Information Processing Letters 115 (2015) 694–698

Contents lists available at ScienceDirect

Information Processing Letters

www.elsevier.com/locate/ipl

Non-deterministic transducer models of retransmission protocols over noisy channels

Jay Thakkar, Aditya Kanade *

Indian Institute of Science, Bangalore, India

ARTICLE INFO

Article history:
Received 21 August 2014
Received in revised form 6 January 2015
Accepted 17 April 2015
Available online 21 April 2015
Communicated by J.L. Fiadeiro

Keywords:
Formal methods
Protocols
Noisy channels
String transducers

ABSTRACT

Retransmission protocols such as HDLC and TCP are designed to ensure reliable communication over noisy channels (i.e., channels that can corrupt messages). Thakkar et al. [15] have recently presented an algorithmic verification technique for *deterministic* streaming string transducer (DSST) models of such protocols. The verification problem is posed as equivalence checking between the specification and protocol DSSTs. In this paper, we argue that more general models need to be obtained using *non-deterministic* streaming string transducers (NSSTs). However, equivalence checking is undecidable for NSSTs. We present two classes where the models belong to a sub-class of NSSTs for which it is decidable.

1. Introduction

Retransmission protocols use cyclic redundancy check and sliding window protocols for error detection and control respectively [14]. TinyOS serial communication protocol, Philips bounded retransmission protocol (BRP), high-level data link control (HDLC), and transmission control protocol (TCP) are examples of widely used retransmission protocols that provide reliable communication over *noisy channels*, i.e., channels that can corrupt messages.

1.1. Motivation for transducer based modeling of retransmission protocols

In a recent work, Thakkar et al. [15] present an approach of transducer based modeling of retransmission protocols over noisy channels. They show that the usual approach of abstracting message contents by symbolic constants (e.g. [11]) is inadequate in this setting. In particular, they illustrate that unless the message contents are modeled as *bit strings*, the noisy channel can give rise to a sequence of message corruptions inducing the receiver to deliver an incorrect sequence of messages to its client (see [15], Section 2). Thus, we need an expressive framework to precisely model retransmission protocols. Modeling them as finite-state machines communicating asynchronously over unbounded FIFO channels however does not yield a decidable framework [6].

They show that *deterministic streaming string transducers* or *DSSTs* [2,3] provide an intuitive and expressive modeling framework of retransmission protocols. DSSTs can model different classes of retransmission protocols such as those based on stop-and-wait, go-back-n and selective-repeat sliding window protocols [14]. In these models, the length of a message string and the number of retransmission rounds can be *unbounded*. Even in the presence of these sources of unboundedness, the protocol verification problem — formalized as equivalence checking problem over DSSTs — is *decidable*.

1.2. Deterministic protocol models

We first review the approach in [15]. In this approach, the protocol components — sender and receiver — and the specification are modeled using deterministic string

* Corresponding author.
E-mail address: kanade@csa.iisc.ernet.in (A. Kanade).

http://dx.doi.org/10.1016/j.ipl.2015.04.009
0020-0190/

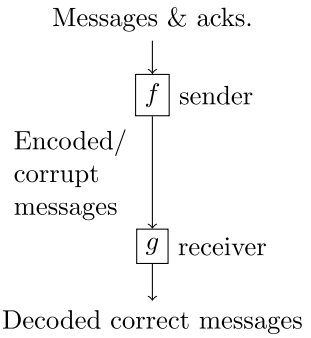

Fig. 1. Sender and receiver transducer models.

transducers. As shown in Fig. 1, the input to a *sender transducer* f is a sequence of strings representing messages to be transmitted as well as acknowledgements sent by the receiver where both messages and acknowledgements are bit strings. The sender's output is the sequence of encoded or corrupt messages that arrive at the receiver *over the noisy channel* across all rounds of transmission. That is, the noisy behavior of the channel and the protocol's retransmission logic are modeled in the output of the sender. The *receiver transducer* g (1) recognizes and discards corrupt messages, and (2) extracts and outputs decoded values of correctly received messages. The *protocol transducer* $p \equiv g \circ f$ is obtained by sequential composition of the sender and receiver transducers where $(g \circ f)(x) = g(f(x))$.

The *specification transducer* h captures the desired end-to-end behavior of the protocol. It requires that (1) the messages be delivered by the receiver to its client in the same order in which they were received by the sender from its client and (2) the protocol delivers exactly those messages that are positively acknowledged (not corrupted by the channel). The verification problem is posed as *functional equivalence* between the transducers h and p, that is, whether $dom(h) = dom(p)$ and for all $w \in dom(h)$, $h(w) = p(w)$. Here, the output of the transducer g is the input to the client of the receiver. Another transducer g' can be constructed, in a similar manner, to model the acknowledgements that the receiver would generate for the sender component and verified separately.

1.3. Limitation of deterministic models

The deterministic models presented in [15] use a *fixed* string *ERR* in the sender's output to capture the noisy behavior of the channel. As an example, suppose the input to the sender transducer is $M\sharp a$, where $\sharp$ is the end-marker of the message string M and a is an acknowledgement. If $a = 0$ then it is a negative acknowledgement indicating that the previously transmitted message was received incorrectly. Otherwise, it indicates correct reception. Following the approach of [15], let the sender transducer f be: $f(M\sharp 1) = M$ and $f(M\sharp 0) = ERR$.[1] The specification transducer is $h(M\sharp 1) = M$ and $h(M\sharp 0) = \epsilon$ where ϵ is the empty string. A receiver transducer $g(ERR) = \epsilon$ and $g(M) = M$ when $M \neq ERR$ is *verified* to be correct as h is equivalent to $g \circ f$.

Now, consider a *non-deterministic* sender transducer f' such that $f'(M\sharp 1) = M$ and $f'(M\sharp 0) = M'$ where M' is an arbitrary corrupt form of M. With this, the protocol model $g \circ f'$ would deliver a *corrupt* message $M' \neq ERR$ to the receiver's client. In other words, with the deterministic sender f, we cannot ascertain (1) whether the receiver g handles all forms of corrupt messages and (2) whether the protocol delivers the messages correctly in the presence of *arbitrary corruption*. To establish these properties, we need a *non-deterministic* model of the sender transducer (similar to f') that emits arbitrarily corrupt messages instead of a fixed error string.

1.4. Our approach

In this work, we propose to use *non-deterministic streaming string transducers* or *NSSTs* [4] for modeling the sender. NSSTs are closed under sequential composition (required to compute the protocol transducer) but equivalence checking for NSSTs is undecidable. For a sub-class of NSSTs, called *functional NSSTs* [4], equivalence checking is PSPACE-COMPLETE. We observe that the receiver transducer is *deterministic* due to the protocol semantics. Unfortunately, the sequential composition of an NSST and a DSST does *not* necessarily yield a functional NSST (see Section 4 for an example).

Nevertheless, we show that for the following two interesting classes of protocols, the senders and receivers are of the form that when composed, they result in functional NSSTs:

1. *Bounded retransmission rounds per message*: Here, the messages can be corrupted arbitrarily but each message can be retransmitted up to a certain *fixed* number of rounds. This class is motivated by Philips bounded retransmission protocol (BRP) [10].
2. *Bounded non-determinism in message corruption*: The messages can be corrupted non-deterministically but only in *finitely* many ways, whereas, the messages can be retransmitted an unbounded number of times.

In both these cases, the message lengths and the total number of messages transmitted are *unbounded*. We show that both these classes result in functional-NSST protocol models and hence, can be verified *algorithmically*.

In practice, the implementations of a protocol may differ in the choice of the number of retransmission rounds or the error detection mechanism. The protocol models we construct in this paper are generic, and different concrete models can be obtained by instantiating them and verified individually.

The expressive power of functional NSSTs is same as that of DSSTs [4]. As a consequence, our results also imply that the above classes of protocols can be modeled directly as DSSTs. However, our *modular* approach of modeling the (non-deterministic) sender and the (deterministic) receiver

[1] In the actual models (and real protocols), a message is encoded with a checksum and a sequence number. We postpone these details to the latter part of the paper.