[9] M. Bojanczyk, C. David, A. Muscholl, T. Schwentick, and L. Segoufin. Two-variable logic on data words. *ACM Trans. Comput. Log.*, 12(4):27, 2011.

[10] B. Bollig and L. Hélouët. Realizability of dynamic MSC languages. In *CSR'10*, volume 6072 of *LNCS*, pages 48–59, Kazan, Russia, 2010. Springer.

[11] B. Bollig, A. Cyriac, P. Gastin, and K. Narayan Kumar. Model checking languages of data words. In *FoSSaCS'12*, volume 7213 of *LNCS*, pages 391–405. Springer, Mar. 2012.

[12] B. Bollig, A. Cyriac, L. Hélouët, A. Kara, and Th. Schwentick. Dynamic communicating automata and branching high-level MSCs. In *(LATA'13)*, volume 7810 of *LNCS*, pages 177–189, Bilbao, Spain, 2013. Springer.

[13] A. Bouajjani and M. Emmi. Bounded phase analysis of message-passing programs. In *TACAS*, volume 7214 of *LNCS*, pages 451–465, 2012.

[14] A. Bouajjani, R. Echahed, and R. Robbana. On the automatic verification of systems with continuous variables and unbounded discrete data structures. In *Hybrid Systems II*, volume 999 of *LNCS*, pages 64–85. Springer, 1994.

[15] L. Clemente and S. Lasota. Timed pushdown automata revisited. In *LICS 2015*, pages 738–749. IEEE, 2015.

[16] L. Clemente, F. Herbreteau, A. Stainer, and G. Sutre. Reachability of communicating timed processes. In *FOSSACS 2013*, volume 7794 of *LNCS*, pages 81–96. Springer, 2013.

[17] S. Demri and R. Lazić. Ltl with the freeze quantifier and register automata. *ACM Trans. Comput. Logic*, 10(3):16:1–16:30, 2009.

[18] A. Finkel. Decidability of the termination problem for completely specified protocols. *Distributed Computing*, 7(3):129–135, 1994. ISSN 0178-2770.

[19] A. Finkel and Ph. Schnoebelen. Well-structured transition systems everywhere! *Theoretical Computer Science*, 256(1-2):63–92, Apr. 2001.

[20] O. Grumberg, O. Kupferman, and S. Sheinvald. Variable automata over infinite alphabets. In *LATA*, volume 6031 of *LNCS*, pages 561–572, 2010.

[21] M. Kaminski and N. Francez. Finite-memory automata. *Theoretical Computer Science*, 134(2):329–363, 1994.

[22] P. Krcál and W. Yi. Communicating timed automata: The more synchronous, the more difficult to verify. In *CAV*, volume 4144 of *Lecture Notes in Computer Science*, pages 249–262. Springer, 2006.

[23] S. La Torre, P. Madhusudan, and G. Parlato. Context-bounded analysis of concurrent queue systems. In *TACAS*, volume 4963 of *LNCS*, pages 299–314. Springer, 2008.

[24] R. Lazic, T. Newcomb, J. Ouaknine, A. W. Roscoe, and J. Worrell. Nets with tokens which carry data. *Fundam. Inform.*, 88(3):251–274, 2008.

[25] M. Musuvathi and S. Qadeer. Iterative context bounding for systematic testing of multithreaded programs. In *PLDI*, pages 446–455. ACM, 2007.

[26] F. Neven, T. Schwentick, and V. Vianu. Finite state machines for strings over infinite alphabets. *ACM Trans. Comput. Logic*, 5(3):403–435, July 2004. ISSN 1529-3785.

[27] S. Qadeer and J. Rehof. Context-bounded model checking of concurrent software. In *TACAS*, volume 3440 of *LNCS*, pages 93–107. Springer, 2005.

[28] F. Rosa-Velardo and D. de Frutos-Escrig. Decidability and complexity of petri nets with unordered data. *Theor. Comput. Sci.*, 412(34):4439–4451, 2011.

[29] N. Tzevelekos. Fresh-register automata. *SIGPLAN Not.*, 46(1):295–306, Jan. 2011.