The חי -Calculus: A Computational Formalism and its Relationship to Biological Processes

Samuel Arbesman
Santa Fe Institute
Brandeis University
arbesman@brandeis.edu

7th August 2003

Biotic systems have been described and analyzed for years. What is attempted here is the creation of a computational formalism which abstracts various principles of these simple systems and provides a consistent structure based on the theory of computation. Existent computational calculi are examined and found to fall short, based on certain criteria related to biological defensibility, which are held to be necessary for a proper abstraction. Therefore, the creation and description of a more biologically defensible calculus, the חי -calculus[1], is undertaken. This calculus is based upon the π-calculus, a formalism whose foundation is interaction, rather than function. The חי -calculus, its relationship to biology, as well as future directions for this new formalism are examined.

Introduction and Statement of the Problem

In modeling biological systems, there is a spectrum of complexity that dictates the nature of the model to be used. On one end there are simple systems with few agents and interactions; these systems are ideally modeled by such mathematical constructs as differential equations. On the other end are systems of an overwhelmingly large number of agents, each with simple states and interactions. Such systems are amenable to stochastic models. However, in between are a multitude of systems where the number of agents is large, the interactions complex, and the number of agent-states significant. For these systems, agent-based models are used. However, unlike the other systems, agent-based model construction is not guided by any mathematical framework or formalism. If such a formalism could be obtained, then this mathematical theory of agents and their interactions could be a powerful way of understanding such systems, including biological ones.

But what would such a formalism encompass and look like? In any biotic system, the interactions of the components of the system are neither capricious nor dictated by an external observer of

[1] חי (*chai*) is pronounced to rhyme with *pi* but with a "ch" sound similar to the one found in "Bach". The word חי is Hebrew for living or alive.

the system. The rules of physics are specified by the makeup of the components themselves. This well-known principle, that structure informs function, is vitally important to all biotic systems, and any computational formalism that attempts to express biology must have this capability. But this capability is an outgrowth of the fact of the existence of the very laws of Nature. Analogously then, these laws within the physical world are equivalent to the *syntax of interactions* within computation, by which I mean the possibilities of interaction and the resulting effects of these interactions. Computation is limited and guided by the syntactic possibilities within the formalism. [2] What is then necessary is a simple syntax of interactions, yet one rich with possibilities.

A computational calculus for biological interactions must be capable of expressing molecular interactions in a formal way. However, this computational formalism for biotic systems must not merely have descriptive capability (for if the nature of a network of interactions is unknown, it will be unable to be expressed). Instead, such a system requires a syntax of interactions that is sufficiently rich and biologically defensible such that biological processes may arise (such as catalysis, metabolism, self-persistence and replication). It should be a computational formalism that is abstract yet expressive of biological intuition. In this manner, biological interaction would emerge rather than be defined.[3]

The model of the lambda-calculus is a functional approach to biology.[4] It is based on input-output abstractions, but in so doing, the internal mechanisms are imperturbable to other functions within the system. In essence, this means that there is a lowest scale of interaction within a lambda-calculus system. On the other hand, process calculi such as the π-calculus are entirely made up of behavior, where every interaction is atomic and there is no inherent possibility of an inability to interact.[8] The lambda-calculus may be viewed as a theory of closed functions while the π-calculus is a theory of processes which may be interacted with at all levels. Viewing biology as processes that can interact all the way down is a powerful abstraction and the π-calculus will be evaluated for its utility as a biological formalism.

The π-Calculus: An Overview

Within the π-calculus, there are four basic components: channel-names, channels, and processes. Each channel has an end upon which communication may either be sent or received - these are referred to as channel-names. A process consists of sending or receiving communications along a channel. When there are processes for both sending and receiving on the same channel (both output and input channel-names), an interaction occurs and a communication occurs. The content of the communication are channel-names. The original processes are then eliminated and replaced with their continuations (their "tails", i.e. what their mission is upon interaction). To abstract then, a process is a context-dependent sequence of potential interactions, channel-names are the sites of interaction and the actual channel-formation and communication is simply the interaction itself.

[2] Within the theory of computer programs, what I term syntax of interactions actually includes both the syntax and semantics, but I would prefer to reserve the term semantics for the understanding of the biotic system, rather than simply the delineation and nature of the interactions.

[3] Others have their own definitions of what is necessary for such biological emergence systems (often viewed within the context of the field of artificial life). For one opinion, see [10].

More formally, the π-calculus is made up of processes. Most simply, the process can be the null process,**0**, which does nothing at all. For the most part, however, each process has a prefix name or co-name (signifying input or output, respectively), followed by a continuation, which is itself a process (preceded by a dot). For instance, the process $\overline{u}v.P$ is an output process (signified by the overline), on the channel u transmitting the channel-name v. However, this process is only able to send a channel-name if there is a concurrent process which is waiting to receive on channel u. If there is such a possibility though, the above process transmits and then continues as process P. An input process is represented like the form $u(x).Q$, where x is the variable which is replaced with whatever it receives (in this case, v), and then this variable is bound throughout the entire continuation Q.

Concurrent processes are separated by the par symbol, |, and two or more concurrent process are considered a single concurrent process. This allows the following to be written: $\overline{u}v.(v(x) \mid \overline{w}z)$ In addition, definitions may be used, allowing the following:

$$X \triangleq \overline{u}v.(v(x) \mid X)$$

In this way, processes can be grouped together and recursive definitions are even possible. Lastly, there are two operators that are relevant to this discussion, the replication operator, !, and the new operator, ν. The replication operator allows infinitely many copies of a process to be made, but only as needed, so that the process is not used up as it synchronizes over a channel. The new operator is used to create a private channel name, which is guaranteed not to be present within the current group of processes. This allows private communication without fear of unwanted interruption.

A brief example of the flow of some π-calculus processes is below:

$$\begin{gathered}
\overline{u}v.v(x).\overline{x}u \mid !u(x).\overline{x}w \\
\downarrow \\
v(x).\overline{x}u \mid \{v/x\}\overline{x}w \equiv \overline{v}w \mid !u(x).\overline{x}w \\
\downarrow \\
\{w/x\}\overline{x}u \equiv \overline{w}u \mid !u(x).\overline{x}w
\end{gathered}$$

For a more formal and comprehensive treatment of the π-calculus, please consult [6][7].

Shortcomings of the π-Calculus

Any computational formalism of a biotic system must be biologically defensible. Due to this requirement, there are a number of aspects of the π-calculus that are problematic. The use of replication is unreasonable as it begs the question of how to come up with replication or a self-maintaining network in such a system. Therefore, replication cannot be explicitly stated in a biologically-informed calculus.

The use of definition (and its resultant mechanism, recursion) is also not a biologically informed

mechanism and should not be used. But two aspects of definition must be examined - defining a grouping of processes, and defining the possibilities of interaction. The first is not biologically realistic because processes are not grouped purely based on fiat. They are grouped together due to the actual interaction of such processes. The second aspect of definition, enumerating the possible interactions, is clearly not a biological intuition. The syntax of interactions within any physical system is a built-in one, and to instead provide the syntax rather than it being an inherent one, is to merely describe biological systems (which is what these systems attempt to do) rather than express them. This eliminates many languages that attempt to model biological systems, such as Aviv Regev's BioSPI and Cosimo Laneve's Core Molecular Biology.[9][1][2]

In addition, the use of the "new" operator to create fresh channel names for private use, while useful in Regev's work to create coupling, is not a biologically acceptable notion either. The reason for this is that biological interactions occur or do not occur due to energetics and location, not due to an allowance or disallowance by commandment of such an interaction. Such exclusivity must arise within the system and not be due to primitives. Therefore, these three properties of the π-calculus as traditionally conceived (replication, definition, and creation of fresh names) must be eliminated as biologically infeasible.

Lastly, an interaction between two processes is dictated solely by whether or not the channel names and co-names are identical. While this is a syntax of interaction for sites of interaction, it is a rather simple one. However, if channel-names were no longer atomic but instead have structure that could dictate possible interactions, there could be some added richness to the system.

The three properties of the π-calculus as traditionally conceived (replication, definition, and creation of fresh names) must be eliminated as biologically infeasible (it is unclear how to deal with the atomicity of channel-names). Due to this pruning of the possibilities within the π-calculus, there are difficulties that preclude the creation of truly biological systems. Within this more limited π-calculus, a difficulty that arises is that of persistence. With the elimination of the replication operator, any process that succeeds in interacting is eliminated and replaced with its continuation. Since there is no possibility of recursion, this process can never again be recreated.

Within simple biomolecular systems, there is a nuanced understanding of persistence. Upon interaction, two molecules are no longer the same. Each has influenced the other and their possible future interactions have been altered as well, yielding new possible interactions. Yet at some fundamental level, there is persistence; neither molecule's components have vanished into thin air upon interaction. Therefore, the persistence of certain primitive reactants, such as atoms, are maintained. This careful (and rather qualitatively described) balance between change and persistence must exist in a proper computational formalism of biology. The persistence of something in the π-calculus is simply dictated by the length of its continuation. Since no continuation may be infinite, there is no persistence and no possibility of a biologically defensible system to arise.

A formalism which remains within the sphere of process calculi yet may be a solution to these problems is the חי -calculus.

The חי -Calculus: An Overview

The חי -calculus is similar to the π-calculus. Within the חי -calculus, there is some change in terminology, so that a process (being executed in parallel with the other concurrent processes) is called the *aggregate process*, or simply the *aggregate*. The processes found between the "dots" are termed the *constituents* or *constituent processes*. For example, $u\bar{v}.w(x).x\bar{y}$ is an aggregate process while $u\bar{v}$, $w(x)$, and $x\bar{y}$ are its constituent processes.

What then defines the חי -calculus? Constituent processes, rather than being eliminated (and replaced by their continuations), are simply rendered inactive. Upon the interaction of an output and an input channel, a "bond" (or couple) is created, which acts as an indicator of inactivation. Upon the action of a new operator within the חי -calculus, u^*, the communication channel u is removed and the two constituent processes at either end are reactivated for further interaction. If a bonded process is reactivated, the channel is also removed and both ends are reactivated for further interaction. Due to the nature of the bonding (processes being interconnected), while the חי-calculus may be expressed using nothing but a collection of strings, it is more easily rendered in a more graphical fashion, with lines connecting the bonded constituent processes.

Active constituent processes are indicated by a line beneath the process, $\underline{u\bar{v}}.w(x)$. Upon inactivation, any bound variable statements ($\{v/x\}$) are passed to the next constituent process within the aggregate, and the inactivated process is freed from its binding (to allow the variables to be bound to different names if it is ever re-activated). If an activation activates an already active process, the two activations merge to be one activated process (a process cannot be activated twice). If an aggregate consists of more than one process ($P \mid Q$), a non-deterministic choice is made as to which process remains with the inactivated portion, and which processes split away to become individual processes.

Appendix A contains a method of encoding the חי -calculus into the π-calculus, as a demonstration that the חי -calculus is sufficiently similar to other process calculi. Appendix B provides an example of how a simple חי -calculus example would be encoded into the π-calculus. But now, an example of how the חי -calculus works.

A חי -Calculus Example

1. $\underline{\bar{u}v}.(v(x).\bar{x}a \mid \bar{a}c) \qquad \underline{a(x)}.u^* \qquad \underline{u(x)}.\bar{x}b.b(y)$ starting conditions
2. $\bar{u}v.\underline{(v(x).\bar{x}a \mid \bar{a}c)} \qquad \underline{a(x)}.u^*$ a bond is formed and variables are bound
   $\mid$
   $u(x).\underline{\{v/x\}\bar{x}b}.b(y)$
3. $\bar{u}v.\underline{\bar{a}c} \qquad \underline{v(x)}.\bar{x}a \qquad \underline{a(x)}.u^*$ the way the processes split is non-deterministically chosen
   $\mid$
   $u(x).\underline{\{v/x\}\bar{x}b}.b(y)$
4. $\bar{u}v.\underline{\bar{a}c} \qquad \underline{a(x)}.u^*$ another bond is formed
   $\mid$
   $u(x).\bar{x}b.\underline{\{v/x\}b(y)}$
   $\mid$
   $v(x).\underline{\{b/x\}\bar{x}a}$
5. $\bar{u}v.\bar{a}c$ — $a(x).\underline{\{c/x\}u^*}$ another bond is formed
   $\mid$
   $u(x).\bar{x}b.\underline{\{v/x\}b(y)}$
   $\mid$
   $v(x).\underline{\{b/x\}\bar{x}a}$
6. $\underline{\bar{u}v}.\bar{a}c$ — $a(x).u^*$ a bond is broken

   $\underline{u(x)}.\bar{x}b.\underline{\{v/x\}b(y)}$
   $\mid$
   $v(x).\underline{\{b/x\}\bar{x}a}$

Analysis and Future Directions

So, what does the חי -calculus do for us? Well, unlike in the limited π-calculus discussed above, persistence in a process calculus is achieved in such a way that a process, instead of being eliminated, is simply inactivated (with the possibility of reactivation). Bonding is a graphical shorthand for the ongoing possibility of reactivation (similar to how an active process is the ongoing possibility of interaction).

In the חי -calculus then, what becomes important are the interactions and structural relationships between the aggregates. Each aggregate will always remain in the system (or reactor), but how it interacts (what it is bonded to, what portions are activated, etc.) is entirely dependent upon its context. This seems then to provide some parallels within biological processes. As mentioned above in regards to persistence, while processes change and the aggregate relationships are not invariant, no constituent (or even aggregate) ever vanishes into thin air. The aggregate processes are then akin to simple molecules (or even atoms) that can be fed into a system, with possibilities for interaction.

In addition, the formalisms of activation and inactivation (as well as bonding and coupling) have clear parallels within biological processes. Activating a process might be something like a conformational change within a protein, or a phosphorylation of a certain site. Bonding two processes inactivates certain regions similar to how two proteins might interact. Upon doing their interaction, the interface is hidden, making it unavailable for further interaction.

Furthermore, the flow of activation along a continuation may be thought of loosely as an energetic flow, an important aspect of biological systems. In any physical system, energy has the general tendency to flow downhill, and without the availability of more free energy, the system will eventually reach a stagnant equilibrium. So too within the חי -calculus, if there are no processes which continually provide for the possibility of reactivation, the system will eventually reach a steady state of inactive processes. This is similar to what happens to many non-biological systems after a certain amount of time.

One further direction for the חי -calculus which is clear is to create an actual computational instantiation for the calculus. As shown in Appendix A, the חי -calculus can indeed be encoded into the π-calculus, allowing the possible use of currently existent programming systems to be used to run the substrate and encoding upon which the חי -calculus may be run. One possible programming framework that could be used would be Lucian Wischik's Fusion Machine implementation of the π-calculus.[12, 11]

Once a חי -calculus system is in place, a חי -reactor could be created to actually see how such a system's interactions and connections would evolve. This would be similar to many artificial chemistry systems (for an overview, see [3]).

An ideal implementation of a חי -reactor would be a form of chemostat, which is a system where there is a continual influx and outflux of chemicals. This is similar to a small bounded area within an open non-equilibrium soup of chemicals. A continual influx of building-block materials seems to be a good methodology to use. As the חי -calculus has a notion of activation and inactivation - an energetics of sorts - it would be logical to provide a continual influx of activated chemicals (ones with unreleased free energy) to be used in order to create further interactions, connective networks, and perhaps even autocatalytic networks.

The resulting interacting systems, as well as the חי -calculus in general, could be analyzed mathematically using the tools already available for process calculi and theories of computation. In this way, a rigorous method of determining the relevant emergent properties of a חי -calculus system could be determined, and filter out those properties which are simply a byproduct of the definition of the calculus.

Lastly, a further direction to take this line of research is to attempt to create a calculus with more possibility for form and function relationships. In the חי calculus (as well as the π-calculus), an interaction between two processes is dictated solely by whether or not the channel names and co-names are identical. While this is a syntax of interaction, it is a rather simple one. However, if channel-names were no longer atomic but instead have structure that could dictate possible interactions, there could be some added richness to the system. Perhaps Girard's Geometry of Interaction, based on his linear logic, might provide this, but it is currently unclear.[5]

Therefore, the חי-calculus seems to be a powerful and promising formalism for simple biotic systems. Unlike other abstractions based on process calculi, it is not simply a formalism for description of molecular biological interactions. Instead, it provides the possibility of demonstrating how biotic processes interact, in a formalized and rigorous manner. An implementation of the חי -calculus seems a logical next step as well as the analysis of the resulting interacting systems.

References

[1] Vincent Danos and Cosimo Laneve. Core formal molecular biology. In *Proceedings of ETAPS 03*, 2003.

[2] Vincent Danos and Cosimo Laneve. Graphs for core molecular biology. In *Proceedings of International Workshop on Computational Methods in Systems Biology*, 2003.

[3] Peter Dittrich, Jens Ziegler, and Wolfgang Banzhaf. Artificial chemistries - a review. *Artificial Life*, 7(3):225–275, 2001.

[4] W. Fontana, G. Wagner, and L. Buss. Beyond digital naturalism, 1994.

[5] Jean-Yves Girard. Towards a geometry of interaction. *Contemporary Mathematics*, (92):69–108, 1989.

[6] R. Milner. The polyadic pi-calculus: a tutorial. In F. L. Bauer, W. Brauer, and H. Schwichtenberg, editors, *Logic and Algebra of Specification*, pages 203–246. Springer-Verlag, 1993.

[7] Joachim Parrow. *Handbook of Process Algebra*, chapter An Introduction to the Pi-Calculus, pages 479–543. Elsevier, 2001.

[8] Benjamin C. Pierce. *The Computer Science and Engineering Handbook*, chapter Foundational Calculi for Programming Languages, pages 2190–2207. CRC Press, 1997.

[9] Aviv Regev. *Computational Systems Biology: A Calculus for Biomolecular knowledge*. PhD thesis, Tel Aviv University, December 2002.

[10] Hideaki Suzuki, Naoaki Ono, and Kikuo Yuta. Several necessary conditions for the evolution of complex forms of life in an artificial environment. *Artificial Life*, 9(2):153–174, 2003.

[11] Lucian Wischik. *Explicit Fusions: Theory and Implementation*. PhD thesis, Computer Laboratory, University of Cambridge, 2001.

[12] Lucian Wischik. Fusion machine prototype. http:// www.wischik.com/ lu/ research/ fusion-machine, 2002.

Appendix A: π-Calculus Encoding of the חי -Calculus

The חי -calculus may be relatively simply encoded into the π-calculus. Each aggregate process (the first constituent is assumed to be the initially active part) is decomposed into its constituent processes, with a definition for each portion (relatively easily generated). The only additional part is to create a series of private channel names to allow the constituents of each aggregate to communicate, with these channels carrying all of the bound names found within the aggregate (in the encoding, this private channel series is common to the entire aggregate). In this way, the חי-calculus may be easily encoded.

A חי -calculus aggregate process P may be thought of being composed as follows (each part is a constituent):

$$P \equiv A_1.A_2.\cdots A_n$$

P may then be encoded into separate definitions for each of its constituents, $\mathbb{A}_1$, $\mathbb{A}_2$,..., $\mathbb{A}_n$. In addition, the list of all bound variables found in the aggregate process is β. [4] The doubled channel names (e.g. A_1A_1) are used to indicate a bonded and inactivated process (this method is used to encode the bond-breaking process, as seen below). The channel name σ is a private name that acts as a synchronizer between bonded constituents so that if one end is activated, the other end is activated as well. [5]

$$\begin{aligned} u^* &\triangleq \overline{uu} \\ u(x) &\triangleq u(x, \varsigma_1, \varsigma_2) \\ \bar{u}v &\triangleq (\nu\sigma_1, \sigma_2)\bar{u}(v, \sigma_1, \sigma_2) \end{aligned}$$

[4] This encoding assumes that each constituent is only one process. If the constituent contains more than one parallel process, however, it is a simple matter to encode the non-deterministic choice of which process remains with the aggregate and which ones go free by using the sum (+) operator. Each possibility of which aggregate the constituent is part of will also be encoded.

[5] In the lower of the two lines in the encoding, σ is either the variable itself or the bound name- depending on whether A_1 is an output or an input channel, respectively.

Three cases:

Output Process:

$$\mathbb{A}_1 \triangleq (\nu x_1)(A_1.(\overline{x_1}\beta \mid \sigma_1().\mathbb{A}_1))$$
$$\mathbb{A}_n \triangleq x_{n-1}(\beta).\mathbb{A}_{tail}$$
$$\mathbb{A}_{tail} \triangleq (\nu x_n, \sigma_1, \sigma_2)(A_n.(\overline{x_n}\beta \mid \sigma_1().\mathbb{A}_{tail} \mid x_{n-1}(\beta).\overline{\sigma_2}.\mathbb{A}_{tail}))$$

Input Process:

$$\mathbb{A}_1 \triangleq (\nu x_1)(A_1.(\overline{x_1}\beta \mid a_1a_1().\overline{\varsigma_1}.\mathbb{A}_1 \mid \varsigma_2().\mathbb{A}_1))$$
$$\mathbb{A}_n \triangleq x_{n-1}(\beta).\mathbb{A}_{tail}$$
$$\mathbb{A}_{tail} \triangleq (\nu x_n)(A_n.(\overline{x_n}\beta \mid \varsigma_2().\mathbb{A}_{tail} \mid a_na_n().\overline{\varsigma_1}.\mathbb{A}_{tail} \mid x_{n-1}(\beta).\overline{\varsigma_1}.\mathbb{A}_{tail}))$$

Bond-Breaking Process:

$$\mathbb{A}_1 \triangleq (\nu x_1)(A_1.\overline{x_1}\beta)$$
$$\mathbb{A}_n \triangleq (\nu x_n)(x_{n-1}(\beta).A_n.(\overline{x_n}\beta \mid \mathbb{A}_n))$$

This simple encoding is able to simulate a group of חי -calculus processes within the π-calculus.[6]

[6]Due to the nature of the encoding and the use of private names, there will be unusable processes remaining in the system. In a reasonable implementation, this would cause the need for a garbage collector.

Appendix B: A Simple System in the חי-Calculus and its π-Calculus Encoding

$$\underline{u(x)}.u^* \qquad \underline{\bar{u}v} \qquad \underline{\bar{v}w}$$

$$\begin{aligned} A_1 =& u(x) \\ A_2 =& u^* \\ B_1 =& \bar{u}v \\ C_1 =& \bar{v}w \end{aligned}$$

$$\begin{aligned} \mathbb{A}_1 \triangleq& (\nu a x_1)(u(x, \varsigma_1, \varsigma_2).(\overline{a x}_1 x \mid u u().\bar{\varsigma}.\mathbb{A}_1 \mid \varsigma_2().\mathbb{A}_1)) \\ \mathbb{A}_2 \triangleq& a x_2().\overline{u u}.(\overline{a x_2} \mid \mathbb{A}_2) \\ \mathbb{B}_1 \triangleq& (\nu b x_1)(\nu b\sigma_1, b\sigma_1)(u(v, b\sigma_1, b\sigma_2).(\overline{b x}_1 x \mid b\sigma_1().\mathbb{B}_1)) \\ \mathbb{C}_1 \triangleq& (\nu c x_1)(\nu c\sigma_1, c\sigma_1)(v(w, c\sigma_1, c\sigma_2).(\overline{c x}_1 \mid c\sigma_1().\mathbb{C}_1)) \end{aligned}$$

The above system will never be inactivated (it continuously regenerates itself).