Theoretical Computer Science ••• (••••) •••–•••

Contents lists available at ScienceDirect

Theoretical Computer Science

www.elsevier.com/locate/tcs

Quantifying communication in synchronized languages

Zhe Dang [a,b], Thomas R. Fischer [b], William J. Hutton III [b,*], Oscar H. Ibarra [c], Qin Li [a]

[a] School of Computer Science and Technology, Anhui University of Technology, Ma'anshan, China
[b] School of Electrical Engineering and Computer Science, Washington State University, Pullman, WA 99164, USA
[c] Department of Computer Science, University of California, Santa Barbara, CA 93106, USA

ARTICLE INFO

Article history:
Received 31 October 2015
Accepted 28 January 2016
Available online xxxx

Keywords:
Communication complexity
Cryptography
Information theory

ABSTRACT

A mutual information rate is proposed to quantitatively evaluate inter-process synchronized communication. For finite-state processes with implicit communication that can be described by a counting language, it is shown that the mutual information rate is effectively computable. When the synchronization always happens between the same two symbols at the same time (or with a fixed delay), the mutual information rate is computable. In contrast, when the delay is not fixed, the rate is not computable. Finally, it is shown that some cases exist where the mutual information rate is not computable.

Published by Elsevier B.V.

1. Introduction

Computer systems often run a set of processes communicating with a provided interface for processes to talk with each other. The system may implement highly complex functions; e.g., a concurrent system. Obviously, communications between two processes contribute significantly to the complexity of the whole system. Therefore, it is desirable to find a way to measure the *quantity* of the communications between two processes. Since such communications are often nondeterministic, the quantity would also be part of an indication of hardness of testability when an analyst tests the system.

However, defining such a quantity is not trivial. Static analysis [3,19,23] would not always work because the quantity we are looking for is a dynamic indicator on how "tightly" two communicating processes are working together. Another idea would be to use the classic theory of communication complexity [13,21,18]. However, this theory aims at studying the minimal communication bit-rate needed for a given communication task instead of analyzing the actual amount of communication involved between given processes [27].

In this paper, we provide a metric to quantify the amount of communication between two communicating processes where the communication mechanism is synchronous. Our ideas are as follows. When a process runs, it demonstrates a behavior, which is a sequence (or a word) of events. In this way, two processes, when run synchronously, demonstrate two parallel aligned sequences. In automata theory, the parallel sequences can be thought of a word of two tracks called a synchronized word. The set of all such synchronized words that are the actual runs of the two communication processes forms a synchronized language, written as L_{12}. The metric we study is defined as the information in bits shared between two tracks (each called a string, and both strings are of the same length) in a synchronized word of the language.

There has been already a definition of the amount of information contained in a word. This definition was proposed by Shannon [22] and later Chomsky and Miller [8], that we have evaluated through experiments [5,11,9,7]. For a number n,

* Corresponding author.
E-mail address: williamhutton@gmail.com (W.J. Hutton).

http://dx.doi.org/10.1016/j.tcs.2016.01.042
0304-3975/Published by Elsevier B.V.

we use $S_n(L)$ to denote the number of words in a language L whose length is n. The *information rate* λ_L of L is defined as $\lambda_L = \lim \lambda_{n,L}$, where $\lambda_{n,L} = \frac{\log S_n(L)}{n}$. When the limit does not exist, we take the upper limit, which always exists for a finite alphabet. Throughout this paper, we use log_2.

The intuition behind Shannon's definition is as follows. $\lambda_{n,L}$ specifies the average number of bits needed per symbol, i.e. bit rate, if one losslessly compresses a word of length n in L, while the information rate λ_L is simply the asymptotic bit rate. In other words, λ_L is the average amount of information per symbol contained in a word in L.

Notice that communication passes information between two processes. Let L_1 and L_2 be the projections of the aforementioned synchronized language L_{12} to the first and second tracks, respectively. Let w_1 and w_2 be the aforementioned two strings that form a two-track word w_{12}, with length n in the synchronized language L_{12}. By definition, $\log S_n(L_2)$ is the number of bits needed (on average) to encode w_2. Those bits are divided into two parts: a) the bits or the information that w_1 "knows" about or shares with w_2 as the result of communication, and b) the bits or the information that w_1 does not share with w_2. There are possibly many occurrences of w_2 in L_2 to pair with w_1 to make a two-track word in L_{12}. Due to nondeterminism, the mapping from w_1 to w_2 is one-to-many. Hence, for part b), what w_1 does not know is which branch to take in the mapping to reach w_2. Notice that $\frac{S_n(L_{12})}{S_n(L_1)}$ is the average branching factor in the mapping. Therefore, $\log S_n(L_{12}) - \log S_n(L_1)$ is the number of bits needed to encode a branch, i.e. the bits in part b). In summary, a) is the amount of information shared between w_1 and w_2 on average, $\log S_n(L_2) - (\log S_n(L_{12}) - \log S_n(L_1))$ which equals $\log S_n(L_1) + \log S_n(L_2) - \log S_n(L_{12})$. Taking its asymptotic form, we now define the mutual information rate to quantify the communication in L_{12}:

$$\eta_{L_{12}} = \lambda_{L_1} + \lambda_{L_2} - \lambda_{L_{12}}. \tag{1}$$

In the paper, we show cases when computing the mutual information rate is effective. These cases assume that the two processes are finite-state but the implicit communication mechanism between the two makes the resulting synchronized language L_{12} rather complex; e.g., a counting language (a regular language constrained by a Presburger formula on counts of individual symbols in a word. Notice that a counting language may be nonregular). We show that when the synchronization always happens between two symbols that are the same and that are at the same time, the mutual information is computable. The proof is quite complex, which involves combinatorial analysis of the synchronized words in L_{12} and properties from reversal-bounded counter machines [14]. Later, we also show that this result can be further generalized to cases when the two symbols are not necessarily synchronized at the same time (with a fixed delay). However, the case of arbitrary delays is not computable. We also present some other uncomputable cases as well.

We note that computing the mutual information rate of L_{12} is not trivial at all. We have cases (see the comment right after Theorem 7) where the information rates of L_1 and L_2 are computable but the information rate of L_{12} (and hence the mutual information rate $\eta_{L_{12}}$) is not computable. We also have cases where the information rate of L_{12} is computable but the information rates of L_1 and L_2 are unknown to be computable or takes some nontrivial effort in proving their computability (as we will show in the proof of Theorem 4).

2. Quantifying communication with information rate

Let Σ be an alphabet and consider two languages L_1 and L_2 on the alphabet Σ. For the purpose of this paper, a word on the alphabet represents an observable behavior of a process, which is a sequence of its observable events. Such an event can be, for instance, a state transition when the states are observable. Suppose that P_i $(i = 1, 2)$ is a process. One can think the two processes as two nodes on a network, two concurrent programs running on a processor, two persons monitored by a surveillance cameras in a building, or simply two people dancing on a stage. When the two processes are observed together, a joint behavior is obtained. To ease our presentation, we assume that, whenever an event (say a) is observed in one process, an event (say b) is also observed in another process. Hence, the two processes, intuitively, run at the same pace by observation. Actually, this assumption is made without loss of generality. This is because, one can always introduce a new "idle" event into Σ. At the time when a is observed in one process and, at this time, if no event is observed in another, we treat the "no event" as the idle event. The two processes still run at the same pace by observation.

With this formulation, a joint behavior is a *synchronized word* α on alphabet Σ^k. Without loss of generality, we take $k = 2$ (i.e. two processes). It is trivial to generalize all the results in the paper to an arbitrary k. In the two process case, a synchronized word, α, is of the form

$$(a_1^1, a_1^2) \cdots (a_n^1, a_n^2) \tag{2}$$

for some n and some words $w_1 = a_1^1 \cdots a_n^1$ and $w_2 = a_1^2 \cdots a_n^2$ in Σ^*. For notational convenience, we often write $\alpha = [w_1, w_2]$ while, implicitly, we assume that the two projections w_1 and w_2 share the same length. In the sequence, w_1 is called the first coordinate of α while w_2 is the second coordinate of α.

The synchronized word α can be thought of as being woven from its first coordinate w_1 and second coordinate w_2. When one thinks of a joint run of two synchronized processes as a synchronized word, a restriction may be placed on the possible α so that not every pair of w_1 and w_2 can be woven into an actual joint run. For instance, under the scenario that an event a in process P_1 must be synchronized with an event $\dot{a}$ in process P_2, the synchronized word $[aba, \dot{a}\dot{a}\dot{a}]$ cannot be a joint run. The exact form of the restriction can be an explicit definition of the communication mechanism used between