On communication as transmission of programs

Robert Rehammar, robert@rehammar.se

Abstract

This paper aims at introducing a new view on communication as transmission of programs. This is contrasting the classical view on communication as transmission of data. By employing this view, a completely new understanding of the process of communication appear. How robust communication schemes should be designed, as well as how language learning is done can be viewed in a completely new light.

Introduction

In this paper a view on communication as *transmission of programs* is employed and investigated. The receiver of a communication message is an interpreter (as in a computer program executing a piece of code). The aim for a transmitter of a communication is always to trigger some kind of action from the receiver. Which action is triggered depends on how the receiver interprets the message being sent. Hence, in the receiving end there have to be an interpreter, and thus the message can be viewed as a program to be executed by the receiver interpreter.

This view on communication is fundamentally different from the traditional view of Shannon where communication consists of *transmission of data*. If communication is transmission of data, the focus of the communication is on the data and how it should be handled (packaged) to yield as robust communication as possible. Error correcting codes are and other mechanisms are introduced to make the data as robust at possible when transmitting over a distorting and noisy channel.

However, if the communication consists of transmitting a program, focus can be put on the receiving interpreter and how to design that to yield a robust execution of the transmitted program given that is was distorted over the distorting and noisy channel. Hence, one would want to design the language in which the programs are communicated in, in such a way that a small distortion of the program, yields a small distortion of the execution output. This design pattern in typically not applied to classical communication. There a message is either decoded or discarded. Distorted messaged are never meant to be interpreted. (One particular exception of this is real time communication, such as talk over a phone or live video. There, messages cannot be re-transmitted without losing the purpose of the communication, and hence, communication is designed exactly in analogy to the paradigm suggested above, where distortion of the execution is accepted as long as it is small.)

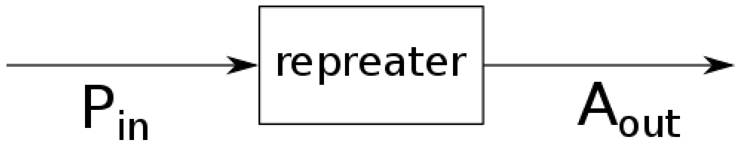
In parallel, every interpretation of a program is a communication. Thus, communication and program execution is equivalent.

Since every receiver is an interpreter, a new entity is introduced, denoted “repreter” and referring to the receiver interpreter. The repreter is in reality nothing new, but the word is used to indicate the new view of every communication action to involve the expectation of a resulting action from the receiver

Analogously, a transmitter is also a programmer, and the new word “transgrammer” is introduced, paralleling repreter. Further the general term or an entity capable of both transgraming and repreating is denoted regrammer (or transpeater).

Receiver design

One way to design a repreater is according to the scheme



Where P_{in} is an input program and A_{out} is the action from the repreater due to P_{in}. Now, if a distortion (or a previously undefined program) is transmitted to the repreater, A_{out} gets distorted. Then, the repreater should be designed in such a way that

$$\frac{\partial A_{\mathrm{out}}}{\partial P_{\mathrm{in}}}$$

is minimized. For this to work, a measure has to be introduced on the spaces which P and A belong to, M_P and M_A respectively. One can expect many such measures to exist. Some measures on M_P are probably be quite easy to define, e.g. edit distance or the Hamming distance. On M_A however, one can expect that defining measures can be fairly challenging. And also, maybe, very context or application dependent.

Transmitter design

A transmitters task is to generate programs. This have to be done using a more or less stringent schema. Communication transmitted over for example a computer bus has a very strict schema with hard coded transgammer and repreater. No distortion is expected over the channel (which has a very good signal-to-noise ratio (SNR)), and in the rare events of a program distortion, there is a large chance that the whole device breaks down. These types of “formal” situations diverse and spread across all fields, not only internal computer busses. Take for example the situation in a court, a business agreement or any other situation where the stakes are high.

On the other hand, is a situation where it is known that the channel has a poor SNR, if the stakes are not high, e.g. in a casual conversation at a club, no one of the participants’ bother in achieving a high-SNR channel, and programs are interpreted loosely (e.g. just smile if you do not hear what the other party is saying).

Communication system design

The idea introduced in the section on repreater design can be extended to the design principle of a communication system. A typical action in a communication system is to transmit another program back to the original transgrammer. In that scenario, $M_A = M_P$.

A communication (a set of repeated transmissions and executions of programs between a number of regrammers where actions, at least to some part, consists of transmitting another program) should then as a whole be designed in a robust manner so that eg.

$$\frac{\partial P_{1\mathrm{out}}}{\partial P_{1\mathrm{in}}}\frac{\partial P_{2\mathrm{out}}}{\partial P_{2\mathrm{in}}} = \frac{\partial P_{2\mathrm{out}}}{\partial P_{1\mathrm{in}}}$$

is minimized. In the above example, it is used that $P_{1out} = P_{2in}$. And so one for longer sequences of communication.

This kind of design pattern is common in natural language, but fairly absent in e.g. computer communication where a package (= program in that communication system) is discarded and a retransmission is requested if the package is in any way distorted, however little. This makes regrammer design easy, but communication non-robust. If instead, the regrammers are designed to manage distorted programs, they become much more complex, but instead, communication becomes robust.

There have been attempts at designing these kind of things, e.g. in fuzzy logics. Another example is a real-time video link. This is usually transmitted over UDP where packages are not retransmitted if decoding fails. Instead the failed decoded video frame is displayed with the decoding failure. Of course, if the videos image has a few pixels which failed, the overall quality of the video stream could be improved by applying some convolution to the video image before displaying. This is a simple repreater design idea which sacrifice output action stringency (displaying a specific picture) in favor of robust execution.

As a comment to the vide-link example above, note that it is not an image that is transmitted, but a program which is designed to render a specific image.

Learning natural language

If all communication is transmission of programs, there is an obvious path to how to learn a language: Transmit, or intercept others actor's transmissions of, programs (communication) and observe them execute. Once a fair amount of such communications have been observed, the learner can simply copy the execution of what was observed. This also gives a clear indication of how misunderstandings can be understood. It is simply communication between two parties where the transgrammer and repreater are not aligned.

This view of language is fairly parallel to Wittgenstein's view of language as games. However, Wittgenstein failed (at least to the authors knowledge) to make the connection between communication as programs. He was more focused on the learning process and how natural language evolve.

It would be very interesting to train a neural net or similar via this method on some simple language and see if it becomes capable of communication.

Another interesting conclusion that can be drawn from this is that a minimal requirement for an entity to be able to learn a language, is that it has to be equipped with sensors that can detect all actions that are possible outcome of a particular language. Of course, in the spirit of Wittgenstein, this whole notion holds in the context of a set of actors interacting with the language at hand, and how these transgrammers looks like (are designed). If they are different, which they most surely are for a natural language, this will inflict impairments on the learning process of the learning entity.

An analogy to test-driven software development

One strong trend in software development is test-driven development where a particular program is run against a set of well-defined initial (IS) and final states (FS). Denote a particular such set S. The software, denote it R (for repreater), should enter the final states given the initial states. This approach is really an application of revering the process of learning language. Here, the (partial) wanted actions from the repreater is known, and the tests exist to verify that the repreter understands the language as expected (note that the initial states are programs).

Viewing repreater design in this fashion, one could imagine to create a “meta”-repreater (a classical programmer), denote it P, that takes a program S and outputs something that is approximately R. An interesting question is then, how large needs the set S be for P to output something that is R or very close to R. If S is a complete basis of the space which R operates on, that probably.

Further, if S is complete in some sense (this excludes things like internal states which of course are vital to human interaction, but might be possible to exclude in a formal description, such as what is relevant if one aims at describing classical software development), it can be used as R.

Conclusion

In this paper, a new view on what communication is has been introduced. When employing the view that communication is transmission of programs between so called regrammers, new design patterns emerge and the understanding of language can be extended. It is believed by that author, that the introductory results presented here lays a foundation for a new paradigm in what communication, language and program execution is.

The concepts here defined can also be used to describe and understand what an actor in communication is, and how artificial intelligence should be modelled. An intelligent entity is one that can interpret messages in a particular language and provide a coherent action outcome. Hence, the concept of intelligence, becomes context dependent. To behave intelligent is to be able to interpret the language used in a particular communication session. This could probably be reformulated as, to be intelligent within a language

$$\frac{\partial P_{\mathrm{j,out}}}{\partial P_{1\mathrm{in}}}$$

can be made small for any j.