Towards a process-calculi approach to study the evolution of biological networks

Alessandro Romanel

CoSBi and Università di Trento, Italy

Over the last decade an increasing interest in using evolutionary approaches to study biological networks has continuously grown. Understanding how networks emerged during evolution can help us to understand their basic properties, such as the role of complexity and the importance of topology and feedback loops.

In [5] we developed a specific framework to allow straightforward study of network evolution based on BlenX [4] and the Beta Workbench [3], a process calculi based programming language and its stochastic simulation engine, respectively. We proposed a framework for simulating the evolution of protein-protein interaction networks where evolution proceeds through selection acting on the variance generated by random mutation events, and individuals replicate in proportion to their performance, referred to as fitness. This follows the idea that in order to simulate evolution by natural selection, we must be able to express populations of individuals, variability and fitness.

BlenX represents a protein as a computational entity, a box, composed by a set of interfaces and an internal program. Interfaces, which represent protein domains, have associated a sort (i.e. representing the structure of the domain) and are the places where a protein interacts with other proteins; the internal program, instead, codifies for the mechanism that transforms an interaction into a protein conformational change, which can result in the modification of other interface sorts.

A BlenX program specifies qualitatively and quantitatively a system of proteins and their interaction capabilities. In particular, the former are specified through a binary relation $\alpha$ on interface sorts (see [4] for details). The graphical notation of boxes and their interaction capabilities we use throughout this paper is depicted in Fig.1(a). Since in our evolutionary framework a system specified by a BlenX program represent and individual (see Fig.1(b)), a population consists of a set of different BlenX programs, each representing an individual composing the population (see Fig.1(c)).

In [5] the evolution of a population is implemented with an evolutionary algorithm which works in four main parts and is iterated for a specified number of steps; each iteration is called generation. The algorithm firstly generates the initial population; the population can be generated randomly, from a predefined network configuration to be used as a starting point, or it can be a network with no interactions. Each individual in the population is
then simulated separately using the Beta Workbench stochastic simulator, and the outputs of the simulations are used to compute the fitness values of the individuals. Like in a real environment, individuals with the highest fitness values are more likely to survive, replicate and produce a progeny that resembles them, being not, however, completely equal to them. This part of the algorithm, indeed, creates a new population with the same number of individuals of the current population, using as a base the current individuals. Depending on a probability proportional to the fitness measure, individuals replicate and pass to the next generation. During the replication, each protein in the individual is given the chance to mutate, according to a probability. The different types of mutations we considered in [5] are based on real biological processes where mutations can happen at DNA and protein level. Variability is achieved by associating each of the considered mutations to a BlenX program modification. The great flexibility of BlenX in the definition of the structure of proteins, indeed, allow us to introduce primitives for mutations used to build domain-based interaction and mutation models. Starting from the study of mutations at a biological level, we end up with some interesting program modifications that permit us to mutate the BlenX representation of proteins in a meaningful and automatic way.

It is clear that the measure of fitness is problem dependent: it varies with the kind of network, with the characteristics a scientist wants to investigate, and so on. This measure can be done in various ways, including stability analysis, integration of the signal, measure of the derivative. In [5] the fitness was computed using integration of protein species time-courses in stochastic simulation results. Here we propose and discuss a theoretical framework, based
on concurrency theory, for computing certain classes of fitness measures. The goal of this paper is to present intuitively and informally most of the ideas we want our framework to be based on.

The dynamics of a system implemented by a BlenX program is described by a stochastic structural operational semantics, which distinguishes between monomolecular reactions (a single box is involved), bimolecular reactions (two boxes interact by means of synchronization or communication) and events (global box rewriting rules), allowing the generation of transition systems with arcs labelled by stochastic rates. This dynamics results exactly in the behaviour we are interested to study, i.e., trajectories in transition systems encode the discrete variation of protein species and protein domains amounts. In particular, we are interested in determining if the dynamics of protein species and protein domains in a given system fits an ideal behaviour, i.e., a certain time-course representing the ideal performance (see Fig. 2). When we say fits, we intrinsically admit an amount of error, hence opening the doors to the realm of approximation methods [1, 2, 8, 6]. These methods aim to bridge the gap between rigid equivalence checking techniques and more relaxed requirements of real systems, and some of them deal with notions of behavioral equivalence for deciding if two systems behave almost (up to small errors or fluctuations) the same or, more formally, for measuring the distance between systems. One well-established approach uses pseudometrics, which give a measure of the similarity of systems that are not equivalent (see e.g. [8, 6]).

![Figure 2: a) Example of ideal behaviour of a single protein species; b) Example of ideal behaviours of a set of protein species.](image)

At this stage, anyway, we are not interested in comparing systems, but only systems against ideal behaviours. We rely on an approximated variant of the simulation preorder notion for deciding if a system almost simulates (fits) a certain ideal behaviour. Denoting with \( S_i \) a \( i \)-th discrete configuration of a system described in BlenX, an ideal behaviour (or more generally a set of ideal behaviours) can be represented as a trace of the form:

\[
T = S_0 \xrightarrow{t_0} S_1 \xrightarrow{t_1} S_2 \xrightarrow{t_2} \cdots \xrightarrow{t_{n-1}} S_n
\]
where \( t_i \) values represent times. It is clear how this trace definition captures the notion of time-course (or more generally of a set of time-courses). However, since we deal with times, it is quite obvious that traces can be used to represent behaviours at different time scales. For simplicity, we start by considering only traces at the same time scale of BlenX systems dynamics.

In order to develop further our fitness definition, we have first to introduce a notion of observables. At this stage we consider the two main properties we are interested to observe in a system configuration, namely the amount of protein species and protein domains. In our BlenX setting, classes of structurally equivalent boxes represent protein species while interfaces with same sorts represent protein domain classes. Denoting with \( S \) the system configuration reported in Fig.1(b), we indicate with \( \text{Obs}_s(S, B_4) \) and \( \text{Obs}_d(S, E) \) observations in \( S \) of, as an example, protein species \( B_4 \) and domains \( E \), respectively, obtaining:

\[
\text{Obs}_s(S, B_4) = 3 \quad \text{and} \quad \text{Obs}_d(S, E) = 11
\]

Note that given a protein species \( B \) and a protein domain \( A \), the functions:

\[
d^B_d(X, Y) = |\text{Obs}_s(X, B) - \text{Obs}_s(Y, B)| \quad \text{and} \quad d^A_d(X, Y) = |\text{Obs}_d(X, A) - \text{Obs}_d(Y, A)|
\]

are metrics on the set of BlenX system configurations. These notions of distance represent two of the ingredients we want to use to develop our approximated simulation relation. Given a bound \( \epsilon \), indeed, we can construct a relation that compares systems and ideal behaviours up-to fluctuations of protein species and protein domains amounts within the range \( \epsilon \). As an example, the following picture shows how a BlenX transition system trajectory match a trace up-to fluctuations in the amount of protein species \( B \) within the range \( \epsilon \).

\[
\begin{align*}
\text{trace} & \quad S_0 \xrightarrow{t_1} S_1 \xrightarrow{t_2} \cdots \xrightarrow{t_{n-1}} S_n \\
\text{trajectory} & \quad S'_0 \xrightarrow{\lambda_0} S'_1 \xrightarrow{\lambda_1} \cdots \xrightarrow{\lambda_{n-1}} S'_n
\end{align*}
\]

It is quite clear how traces differ from BlenX transition systems trajectories by the presence of times instead of stochastic rates on arc labels. Here we relate traces and trajectories by considering the expected values of the negative exponential distributions with parameters \( \lambda_i \). Given a bound \( \sigma \), if we have that the distances:

\[
|\Delta_1 = t_1 - \frac{1}{\lambda_1}| , \quad |\Delta_2 = \Delta_1 + t_2 - \frac{1}{\lambda_2}| , \quad \cdots , \quad |\Delta_{n-1} = \Delta_{n-2} + t_{n-1} - \frac{1}{\lambda_{n-1}}|
\]

are all less than \( \sigma \), then we know that the trajectory has an average time behaviour which is close to the one of the trace up-to fluctuations within the range \( \sigma \):

\footnote{Obviously, in formalizing these ideas we have to be sure that the size of \( \sigma \) respects some size relation with the traces times in order not to create inconsistencies in our calculations.}
At this point, given a set of protein species and protein domains we are interested to observe, a BlenX system $S'$ and a trace $T$, we can combine the two just given notions and say that $S'$ simulates trace $T$ with precisions $\epsilon$ and $\sigma$, only if $S'$ can match all the moves of $T$ up-to fluctuations in observations and time values within ranges $\epsilon$ and $\sigma$, respectively; Fig.3 gives graphical intuitions of what this relation looks like. By finding the tightest $\epsilon$ and $\sigma$ such that $S'$ simulates $T$ we can build a notion of distance (metric) between BlenX systems and traces; this allows to compare different systems with respect to the distance they have from a trace, given the set of observations we are interested in. This concept fits perfectly in our evolutionary framework and can be rephrased in terms of comparing different individuals with respect to the distance they have from an ideal behaviour, i.e., the notion of approximated simulation can be used to compute the fitness of an individual. Tightest is the precision of the simulation, greater is the fitness value. Obviously, since the precision depends on two values, we can think of finding a formula that combines them (e.g. a linear combination) allowing for the weighting of the precisions importance.

![Diagram](image)

Figure 3: a) Example of simulation where only a protein species is observed; b) Example of simulation where a set of protein species is observed.

Note, anyway, that the given definition of simulation says only that in the transition system of $S'$ there exists a trajectory that match the trace up-to fluctuations, but says anything about the probability to follow this trajectory with respect to all the possible trajectories.
of length \( n \) in the transition system (\( n \) is the length of the trace). This is quite annoying, because in our fitness computation we would like to consider not only that a system almost simulates an ideal behaviour, but also that the simulation is not a rare event. In order to obtain this, the idea is to use the previous definition of simulation and compute the probability \( p \) that a system \( S' \) has to generate a trajectory that simulates \( T \) with given precisions \( \epsilon \) and \( \sigma \). Also in this case we can develop a notion of distance between systems and traces by combining not only precision values, but also the probability \( p \); the idea is to find the tightest \( \epsilon \) and \( \sigma \) and at the same time maximize \( p \). Like before, we want a combination that allows weighting the three parameters. Note that by giving a weight zero to the probability value we should be able to recover the previous approximated simulation definition.

Although useful as a starting point to develop a theoretical framework for fitness computation, we think that this approach represents also a first step towards the achievement of certain kind of description of neutrality [9] in terms of some process-algebraic definition of functional or behavioural simulation or equivalence meaningful in the biological domain. This will clearly be of help in applying concurrency theory to study evolutionary robustness and evolvability, concepts recently recognized as crucial to the understanding of evolution [9]. We want to finish by mentioning a different process-calculi based work presented in [7], which aims to combine a variant of \( \pi \)-calculus, the continues \( \pi \)-calculus, and model-checking techniques as an approach to study robustness and evolvability of biological networks.

References


