Unidirectional Channel Systems Can Be Tested

Petr Jančar, Prateek Karandikar, Philippe Schnoebelen

Unidirectional channel systems can be tested

P. Jančar[1]⋆, P. Karandikar[2]⋆⋆, and Ph. Schnoebelen[3]⋆⋆⋆

[1] Techn. Univ. Ostrava
[2] Chennai Mathematical Institute
[3] LSV, ENS Cachan, CNRS

Abstract. "Unidirectional channel systems" (Chambart & Schnoebelen, CONCUR 2008) are systems where one-way communication from a sender to a receiver goes via one reliable and one unreliable (unbounded fifo) channel. Equipping these systems with the possibility of testing regular properties on the contents of channels makes verification undecidable. Decidability is preserved when only emptiness and nonemptiness tests are considered: the proof relies on a series of reductions eventually allowing us to take advantage of recent results on Post's Embedding Problem.

1 Introduction

Channel systems are a family of computational models where several, usually finite-state, agents communicate via usually unbounded fifo communication channels [1]. These models are well-suited to the formal specification and algorithmic analysis of asynchronous communication protocols [2–5]. They are sometimes called *queue automata* when there is only one agent using the channels as fifo memory buffers.

A particularly interesting class of channel systems are the *lossy channel systems*, "LCS" for short, popularized by Abdulla, Bouajjani, Jonsson, Finkel, *et al.* [6–8]. Lossy channels are unreliable and can lose messages nondeterministically and without any notification. A bit surprisingly, this makes lossy systems easier to analyse: safety, inevitability and several more properties are decidable for this model [6, 7, 9–11] while they are undecidable when channels are reliable.

It should be stressed that LCS's have also been very useful outside the field of communicating systems and distributed computing. During the last decade, they have been used to show the decidability, or (more often) the hardness, of problems on Timed Automata, Metric Temporal Logic, modal logics, etc. [12–16]. With other unreliable computational models, lossy channel systems are now an important tool for the complexity analysis of algorithms that rely on well-quasi-ordering theory [17–19].

Unidirectional channel systems, "UCS" for short, are a variant of LCS's where a Sender process communicates to a Receiver process via one reliable and one lossy channel. Fig. 1 gives an example.

⋆ Supported by the European Regional Development Fund in the IT4Innovations Centre of Excellence project (CZ.1.05/1.1.00/02.0070), and by the project GAČR:P202/11/0340.
⋆⋆ Partially funded by Tata Consultancy Services.
⋆⋆⋆ Currently visiting Oxford Univ. Comp. Sci. Dept, supported by Grant ANR-11-BS02-001 and by the Leverhulme Trust.

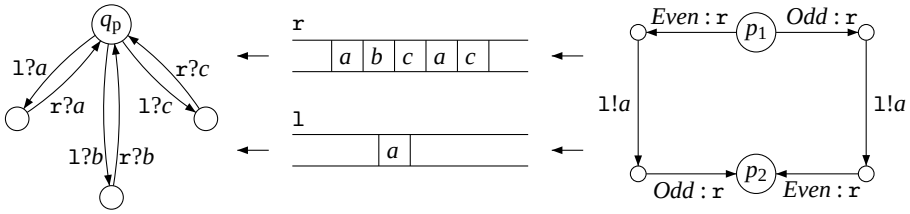

Fig. 1. Unidirectional channels: Sender on the right, Receiver on the left

The presence of one reliable channel put UCS's beyond plain LCS's. On the other hand, the unidirectionality (there is no channel from Receiver to Sender) is a limitation that LCS's do not share.

UCS's were first studied by Chambart and Schnoebelen who considered *mixed* channel systems (i.e., communicating systems using both reliable and lossy channels in arbitrary combinations) and showed how to reduce safety and reachability problems for arbitrary network topologies to reachability problems on either queue automata (undecidable), or LCS's (decidable), or the previously unidentified UCS's [20].

The reachability problem for UCS's is quite challenging: it was proved decidable by reformulating it more abstractly as PEP, aka the *Regular Post Embedding Problem*, which is easier to analyze [21–23]. We want to stress that, while PEP is a natural variant of Post's Correspondence Problem, it was only identified through questions on UCS's. Recently, PEP has proved useful in other areas, starting with Graph Logics [24].

Testing channel contents. Basic channel machines are not allowed to inspect the contents of the channels. However, it is natural to enrich the basic setup with tests (conditions on channel contents) as in, e.g., Fig. 1 where some sender's actions depend on (the parity of) the number of messages currently in r. Adding tests goes smoothly and painlessly for LCS's where the main decidability results extend directly with almost unchanged algorithms [10, sect. 3.3]. Adding even simple tests to UCS's is a completely different story, as we discovered. One meets two obstacles when trying to extend the approach that worked for UCS's:

1. The "reformulation" of UCS reachability as a Post Embedding Problem is a nontrivial reduction that *reorders* the events in a UCS run, relying on the independence (in a concurrency-theoretical sense) of sendings wrt readings. Tests on channel contents introduce global dependencies that are not reflected in PEP problems.
2. One is then led to consider *extensions* of PEP where said dependencies can be reflected, raising a new question: how to show the decidability of these extensions?

Our contribution. We extend UCS's with the possibility of testing channel contents with simple regular predicates. This makes reachability undecidable even with restricted sets of simple tests. Our main result is that reachability remains decidable when only emptiness and non-emptiness tests are allowed. The proof goes through a series of reductions that leave us with UCS's extended by only emptiness tests on a single side of a single channel (called "Z_1^1 tests"). This minimal extension can then be reformulated as

$\mathsf{PEP}^{\text{partial}}_{\text{codir}}$, or "PEP with partial codirectness", a nontrivial extension of PEP that was recently proved decidable [25].

Outline of the paper. Unidirectional channel systems with tests are defined in Section 2. Section 3 shows how undecidability creeps in when regular tests are allowed. Section 4 presents protocols for simulating (non-)emptiness in UCST's with only emptiness tests by Sender, thus reducing UCST to UCST$[Z_1]$. Section 5 proves decidability for UCST$[Z_1^1]$ by reducing to $\mathsf{PEP}^{\text{partial}}_{\text{codir}}$. This is then leveraged in section 6 to account for the whole of UCST$[Z_1]$. Finally, Section 7 proves that (non-)emptiness tests strictly enrich the basic UCS model.

2 Unidirectional Channel Systems

Structure. Formally, a *UCST* (for *Unidirectional Channel System with Tests*) is a tuple $S = (\mathtt{Ch}, \mathtt{M}, Q_1, \Delta_1, Q_2, \Delta_2)$, where $\mathtt{M}$ is a finite alphabet of *messages*, Q_1, Q_2 are disjoint finite sets of *states* of Sender and Receiver, respectively, and Δ_1, Δ_2 are finite sets of *rules* of Sender and Receiver, respectively. $\mathtt{Ch} = \{\mathtt{r}, \mathtt{l}\}$ is a fixed set of channel (names), $\mathtt{r}$ being reliable and $\mathtt{l}$ unreliable, so called "lossy".

A rule $\delta \in \Delta_i$ is a tuple $(q, \mathtt{c}, \alpha, q') \in Q_i \times \mathtt{Ch} \times Act \times Q_i$ where the set of actions Act contains *tests* R (checking whether the contents of $\mathtt{c} \in \mathtt{Ch}$ belongs to R, a regular language) and *communications* w (sending a sequence of messages to $\mathtt{c}$ in the case of Sender's actions, reading it for Receiver's) and is thus given by $Act \stackrel{\text{def}}{=} \mathsf{Reg}(\mathtt{M}) \cup \mathtt{M}^*$.

We write $q \xrightarrow{R:\mathtt{c}} q'$ for a rule where the action is a test on $\mathtt{c}$, and $q \xrightarrow{\mathtt{c}!w} q'$ (resp., $q \xrightarrow{\mathtt{c}?w} q'$) when the action is a communication by Sender (resp., by Receiver).

In graphical representations like Fig. 1, Sender and Receiver are depicted as two disjoint directed graphs, where states appear as nodes and where rules $q \xrightarrow{\alpha} q'$ appear as edges from q to q' with action and channel name labeling the edge. We may omit the label, or just use $\top$, for trivial tests, $R = \mathtt{M}^*$, or empty communications, $w = \varepsilon$.

Remark 2.1 (On separating tests from communications). Our definition requires that an action is a test *or* a communication. It does not allow performing both atomically inside a single step (but they can be chained using intermediary states). This choice, which is no real loss of generality, lets us focus on simulating tests by other constructs (or other tests) without having to account for accompanying communications. □

Operational Semantics. The behaviour of S is defined via an operational semantics defined along standard lines. A *configuration* of $S = (\mathtt{Ch}, \mathtt{M}, Q_1, \Delta_1, Q_2, \Delta_2)$ is a tuple $C \in Conf_S \stackrel{\text{def}}{=} Q_1 \times Q_2 \times \mathtt{M}^* \times \mathtt{M}^*$. In $C = (q_1, q_2, u, v)$, q_1 and q_2 are the current states of, respectively, Sender and Receiver, while u and v are the current contents of, respectively, $\mathtt{r}$ and $\mathtt{l}$.

Rules give rise to transitions in the expected way. We start with so-called "reliable" steps where the effect of a rule is deterministic. Formally, given two configurations $C = (q_1, q_2, u, v)$, $C' = (q_1', q_2', u', v')$ and a rule $\delta = (q, \mathtt{c}, \alpha, q')$, there is a reliable step denoted $C \xrightarrow{\delta}_{\text{rel}} C'$ if, and only if, the following four conditions are satisfied:

states: $q = q_1$ and $q' = q'_1$ and $q_2 = q'_2$ (for Sender rules), or $q = q_2$ and $q' = q'_2$ and $q_1 = q'_1$ (for Receiver rules);

tests: if δ is a test rule $q \xrightarrow{R:\mathtt{c}} q'$, then $\mathtt{c} = \mathtt{r}$ and $u \in R$, or $\mathtt{c} = \mathtt{l}$ and $v \in R$, and furthermore $u' = u$ and $v' = v$;

writes: if δ is a writing rule $q \xrightarrow{\mathtt{c}!w} q'$, then $\mathtt{c} = \mathtt{r}$ and $u' = uw$ and $v' = v$, or $\mathtt{c} = \mathtt{l}$ and $u' = u$ and $v' = vw$;

reads: if δ is a reading rule $q \xrightarrow{\mathtt{c}?w} q'$, then $\mathtt{c} = \mathtt{r}$ and $u = wu'$ and $v' = v$, or $\mathtt{c} = \mathtt{l}$ and $u' = u$ and $v = wv'$.

Now to unreliable, aka lossy, steps denoted $C \xrightarrow{\delta}_{\text{los}} C'$. As is standard, a lossy step is defined as a combination of message losses (where the contents of $\mathtt{l}$ may be replaced with a subword) with a reliable step. For $v_1, v_2 \in \mathtt{M}^*$, we write $v_1 \sqsubseteq v_2$ when v_1 is a subword of v_2, i.e., a (scattered) subsequence. In particular, $\varepsilon \sqsubseteq v_2$ and $v_2 \sqsubseteq v_2$ for any v_2. This is extended to configurations and we write $C \sqsubseteq D$ when $C = (q_1, q_2, u, v)$ and $D = (q_1, q_2, u, v')$ with $v \sqsubseteq v'$.[4] We now define:

$$C \xrightarrow{\delta}_{\text{los}} C' \overset{\text{def}}{\Leftrightarrow} \exists D, D' : C \sqsupseteq D \wedge D \xrightarrow{\delta}_{\text{rel}} D' \wedge D' \sqsupseteq C' \,. \tag{1}$$

In other words, a lossy step is a reliable step sandwiched between arbitrary message losses on $\mathtt{l}$. In particular, reliable steps are a special case of lossy steps. In the rest of this paper, we consider reachability via lossy steps, and often write simply $C \xrightarrow{\delta} C'$ without a "los" subscript. (When we refer to reliable steps and runs, we always use "rel" subscript.)

Remark 2.2 (On reliable steps). As is usual with lossy channel systems, the reliable semantics plays a key role even though the object of our study is reachability via unreliable steps. First $\to_{\text{rel}}$ is a normative yardstick from which the unreliable semantics depart: $\to_{\text{los}}$ is defined as a modification of $\to_{\text{rel}}$. Then many hardness results on lossy systems are proved with reductions where a lossy system simulates in some way the reliable (and Turing-powerful) behaviour. □

A *run* from C_0 to C_n is a sequence of chained steps $C_0 \xrightarrow{\delta_1} C_1 \xrightarrow{\delta_2} C_2 \cdots \xrightarrow{\delta_n} C_n$, abbreviated as $C_0 \xrightarrow{*} C_n$ (or $C_0 \xrightarrow{+} C_n$ when we rule out zero-length runs).

Definition 2.3. *The* Reachability Problem *is the question, given a UCST S and some states $p_{\text{in}}, p_{\text{fi}} \in Q_1$, $q_{\text{in}}, q_{\text{fi}} \in Q_2$, whether S has a (lossy) run $C_{\text{in}} = (p_{\text{in}}, q_{\text{in}}, \varepsilon, \varepsilon) \xrightarrow{*} C_{\text{fi}} = (p_{\text{fi}}, q_{\text{fi}}, \varepsilon, \varepsilon)$.*

The Extended Reachability Problem *asks, further given regular languages $U, V, U', V' \subseteq \mathtt{M}^*$, whether there exist $u \in U$, $v \in V$, $u' \in U'$, and $v' \in V'$ such that S has a (lossy) run $(p_{\text{in}}, q_{\text{in}}, u, v) \xrightarrow{*} (p_{\text{fi}}, q_{\text{fi}}, u', v')$.*

In the following we only consider reachability problems with empty channels in C_{in} and C_{fi} since this is technically convenient. There is no loss of generality:

Lemma 2.4. *The extended reachability problem many-one reduces to the reachability problem.*

[4] Note that $(Conf, \sqsubseteq)$ is *not* a well-quasi-order since $C \sqsubseteq D$ requires equality on channel $\mathtt{r}$.

Roughly speaking, we can transform an instance of the extended reachability problem to an "empty-channel" instance by letting Sender start with generating some $u \in U$, $v \in V$ into the channels, and by letting Receiver read some $u' \in U'$, $v' \in V'$ in the end.

The two problems are thus equivalent. Moreover, the reduction does not need to introduce any new tests, and the equivalence thus also holds for UCST's with restricted sets of tests which we will consider.

3 Testing channels and the undecidability of reachability

Despite their similarities, UCS's and LCS's (lossy channel systems) behave differently. The algorithms deciding reachability for LCS's can easily accommodate regular (or even more expressive) tests [10, Sect. 3.3]. By contrast, this section gives several versions of the following result:

Theorem 3.1. *Reachability is undecidable for UCST.*

3.1 Simulating queue automata

We now show how even simple tests lead to undecidability. The main technique we use is to simulate queue automata which are a Turing-powerful model already with a single reliable channel.

UCS's already have a reliable channel but Sender (or Receiver) cannot both read *and* write from/to it. If Sender could somehow read from the head of `r` as well as write to its tail, it would be as powerful as a queue automaton. Now, with regular tests on channels, there exists a simple protocol making Receiver act as a proxy for Sender and implement read actions on its behalf.

Described informally, the protocol is the following[5]:

1. Channel `l` is initially empty.
2. In order to "read" from `r`, Sender checks and records whether the length of the contents of `r` is odd or even, using a regular test on `r`.
3. It then sends on `l` the message, say a, that it wants to read.
4. It checks that (equivalently, waits until) the parity of the contents of `r` has changed, and on detecting this change, concludes that the read was successful.
5. Receiver waits in its initial q_{proxy} (or q_p) state and tries to read from `l`. When it reads a message a from `l`, it understands it as an request telling it to read a from `r` on behalf of Sender. Once it has performed this read on `r`, it returns to q_{proxy} and waits for the next instruction.
6. `l` is now empty and the simulation of a read by Sender is concluded.

If no messages are lost on `l`, the protocol allows Sender to read on `r`. If a message is lost on `l`, the protocol deadlocks. Also, Sender deadlocks if it attempts to read a message that is not at the head of `r` (it has to guess correctly). We note that these deadlocks do not make the simulation incorrect since we are only concerned with reachability.

[5] We describe the protocol informally but Fig. 1 page 2 depicts exactly how Receiver implements a proxy on $\mathtt{M} = \{a,b,c\}$ and how Sender simulates a rule $p_1 \xrightarrow{\mathtt{r}?a} p_2$ for a queue automaton.

3.2 Restricted sets of tests

In the above reduction only parity tests were used. When $T \subseteq \mathsf{Reg}(\mathtt{M})$, we write UCST[$T$] to denote the class of UCST's where only tests belonging to T are allowed. Thus UCST and UCS coincide with UCST[$\mathsf{Reg}(\mathtt{M})$] and UCST[$\varnothing$], respectively.

More interestingly, defining $\mathit{Odd}, \mathit{Even} \in \mathsf{Reg}(\mathtt{M})$ with $\mathit{Even} \stackrel{\text{def}}{=} (\mathtt{M}.\mathtt{M})^*$ and $\mathit{Odd} \stackrel{\text{def}}{=} \mathtt{M}.\mathit{Even}$, and letting $P \stackrel{\text{def}}{=} \{\mathit{Even}, \mathit{Odd}\}$ denote the parity tests, section 3.1 shows that reachability is undecidable already for UCST[P].

We further observe that in Fig. 1 only the sender uses tests, and only $\mathtt{r}$ is submitted to tests. We denote such restricted uses of tests by qualifying test sets like T with a subscript 1 (for Sender) or 2 (for Receiver), and/or by a superscript $\mathtt{r}$ or $\mathtt{l}$. We can now state the following stronger form of Theorem 3.1:

Theorem 3.2. *Reachability is undecidable for UCST[$P_1^{\mathtt{r}}$].*

In the rest of this paper, we single out other simple test sets by letting:

$$Z \stackrel{\text{def}}{=} \{\varepsilon\}, \qquad N \stackrel{\text{def}}{=} \mathtt{M}^+, \qquad H_a \stackrel{\text{def}}{=} a.\mathtt{M}^*, \qquad H \stackrel{\text{def}}{=} \{H_x \mid x \in \mathtt{M}\}.$$

In other words, Z is the *emptiness* (or "zero") test, N is the *non-emptiness* test and H are the *head* tests (that allow checking what is the first message in a channel without consuming it). Note that non-emptiness tests can be simulated with head tests, hence are weaker. Below we abuse notation and, when $R, R' \in \mathsf{Reg}(\mathtt{M}^*)$, we write UCST[$R$] and UCST[$R, R'$] rather than UCST[$\{R\}$] and UCST[$\{R, R'\}$].

One difference with parity tests and the Z, N, H tests is that parity tests are "global" in that their outcome depends on the entire contents of a channel. With H tests, only one message at the head needs be scanned. Still, "local" H tests are sufficient for undecidability:

Theorem 3.3. *Reachability is undecidable for UCST[$H_1^{\mathtt{r}}$].*

Proof (Idea). Sender can simulate parity tests $P_1^{\mathtt{r}}$ by using two copies of the message alphabet, say using different colors. It alternates strictly between the two colors when writing on $\mathtt{r}$. This requires an extra bit of memory, encoded in local states. Then the parity of the length of $\mathtt{r}$ contents can be tested by looking at the first message, using $H_1^{\mathtt{r}}$ tests, and comparing with the color of the last written message. (This assumes that $\mathtt{r}$ is never completely emptied, otherwise deadlocks will occur, but this is no loss of generality.) □

4 Simulating UCST[Z, N] by using Sender's emptiness tests only

This section describes two simulations that, put together, entail Theorem 4.1.
Remark. The simulations are tailored to the reachability problem. They may not preserve, e.g., termination or deadlock-freedom.

Theorem 4.1. *Reachability for UCST[Z,N] many-one reduces to reachability for UCST[Z_1].*

4.1 Reducing UCST[Z,N] to UCST[Z_1,N_1]

We now explain how to eliminate Z and N tests by Receiver. W.l.o.g. we assume that x in $\mathtt{c}!x$ and $\mathtt{c}?x$ is always one symbol ($x \in \mathtt{M}$), and we use two special new messages, "$\mathtt{z}$" and "$\mathtt{n}$", with which Sender will signal to Receiver about the status, empty or not, of the channels.

Formally, for $S \in$UCST[Z,N], where $S = (\{\mathtt{r},\mathtt{l}\},\mathtt{M},Q_1,\Delta_1,Q_2,\Delta_2)$, we construct S' arising from S as follows (see Fig. 2):

- S' uses the special new messages $\mathtt{z},\mathtt{n}$, and it thus has alphabet $\mathtt{M}' \stackrel{\text{def}}{=} \mathtt{M} \cup \{\mathtt{n},\mathtt{z}\}$;
- for each channel $\mathtt{c} \in \{\mathtt{r},\mathtt{l}\}$ and each sender state $p \in Q_1$ we add new states $p^1_\mathtt{c}$, $p^2_\mathtt{c}$ and an "*(emptiness) testing loop*" $p \xrightarrow{Z:\mathtt{c}} p^1_\mathtt{c} \xrightarrow{\mathtt{c}!\mathtt{z}} p^2_\mathtt{c} \xrightarrow{Z:\mathtt{c}} p$;
- for every sender rule θ of the form $p \xrightarrow{\mathtt{c}!x} p'$ we add a new state p_θ, and the rule is replaced in S' by the following three rules: $p \xrightarrow{\top} p_\theta$, $p_\theta \xrightarrow{\mathtt{c}!\mathtt{n}} p_\theta$ (a "*padding loop*"), and $p_\theta \xrightarrow{\mathtt{c}!x} p'$;
- every receiver rule $q \xrightarrow{Z:\mathtt{c}} q'$ testing emptiness of $\mathtt{c}$ is replaced by $q \xrightarrow{\mathtt{c}?\mathtt{z}} q'$;
- every receiver rule $q \xrightarrow{N:\mathtt{c}} q'$ testing non-emptiness of $\mathtt{c}$ is replaced by $q \xrightarrow{\mathtt{c}?\mathtt{n}} q'$.

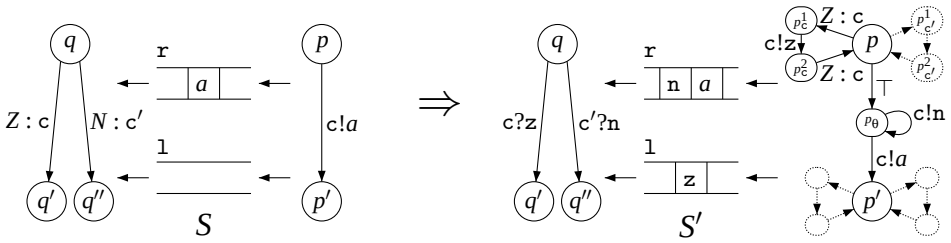

Fig. 2. From S to S': eliminating Receiver's N and Z tests

Lemma 4.2 (Correctness of the reduction). *S has a run $C_{\text{in}} \xrightarrow{*}_{\text{los}} C_{\text{fi}}$ if, and only if, S' has a run $C_{\text{in}} \xrightarrow{*}_{\text{los}} C_{\text{fi}}$.*

Proof (Sketch). The "$\Rightarrow$" direction. Suppose a run $C_{\text{in}} \xrightarrow{*}_{\text{los}} C_{\text{fi}}$ of S. For each concrete occurrence o of a message $x \in \mathtt{M}$ which is written to a channel $\mathtt{c}$ there is a number k_o such that the run uses k_o steps where Receiver tests $\mathtt{c}$ for non-emptiness (i.e., performs transitions $q \xrightarrow{N:\mathtt{c}} q'$) in the situation when o is the first symbol (the head) in $\mathtt{c}$. We can use this for constructing a run of S' which mimics the above run of S. Any original Sender's step $p \xrightarrow{\mathtt{c}!x} p'$, writing an occurrence o of x to $\mathtt{c}$, is replaced by $p \xrightarrow{\top} p_\theta \xrightarrow{\mathtt{c}!\mathtt{n}} p_\theta \xrightarrow{\mathtt{c}!\mathtt{n}} \cdots \xrightarrow{\mathtt{c}!\mathtt{n}} p_\theta \xrightarrow{\mathtt{c}!x} p'$ where the padding loop is used k_o times; any Receiver's step $q \xrightarrow{N:\mathtt{c}} q'$ is replaced by $q \xrightarrow{\mathtt{c}?\mathtt{n}} q'$. Any original Receiver's step $q \xrightarrow{Z:\mathtt{c}} q'$, when Sender is in state p, is replaced with the sequence of steps corresponding to $p \xrightarrow{Z:\mathtt{c}} p^1_\mathtt{c} \xrightarrow{\mathtt{c}!\mathtt{z}} p^2_\mathtt{c}$, $q \xrightarrow{\mathtt{c}?\mathtt{z}} q'$,

$p^2_{\mathtt{c}} \xrightarrow{Z:\mathtt{c}} p$. The inserted `n`'s (and `z`'s) are never lost; other message losses are the same as originally.

The "$\Leftarrow$" direction. Suppose a run $C_{\mathrm{in}} \xrightarrow{*}_{\mathrm{los}} C_{\mathrm{fi}}$ of S'. It is convenient to consider the run as a sequence of *fine-grained steps*, i.e., $C_{\mathrm{in}} \xrightarrow{\delta_1} C_1 \xrightarrow{\delta_2} C_2 \xrightarrow{\delta_3} \cdots C_{n-1} \xrightarrow{\delta_n} C_{\mathrm{fi}}$, where each step is either a reliable step $C_{i-1} \xrightarrow{\delta_i}_{\mathrm{rel}} C_i$ or the loss of a single message. The idea is to repeatedly switch two consecutive steps conveniently so that the validity of the obtained (fine-grained) runs is kept, with the aim to achieve a "*straight run*" which can be easily translated to a run $C_{\mathrm{in}} \xrightarrow{*}_{\mathrm{los}} C_{\mathrm{fi}}$ of S. Imagine first that we give the priority to Sender's (reliable) steps: whenever some (current) δ_i is a Receiver's step or a loss and δ_{i+1} is a Sender's step then we switch the steps if the result is still a valid run. It is easy to observe that the writing steps in the resulting run (in which no above switches are possible) are only in the segments corresponding to $p \xrightarrow{\top} p_\theta \xrightarrow{\mathtt{c!n}} p_\theta \xrightarrow{\mathtt{c!n}} \cdots \xrightarrow{\mathtt{c!n}} p_\theta \xrightarrow{\mathtt{c!x}} p'$ (uninterrupted by Receiver or message losses). Regarding the steps corresponding to testing loops, we get segments $p \xrightarrow{Z:\mathtt{c}} p^1_{\mathtt{c}} \xrightarrow{\mathtt{c!z}} p^2_{\mathtt{c}}, \sigma, p^2_{\mathtt{c}} \xrightarrow{Z:\mathtt{c}} p$ where σ is a sequence of Receiver's steps and/or message losses. Now we can switch (anyhow) inside such a segment, with the aim to get $\sigma_1, p \xrightarrow{Z:\mathtt{c}} p^1_{\mathtt{c}} \xrightarrow{\mathtt{c!z}} p^2_{\mathtt{c}}, \sigma_2, p^2_{\mathtt{c}} \xrightarrow{Z:\mathtt{c}} p$ for a shortest σ_2. It turns out that σ_2 is, in fact, one step, either $q \xrightarrow{\mathtt{c?z}} q'$ or a loss of `z`. There is a final issue: we arrange that the finally achieved run is also "*head-lossy*", i.e. any loss-step loses the first message (the head) of `l`; thus we never have only `n`'s in `l` when Sender tests the non-emptiness of `l`. It is then straightforward to translate the finally achieved run of S' to the corresponding run of S. □

4.2 Reducing UCST[Z_1, N_1] to UCST[Z_1]

When there are no receiver tests, N_1 tests can be eliminated by a buffering technique on Sender's side. With any $S \in$UCST$[Z_1, N_1]$ we associate a derived system S' as follows:

For each channel $\mathtt{c} \in \mathtt{Ch}$, S' uses an auxiliary 1-place buffer between Sender and the channel `c`. In any S' configuration, a buffer is empty (containing no messages) or full (containing a single message). Now the sender does not write to the channels, it can only directly write to the auxiliary buffers, and it may only write to a buffer when it is empty, making it full. Buffers may be nondeterministically flushed at any time, transferring their contents to the actual channel (in a potentially lossy way for `l`). Finally, the buffers are not actual extra peripherals, rather they are encoded in the finite control of Sender, which also simulates the lossy behavior of writing to channel `l`. Within this setup, an $N^{\mathtt{c}}_1$ test translates to "`c`'s auxiliary buffer is full", and a $Z^{\mathtt{c}}_1$ test translates to "`c`'s buffer is empty *and* `c` is empty".

Finally, $S' \in$UCST$[Z_1]$ simulates S without any need of N_1 tests, as stated by the following lemma.

Lemma 4.3 (Correctness of the reduction). *S has a run $C_{\mathrm{in}} \xrightarrow{*}_{\mathrm{los}} C_{\mathrm{fi}}$ if, and only if, S' has a run $C'_{\mathrm{in}} \xrightarrow{*}_{\mathrm{los}} C'_{\mathrm{fi}}$ (where C'_{in} and C'_{fi} are the configurations in S' corresponding to C_{in} and C_{fi} with empty auxiliary buffers).*

5 Reachability for UCST[Z_1^1] via Post's Embedding Problem

This section develops a many-one reduction from the reachability problem for UCST[Z_1^1] to $\mathsf{PEP}_{\text{codir}}^{\text{partial}}$, a generalization of Post's Embedding Problem.

Definition 5.1 (Post embedding with partial codirectness [25]). $\mathsf{PEP}_{\text{codir}}^{\text{partial}}$ *is the question, given two finite alphabets* Σ,Γ*, two morphisms* $u,v:\Sigma^*\to\Gamma^*$*, and two regular languages* $R,R'\in\mathsf{Reg}(\Sigma)$*, whether there is* $\sigma\in R$ *(called a* solution*) such that* $u(\sigma)\sqsubseteq v(\sigma)$*, and such that furthermore* $u(\sigma')\sqsubseteq v(\sigma')$ *for all suffixes* σ' *of* σ *that belong to* R'*.*[6]

The above definition uses the same subword relation, "$\sqsubseteq$", that captures message losses. $\mathsf{PEP}_{\text{codir}}^{\text{partial}}$ can be compared with Post's Correspondence Problem, where the question is whether there exists $\sigma\in\Sigma^+$ such that $u(\sigma)=v(\sigma)$.

Since $\mathsf{PEP}_{\text{codir}}^{\text{partial}}$ is decidable [25], we deduce:

Corollary 5.2. *Reachability is decidable for UCST[*Z_1^1*].*

The reduction from UCST[Z_1^1] to $\mathsf{PEP}_{\text{codir}}^{\text{partial}}$ extends an earlier reduction from UCS to PEP [22]. Here the presence of Z_1^1 tests creates new difficulties.

We fix an instance $S=(\{\mathtt{r},\mathtt{l}\},\mathtt{M},Q_1,\Delta_1,Q_2,\Delta_2)$, $C_{\text{in}}=(p_{\text{in}},q_{\text{in}},\varepsilon,\varepsilon)$, $C_{\text{fi}}=(p_{\text{fi}},q_{\text{fi}},\varepsilon,\varepsilon)$ of the reachability problem for UCST[Z_1^1]. (We again assume $x\in\mathtt{M}$ in each $\mathtt{c}!x$, $\mathtt{c}?x$.) We construct a $\mathsf{PEP}_{\text{codir}}^{\text{partial}}$ instance $\mathcal{P}=(\Sigma,\Gamma,u,v,R,R')$ intended to express the existence of a run from C_{in} to C_{fi}.

We first put $\Sigma\stackrel{\text{def}}{=}\Delta_1\cup\Delta_2$ and $\Gamma\stackrel{\text{def}}{=}\mathtt{M}$ so that words $\sigma\in\Sigma^*$ are sequences of UCST rules and their images $u(\sigma),v(\sigma)\in\Gamma^*$ are sequences of messages. With any $\delta\in\Sigma$, we associate $\mathit{write}\_\mathtt{r}(\delta)$ defined by $\mathit{write}\_\mathtt{r}(\delta)=x$ if δ is a sender rule of the form $\cdot\xrightarrow{\mathtt{r}!x}\cdot$, and $\mathit{write}\_\mathtt{r}(\delta)=\varepsilon$ in all other cases. This is extended to sequences with $\mathit{write}\_\mathtt{r}(\delta_1\cdots\delta_n)=\mathit{write}\_\mathtt{r}(\delta_1)\cdots\mathit{write}\_\mathtt{r}(\delta_n)$. In a similar way we define $\mathit{write}\_\mathtt{l}(\sigma)\in\mathtt{M}^*$, the sequence written to $\mathtt{l}$ by the sequence σ, and $\mathit{read}\_\mathtt{r}(\sigma)$ and $\mathit{read}\_\mathtt{l}(\sigma)$, the sequences read by σ from $\mathtt{r}$ and $\mathtt{l}$, respectively. We define $E_{\mathtt{r}}\in\mathsf{Reg}(\Sigma)$ where $E_{\mathtt{r}}\stackrel{\text{def}}{=}E_1\cup E_2$ and

$$E_1\stackrel{\text{def}}{=}\{\delta\in\Sigma\mid \mathit{write}\_\mathtt{r}(\delta)=\mathit{read}\_\mathtt{r}(\delta)=\varepsilon\}\,,$$
$$E_2\stackrel{\text{def}}{=}\{\delta_1\cdot\delta_2\in\Sigma^2\mid \mathit{write}\_\mathtt{r}(\delta_1)=\mathit{read}\_\mathtt{r}(\delta_2)\neq\varepsilon\}\,.$$

In other words, E_1 gathers the rules that do not write to or read from $\mathtt{r}$, and E_2 contains all pairs of sender/receiver rules that write/read a same letter to/from $\mathtt{r}$.

Let now $P_1\subseteq\Delta_1^*$ be the set of all sequences of sender rules of the form $p_{\text{in}}=p_0\xrightarrow{\cdots}p_1\xrightarrow{\cdots}p_2\cdots\xrightarrow{\cdots}p_n=p_{\text{fi}}$, i.e., sequences which take the sender state from p_{in} to p_{fi}.[7] Similarly, let $P_2\subseteq\Delta_2^*$ be the set of all sequences of receiver rules which take the receiver component from q_{in} to q_{fi}. Since P_1 and P_2 are defined by finite state systems, they are regular languages. We write $P_1\|P_2$ to denote the set of all interleavings (shuffles) of

[6] This problem is actually called $\mathsf{PEP}_{\text{codir}}^{\text{partial}}$ in [25].

[7] I.e., all paths from p_{in} to p_{fi} in the directed graph of the sender, seeing rules as directed edges.

a word in P_1 with a word in P_2. This operation is regularity-preserving, so $P_1 \| P_2 \in \mathsf{Reg}(\Sigma)$. Let $Z_1 \subseteq \Delta_1$ be the set of all sender rules which test the emptiness of $\mathtt{l}$ (which are the only test rules in S). We define R and R' as the following regular languages:

$$R = E_{\mathtt{r}}^* \cap (P_1 \| P_2), \qquad\qquad R' = Z_1 \cdot \big(\Delta_1 \cup \Delta_2\big)^*.$$

Finally, the morphisms $u, v : \Sigma^* \to \Gamma^*$ are given by $u \stackrel{\text{def}}{=} \mathit{read}\_\mathtt{l}$ and $v \stackrel{\text{def}}{=} \mathit{write}\_\mathtt{l}$.

Lemma 5.3 (Correctness). *S has a run $C_{\text{in}} \stackrel{*}{\to} C_{\text{fi}}$ iff $\mathcal{P}$ has a solution.*

Proof. We first introduce a notion bridging the difference between runs of S and solutions of $\mathcal{P}$. We call $\sigma \in (\Delta_1 \cup \Delta_2)^*$ a *pre-solution* if all the following conditions hold:

1. $\sigma \in P_1 \| P_2$;
2. $\mathit{read}\_\mathtt{r}(\sigma) = \mathit{write}\_\mathtt{r}(\sigma)$;
3. $\mathit{read}\_\mathtt{r}(\sigma_1)$ is a prefix of $\mathit{write}\_\mathtt{r}(\sigma_1)$ for each prefix σ_1 of σ;
4. $\mathit{read}\_\mathtt{l}(\sigma) \sqsubseteq \mathit{write}\_\mathtt{l}(\sigma)$;
5. for each factorization $\sigma = \sigma_1 z \sigma_2$ where $z \in Z_1$ we have $\mathit{read}\_\mathtt{l}(\sigma_2) \sqsubseteq \mathit{write}\_\mathtt{l}(\sigma_2)$.

A pre-solution σ has a *receiver-advancing switch* if $\sigma = \sigma_1 \delta \delta' \sigma_2$ where δ is a sender rule, δ' is a receiver rule, and $\sigma' = \sigma_1 \delta' \delta \sigma_2$ is a pre-solution. A *receiver-postponing switch* is defined analogously, for δ being a receiver rule and δ' being a sender rule.

It is obvious that if there is a pre-solution σ then there is an *advance-stable pre-solution* σ', which means that σ' has no receiver-advancing switch; there is also a *postpone-stable pre-solution* σ'' which has no receiver-postponing switch.

Claim. Any advance-stable pre-solution σ is in $E_{\mathtt{r}}^*$, and it is thus a solution of $\mathcal{P}$.

Proof of the claim. Let us write an advance-stable pre-solution σ as $\sigma_1 \sigma_2$ where σ_1 is the longest prefix such that $\sigma_1 \in E_{\mathtt{r}}^*$; hence $\mathit{read}\_\mathtt{r}(\sigma_1) = \mathit{write}\_\mathtt{r}(\sigma_1)$ by the definition of $E_r = E_1 \cup E_2$. Now suppose $\sigma_2 \neq \varepsilon$. Then $\sigma_2 = \delta_1 \delta_2 \cdots \delta_k$ where $\delta_1 \notin E_1$. Since now $\sigma_1 \in E_{\mathtt{r}}^*$, hence $\mathit{read}\_\mathtt{r}(\sigma_1) = \mathit{write}\_\mathtt{r}(\sigma_1)$, δ_1 must be of the form $. \stackrel{\mathtt{r}!x}{\longrightarrow} .$ (to keep 3.). Let us pick the smallest ℓ such that $\delta_\ell = . \stackrel{\mathtt{r}?x}{\longrightarrow} .$ (which must exist by 2.) and note that $\ell \geq 3$ since $\sigma_1 \delta_1 \delta_2 \notin E_{\mathtt{r}}^*$. If we now pick the first j with $1 \leq j \leq \ell - 1$ and such that δ_j is a sender rule and δ_{j+1} is a receiver rule, switching δ_j, δ_{j+1} leads again to a pre-solution (as can be checked by inspecting 1.–5.). This contradicts the assumption that σ is an advance-stable pre-solution.

Claim. Any postpone-stable pre-solution σ corresponds to a run $C_{\text{in}} \stackrel{*}{\to} C_{\text{fi}}$ of S.

Proof of the claim. Consider a presentation $\sigma = \sigma_1 \sigma_2 z \sigma_3$ where $z \in Z_1$, σ_2 contains no rules from Z_1, and σ_1 is either empty or finishes with some $z' \in Z_1$; recall that $\mathit{read}\_\mathtt{l}(\sigma_2 z \sigma_3) \sqsubseteq \mathit{write}\_\mathtt{l}(\sigma_2 z \sigma_3)$ and $\mathit{read}\_\mathtt{l}(\sigma_3) \sqsubseteq \mathit{write}\_\mathtt{l}(\sigma_3)$. We then must have $\mathit{read}\_\mathtt{l}(\sigma_2) \sqsubseteq \mathit{write}\_\mathtt{l}(\sigma_2)$: otherwise we had $\sigma_2 = \sigma' \delta \sigma''$ where δ is of the form $. \stackrel{\mathtt{l}?x}{\longrightarrow} .$, σ'' contains no $\mathtt{l}$-reading rules and $\mathit{read}\_\mathtt{l}(\delta \sigma'' z \sigma_3) \sqsubseteq \mathit{write}\_\mathtt{l}(\sigma_3)$; then switching the leftmost receiver-sender pair in $\delta \sigma'' z$ would lead to a pre-solution, as can be easily checked. Moreover, in σ_2 each sender rule precedes all receiver rules. It is now easy to verify that there is a run $C_{\text{in}} \stackrel{\delta_1}{\to} C_1 \stackrel{\delta_2}{\to} \cdots C_{n-1} \stackrel{\delta_n}{\to} C_{\text{fi}}$ of S where $\delta_1 \delta_2 \cdots \delta_n = \sigma$.

Finally we observe that if $C_{\text{in}} \xrightarrow{\delta_1} C_1 \xrightarrow{\delta_2} \cdots C_{n-1} \xrightarrow{\delta_n} C_{\text{fi}}$ is a run of S then $\sigma = \delta_1\delta_2\cdots\delta_n$ is a pre-solution; then there is also an advance-stable pre-solution, i.e. a solution of $\mathcal{P}$. On the other hand, if σ is a solution of $\mathcal{P}$ then σ is a pre-solution, and then there is a postpone-stable pre-solution, which corresponds to a run $C_{\text{in}} \xrightarrow{*} C_{\text{fi}}$ of S. □

Actually, $\mathsf{PEP}^{\text{partial}}_{\text{codir}}$ and UCST[Z_1^1] are equivalent (inter-reducible) problems:

Theorem 5.4. $\mathsf{PEP}^{\text{partial}}_{\text{codir}}$ *many-one reduces to the Reachability Problem for UCST[*$Z_i^{\mathtt{c}}$*] for any* $i \in \{1,2\}$ *and* $\mathtt{c} \in \mathtt{Ch}$.

Proof (Idea). These reductions are easy and follow basically the same pattern: a UCST system nondeterministically guesses a solution and validates it. As an example, let us informally describe the simplest one and show how to solve a $\mathsf{PEP}^{\text{partial}}_{\text{dir}}$ instance with a UCST[$Z_1^{\mathtt{r}}$] system. We recall from [25] that $\mathsf{PEP}^{\text{partial}}_{\text{dir}}$ is the question whether there is a $\sigma \in R$ such that $u(\sigma) \sqsubseteq v(\sigma)$ and furthermore $u(\sigma') \sqsubseteq v(\sigma')$ for all *prefixes* of σ that belong to R' (thus $\mathsf{PEP}^{\text{partial}}_{\text{dir}}$ and $\mathsf{PEP}^{\text{partial}}_{\text{codir}}$ are equivalent problems and one switches from one to the other by taking the mirror images of u, v, R, R').

Given $(\Sigma, \Gamma, u, v, R, R')$ we build an UCST where Sender nondeterministically generates a $\sigma \in R$, sending $u(\sigma)$ on channel $\mathtt{r}$ and $v(\sigma)$ on channel $\mathtt{l}$. A subword of $v(\sigma)$ is written on $\mathtt{l}$. Receiver checks that $\mathtt{l}$ and $\mathtt{r}$ contain exactly the same sequence of messages, that is, $u(\sigma)$. Whenever the prefix of σ generated so far (call it σ') is in R', Sender waits for $\mathtt{r}$ to be empty before going on with the generation of σ. This forces Receiver to match $u(\tau)$ with a prefix of $v(\tau)$, or more precisely, with a prefix of the subword of $v(\tau)$ that ends up in $\mathtt{l}$ after message losses may have occurred.

The other three reductions are similar. □

6 Reducing UCST[Z_1] to UCST[Z_1^1]

In this section we prove the decidability of reachability for UCST[Z_1] by reducing to UCST[Z_1^1]. Since this involves eliminating Z tests on $\mathtt{r}$, the configurations in which $\mathtt{r}$ is empty are of interest. For a UCST S, we let $Conf_{\mathtt{r}=\varepsilon}$ be the subset of configurations (p,q,ε,v) in which $\mathtt{r}$ is empty. We abuse terminology and say that a subset $W \subseteq Conf_{\mathtt{r}=\varepsilon}$ is *regular* if there are some state-indexed regular languages $(V_{p,q})_{p\in Q_1, q\in Q_2}$ in $\mathsf{Reg}(\mathtt{M})$ such that $W = \{(p,q,\varepsilon,v) \mid v \in V_{p,q}\}$. Such regular subsets of $Conf_{\mathtt{r}=\varepsilon}$ can be finitely represented using, e.g., regular expressions or finite-state automata.

We have put $C = (p,q,u,v) \sqsubseteq C' = (p',q',u',v')$ iff $p = p'$, $q = q'$, $u = u'$, and $v \sqsubseteq v'$. $Conf_{\mathtt{r}=\varepsilon}$ is thus a well-quasi order under $\sqsubseteq$, unlike $Conf$.

$W \subseteq Conf_{\mathtt{r}=\varepsilon}$ is *upward-closed* (in $Conf_{\mathtt{r}=\varepsilon}$) if $C \in W$, $C \sqsubseteq C'$ and $C' \in Conf_{\mathtt{r}=\varepsilon}$ imply $C' \in W$. It is *downward-closed* if $Conf_{\mathtt{r}=\varepsilon} \setminus W$ is upward-closed. The upward-closure $\uparrow W$ of $W \subseteq Conf_{\mathtt{r}=\varepsilon}$ is the smallest upward-closed set that contains W. A well-known consequence of Higman's Lemma is that upward-closed and downward-closed subsets of $Conf_{\mathtt{r}=\varepsilon}$ are regular, and that upward-closed subsets can be canonically represented by their finitely many minimal elements.

For $W \subseteq Conf_{\mathtt{r}=\varepsilon}$, we let $\mathrm{Pre}^*(W) \stackrel{\text{def}}{=} \{C \in Conf_{\mathtt{r}=\varepsilon} \mid \exists D \in W : C \xrightarrow{*} D\}$: observe that $\mathrm{Pre}^*(W)$ only contains configurations with empty $\mathtt{r}$.

Lemma 6.1. *If W is an upward-closed subset of $Conf_{\mathtt{r}=\varepsilon}$ and if S is a UCST[Z_1^1], then* $\mathrm{Pre}^*(W)$ *is upward-closed and is computable uniformly from S and W.*

Proof (Sketch). That $\mathrm{Pre}^*(W)$ is upward-closed is an immediate consequence of the definition of lossy steps in Eq. (1). That it is computable from S and W is more interesting: this is an application of the VJGL Lemma: "an upward-closed set U is computable if one can decide $C \in U$ and $V \cap U \neq \varnothing$ for arbitrary configurations C and regular sets V" (see [26, Theorem 2] for details). Here the two questions, "$C \in U$?" and "$V \cap U \neq \varnothing$?", i.e., "$C \xrightarrow{*} W$?" and "$U \xrightarrow{*} W$?", reduce to instances of the extended reachability problem for UCST[Z_1^1], hence are decidable. □

Theorem 6.2. *Reachability is decidable for UCST[Z_1].*

Proof (Sketch). Given a UCST[Z_1] S, a run $\pi = C_{\mathrm{in}} \xrightarrow{*} C_{\mathrm{fi}}$ can be presented in the form

$$(C_{\mathrm{in}} =)\ C_0 \xrightarrow{*}_{S'} D_1 \xrightarrow{Z:\mathtt{r}} C_1 \xrightarrow{*}_{S'} D_2 \xrightarrow{Z:\mathtt{r}} C_2 \cdots \xrightarrow{*}_{S'} D_m\ (= C_{\mathrm{fi}})$$

where the $D_i \to C_i$ steps gather all occurrences of $Z_1^{\mathtt{r}}$ tests: note that necessarily D_i and C_i are in $Conf_{\mathtt{r}=\varepsilon}$. The $C_{i-1} \xrightarrow{*}_{S'} D_i$ subruns can be seen as runs of a new system S', which is obtained from S by removing all $Z_1^{\mathtt{r}}$ testing rules from Δ_1. The point is that we can apply Lemma 6.1 to S' since it is a UCST[Z_1^1].

So for $k = 0, 1, \ldots$, we define T'_k and T_k by letting $T'_0 = {\uparrow}C_{\mathrm{fi}}$, $T_k = \mathrm{Pre}^*_{S'}(T'_k)$ and $T'_{k+1} = T'_k \cup \{C \mid \exists D \in T_k : C \xrightarrow{Z:\mathtt{r}} D\}$ (note that T_k is defined with a $\mathrm{Pre}^*_{S'}$ restricted to S'). T_k collects all configurations $C \in Conf_{\mathtt{r}=\varepsilon}$ from which one can reach T_0 with at most k uses of a $Z : \mathtt{r}$ test. We observe that all T'_k, T_k are upward-closed subsets of $Conf_{\mathtt{r}=\varepsilon}$, that T_k is computable from T'_k by Lemma 6.1, and that T'_{k+1} is obviously computable from T_k and T'_k. Furthermore, the T_k's are increasing: $T_0 \subseteq T_1 \subseteq \cdots T_k \subseteq T_{k+1} \cdots$. Since they are upward-closed, they eventually stabilize by the well-quasi-ordering property: letting $T_\omega \stackrel{\mathrm{def}}{=} \bigcup_{k\in\mathbb{N}} T_k$, there is n such that $T_n = T_{n+1} = T_\omega$. Since there is a run $C_{\mathrm{in}} \xrightarrow{*} C_{\mathrm{fi}}$ of S iff $C_{\mathrm{in}} \in T_\omega$, the proof is finished. □

Observe that Lemma 6.1 and Theorem 6.2 exhibit a Turing reduction (from reachability for UCST[Z_1] to reachability for UCST[Z_1^1]) and not a many-one reduction like all the other reductions in this paper.

With the results of sections 4 and 5, one obtains the following corollary.

Theorem 6.3. *Reachability is decidable for UCST[Z,N].*

Remark 6.4 (On complexity). Based on known results on the complexity of $\mathsf{PEP}^{\mathrm{partial}}_{\mathrm{codir}}$ (see [17, 25]), our reductions prove that reachability for UCST[Z,N] is at level $\mathcal{F}_{\omega^\omega}$ in the extended Grzegorczyck hierarchy, and at level $\mathcal{F}_{\omega^{m-1}}$, where $m = |\mathtt{M}|$, when we restrict to systems with a fixed-sized alphabet of messages. □

7 Some undecidable problems for UCST[Z,N]

The main result of this paper is that reachability is decidable for UCST[Z,N] (Theorem 6.3). In this section we argue that emptiness and non-emptiness tests strictly add

to the expressive power of UCS's. This point is made in two different ways.

We start with recurrent reachability. Formally, the Recurrent Reachability Problem asks whether a given S has an infinite run $C_{\text{in}} \xrightarrow{+} (p,q,u_1,v_1) \xrightarrow{+} (p,q,u_2,v_2) \xrightarrow{+} \cdots$ visiting infinitely often a given control pair $(p,q) \in Q_1 \times Q_2$ (but with no constraints on channel contents).

Theorem 7.1. *Recurrent reachability is undecidable for UCST[$Z_1^{\mathtt{r}}$].*

Proof (Idea). We prove Theorem 7.1 by reducing from the undecidable question whether a length-preserving string rewrite system (aka semi-Thue system) has a loop $x \xrightarrow{+} x$. We design a UCST S where Sender guesses a word y_0, writes it on `l`, and then guesses pairs x_i, y_i for $i = 1,2,\ldots$ such that each $x_i \to y_i$ is a rewrite step. It writes x_i on `r` and y_i on `l`. Receiver's job is to check that $y_{i-1} = x_i$. With $Z_1^{\mathtt{r}}$ tests, Sender can wait for a check on x_i to be concluded before issuing the next pair. This way we ensure progress of the checking phase and avoid confusion between pairs if a separator is lost. Since the rewrite system is length-preserving, any infinite run of S must eventually stop losing messages and witness a loop. □

Since recurrent reachability is decidable for UCS (see [22]), Theorem 7.1 shows that Z tests, even just $Z_1^{\mathtt{r}}$ tests, cannot be simulated in UCS's.

As another illustration, we consider UCST's with *write-lossy semantics*, that is, UCST's with the assumption that messages are only lost during steps that (attempt to) write them to `l`. Once they are in `l`, they are never lost. This is formalized via a new transition relation $C \to_{\text{wrlo}} C'$ (definition omitted, but as expected) that is intermediary between $\to_{\text{rel}}$ and $\to_{\text{los}}$.

In many cases the two lossy semantics coincide:

Lemma 7.2. *Assume S is a UCST[Z] system. Then $C_{\text{in}} \xrightarrow{*}_{\text{los}} C_{\text{fi}}$ iff $C_{\text{in}} \xrightarrow{*}_{\text{wrlo}} C_{\text{fi}}$.*

Proof (Idea). Prove that $C \xrightarrow{\delta}_{\text{los}} C'$ iff $D \xrightarrow{\delta}_{\text{wrlo}} C'$ for some $D \sqsubseteq C$. Deduce $C_{\text{in}} \xrightarrow{n+1}_{\text{los}} C'$ iff $C_{\text{in}} \xrightarrow{n+1}_{\text{wrlo}} C'$ by induction on n. See [18, App. A]. □

Corollary 7.3. *Reachability is decidable for UCST[Z] with write-lossy semantics.*

Remark 7.4. Write-lossy semantics is meaningful when modeling unreliability of the writing actions as opposed to unreliability of the channels. However, in the literature, write-lossy semantics is mostly used as a way of restricting the nondeterminism of lossy channel systems without losing any essential generality, as stated by Lemma 7.2. □

Write-lossy and (plain) lossy semantics do not coincide when N tests are allowed. In fact, Theorem 6.3 does not extend to write-lossy systems.

Theorem 7.5. *Reachability is undecidable for UCST[Z_1^1, N_1^1] with write-lossy semantics.*

Proof (Idea). As before, Sender simulates queue automata using tests and the help of Receiver. See Fig. 3. Channel `l` is initially empty. To read a from `r`, Sender does the following: (1) write a on `l`; (2) check that `l` is nonempty (hence the write was not lost); (3) check that, or wait until, `l` is empty. Meanwhile, Receiver reads identical letters from `r` and `l`. □

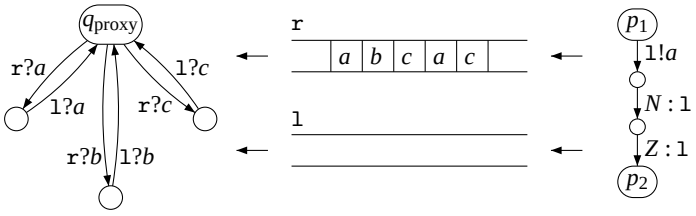

Fig. 3. Write-lossy Sender simulates "$p_1 \xrightarrow{\mathtt{r}?a} p_2$" with N and Z tests and proxy Receiver

Thus, at least in the write-lossy setting, we can separate UCST[Z] and UCST[Z, N_1^1] w.r.t. decidability of reachability.

8 Conclusion

UCS's are communicating systems where a Sender can send messages to a Receiver via one reliable and one unreliable, lossy, channel, but where no direct communication is possible in the other direction. We introduced UCST, an extension of UCS's where steps can be guarded by tests, i.e., regular predicates on channel contents. This extension introduces limited but real possibilities for synchronization between Sender and Receiver. For example, Sender (or Receiver) may use tests to detect whether the other agent has read (or written) some message. As a consequence, adding tests leads to undecidable reachability problems in general. Our main result is that reachability remains decidable when only emptiness and non-emptiness tests are allowed. The proof goes through a series of reductions from UCST[Z, N] to UCST[Z_1^1] and finally to $\mathsf{PEP}_{\text{codir}}^{\text{partial}}$, an extension of Post's Embedding Problem that was motivated by the present paper and whose decidability was recently proved by the last two authors [25].

We see two main directions for future works:

1. The limits of decidability: is it possible to characterize precisely the families of tests $T \subseteq \mathsf{Reg}(\mathtt{M})$ for which UCST[T] has a decidable reachability problem? We gave positive and negative examples, but a precise characterization would help understand the phenomenon at hand.
2. Beyond reachability: we focused on reachability questions since they are the most natural starting point as far as verification is concerned. However several other natural verification problems, e.g., termination, are known to be decidable for UCS's.

References

1. D. Brand and P. Zafiropulo. On communicating finite-state machines. *J. ACM*, 30(2):323–342, 1983.
2. B. Boigelot and P. Godefroid. Symbolic verification of communication protocols with infinite state spaces using QDDs. *Formal Methods in System Design*, 14(3):237–255, 1999.
3. A. Bouajjani and P. Habermehl. Symbolic reachability analysis of FIFO-channel systems with nonregular sets of configurations. *Theor. Comp. Sci.*, 221(1–2):211–250, 1999.
4. M. F. Atig, A. Bouajjani, and T. Touili. On the reachability analysis of acyclic networks of pushdown systems. In *CONCUR 2008*, LNCS 5201, pp. 356–371. Springer, 2008.
5. A. Muscholl. Analysis of communicating automata. In *LATA 2010*, LNCS 6031, pp. 50–57. Springer, 2010.
6. G. Cécé, A. Finkel, and S. Purushothaman Iyer. Unreliable channels are easier to verify than perfect channels. *Inf. Comp.*, 124(1):20–31, 1996.
7. P. A. Abdulla and B. Jonsson. Verifying programs with unreliable channels. *Inf. Comp.*, 127(2):91–101, 1996.
8. P. A. Abdulla, A. Collomb-Annichini, A. Bouajjani, and B. Jonsson. Using forward reachability analysis for verification of lossy channel systems. *Formal Methods in System Design*, 25(1):39–65, 2004.
9. P. A. Abdulla, N. Bertrand, A. Rabinovich, and Ph Schnoebelen. Verification of probabilistic systems with faulty communication. *Inf. Comp.*, 202(2):141–165, 2005.
10. C. Baier, N. Bertrand, and Ph. Schnoebelen. On computing fixpoints in well-structured regular model checking, with applications to lossy channel systems. In *LPAR 2006*, LNCS 4246, pp. 347–361. Springer, 2006.
11. C. Baier, N. Bertrand, and Ph. Schnoebelen. Verifying nondeterministic probabilistic channel systems against ω-regular linear-time properties. *ACM Trans. Comput. Logic*, 9(1), 2007.
12. P. A. Abdulla, J. Deneux, J. Ouaknine, and J. Worrell. Decidability and complexity results for timed automata via channel machines. In *ICALP 2005*, LNCS 3580, pp. 1089–1101. Springer, 2005.
13. J. Ouaknine and J. Worrell. On metric temporal logic and faulty Turing machines. In *FOSSACS 2006*, LNCS 3921, pp. 217–230. Springer, 2006.
14. A. Kurucz. Combining modal logics. In P. Blackburn, J. van Benthem, and F. Wolter, editors, *Handbook of Modal Logics*, volume 3, chapter 15, pp. 869–926. Elsevier Science, 2006.
15. B. Konev, R. Kontchakov, F. Wolter, and M. Zakharyaschev. Dynamic topological logics over spaces with continuous functions. In *Advances in Modal Logic, vol.6*, pp. 299–318. College Publications, 2006.
16. S. Lasota and I. Walukiewicz. Alternating timed automata. *ACM Trans. Computational Logic*, 9(2), 2008.
17. S. Schmitz and Ph. Schnoebelen. Multiply-recursive upper bounds with Higman's lemma. In *ICALP 2011*, LNCS 6756, pp. 441–452. Springer, 2011.
18. P. Chambart and Ph. Schnoebelen. The ordinal recursive complexity of lossy channel systems. In *LICS 2008*, pp. 205–216. IEEE Comp. Soc. Press, 2008.
19. S. Haddad, S. Schmitz, and Ph. Schnoebelen. The ordinal-recursive complexity of timed-arc Petri nets, data nets, and other enriched nets. In *LICS 2012*, pp. 355–364. IEEE Comp. Soc. Press, 2012.
20. P. Chambart and Ph. Schnoebelen. Mixing lossy and perfect fifo channels. In *CONCUR 2008*, LNCS 5201, pp. 340–355. Springer, 2008.
21. P. Chambart and Ph. Schnoebelen. Post Embedding Problem is not primitive recursive, with applications to channel systems. In *FST&TCS 2007*, LNCS 4855, pp. 265–276. Springer, 2007.

22. P. Chambart and Ph. Schnoebelen. The ω-Regular Post Embedding Problem. In *FOSSACS 2008*, LNCS 4962, pp. 97–111. Springer, 2008.
23. Pierre Chambart and Philippe Schnoebelen. Pumping and counting on the regular Post embedding problem. In *ICALP 2010*, LNCS 6199, pp. 64–75. Springer, 2010.
24. P. Barceló, D. Figueira, and L. Libkin. Graph logics with rational relations and the generalized intersection problem. In *LICS 2012*, pp. 115–124. IEEE Comp. Soc. Press, 2012.
25. P. Karandikar and Ph. Schnoebelen. Cutting through regular Post embedding problems. In *CSR 2012*, LNCS 7353, pp. 229–240. Springer, 2012.
26. J. Goubault-Larrecq. On a generalization of a result by Valk and Jantzen. Research Report LSV-09-09, Laboratoire Spécification et Vérification, ENS Cachan, France, May 2009.