Computational Nature of Processes Induced by Biochemical Reactions

Andrzej Ehrenfeucht[1], Grzegorz Rozenberg[1,2]

[1] University of Colorado at Boulder, USA
[2] Leiden University, The Netherlands
rozenber@liacs.nl

Natural computing is concerned with human-designed computing inspired by nature as well as with computations taking place in nature, i.e., it investigates phenomena taking place in nature in terms of information processing.

Well-known examples of the first strand of research are evolutionary computing, neural computation, cellular automata, swarm intelligence, molecular computing, quantum computation, artificial immune systems, and membrane computing.

Examples of research themes from the second strand of research are computational nature of self-assembly, computational nature of developmental processes, computational nature of bacterial communication, computational nature of brain processes, computational nature of biochemical reactions, and system biology approach to bionetworks.

While progress in the first line of research often contributes to important progress in Information and Communication Technology (ITC), advances in the second line of research often remind the general scientific community that computer science is also the fundamental science of information processing, and as such a basic science for other scientific disciplines such as, e.g., biology.

The research we present is concerned with the computational nature of biochemical reactions in living cells. In particular we investigate the computational processes inspired (based on) biochemical reactions.

On the level of abstraction that we adopt, the functioning of a biochemical reaction is based on facilitation and inhibition: a reaction can take place if all of its reactants are present and none of its inhibitors is present. If a reaction takes place, then it produces its product. Therefore a *reaction* is defined as a triplet $a = (R, I, P)$, where R, I, P are finite sets called the *reactant set of* a, the *inhibitor set of* a, and the *product set of* a, and denoted by R_a, I_a, and P_a, respectively. If S is a set such that $R, I, P \subseteq S$, then we say that a is a *reaction in* S.

Then a reaction a takes place (in a given state – a given molecular soup) if all of its reactants are present and none of its inhibitors is present. Consequently, for a finite set (state) T, a is enabled by T if $R_a \subseteq T$ and $I_a \cap T = \emptyset$. The result of a

on T, denoted by $res_a(T)$, is defined by: $res_a(T) = P_a$ if a is enabled on T, and $res_a(T) = \emptyset$ otherwise.

For a set A of reactions, the *result of A on T*, denoted $res_A(T)$, is defined by:

$$res_A(T) = \bigcup_{a \in A} res_a(T).$$

Finally, a *reaction system*, abbreviated rs, is an ordered pair $\mathcal{A} = (S, A)$ such that S is a finite set, called the *background set of $\mathcal{A}$*, and A is a set of reactions in S, called the *set of reactions of $\mathcal{A}$*. For a finite set (state) $T \subseteq S$, the *result of $\mathcal{A}$ on T*, denoted $res_{\mathcal{A}}(T)$, is defined by:

$$res_{\mathcal{A}}(T) = res_A(T).$$

The framework of reaction systems sketched above and motivated by organic chemistry of living organisms is based on assumptions that are very different from (and mostly orthogonal to) underlying assumptions of majority of models in theoretical computer science. We will discuss now some of these assumptions.

If a reaction a is enabled by a state T, then the result $res_a(T)$ is "locally determined" in the sense that it depends on R_a only. However, the effect of applying a to T is "dramatically global", because the whole set $T - P_a$ vanishes (to visualize this effect assume that the cardinalities of T, R_a, and P_a are 10000, 3, and 2 only; then 9998 elements of T will vanish while a has seen/used only 3 elements of T!!!). This is really orthogonal to models such as, e.g., Petri nets, and it affects our assumption that there is no permanency of elements: an element of a global current state will vanish unless it is sustained by a reaction.

When a set of reactions A is applied to a state T, the result of application is cumulative: it is the union of the results of all individual reactions from A. Note that we do not have here a notion of conflict between reactions in A: even if $R_a \cap R_b \neq \emptyset$ for some $a, b \in A$, then still both a and b contribute to $res_A(T)$ – there is no conflict of resources here. Again this is in strong contrast to standard models in theoretical computer science such as, e.g., Petri nets. This reflects our assumption about the "threshold supply": either an element is present, and then there is "enough" of it, or an element is not present. Therefore, there is no counting in reaction systems, and consequently, reaction systems is a qualitative rather than a quantitative model.

Finally, we note that in reaction systems reactions are primary while structures are secondary. We do not have permanency of elements and consequently, in transitions from state to state, reaction systems *create* states (rather than they transform states). Therefore, reaction systems do not work in an environment, but rather they create an environment.