The barrier of objects: From dynamical systems to bounded organizations

Walter Fontana

Theoretical Chemistry
University of Vienna
Währingerstraße 17
A-1090 Vienna, Austria
and
International Institute for Applied
Systems Analysis (IIASA)
Schloßplatz 1
A-2361 Laxenburg, Austria

walter@leonardo.tbi.univie.ac.at

Leo W. Buss

Department of Biology
and
Department of Geology and
Geophysics
Yale University
New Haven, CT 06520-8104, USA

buss@muggiea.biology.yale.edu

Contents

Overview **3**

1 The barrier of objects **4**

2 Towards a specification language for chemistry **8**

2.1 Minimal Chemistry Zero 11

2.1.1 Ontological commitment, resultant metaphor and formal representation 11

2.1.2 Model 12

2.1.3 Main results 14

2.1.4 Main limits 20

2.2 Minimal Chemistry One 21

2.2.1 Shape and action 21

2.2.2 What is a type? 23

2.2.3 Improved metaphor 24

2.2.4 Model and preview of results 25

2.3 Minimal Chemistry Two 25

2.3.1 From λ-calculus to proof-theory 27

2.3.2 Ontological commitment, resultant metaphor, and formal representation 29

2.3.3 Addressing prior limits in the linear logic framework . . . 33

2.4 A Roadmap from chemistry to proof-theory 38

3 From dynamical systems to bounded organizations: The thread from chemistry ... **40**

3.1 ... to the "object problem" 41

3.2 ... to the foundations of mathematics 42

3.3 ... to concurrency and self-organization 47

3.4 ... to biology and beyond 50

References **52**

Appendix **59**

A λ-calculus for tourists **59**

A.1 Conceptual 59

A.2 Instant Syntax and Semantics 60

A.3 Beyond λ 64

B Types for tourists **66**

B.1 The chemistry of types 66

B.2 Polymorphism 67

B.3 Type inference 68

C Logic background **69**

C.1 The Curry-Howard isomorphism 69

C.2 Sequent calculus 72

C.3 Linear logic for tourists 76

C.3.1 The rules of the game 80

C.3.2 Proof-nets 81

Overview

Self-maintaining natural systems include the global climate system, all living organisms, many cognitive processes, and a diversity of human social institutions. The capacity to construct artificial systems that are self-maintaining would be highly desirable. Yet, curiously, there exists no readily identifiable scientific tradition that seeks to understand what classes of such systems are possible or to discover conditions necessary to achieve them. Given the ubiquity of such systems naturally and the desirability of self-maintenance as a feature of design, any credible approach to establishing such a tradition merits serious attention.

We have recently developed and implemented a framework for approaching the problem [26, 27]. It is based on the premise that *the constituent entities of a self-maintaining system characteristically engage in interactions whose direct outcome is the* **construction** *of other entities in the same class. Self-maintenance, then, is the consequence of a constructive feed-back loop: it occurs when the construction processes induced by the entities of a system permit the continuous regeneration of these same entities* [88]. The specific functional relationships between entities which collectively insure their continuous regeneration, we define as an **organization**. A theory of organization, so defined, is a theory of self-maintaining systems. A prototypical instance of entities are molecules. And organisms are a particularly interesting class of self-maintaining systems generated by their constructive interactions. The atmosphere is another example. And so, perhaps, is the sun at the nuclear level.

The overarching long-term goal of our program is *to develop a formal understanding of self-maintaining organizations.* Our efforts in doing so, which we summarize here, have led us to appreciate a fundamental problem in methodology: the traditional theory of "dynamical systems" is not equipped for dealing with constructive processes. Indeed, the very notion of "construction" requires a description that involves the structure of objects. Yet, it was precisely the elimination of objects from the formalism that make dynamical systems approaches so tremendously successful. We seek to solve this impasse by connecting dynamical systems with fundamental research in computer science, whose theoretical foundations are about "objects" and their constructive interrelations. Our long-term goal, then, becomes equivalent to the task of expanding dynamical systems theory to include object construction, to become what we have come to call *constructive dynamical systems* [26].

1 The barrier of objects

The vast bulk of knowledge base of classical physics has been earned by application of the tools of dynamical systems theory. It began with Newton, and became a powerful tool-kit with Hamilton, Jacobi, and Poincaré. Like all major perspectives in science, its power derives from a useful decision about what constitutes "the system" and what belongs to "the rest of the world." The characteristic feature of dynamical systems theory is to conceptualize "the system" as existing exclusively in terms of *quantifiable* properties (e.g., position, concentration) of interacting entities (real or abstract). The distinction in representing interaction between entities via their properties as opposed to some appropriate theory of the entities themselves will play a major role in what follows. The point is subtle. In a dynamical system, it is *not* the interacting entities that participate *as objects* in the formal constitution of "the system", but rather their quantitative properties and couplings. As a consequence, interaction is understood as the temporal or spatial change in the numerical value of variables. This change is captured by a set of (deterministic or stochastic) differential (or difference) equations. The solutions of these equations may then be viewed as a flow in phase space. Analytical and numerical tools exist which permit the characterization of that flow and its change as parameters are varied (e.g., invariant subspaces, attractors and repellors, basins of attraction, bifurcations). In the centuries since Newton, our own century most prominently, the power and efficacy of this cognitive style has been established beyond all question.

The success of this framing in physical systems has fueled an inexorable export of the dynamical systems approach from physics to virtually every domain of biological, cognitive and social science. The record of achievement in these other domains has been mixed at best. To what may we attribute this apparent "limit to scientific knowledge"? A variety of attributions to both specific and general failure are so commonplace as to have become tiring to once again repeat. Many failures in domains of biological (e.g., development), cognitive (e.g. organization of experience), social (e.g., institutions), and economic science (e.g., markets) are nearly universally attributed to some combination of high dimensionality and nonlinearity. Either alone won't necessarily kill you, but just a little of both is more than enough. This, then, is vaguely referred to as "complexity".

Laying the blame for scientific limits in this common waste bin, however, has an uncomfortably facile texture. After all, there are examples of wildly successful application of dynamical systems approaches to problems that must have seemed no less daunting at the time than, say, predicting the evolution of the telecommunications market or the global climate looks today. Consider the efforts of R. A. Fisher, whose application of dynamical systems to the problem of combining Mendel's genetics with Darwin's evolution yielded his "genetical theory of

natural selection" (i.e., what we now know as the field of population genetics). Fisher accomplished his task via an act of abstraction. His genius was to claim that organisms were an utter distraction and irrelevancy, simply not a part of the problem. The concordance of Darwinism and Mendelism required only a population and genes; the concentration of the latter in the former is the relevant variable, and its behavior may be had by solving differential equations wherein the frequency of a gene is jointly determined by Mendel's transmission rules and Darwin's selection. Fisher's accomplishments belie the conventional mantra of "too complex". Perhaps the "limits to scientific knowledge" are simply a deficit of genius. None of us retain into adulthood a capacity to *seriously* attend for prolonged intervals to an imagined system of abstract entities; whereas every youngster quite seriously attends to the closetful of monsters that appear each nightfall. We rightly celebrate as genius the (first) man who saw genes disembodied from the organism!

In seeking to understand why dynamical systems have had only modest impact in some sciences, the usual explanations are, in some considerable degree, internal to a dynamical systems representation itself. The failures are cast as failures in applying the dynamical systems approach - either a failure of insight in imagining abstract entities appropriate to the system or a failure in tools for the qualitative analysis of high-dimensional, nonlinear differential equations. However, no less real a limit is our ability to stand outside a dynamical systems perspective for a moment and to seriously ask what is it good at and what is it not. Perhaps, then, we might augment the cognitive style itself to render it more tractable in those domains where its achievements have been heretofore limited. This is our intent.

What is left out of thinking about the physical universe as one massive dynamical system with our understanding of it limited solely by insights in framing abstractions well-suited to carving off soluble subsystems? Perhaps what is being too easily overlooked is the fact that dynamical systems never deal with objects themselves [83]. Objects are never represented as entities with a distinct internal structure giving rise to behavior. Rather, objects disappear into arrays of structureless variables confined to holding numerical values that quantify properties of an object class, such as the frequency of a gene, the concentration of a chemical, the density of an electromagnetic field, the position and velocity of an aircraft, the pressure of a gas, the earnings of a firm. The moon, for example, is never represented as an object in the equations that express its orbit; the "moon" is defined as a time-dependent vector of numbers specifying position and momentum. Numerical values are indeed an appropriate abstraction, but *only as long as objects don't change*. Planets interacting gravitationally or Fisher's genes interacting in accord with transmission and selection serve as examples. The situation is quite different when objects possess an internal structure that is

subject to change, particularly when that change is endogenous to the universe of objects considered, i.e., when the internal structure of an object causes specific actions to occur that modify (or create) other objects.

Conventional dynamical systems, then, are well-suited to treat changes in the magnitudes of quantitative properties of fixed object species, but ill-suited to address interactions that change the objects themselves. The latter is challenging in the dynamical systems context. The relevant "variables" would have to hold objects, rather than the familiar numerical values. But if the objects become the variables of the system, we would need a "calculus of objects" like we have a differential calculus for numerical values. This places a high premium on the difficult task of abstracting objects without losing the link between their action and structure (i.e., without losing the objects). Perhaps herein lies our seeming deficit of genius.

In Nature, interaction involves objects directly and never by a numerical value describing them. Stepping outside of conventional dynamical systems requires taking this observation seriously. Stated less rhetorically, the occurrence of objects that possess a distinct internal structure of a combinatorial kind has two implications. First, there are substantially more possible objects than can be realized at any given time. It is this which gives meaning to the notion of a "*space of objects*"[1]. Second, and most importantly, when the interaction among objects *causes* the construction of further objects, relations of production tied to their internal structure become possible. This never appears in a conventional setting: it can only arise as a consequence of a causal linkage between the internal structure of an object and the actions through which it participates in the construction of others. A theory of such linkage is what a "calculus of objects" would have to accomplish. If we throw out the *constructive* component, we throw out the capacity of a system to *endogenously* induce a *motion* in its "space of possible objects"[2].

What is gained may be seen by analogy to conventional dynamical systems. We imagine construction relations (the analogue of the differential operator) to induce a flow in a "space of possible objects " (the analogue of phase space). The intuition is that this flow will have a structure where collectives of objects implementing particular production relations form "attractors" (i.e., "fixed-points", "limit-cycles", and the like) with corresponding "basins". If so, then objects

[1]Eigen [21, 22] has introduced this notion for the special case of nucleic acid sequences - the "sequence-space". Maynard-Smith [60] thought of the same in the context of proteins.

[2]Throwing out construction still leaves room for chance events, such as mutation, to induce a motion in object space. The deeper theoretical and conceptual issues arise when the construction of objects derives from the *interaction* among existing ones, not from their variation by chance. The former makes the motion in object space endogenous, while the latter makes it exogenous to the system. Mutation is to construction like perturbation is to dynamics.

which change one another upon interaction - as surely is the habit of elementary particles, molecules, neurons, firms and governments - have the potential of being characterized and studied as organized collectives of construction relations. The question becomes: Do such organized collectives exist? If so, what are they and what are their properties? Are some self-maintaining, self-repairing, and capable of extension? Is their extension constrained by their internal structure, their history of extension, or both? Are they helpful in filling the void that steadfastly remains in the biological and social sciences, despite the wholesale importation of dynamical systems approaches into domains so manifestly rich in object construction and transformation?

The issue posed above mandates that the constructive aspect of interaction be brought into the picture. This necessarily requires the representation of objects. In seeking formalisms appropriate to facing the issue of object construction and transformation, one is invariably drawn to the foundations of computation. The computational sciences deal explicitly with syntactical entities, and, thus, with the possible representations of objects and their construction. This defines, then, our specific approach to the general problem. We are obligated to define objects, using formalisms borrowed (at least at the outset) from theoretical computer science, to animate their interaction in an appropriate dynamical setting, and to thereby generate a "motion in a space of objects", the features of which we desire to explore.

All that follows is but a progressive refinement of research tactics we are employing in an attempt to explore this larger question in a specific instance. We concern ourselves with the biological domain, specifically thinking of organisms as self-maintaining chemical collectives. Hence we treat molecules-as-objects and search a corresponding "space of objects" for self-maintaining collectives. We first motivate this choice and show how the simplest abstraction of molecules as agents of construction does indeed generate collectives with a distinctively biological flavor. From this basis, we outline progressive refinements in our abstract chemistry in the form of alternative syntactical systems with the aim of closing the distance from our simplest abstraction of chemistry to something more respectful of chemistry as we know it. After documenting a concrete implementation of the broader perspective in the specific instance of chemistry, we return in conclusion to the larger issues. The reader is urged not to lose sight of the larger goal while immersed in the specific instance: the long chemical excursion is but a logbook of data in support of the utility of the broader view. The proffered "motion in a space of objects" and associated universe of organizations composed of such objects is hardly exclusive to the objects of chemistry and their resultant biological organizations. To make substantive progress - whether in biology or in much of what is beyond biology - we must distinguish and capture the fundamentally different consequences that arise when change is about the objects

themselves, as opposed to the magnitude of prespecified quantitative properties describing them.

2 Towards a specification language for chemistry

Our overall goal is more readily grasped and the methodological challenges more concretely framed when stated in the context of a specific class of objects-that-change-objects and a specific organized collective of such objects. Our starting point will be chemistry; the relevant entities are molecules and organizations are self-maintaining chemical collectives. Our motivation in this choice is twofold. First, we chose chemistry because it is solid ground: we know molecules and their interactions far better than any other object class claimed to participate in the construction of self-maintaining organizations (contrast the challenge of molecules versus the challenge of cognitive entities generating markets or firms, for example). Second, *we believe that biology, particularly molecular biology, has a pressing need for support from a new kind of theoretical chemistry.* Current quantum and structural chemistry are burdened with information that is not relevant to the molecular biologist. The level of detail and the kind of description offered by these approaches necessarily put the focus on single molecules or individual reactions and away from their functional context within organized systems of molecules or reactions. What chemistry lacks is a high level *specification language* focused on the abstract operational aspect of molecules and capable of describing reaction networks and their *algebraic* behavior. The molecular biologist needs a tool for abstracting molecular actions, for plugging them together (like electronic components), and for generating and analyzing the network closures of these actions under a variety of boundary conditions.

While absence of such a specification will be all-too-apparent to biologists, an example may prove useful to others. Let us consider the role that a yet-unrealized theory of network construction and maintenance might play in understanding how self-maintaining molecular organizations evolve. The scenario is conventional: a mutation occurs, which results in a new_1 gene sequence coding for a new_2 protein whose interaction with the chemical machinery of the cell, set up by the remaining gene products, triggers a cascade of new_3 chemical reactions resulting in a new_4 extension of a metabolic pathway which enables the utilization of a new_5 resource.

Each time the word "new" denotes a different kind of novelty, because each time different kinds of constraints are in effect:

1. novelty_1: A sequence is a combinatorial object with the simplest possible structure: a linear concatenation of symbols. The syntactic category of

"sequence" entails a space of possible variations. A chain of 200 positions over an alphabet of four symbols has 4^{200} ($= 10^{120}$) realizations - more than the number of bosons in the universe. At this level the generation of novelty is virtually unconstrained. Any random replacement of any symbol at any position yields a new_1 sequence.

2. novelty_2: A protein is more than a sequence of symbols. It is a sequence that folds into a shape as a consequence of interactions between symbols along the chain. Three-dimensional space and the nature of intramolecular forces severely constrain which shapes are possible. At chemically relevant levels of resolution these constraints result in considerably fewer stable shapes than sequences. Not every novelty_1 is a novelty_2.

3. novelty_3: The types of functional groups and their disposition within a molecule define its "domain of interaction" - its capacity to participate in specific chemical action (i.e., the breaking and making of bonds). Novelty_3 is a matter of chemistry.

4. novelty_4: The constraints and opportunities of interaction within a given network of chemical pathways determine which new_4 network roles a new_3 molecular agent can participate in. How (or, even, whether) a network forms depends on its molecular components, the types of reactions induced by them, the connectivity of these reactions and their kinetics.

5. novelty_5: The innovated_4 metabolic network is characterized by constituent molecules and their relationships. What is regarded, however, as a new_5 "resource" or as new_5 "waste" is a matter of the coupling between this network and other such networks either within the same, or between it and other, levels of biological organization. Indeed, it is the joint construction and maintenance of a chemical reality composed of a large number of linked metabolic networks which defines the biotic element of an environment.

It is plain that novelty_5 cannot occur unless novelty_1 occurs. There is, however, a gap between novelty_1 and novelty_5 which theory is presently unable to bridge. We perfectly understand the "abstract space of possibilities" for novelty_1: it's the space of words over an alphabet. Yet we have basically no clue as to even the nature of the abstract space of possibilities for novelty_5. The two loose ends of the problem circulate in biology under the key-words "genotype" (novelty_1) and "phenotype" ($\text{novelty}_{i>1}$). The evolutionary process is perceived roughly as the conjunction of two factors: the modification of "phenotypes" by chance events at the level of "genotypes", and a dynamics which results in the selective amplification of "genotypes" based on the differential reproductive success conveyed by their "phenotypes". Novelty_1 is as simple as a throw of the dice. However, once it has occurred, we lack utterly the capacity to assess its likelihood

of giving rise to novelty$_5$. Yet, this likelihood is defined by a molecular society, the constituent interactions of which have a lawful - even largely deterministic - character grounded in physics and chemistry. Might there not be an abstraction of chemistry appropriate to such questions?

We claim that *any serious attempt to mathematize such questions requires an abstract characterization of chemical processes.* This stance defines our more specific goals:

- to develop an "abstract chemistry" in which molecules are viewed as computational processes supplemented with a minimal reaction kinetics, and
- to develop a theory of the self-organization, maintenance, and variation of networks based on such processes.

Situating these specific goals in the broader perspective, we believe that an adequate abstraction of chemistry is as crucial in extending the theoretical foundations of biology as was an adequate abstraction of motion in founding a formal basis for physics. The parallel, however daunting, is one we make seriously. Roughly, "motion" in physics is conceived as the temporal change in the value of a state variable (the position, say). This motion is formalized by infinitesimal calculus; a theory of the derivative $d \cdot /dt$. We would like to think of chemical reactions as a kind of "motion" as well - but as a motion in a *space of objects.* The key difference is that, mathematically, such objects are not numerical quantities, they are syntactical entities, to wit: molecules. The chemist denotes that motion with "$\longrightarrow$", as in $CH_3OH+CH_3COOH \longrightarrow CH_3COOCH_3+H_2O$. The objects on the left are replaced by those on the right. But these may interact further with other molecular agents present in the reaction vessel or the cell, thereby keeping its contents changing over time, that is, "moving in object space". What is needed is a theory of the motion generator "$\longrightarrow$" competent to define a universe of self-maintaining organizations of such objects. If the broader perspective is correct and the specific implementation sufficiently exact, within this universe of organizations will be found specific organizations known to us as living biological systems. The reader will find grounds in our simplest implementation in support of the validity of the broad claim, but will find manifest inadequacies in the precision of the specific chemical implementation presented in section 2.1. Optimism inspired by success in the former has motivated the series of refinements summarized in sections 2.2 and 2.3 in attempt to improve upon the deficits of the latter.

2.1 Minimal Chemistry Zero

2.1.1 Ontological commitment, resultant metaphor and formal representation

In view of the above discussion, the principal question is "how to frame chemistry?" The problem is one of focus – how close-up? how distant? – and one of scope – how wide? how narrow?. We are forced to make ontological choices. To begin with, we choose an absolutely minimalist view of chemistry.

1 - Syntactical. Molecules are treated as discrete structures of symbols, defined inductively. A molecule is an atom or a combination of molecules.

2 - Constructive. Reactions are seen as events where such symbolic structures "interact" to construct new symbolic structures.

3 - Substitution. The basic mechanism of a reaction is the exchange of one group of symbols (a substructure) by another, i.e., a substitution.

4 - Equivalence. Different combinations of reactants can yield the same product.

5 - Deterministic. When particular functional groups in a molecule initiate a reaction, the product is determined.

This minimalist view coincides with the description of a mathematical function as a rule, rather than a set. In the former case a function is a suite of operations that generate an output when applied to an input. In contrast, the latter case views a function as a look-up table, i.e., a set of input/output pairs. To express rules, a syntax is needed (point 1). Functions-as-rules can be applied to arguments which can themselves be functions, returning a new function as a result (point 2). For example, take a polynomial and "apply" ($\circ$) it to another one: $(x^2 + 4x + 2) \circ (y - 1) \longrightarrow (y-1)^2 + 4(y-1) + 2 \longrightarrow y^2 - 2y + 1 + 4y - 4 + 2 \longrightarrow \cdots \longrightarrow y^2 + 2y - 1$. This also illustrates that the process of evaluating the application of a function to an argument is done by repeated substitutions, where the formal variable of a function is replaced by the literal text of the argument (point 3). The schemes which govern this process define a calculus. Furthermore, different function/argument combinations can return the same result. Trivially, $7 + 3 = 20/2$, but "$7 + 3$" is not the same object as "$20/2$". To justify the equality a syntactical manipulation - a computation - must occur that puts each object into its canonical form: $7+3 \longrightarrow 10 \longleftarrow 20/2$ (point 4). Furthermore, the application of a particular function to a particular argument always yields the same result (point 5), although it may proceed via different routes (depending on which subexpressions are evaluated first).

This, then, is summarized by the following metaphor:

chemistry ·····	**a calculus**
physical molecule ·····	symbolic representation of an operator
molecule's behavior ·····	operator's action
chemical reaction ·····	evaluation of functional application

To put the metaphor to work, it must be made precise. Any formal system that is a candidate for an abstraction of chemistry at this level must make the same ontological choices. The *only* canonical system known that formalizes the notion of a function as a rule, and is both based on substitution and naturally yields a theory of equality is λ-calculus. λ-calculus was invented in the 1930's [12, 13, 14, 82], and has since become of foundational importance in the computational sciences. We find it remarkable that there is a system at all - and even such a central one - that fits so well. The correspondence with λ-calculus will later enable substantial refinements of the chemistry/calculus metaphor, thereby providing an *ex post* justification of this choice.

Although not strictly necessary to grasp most of the remaining chapter at an intuitive level, the reader unfamiliar with λ-calculus is invited to appendix A.

2.1.2 Model

Motivated by biological problems akin to that sketched at the beginning of section 2, we have developed and implemented a toy model aimed at exploring the conjunction of the two interaction modes – construction and dynamics – introduced in section 1.

Our abstract molecules are symbolic operators expressed in λ-calculus. We consider a "flow reactor" of N such abstract molecules, each one a λ-expression. In this setting a given expression may occur in multiple instances, just as in a test tube a number of molecules may be instances of the same chemical formula. We think now of the expressions as if they were particles floating within a well stirred solution where they collide at random. Upon contact, two expressions interact by functional application, such that one expression assumes the role of operator, and is applied to the other expression which assumes the role of argument. The evaluation of this interaction yields as its result a new expression. Thus, the canonical calculus realizes the desired constructive component – collisions (i.e., "applications" followed by reduction to normal form; see A.2) *are* production relations among abstract molecules – and these occur in a particular dynamical setting (i.e., the flow reactor) such that construction is coupled to changes in the

concentration of the expressions.

We omit details of our implemention not essential for the purpose of this overview. They can be found in [26, 27]. One issue is, however, immediately germane. Recall that we wish to induce a "motion in object space", with that motion settling upon self-maintaining systems of objects. To achieve the latter, we impose a generic selection constraint on object motion through a choice of reaction kinetics and a restriction on reactor size. Specifically, we make two assumptions:

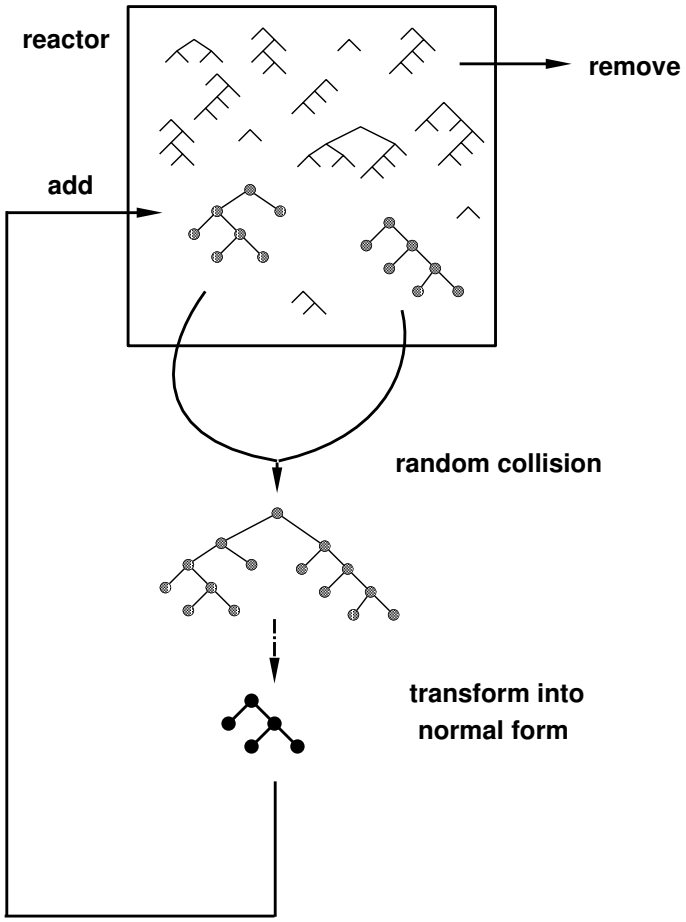


Figure 1: λ-calculus flow reactor. Two expressions A and B are chosen at random and a new object, $(A)B$, is constructed by "application" (see appendix A.2). Putting $(A)B$ into its normal form by β-reduction (see appendix A.2), effectively decides which object species (i.e., "stable molecular formula") the new object is an instance of.

- Reactants are *not* used up in a reaction:

$$A \ + \ B \ \longrightarrow \ C \ + A \ + \ B \tag{1}$$

where C is the normal form result that is contingent on the application of operator A to argument B: $(A)B \rightarrow C$ in λ-calculus. In this way the total number of expressions increases by one with each reactive collision.

• Each time a new expression has been produced, a randomly chosen one, X, is removed from the reactor:

$$X \longrightarrow \emptyset \tag{2}$$

The overall number of expressions N is thereby kept exactly constant. This means that each expression has a finite life time, even though it is not consumed at the moment of a reaction. Moreover, since any two expressions interact to produce a particular third expression with a frequency proportional to their concentration, the reaction scheme together with fixed reactor size act to favor convergence to a population of expressions whose relations of production yield expressions extant within the reactor – the motion in object space settles upon a set of objects that produce one another.

The reaction scheme, however, does obvious violence to the chemical metaphor. Indeed, the present metaphor and its instantiation through λ-calculus have a number of limitations which we discuss further in section 2.1.4 and which largely motivate the refinements in methodology outlined in sections 2.2 and 2.3.

2.1.3 Main results

Self-organized algebras and kinetic confinement

The intended motion-in-a-space-of-objects settling upon self-maintaining-sets-of-objects was observed [26, 27]. We focus first on the different kinds of λ-expressions in the reactor. As reactions proceed new expressions are generated, while others disappear due to the removal flow. Depending on the initial conditions, and after many interactions have occurred, the system frequently converges on an ensemble of λ-expressions that

(i) maintain each other in the system by mutual production pathways, and that

(ii) share invariant syntactical and algebraic regularities.

The latter means that the contents of the reactor have reached a particular (possibly infinite) subset of the space of λ-expressions that is invariant (closed) under interaction.

Syntactical regularities are made explicit by parsing expressions into two kinds of building blocks, called terminal elements and prefixes[3] [26, 31]. Terminal elements are closed λ-expressions (also called combinators, see appendix A). Prefixes are

[3]We define a terminal element to be the smallest closed subexpression reading from the end

not complete λ-expressions. However, prefixes form closed expressions when they precede a terminal element. The invariant subspace contains only expressions that are made from a characteristic set of such building blocks.

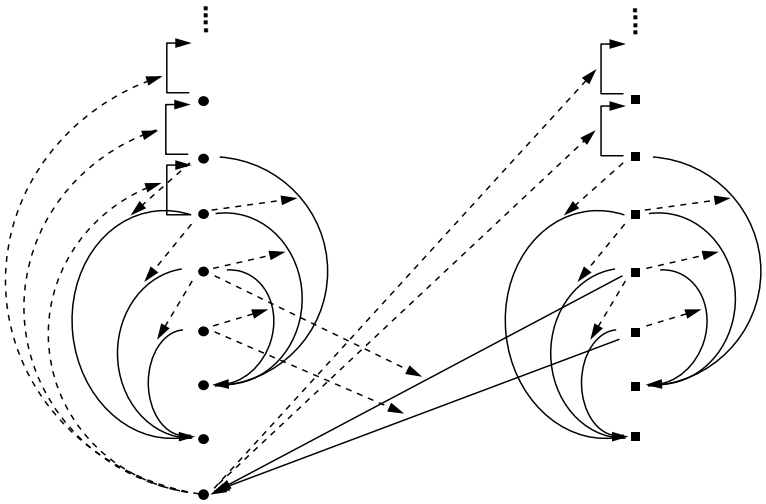


Figure 2: A simple self-maintaining organization. The dots (left) and the squares (right) represent λ-expressions with a particular grammatical structure. They are made of one prefix $[\lambda x.(x)]$ and two terminals $[T_1 \stackrel{\text{def}}{=\!=} \lambda x.x$ and $T_2 \stackrel{\text{def}}{=\!=} \lambda x.\lambda y.(y)\lambda z.(z)x]$. From bottom to top, the dots (left) are expressions consisting of an increasing number of prefixes (starting with 0 at the bottom) terminated by T_2. The same holds for the squares (right), except that they are terminated by T_1. A solid arrow indicates the transformation of an argument (tail) to a result (tip) by an operator (dotted arrow). For clarity, only a subset of the possible interrelations is shown. Notice the connectivity enables kinetic confinement. Most transformations yield objects at the bottom (leading to an increasing concentration profile from top to bottom). Some operations, however, yield objects up the "ladder", thus establishing self-maintenance. Both syntactical families depend on each other for maintenance as indicated by the "cross-family" connections.

Algebraic laws are a description of the specific action(s) associated with each building block. This action may depend on the context of a building block within an expression. The characterization of the functional relationships among the blocks yields a system of rewrite equations [50]. This system can, in many cases, be exhaustively specified using Knuth-Bendix (and related) completion

of a λ-expression. A prefix is a smallest closed sub*structure*. It need not be (and typically is not) a well-formed expression.

techniques [51, 52][4]. Rewrite systems which complete, permit a finite specification of all interactions among the expressions of the subspace. They implicitly determine a grammar for its (normal form) expressions.

The rewrite system cast in terms of building blocks is a *description of the converged reactor system in which all reference to the underlying λ-calculus has been removed.* In other words, the generic λ-calculus can be replaced by another formalism specific to the self-maintaining ensemble of expressions in the reactor, that is, a particular algebraic structure.

The expressions of the invariant subspace are the carrier set of the algebra. Very often, but not always, that set is infinite. Although the reactor has only a very small capacity (1000 or 2000 expressions), the algebra persists through a fluctuating, yet stably sustained, finite set of expressions. This occurs whenever the connectivity of the transformation network is such that it channels most of the production flow to a core set of expressions. This we call *kinetic confinement*. An example is shown in figure 2.

Organization

The main conceptual result is a useful working definition of what we mean by an "organization": *an organization is a kinetically self-maintaining algebraic structure.* Self-maintenance has here two aspects which reflect the two modes of interaction: (i) algebraic, a network of mutual production pathways that is a fixed-point under applicative interaction, and (ii) kinetic, the concentrations of the expressions in the network core are maintained positive. The former is a necessary, but not a sufficient condition for the latter (i.e., a network can be algebraically a fixed-point - every expression being produced within the network - but its particular connectivity may not suffice to sustain non-zero concentrations of its core components under flow-reactor conditions as specified by equations (1) and (2)).

Organizations of differing algebraic structure are obtained by varying the set of λ-expressions used to seed the reactor. An infinity of such organizations are possible. Developing a taxonomy of their structure and properties remains a long-term goal of our program.

[4]Some rewrite systems induce an infinite recursion and defy completion. In our system, this is manifested as building blocks whose action upon one another is to generate new building blocks with the same property. The failure of some rewrite systems to complete is a consequence of the unsolvability of the universal word problem.

Self-repair and constrained variation

Two prominent properties of these organizations are their resilience to the subtraction of existing components and resistance to the addition of new expressions. Organizations often repair themselves following removal of even large portions of their component expressions. Some organizations are even indestructible: they regenerate themselves from any component. The reason for this robustness is the existence of *generators* of the algebra. These are sets of expressions whose repeated interactions rebuild piece by piece the entire organization; if they are retained, the system regenerates.

The link with algebra also clarifies an organization's response to the addition of new expressions, but for a different reason. The grammatical and algebraic invariances can be viewed as abstract *boundaries* of the organization. They determine membership. An expression which does not conform with that organization's particular grammar cannot be a member of the organization. Despite having an independent description, an organization is embedded in the larger λ-universe, and a non-member expression may perturb the organization algebraically (and grammatically), generating further expressions "outside" of it. The perturbing expression can, in some cases, be stably integrated, leading, for example, to a *self-maintaining extension* of the original organization. Alternatively, the perturbing expression may be diluted out of the system leaving the organization unaltered. The algebraic relationships which define an organization also determine *specific* opportunities for its extension. Biological interpretations are many. As but one example, Morowitz suggests that nonenzymatic precursor networks of the cellular core metabolism have evolved via distinct extensions [72].

Organization within organization

Organizations can have a quite complex substructure. To explain what we mean by substructure we need two iterated mappings. One is an "expansion" of a set $\mathcal{A}_i$ of expressions:

$$\mathcal{A}_{i+1} = \overline{\Omega}(\mathcal{A}_i) \stackrel{\text{def}}{=} (\mathcal{A}_i \circ \mathcal{A}_i) \cup \mathcal{A}_i \tag{3}$$

where $\mathcal{A} \circ \mathcal{B}$ means the set resulting from applying every expression in $\mathcal{A}$ to every expression in $\mathcal{B}$. The other is a "contraction" of a set:

$$\mathcal{A}_{i+1} = \underline{\Omega}(\mathcal{A}_i) \stackrel{\text{def}}{=} (\mathcal{A}_i \circ \mathcal{A}_i) \cap \mathcal{A}_i \tag{4}$$

Given an organization $\mathcal{O}$, generated in our experimental reactor, we take each expression i in $\mathcal{O}$ and iterate $\overline{\Omega}$ T times to obtain an expansion of i: $\mathcal{B} = \overline{\Omega}^T(\{i\})$. After this we contract $\mathcal{B}$ until we have found a fixed point: $\mathcal{O}_{\{i\}} = \underline{\Omega}^{T+1}(\mathcal{B})$.

In sum, $\mathcal{O}_{\{i\}} = \underline{\Omega}^{T+1}(\overline{\Omega}^{T}(\{i\}))$. If $\mathcal{O}_{\{i\}}$ is not empty, we have obtained a self-maintaining suborganization contained in $\mathcal{O}$ that has been generated by the single expression i.

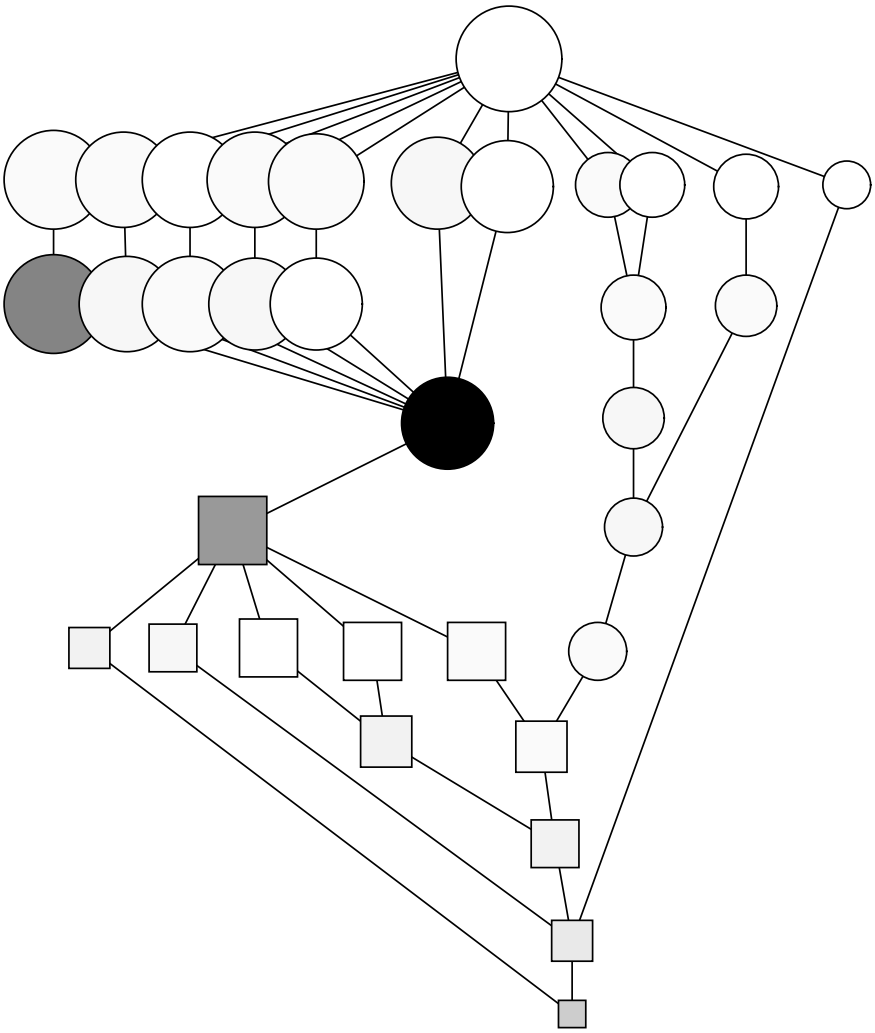


Figure 3: The substructure of an organization. Each node (circle or square) represents a self-maintaining set. Circles denote self-maintaining subspaces with a potentially infinite number of expressions, while squares represent finite self-maintaining subsets. When two nodes are connected by an edge, the lower one represents a set that is contained in the upper one. The size and grey level of a node reflects that node's share of the overall diversity and total number of expressions in the reactor, respectively. See text for further discussion. (Figure and analysis courtesy Harald Freund.)

The relationships between all suborganizations generated by individual expres-

sions of an organization can be visualized in a lattice partially ordered by inclusion. An example is shown in figure 3. The topmost node represents the entire organization. It is a combination of 11 suborganizations located at the next lower level in the diagram. The leftmost suborganization, for example, is an extension of the organization below it (darker node), which in turn is an extension of the black node. Since the black node is contained in a number of organizations above it, these organizations necessarily overlap (i.e., they share some members). The bottom node is a small closed self-maintaining set contained in all others. Despite its apparent complexity, only three interaction laws involving only one terminal element and two prefixes are required to describe the system.

The substructure of an organization reflects only the algebraic aspects of the organization. Any physical realization of such an organization is also a matter of dynamic stability. Structure and dynamics jointly define organization-specific properties with respect to robustness and evolvability.

Higher-order organizations

We can combine *disjoint* organizations that have been obtained independently. In some instances they build a stable higher-order organization that contains the component organizations in addition to a set of products arising from their cross-interactions. This set is not self-maintaining, yet it is crucial in stabilizing their integration into a new unit. We call such a set of objects a *glue*. Biologists will recognize this as an issue of some importance in history-of-life [8, 61], e.g., the mitochrondria and chloroplasts of eucaryotic cells are descendants of cells with an independent procaryotic ancestry.

Copy functions and the emergence of organization

Our model universe invites experimentation on the conditions which facilitate or impede the emergence of organization. An example of one such condition involves the role of replicating objects, that is, λ-expressions that copy.

Replication is a term usually used to denote an autocatalytic *kinetic* role, i.e., an agent whose change in concentration is proportional to its own concentration. In addition to its kinetic aspect, the present model makes the *operational* role of a replicator explicit. A replicator is the fixed-point of some interaction. f is a replicator, if the system contains some g (including $g = f$) such that f is a "left" or "right" fixed-point of its interaction with g: $(g)f = f$ or $(f)g = f$. Notice that g may turn f, but not another h, into a replicator (unless g is the trivial identity function). Replicators, then, need not be universal copiers. They may act to both copy and construct depending upon the expressions they take

as arguments.

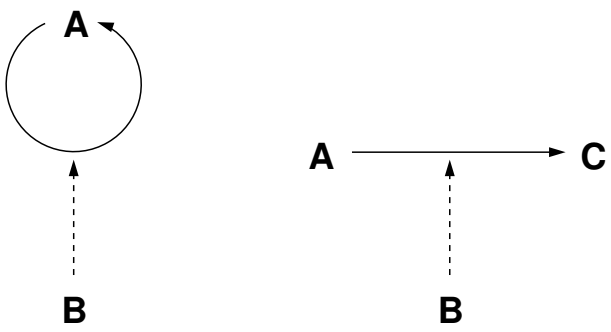


Figure 4: A basic alternative: copy actions (left) and non-copy actions (right). Closure of the former yields hypercycles, or "Level 0" in our nomenclature [26]. Self-maintaining closure of the latter (in the absence of the former) yields "Level 1" organizations [26]. A middle ground, copiers that also participate in constructive interactions, impede the development of hypercycles, favoring "Level 1" organization.

The distinction between the kinetic and operational aspects of replication is key to understanding an essential condition for organization. Self-maintaining structures capable of sustaining themselves *solely* on the basis of their copy actions (i.e., without constructive interactions) are easily encountered in our system (examples labelled as "Level 0" in [26]). Such structures are *hypercycles*, just as Eigen and Schuster discovered some time ago [23]. If replicators are disabled or if their operational role involves both constructive and copy interactions, the system will organize (examples labelled as "Level 1" in [26]). "Level 1" organizations differ fundamentally from hypercycles in their self-repair and extensibility properties.

2.1.4 Main limits

The results summarized above clearly illustrate that the merger of a dynamical system (the flow reactor) with a universe of objects that entertain constructive interrelations (λ-expressions) does indeed achieve the desired objective. A motion-in-the-object-space is induced, such that self-maintaining structures, characterized by an invariant pattern of transformations, arise. Moreover, these organizations possess properties – regeneration, structure-dependent extension, complex substructure, capacity for hierarchical nesting – akin to properties of living organisms. Yet, λ-expressions are far from molecules and our organizations far from organisms. The major limitations of Minimal Chemistry Zero (MC0) are enumerated below.

1. **Shape:** Molecules interact selectively. Violated in MC0, because λ-operators can act on one another indiscriminately.

2. **Symmetry:** Reaction is a symmetric event. Violated in MC0, because functional application is not commutative.

3. **Mass action:** With respect to a reaction event, molecules are resources and are used up. Furthermore, atom types and number are conserved during a reaction event. Violated twice in MC0, first by the kinetic scheme (1), and second microscopically - which is far more serious - by the multiple occurrence of the same bound variable in λ-expressions. To make the latter clear: when supplying the argument 5, say, to the function $f(x) = x^2 + 2x + 3$, the 5 gets used twice; once when substituting in x^2 and once in $2x$. Where does the second 5 come from? In chemistry, a reaction has only as many atoms as are present in the reactants.

4. **Reaction classes:** Chemical reactions proceed according to a variety of distinct schemes, such as substitutions, additions, and eliminations. In particular, individual reactions can yield several molecules on the product side. Violated microscopically in MC0, because application in λ-calculus yields at most one normal form (product). (Note that the reaction scheme (1) is an exogenous condition we impose.)

5. **Rate constants:** In chemistry reactions proceed with different velocities, which leads to a separation of time scales in reaction networks. Violated in MC0, because every reaction event has the same unit rate constant.

These limitations are substantial and motivate the improvements to which we now turn.

2.2 Minimal Chemistry One

We consider an extension of MC0 designed to address the issue of "shape" (item 1 in section 2.1.4). Pure λ-expressions are strings of characters that represent functions with no specific domain of definition (i.e., they can act on any expression). Shape enforces a specificity upon interaction.

2.2.1 Shape and action

The virtues of MC0 lie in the transparency of λ-calculus and the connections its use provides to abstract algebra and rewrite systems. It is difficult to imagine how a 3-dimensional interpretation could be given to the actions of λ-expressions

in a canonical way. An explicit spatial representation would seem to be required. However, a price for capturing shape would surely be paid in transparency of the resulting model. Might there not be an abstraction of shape that evades the costs of explicit spatial imitation?

Molecular shape derives from a self-consistent balance of nuclear and electronic motions influenced by each other's field. At the same time the resultant distribution of electronic and nuclear densities gives rise to specific chemical properties. In this sense shape and chemical action are two sides of the same coin. In a slightly more abstract sense, the specificity of chemical action between molecules results from (i) the *complementarity of chemical properties* between reacting functional groups and (ii) their spatial disposition. The first aspect means, for example, that an electron donor group on one molecule must meet an electron acceptor group on the other for an action between them to occur. To put it in a cartoonish way, chemical complementarity emphasizes that action occurs when one functional group is of the type "`if I'm given an` x`, then I yield a` y", while the other group is of the "`I'm an` x"-kind. If the latter were an "`I'm a` z", no action would take place. It is clear at once that interaction selectivity, though invariably tied to space in real chemistry, does not require space to be expressed abstractly.

When a reaction involves more than one chemically complementary group, their spatial disposition further contributes to specificity by excluding those reaction partners that have the right groups at the wrong places. However, this is a combinatorial aspect that is neither unique to spatial extension, nor one that fundamentally alters the nature of specificity caricatured above.

A rather different issue is raised by non-reactive molecular interactions based on shape. There, the *geometric* aspect of spatial form is essential in giving rise to supramolecular morphologies, such as membranes or viral capsids. This aspect necessarily escapes a formalization cast in a non-geometrical syntactical system; it is as much outside the calculus-metaphor as is the flow-reactor kinetics. At this stage of our program, however, we dispense from further physical embeddings (beyond kinetics); our interest being in transferring as much as possible of what appears to be physical to an abstract computational domain.

Two aspects of molecular form, shape-as-conditional-action and shape-as-geometry, are together responsible for chemical interaction specificity. Here we formalize only the first aspect, taking the stance that it is not the molecule's shape-as-a-coordinate-list that counts, but rather how the spatial configuration is parsed into basic reaction classes. (A virtuoso synthetic chemist looks at a molecular configuration in much the same way that a grand master looks at a chess configuration, perceiving the molecule in terms of what can be *done* with it, i.e., which features can be exploited to make or break bonds with respect to a syn-

thesis goal.) Thus, to the extent that shape is abstracted as a suite of lawful restrictions on permissible actions, it is plausible to capture its role by imposing a suitable discipline upon λ-interaction. This is done by augmenting the notion of function in λ-calculus with the constructive analogue of a "domain of definition" and a "range".

2.2.2 What is a type?

Minimal Chemistry One employs the use of typed versions of λ-calculus, where the system of types serves as an abstraction of restrictions on chemical action. Here, we briefly introduce the notion of a "type". A more detailed but still expository overview can be found in appendix B. For a rigorous treatment the reader should consult the literature [9, 10, 38, 55, 77].

A type is a statement about overall action. To appreciate this, consider an *untyped* universe, such as a computer at the level of memory cells [10]. It appears as an unstructured array of binary strings undergoing transformations. When looking at these strings we typically have no way of telling what is being represented. In contrast, a *typed* universe, such as a programming language, provides frames of interpretation for the digital contents of computers by imposing a kind of semantics defining intended use. Such frames work by offering a repertoire of behavioral *types*, such as variables, arrays, pointers, procedures, and control structures. Furthermore, variables themselves are often distinguished according to the *type* of value they are meant to hold: boolean, real, integer, character, and so on. *The effect of such constructs is basically to enforce a discipline of interaction.* For example, the interpreter of a programming language rejects the application of a function that removes blanks from character strings to an "inappropriate" object such as a vector of numbers. In essence, a type is but an object's "interface" that regulates with what it may communicate.

In programming languages a type serves as a *specification*, that is, it provides partial information about what an operator (a program) abstractly does. In chemistry, however, there are no external reference frames, no intentionally defined "integers", "character strings", or "vector products". The lawful behavior of chemistry is internally defined by the underlying physics. It is, therefore, important to understand that the abstract notion (and theory) of types is *independent* of any particular meanings. A representation of chemistry at a chosen level of resolution could be defined by a repertoire of primitive objects with assigned behaviors. Their internal structure is suppressed, and the behaviors are defined reciprocally. This is what computation theory calls an "abstract data type". Primitive objects of this sort could be atoms, or functional groups, such as hydroxy, amino, carbonyl groups, etc., or they might be further abstracted entities, those which carry "oxidizing"-behavior, others with "reducing"-behavior,

"acid"-behavior, "base"-behavior and so on. The action of a chemical group as a primitive could be specified by indicating which other groups it interconverts, without indicating how this is done (e.g., mapping a keto group into a hydroxy group under certain conditions). To turn this into a chemistry, a mechanism for building complex objects from primitive ones and for interconverting them is needed. That is what typed λ-calculus provides. Admitting primitives with a specific behavioral interpretation amounts to defining constraints as to which objects can be built and, therefore, which reactive combinations are possible.

2.2.3 Improved metaphor

We have implemented a simple standard type system for λ-calculus [15, 63], following the path laid out by a very useful prototype [59, pp. 97–113]. The system is explained in appendix B. Here we emphasize only those conceptual features that are important for our chemical agenda.

• **Syntactical structure and type are coupled.** A type is not arbitrarily attached to a λ-expression. It is derived from its syntactical structure by means of inference rules in a process called *type synthesis*. If an expression is modified, its type changes accordingly. The requirement to possess a type constrains the syntactical structure, and excludes some of the expressions that were possible in the untyped case. These constraints are interpreted to reflect the fact that a molecule's specific domain of action is based on its structure and composition, and that the properties of atoms constrain what kinds of molecules there can be.

• **Type polymorphism and boundary conditions.** Types can convey different degrees of specificity. A particular type may constrain an expression to act on one sort of argument only, while another may not discriminate at all. This is *type polymorphism* (see appendix B). The degree of polymorphism is controlled by assigning *basis types* of chosen specificity to the variables (and constants, if any) of the λ-system. *The set of basis types constitutes a new boundary condition.* It permits the tuning of the overall reactivity of our abstract chemistry and the definition of primitives with specifically chosen interrelations. It is in the definition of the basis set that an abstraction of molecular shape-as-conditional-action (or any other intended restriction upon action) succeeds or fails.

• **Interaction specificity.** An expression that represents a map sending objects of type τ into objects of type σ can act only upon arguments type τ. To decide when an interaction can occur is not as trivial as it seems. The type-expression can be viewed as describing a domain whose size reflects its degree of polymorphism. Whether the polymorphic types of two colliding λ-expressions match properly is not a mere syntactic comparison, but involves detecting whether one type is an instance of the other. The decision procedure is outlined in appendix

B.

The present formalization treats the abstract essence of "shape" as a statement about a molecule's domain of action. It bears emphasis that in this formalization the λ-term continues to be the object corresponding to the physical molecule. The type-expression derived from the λ-term is but a device to enforce an interaction specificity. If the types of two colliding objects permit their interaction, the syntactical manipulations follow the λ-calculus. It is good conceptual hygiene not to confuse the type with the object. This is plain in chemistry, the shape of a molecule is not the molecule.

These features, in aggregate, define the metaphor underlying Minimal Chemistry One:

chemistry	**typed calculus**
physical molecule	symbolic representation of an operator
molecule's behavior	operator's action
specificity of interaction	type discipline
chemical reaction	evaluation of a functional application

2.2.4 Model and preview of results

The reactor with Minimal Chemistry One is schematically shown in figure 5.

Our results with the MC1 model have yet to be exhaustively reported in the primary literature and, accordingly, we will not provide as detailed a summary of results as presented for MC0. The major consequence, however, of the improved model is that organizations are once again achieved and display properties akin to those documented for MC0. Organization, however, is considerably more difficult to achieve than in the untyped case, as may be expected by a restriction on interaction. The degree of difficulty is related to the degree of polymorphism, and, therefore, to the type basis.

2.3 Minimal Chemistry Two

Only one of the limitations inherent in Minimal Chemistry Zero, section 2.1.4, is addressed by our abstraction of shape as a lawful discipline upon interaction and our accompanying implementation of that abstraction in typed λ-calculus. Problems with MC0 regarding symmetry, resource accountability, reaction classes and rate constants remain in MC1. Indeed, one might even contend that our notion of types-as-shape is not mature *until* these problems are solved - that is, until

we succeed in defining a basis set that generates, for example, the appropriate classes of reaction in some restricted chemical domain.

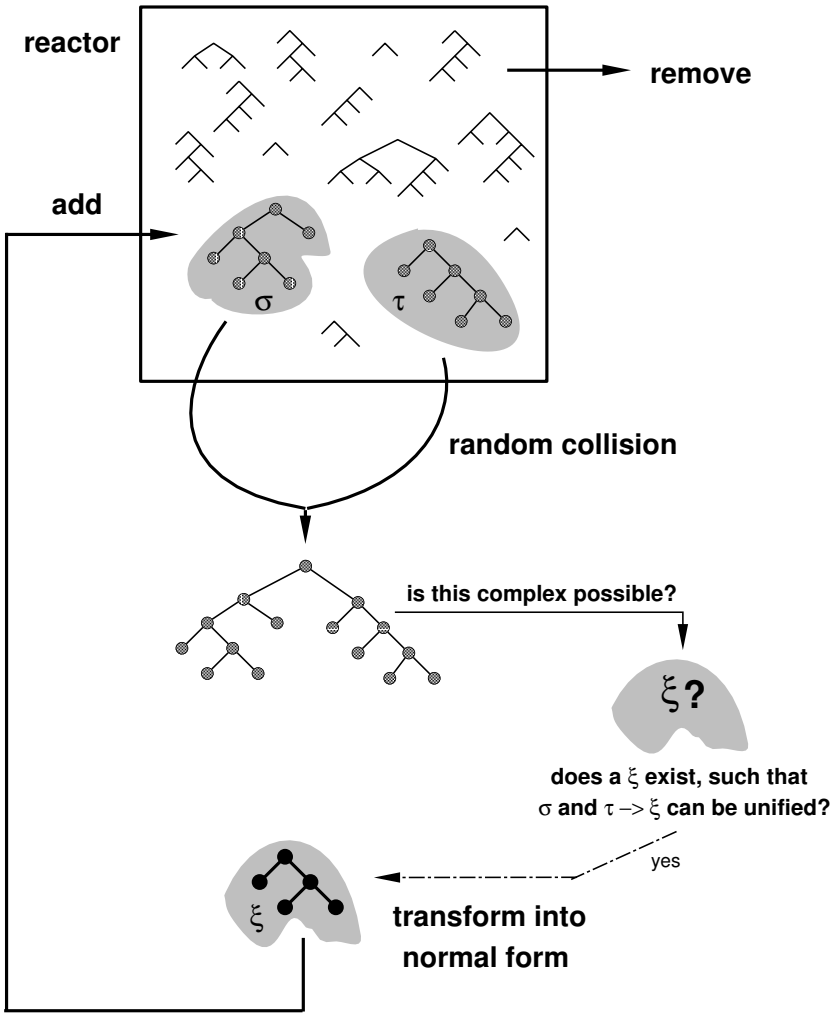


Figure 5: The λ-calculus flow-reactor with function-particles that discriminate among interaction partners on the basis of a type system. Two randomly chosen expressions with types σ and τ (represented as shaded regions) collide. The validity of the interaction complex depends on whether a type can be assigned to it. The procedure is explained in appendix B. If the interaction complex is typable, the reaction proceeds by normalizing the complex. Otherwise, the types σ and τ are incompatible for interaction, and the collision is regarded as elastic.

The issue is one of the level of abstraction we chose. A critic might well contend

that our level of abstraction is so high as to willfully preclude eventual maturation from an abstract to an actual chemistry. This, however, would be a misreading of our intent, see for example [84]. The retention of a high level of abstraction in the transition from MC0 to MC1 is anything but a resistance of the actual. Rather it represents a strategic claim that the benefits of a high level of abstraction exceed the costs of distance from actuality. A principal benefit lies in facilitating the transition *between related formal systems.* A return on costs will be realized if we are led to alternative formalisms uniquely well-suited to stepwise refinement of the original metaphor. Minimal Chemistry Two is vindication of that strategy.

MC2 differs from the advance of MC1 over MC0 in two ways. First, in MC1 we retained the core elements of the MC0 metaphor, merely refining it to include shape. In MC2, we tinker with the ontology itself. Here we abandon λ-calculus as the chosen formalism and are empowered to do so *without* loss of progress gained in the λ framework *by virtue of an isomorphism between formalisms.* Second, unlike MC0 and MC1, MC2 has yet to be implemented. Hence, we limit ourselves below to the task of sketching, sequentially, how the typed λ-calculus leads naturally to formalisms in proof-theory, how the chemical metaphor might be translated to and enriched by the proof-theoretic connection, and what limitations of a λ-based artificial chemistry this translation permits us to address.

2.3.1 From λ-calculus to proof-theory

The Curry-Howard isomorphism

Within MC1, a small number of plausible basis sets were implemented and characterized. These established that the typed system retained the capacity to yield self-maintaining organizations first established in MC0. A multitude of plausible basis sets remains unexplored. Exploring them with chemical plausibility in mind, however, requires insight into how type construction may be used to impose, for example, resource accounting *within* λ-calculus. While not impossible, the task would clearly be vastly simplified if the syntax itself imposed such a discipline of stoichiometry. Syntactical systems of this sort exist. One is led to them through a mapping between typed λ-calculus and proof-theory. Indeed, the mapping lies at the core of a deep connection between computation and logic (see appendix C.1).

Typed λ-calculus can be viewed as a syntax for the derivation of logical formulae. The mapping is known as the Curry-Howard isomorphism [38, 45] and may be stated informally as (see C.1 for a more detailed introduction):

$$\text{type } \sigma \longleftrightarrow \text{logical formula (proposition) } \sigma$$

$$\lambda\text{-term of type } \sigma \longleftrightarrow \text{proof of } \sigma$$

Since we use λ-calculus to define an abstract chemistry, any rigorous link between typed λ-calculus and other areas of mathematics extends the chemical metaphor. Roughly:

$$\text{“shape”} \longleftrightarrow \text{logical formula (proposition)}$$
$$\text{molecule with that shape} \longleftrightarrow \text{proof of that formula}$$

Proof-theory is a large domain, characterized by an initially bewildering diversity of syntactical systems. Our task, to which we now turn, is to situate our approach within this diversity, that is, to specify chemical interpretations of a chosen syntax which are both consistent with the isomorphism and which have potential in ameliorating the deficits of the λ-syntax.

What do proofs have to do with it?

The connection between typed λ-calculus and proof-theory and our imputed connection between both formalisms and chemistry will not demand that a reader have a rich appreciation of logic. Some prefatory remarks are, nonetheless, in order.

A logical formula is built from atomic formulae using *connectives* like `and` ($\wedge$), `or` ($\vee$), `implies` ($\rightarrow$), `negation` ($\neg$), and universal ($\forall$) and existential ($\exists$) quantifiers. There are two basic questions we may put to a logical formula. We may ask for the truth value (`true` or `false`) of a formula, or we may ask for its validity. Contrary to widespread folklore outside of logic, logic is not exhausted by “truth-tables” that permit reading off the truth-value of compound statements, such as $A \wedge B, A \vee B, A \rightarrow B, \neg A$, given the truth-values of the propositions A and B. Logic is far richer and much of the richness lies in the latter of these questions.

A taste of the relation between truth assessments and validity may be introduced by considering Frege's [30] distinction between the *sense* and *denotation* of a logical formula. In $10/2 = 1 + 4$ the denotation of both $10/2$ and $1 + 4$ is 5. Hence, the denotation of $10/2 = 1 + 4$ is `true`. How, though, does one know that $10/2$ has the same denotation as $1 + 4$? As Girard points out [36, 38], it is *not* obvious that $10/2 = 1 + 4$, for if it was we would need neither symbolism for division and addition, nor, even find need to state an equality. This is what is meant, then, by saying that $10/2$ has a different sense than $1 + 4$.

Frege's distinction tells us that what matters in logical systems is not the denotation (i.e., the content) of the propositions, but rather their relational structure. When Aristotle says that “*All men are mortal; all Athenians are men; hence all Athenians are mortal*”, he is saying “*All B is C; all A is B; hence all A is C*”, he is not saying “*true; true; hence true*” [36]! Logic manipulates the sense, not the denotation.

How then does one proceed from sense to denotation? Let us return to $10/2 = 1 + 4$. In this example, it is clear. Something must be done to show that they have the same denotation and here that "something" is a computation.

Just as different logical systems are endowed with different connectives, so must proof-theory come in a diversity of flavors. The flavors of proof-theory germane to our project are those which bear a correspondence with computational operations. These comprise a class known as constructive proof-theory or constructive logic. The idea behind constructive proof-theory is that the meaning of a formula is the set of its proofs, where the proofs are objects of an effective calculus, i.e., proofs are seen as construction scaffolds for a formula. Loosely speaking, the meaning of $\alpha \wedge \beta$ is to exhibit a proof of α and a proof of β. The meaning of $\alpha \rightarrow \beta$ is to provide a *function* which transforms proofs of α into proofs of β. Framed in this way, constructive logic appears not so much as a tool for reasoning about computation than a computational activity itself. Indeed, this is the essence of what is established by the Curry-Howard isomorphism.

2.3.2 Ontological commitment, resultant metaphor, and formal representation

Amongst the diversity of constructive logics is a recent innovation known as *linear logic* [33]. We use it here to illustrate how the proof-theoretic context refines the chemical metaphor. We do so in an intuitive fashion, building from our (similarly intuitive) characterization of typed and untyped λ-calculus. As in our previous treatments, the reader is referred to Appendix (C.3) for a more detailed, but still gentle, treatment.

In linear logic, indeed with proof-theory generally, the properties of a logical connective are defined algebraically (rather than by truth-tables), through rules stating how a connective can be inserted into and removed from a formula. Proof-theory emphasizes what actions must be taken to construct a formula using a set of rules governing introduction and elimination.

The introduction rules, elimination rules and connectives of a fragment of linear logic appear in appendix C.3.1. Relative to the sparsity of axiomatic operations in λ-calculus, the rules in C.3.1 reveal a proliferation of syntactical operators. This meets our intent to progressively refine our chemistry (after all, one needs a richer syntax to describe an atom in quantum mechanical terms than to describe it as "a little solar system"). As the basis for the following discussion we use here only the smallest fragment of linear logic, so-called multiplicative linear logic (MLL), in sequent notation. For details see Appendix C.2, but for those who are in a rush $\vdash$ means "logical consequence", ψ and ϕ are propositions, capital greek letters are sets of propositions, and $\vdash \psi, \phi$ stands for a proof of the disjunction

ψ "or" (comma) ϕ using the inference rules of the proof-system. These rules are represented by a horizontal bar separating the premises (above) from the conclusion (below). An axiom is a conclusion without premises. By virtue of the *par*-rule the comma ("or") behaves like a $\parr$.

$$\frac{}{\vdash \phi, \phi^{\perp}}\ axiom \qquad\qquad \frac{\vdash \Gamma, \phi \qquad \vdash \phi^{\perp}, \Delta}{\vdash \Gamma, \Delta}\ cut \tag{5}$$

Connectives

$$\frac{\vdash \Gamma, \phi \qquad \vdash \psi, \Delta}{\vdash \phi \otimes \psi, \Gamma, \Delta}\ times \qquad\qquad \frac{\vdash \Gamma, \phi, \psi}{\vdash \phi \parr \psi, \Gamma}\ par \tag{6}$$

with negation, $\perp$, defined as:

$$\phi^{\perp\perp} \stackrel{\text{def}}{=\!=} \phi \tag{7}$$

$$(\phi \otimes \psi)^{\perp} \stackrel{\text{def}}{=\!=} \phi^{\perp} \parr \psi^{\perp} \tag{8}$$

What are the imputed chemical interpretations of this syntax and how do they relate to that explored in MC0 and MC1? Recall the core of our original MC0 metaphor:

physical molecule ..... λ-expression (an operator)
molecule's behavior ..... operator's action
chemical reaction ..... evaluation of a functional application

Our approach in MC1 merely extended the metaphor by modulating a molecule's behavior to include:

molecule's shape ..... type

Composing this with the Curry-Howard isomorphism yields:

molecule's shape ..... type σ ..... logical formula σ
physical molecule ..... normal λ-term of type σ ..... proof of σ

The interpretation of a molecule as a proof, compelled by the isomorphism, leads to the first refinement. In linear logic (see appendix C.3), a proof is the construction of a *multi-set* of logical formulae, known as a "sequent" (whose *raison d'être* is explained in appendix C.2). Recall from Curry-Howard that a formula is equivalent to the type of a λ-expression, and that in MC1 a type was the specification of a reaction site. A proof of the sequent $\vdash \phi_1, \ldots, \phi_n$, then, represents a molecule with n possible reaction sites $\phi_1, \ldots, \phi_n$, rather than a single one as in MC1.

Now that we have logical formulae as descriptions of potential reaction sites, we need an interpretation of a chemical reaction. Like in MC1 a chemical reaction involves two issues: (i) which sites are allowed to interact (i.e., specificity), and (ii) what happens once two sites do interact (i.e., action). For this we turn to linear logic and its syntactical machinery, rules (5, 6).

Let us again proceed intuitively. If we view a logical formula as a potentially reactive site, then we must look for the logical counterpart to "chemical complementarity" (see the discussion in section 2.2.1). In linear logic, this is provided by *negation*, $\perp$. The fact that negation is defined in terms of a duality, rules (7, 8), fits quite naturally with chemistry, where the encounter of chemical "duals" (i.e., acid vs. base, oxidizing vs. reducing) is required for a reaction to occur.

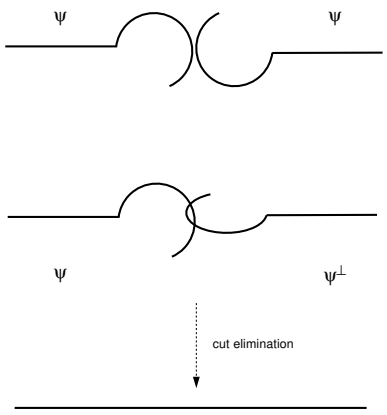


Figure 6: A cartoon of negation, cut, and cut-elimination in linear logic.

In this logic, duality appears at the level of primitive building blocks. These are axioms, i.e., syntactical entities of the form $\overline{\vdash \psi, \psi^{\perp}}$. The structure means that a primitive object always enters the scene as a pair of connected dual "flavors", ψ and $\psi^{\perp}$. More complicated sequents (representing the reactive options of a molecule) are constructed from such "atoms" using the rules (6) for the connectives.

When two such sequents (molecules) meet, it is the cut rule that tells us whether they can interact:

$$\textbf{cut}\ \frac{\vdash \Delta, \phi^{\perp} \qquad \vdash \phi, \Delta'}{\vdash \Delta, \Delta'} \tag{9}$$

The rule states that for an interaction to occur, each molecule must possess the *same* formula ϕ, but in a *complementary* flavor. The cut-rule, therefore, emphasizes the role of a logical proposition as an *address*, i.e., a structured name enabling communication with a specific other address (of dual kind)[5]. Girard's

[5]In this context, the reader is urged to resist giving a logical proposition an interpretation

[34] metaphor of a *plug*, as sketched in figure 6, renders the situation best. A proof-system, then, functions as a formal "physics" in which such addresses are constructed endogenously[6].

The cut-rule states *which* objects can interact, but it doesn't state what happens once they do so. The fact that one has a lamp and an electrical outlet does not mean one has light. All that cut does, is to *initiate* a chemical reaction (the "plugging"). An action is still required. Mathematically, the cut-rule enables two proofs to be joined into a new proof on the basis of a "trade"; one proves (ϕ) what the other assumes ($\phi^{\perp}$). The cut rule is formally equivalent to functional application in λ-calculus which was used to initiate a chemical reaction in MC0 and MC1. Indeed, in correspondence with reduction to normal form, Gentzen [32] showed that a cut can always be removed from a proof. Crucially, Gentzen exhibited an effective process that performs this elimination by rearranging the proof structure (see appendix C.2). It is this process, then, which is triggered by cut and that provides the associated action completing a reactive encounter. Cut-elimination yields a proof in normal form (i.e., a "direct" proof of a sequent, where the previously involved intermediate "lemma" ϕ has been removed). Completing the synthesis of a molecule is seen as "removing the intermediates" (by letting them react) yielding the product molecule as a "normal form proof".

Thus, via negation, cut and cut-elimination we have specified an interpretation of a chemical reaction and thereby completed our translation. The MC2 metaphor follows:

chemistry $\cdots\cdots$	**proof-theory**
chemical properties of bonds $\cdots\cdots$	algebraic properties of connectives
reaction site σ $\cdots\cdots$	logical formula (proposition) σ
stable molecule with site σ $\cdots\cdots$	cut-free proof of σ
chemical complementarity of sites $\cdots\cdots$	negation (σ and $\sigma^{\perp}$)
chemical reaction $\cdots\cdots$	proof with cut between σ and $\sigma^{\perp}$

This may be summarized (in words that might be chosen only by those deep within its grip) in its barest essentials as follows. The proof-theoretic metaphor

linked to "human reasoning". It will not prove edifying. To further clarify the address issue, consider conditions under which *nothing* happens. For example, both objects carry the same flavor of sort ψ (proton-donor meets proton-donor), or both objects carry different flavors but not of the same sort (proton-donor, say ψ, meets electron-acceptor, say $\pi^{\perp}$).

[6] Chemistry may be thought of as a system for constructing addresses that enable reactions that further construct addresses. This perspective recalls the π-calculus paradigm of Milner [67, 68, 69] (Appendix A.3).

views a molecule as the cut-free proof of a "shape", i.e., a chemical action. The cut-rule corresponds to a chemical reaction at site ϕ. Complementary chemical propensities are mirrored by negation, ϕ and $\phi^{\perp}$. The coming into contact of these complementary types creates an instability (the cut). In chemistry this initial site of instability is propagated through the molecular skeleton, possibly breaking old bonds and making new ones, until the stable product molecule results. This reaction progress is mirrored in the mechanics of cut-elimination that propagates the "unstable" cut-site through the proof-skeleton until normal form is attained.

2.3.3 Addressing prior limits in the linear logic framework

MC0 is an absolutely minimal chemistry, consistent with our intent in assessing the potentials inherent in adding constructive interactions to a dynamical setting. The success of MC0 in yielding self-maintaining organizations with properties seemingly so reminescent of biological systems renders its minimality a severe impediment to progress. The deficiencies enumerated earlier (section 2.1.4) were: (i) molecules had no specificity of interaction (i.e., no shape), (ii) reactions were asymmetric by virtue of the asymmetry of application in λ-calculus, (iii) mass-conservation (stoichiometry) was violated (i.e., no resource accountability), and both (iv) reaction classes and (v) rate constants were lacking. Content that the first issue has been addressed with typed λ-calculus and progress carried over to the proof-theory framework via Curry-Howard, we here attempt to sketch the potential of the proof-theoretic representation in addressing the remaining issues. We emphasize, however, that MC2 has yet to be implemented and the following must therefore be regarded as "a transcript from a lab notebook" recorded well before the first reagents have been mixed.

Symmetry

In both MC0 and MC1 a chemical reaction was represented as functional application. Application in λ-calculus is non-commutative, hence asymmetric reactions come as a fixed deficit of the formalism. In the proof-theoretic frame a chemical reaction is modeled by the cut-rule, which is, in turn, dependent upon negation. Negation in linear logic is defined via deMorgan-like dualities (see equation (8) and section C.3) and is involutive: $A^{\perp\perp} = A$ (like a matrix transposition). This permits a completely *symmetric* reaction. By this we mean that in a reaction between sites ϕ and $\phi^{\perp}$ it doesn't matter which of the molecules carries the ϕ-site and which the $\phi^{\perp}$-site: the result is the same. Symmetry of reaction is as generic to linear logic as is asymmetry in λ-calculus.

Resource Accountability

Violation of mass-conservation is rampant within both MC0 and MC1. Recall that a failure in resource accounting obtains whenever the same variable occurs multiple times in a λ-expression. When, for example, a reaction between $\lambda\mathtt{Cl}^-.(\mathtt{Cl}^-)\lambda y.(\mathtt{Cl}^-)y$ and some $\mathtt{OH}^-$ occurs, the $\mathtt{OH}^-$ is used twice during normalization: $(\lambda\mathtt{Cl}^-.(\mathtt{Cl}^-)\lambda y.(\mathtt{Cl}^-)y)\mathtt{OH}^- \rightarrow (\mathtt{OH}^-)\lambda y.(\mathtt{OH}^-)y \rightarrow \cdots$. The meaning and extent of this difficulty is apparent. We are effectively permitting the *same* OH^--group to substitute for two distinct Cl^- ions, or, equally problematic, having the *same* Cl^- be at two different places in the molecule. Moreover, if a variable is declared that never appears in the term, a reaction would simply "annihilate" one reactant, such as in: $(\lambda\mathtt{Cl}^-.\lambda y.y)\mathtt{OH}^- \rightarrow \lambda y.y$. Serious attention to syntactical resource accounting is required in a mature artificial chemistry and no such tools are inherent in the syntax of λ-calculus.

Resource sensitivity is hardly a feature of classical logic, but several varieties of constructive logic - including linear logic - have this attribute. In classical logic, formulae are not viewed as physical entities (or tokens) - like chemicals (or money) - that are consumed when they are deployed to cause effects. A lemma proven by mathematician X need not to be reproven after mathematician Y has used it in proving a theorem. The lack of a resource problem in classical logic derives from the contraction and the weakening rules (see section C.2) which state that in the manipulation of proofs, available formulae can be copied or surplus instances erased arbitrarily. For example, the classical conjunction of twice the same formula, $\phi \wedge \phi$, is equivalent to ϕ. The problem with the real world is that if ϕ stands for some fact like a dollar, then classical logic states that 2 dollars - and, hence, any number of dollars - are equivalent to one dollar. Or that one molecule of a substance has the same effect as an Avogadro such molecules. Here classical logic departs radically from the physical world.

Several constructive logics are resource sensitive, linear logic amongst them. Linear logic achieves resource sensitivity by placing weakening and contraction under explicit control. Depending on the tightness of the control several variant logics are obtained (e.g., [78]). The basic idea shared by each system, however strict its accounting, is to view formulae as "assets" that are consumed when they are used. For our purposes here it is sufficient to note that the syntax of linear logic permits resource accountability; attention is paid to enforce that no formula may be used that has not first been generated and that, no formula, once used, may be used again without generating it anew. This clearly permits an artificial chemistry embedded in linear logic to escape another of the deficits inherent in λ-representation.

Reaction classes

The reaction scheme of MC0 and MC1 shares a deficit beyond that of resource accounting. Specifically, the λ-framework explored only a very restricted set of chemical reaction classes. Moreover, these were exogenously imposed. The restriction is apparent in noting that a λ-expression (a reactant) applied to another can yield either a single product or no product at all. How are commonplace chemical reactions with multiple products, e.g., RCOOH+R′OH→RCOOR′+H_2O, handled in the λ-framework? They are not.

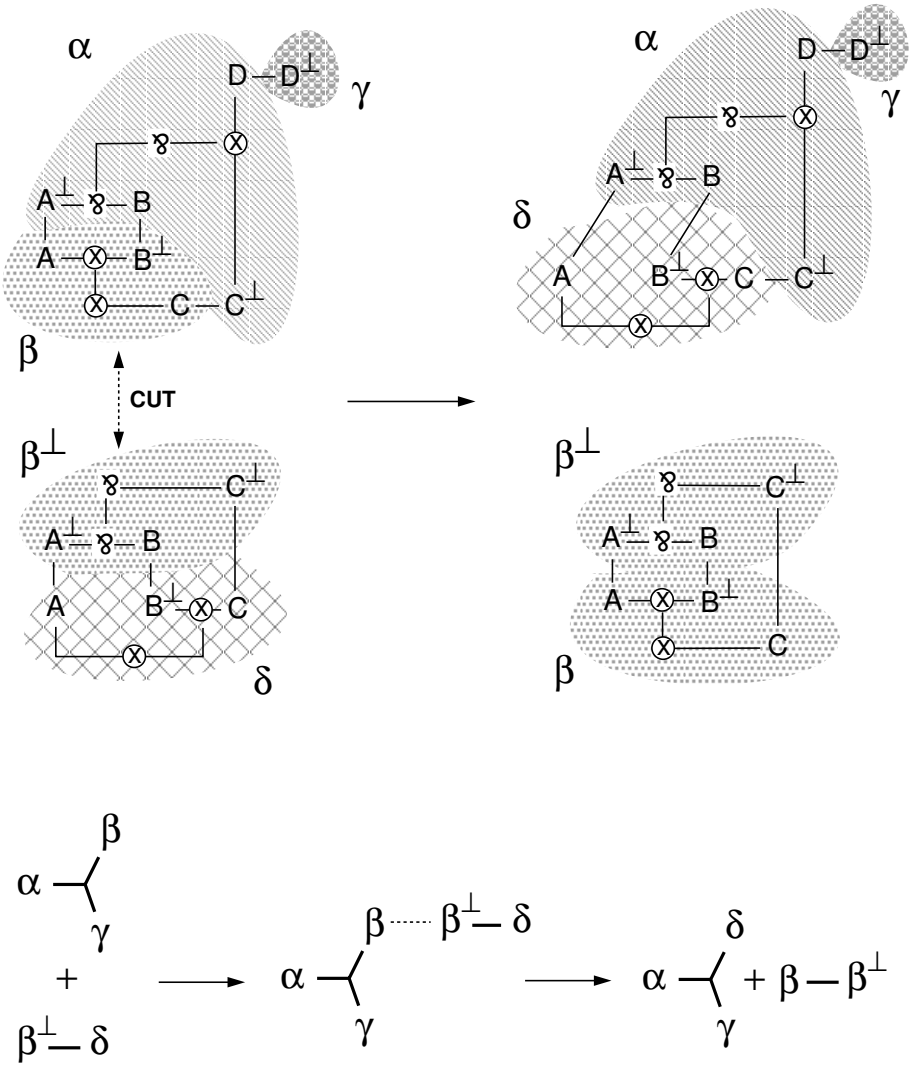


Figure 7: A chemical reaction with linear logic proof-nets. See text and section C.3.2 for details.

Accomodating this deficit within the linear logic framework is the challenge that

most limits implementation of MC2. In the case of λ-calculus, the limitation was inherent in the formalism. In contrast, linear logic - as well as alternative formalisms motivated by linear logic (e.g., [54]) - provide a broad set of options. They raise, however, important issues of chemical interpretation (much like certain chemically motivated extensions to the logical framework raise interesting issues of logical interpretation). For the purpose of this contribution there is little to be gained in fully developing the various candidate schemes. It will suffice to introduce an example to illustrate the power of this route to MC2. We chose this particular example because it makes especially apparent a larger issue which lurks within any choice of chemical interpretation.

Recall that we identify a molecule with a proof, initiate a reaction with the inference rule "cut" and complete the reaction by cut-elimination. As with λ-calculus, it is a simple matter to generate a single product. Figure 7 illustrates a simple scheme that stretches the usual inference by yielding multiple products (i.e., more than one sequent as a conclusion). The reaction is presented in Girard's [33] proof-net notation, a concise presentation mode detailed in appendix C.3. Mastery of proof-net notation is not required to appreciate the reaction intuitively, however. It is sufficient to know that each structure is a valid proof and that the shaded regions are propositional formulae. Imagine the figure to represent esterification of a carbonic acid with an alcohol: RCO$\boxed{\text{OH}}$+R$'$O$\boxed{\text{H}}\rightarrow$RCOOR$'$+$\boxed{\text{H}-\text{OH}}$ with $\boxed{\text{OH}}$ labelled (as in figure 7) as β and $\boxed{\text{H}}$ as $\beta^{\perp}$. The cut performed on β and $\beta^{\perp}$ is now seen as the chemical action between an acid and its complementary base. The by-product of the esterification is water, captured here as the combination of the two cut-formula flavors, H and $\text{H}^{\perp} \equiv$OH, in a single molecule corresponding to $\beta - \beta^{\perp}$ in figure 7. Clearly, exchange of shaded components between the reactant proofs has yielded the desired disconnected product proofs. The interpretation presented in figure 7 is illustrative of how particular issues of chemical interpretation are raised. In the cut rule (9), the single conclusion sequent has lost (both flavors of) the cut formula. Yet, in our figure, we have simply collected the garbage that would otherwise be thrown away during cut-elimination. One might justify this practice on the grounds that in real chemistry, nothing is ever annihilated. The attribution purchased by this justification is that of imparting a physicality to the cut-formula.[7]. The process shuffles a reaction site from one

[7]The attribution has consequences. (i) In linear logic, the cut formula is erased. Its erasure makes cut irreversible. Keeping the latter in our interpretation (see figure 7) preserves all information necessary for its reversal. Yet, we cannot reverse it, because the reverse reaction does not proceed via the cut rule which we took as the formal definition of a reaction. (ii) Treating the cut-formula as an object, say the functional group OH, makes it impossible to subsequently cut within that object, e.g., separating the H from the O. (iii) Our interpretation only implements substitution reactions of chemistry. The scheme is unable to introduce, or eliminate, connectives (i.e., types of chemical bonds). The rules for the connectives (6) are not suited to model such reactions, because they don't provide interaction specificity (see the discussion of cut in section 2.3.2).

molecule to another. This, then, is a clean interpretation of the reaction class known in chemistry as *substitution reactions*.

Treating the cut-formula as a physical object introduces a larger issue. Disarticulating the physical - whether it be an object like a molecule or a memory location in a computer - from the syntactical rules used to perform computation is a principle challenge within the domain of computer science addressing concurrency. Linear logic is one candidate formalism for concurrency. Our interpretation of the cut formula in linear logic as representing a physical object is one of several possible representations of the physical[8]. The diversity of paths within MC2 for addressing the limitation of reaction classes in MC0 and MC1 reflects this larger issue. We return to the general topic in later discussion.

Rate constants

Rate constants are an aspect of chemical reactivity with important consequences for reaction networks. In MC0 and MC1, all reactions proceed at an identical rate. A plausible foundation to endogenize rate constants within the λ-syntax would be to interpret as a rate constant the number of reduction steps needed to obtain normal form. For this to make chemical sense, reduction must be expressed in terms of true "unit-time" events, i.e., truly elementary steps of reconfiguration. This is not the case with standard definitions of β-reduction. The linear logic framework, however, has made it possible to resolve reduction into elementary events [17, 35]. We could imagine them as taking place at the beat of an external clock. Two simultaneously occurring interactions would terminate after different periods of time, thus yielding different "rate constants".

We have focussed above on the power accompanying the shift in formalism enabled by the Curry-Howard isomorphism. It is remarkable that linear logic appears to have the capacity to addresss *each* and to solve some of the principal lim-

Other attributions to the cut formula are no less plausible, but carry their own suite of consequences. One might argue that the cut-formula has no physicality, yet retain chemical plausibility by holding that what is cut are the bonds themselves and not "that which is bound". A variant of linear logic, known as linear logic with MIX (direct logic) [2, 16, 25], readily permits both connected and disconnected proofs, and may be employed in generating multiple products. This attribution, however, purchases its power at the cost of (i) losing a strict control over weakening (and, thereby, a relaxation of strict resource accounting), as well as (ii) a far greater complexity in implementing cut-elimination.

[8]Attempting to give chemical processes a logical interpretation feeds back to logic itself. We interpret here a chemical reaction as being a logical inference (governed by a rule such as cut). But what *logical* interpretation should be given to chemical reactions that yield multiple products? That is, what is the meaning of a rule of inference that splits proofs, i.e., that generates two or more conclusion-*sequents* at the same time?

itations of MC0 and MC1. In focussing upon comparison with the λ- framework, we may have given the inadvertent impression that addressing these limitations exhausts the utility of the linear logic frame. Indeed, it may well prove to be the case that features of linear logic unrelated to the limitations of MC0 and MC1 will ultimately provide most important.

2.4 A Roadmap from chemistry to proof-theory

We pause to ask: How natural is the analogy between a molecule and a proof? Suppose Dr. X claims that a certain chemical action σ is possible. Prof. Y challenges her to prove that claim. X returns a few years later and exhibits an actual molecule that provides that action, thereby proving her claim. Although the molecule does what X claimed it to do, Y will rightly wonder: "How did you produce it?". The question is fair, because by just looking at a molecule it isn't clear *how* it was produced, despite being evident *that* it was produced. The reason is that in chemistry we typically can't just stick atoms together one by one like in a Ball-&-Stick model. A molecule tells little about its synthesis path, just as a mathematician's direct proof from first principles does not convey the insight that led to it. Indeed, a mathematician usually proceeds by intermediate steps, proving lemmas, and then combining them with the cut rule (9) to achieve the desired theorem. Cut permits a proof to be factored into generic modules ("subroutines"), thereby preserving the proof idea. A mathematician, therefore, rarely normalizes the proofs. The situation in chemistry is subtly different. First, a direct synthesis of a molecule by plugging atoms together is almost never feasible. Thus, a chemist is *forced* to synthesize a molecule by using other molecules as intermediates, i.e., he cannot fully exploit what logicians know as the subformula property. Second, as soon as the chemist mixes the reactants, the reaction proceeds, i.e., the mixture spontaneously "normalizes", courtesy of thermodynamics. The self-maintaining organizations we found with MC0 and MC1 appear from this standpoint as specific ensembles of molecules that collectively retain their synthesis pathways, because every molecule is both "final product" and "intermediate". Since in MC2 molecular actions are seen as theorems whose proofs are the molecules that perform those actions, MC2-organizations - had we produced them - would appear as sets of theorems that are closed with respect to inference. The technical word for such sets is a *theory*.

From the initial observation that dynamical systems evade construction and that computation is construction, we have arrived at a representation of chemistry as proof-theory. This path has proven sufficiently surprising to us so as to expect that others might find a roadmap helpful.

Table 1: **The Chemistry / Proof-Theory Roadmap**

Chemistry	λ-calculus	Linear logic
molecule as physical object (the nature of atoms and their bonds)	λ-term in normal form	proof-net in normal form (multiplicative fragment of linear logic)
shape as action (domain of interaction) (with whom and how a molecule interacts, as determined by its shape and the nature of its reactive groups)	type (*specification of an action:* a description of what the term does at a particular level of resolution)	multiset of propositional formulae (theorem) (a set of "actionable" addresses, "interfaces", or "plugging specifications")
bonds (connectors of molecular parts)	abstraction and application	left and right rules of logical connectives
initiation of a reaction	application	rule of inference: cut (reactants are the premises, products are the conclusions)
completion of a reaction (structural rearrangements into stable products)	normalization	cut elimination
branching reaction (multiple reaction pathways among the reactants)	—	multiple cut options (a sequent is a multiset)
synthesis pathway (representation of a molecule as a suite of reactions between intermediates which enable its synthesis)	λ-term containing redexes	proof-net with cut(s) (a proof making use of intermediate results – lemmas a.k.a cut-formulas)
synthesis planning (how do we break up a molecule into achievable subgoals for synthesis?)	theorem proving (in typed λ-calculus only, e.g., Automath [18]; see also linear logic)	theorem proving (how do we break up a formula into achievable subgoals (lemmas)?)
determinism (for a particular reactive encounter the products are determined)	Church-Rosser (reduction is confluent; in addition, some type systems have *strong normalization* properties, i.e. any reduction sequence always terminates)	Church-Rosser (cut elimination is confluent and obeys *strong normalization* in MLL)

Table 1: **The Chemistry / Proof-Theory Roadmap (continued)**

chemistry	λ-calculus	Linear logic
duality (dualities apparent in stereochemical complementarity, hydrophilic/hydrophobic, proton donor/acceptor, reductant/oxidant)	—	duality (linear negation is involutive, defined by DeMorgan dualities)
symmetry (reactive interaction is symmetric)	asymmetric (functional application is not commutative)	symmetric (assumption and conclusion are dual - i.e., to switch is to negate)
resource sensitive (obeys mass conservation)	—	resource sensitive (weakening and contraction are absent in MLL, or controlled in full linear logic)

The match is suggestive, and one cannot fail to wonder whether the level of metaphor might someday be trespassed. If so, there would be a level of explanation at which chemistry would effectively *be* logic. This possibility is one we do not dismiss. It is no more ludicrous than, say, becoming accustomed to regarding physical space to "be" the threefold cartesian product of the real numbers, $\mathbb{R}^3$.

3 From dynamical systems to bounded organizations: The thread from chemistry ...

The long excursion into the specific case of chemistry serves to illustrate that it is indeed possible to move beyond the limits of dynamical systems claimed at the outset, section 1. Here we return to the general point, reiterating its major features in the context both of our own attempt and those of others grappling with related issues in often quite different settings. Our treatment of related issues is admittedly eclectic, representative only of our own interests and backgrounds.

The following define our conceptual coordinates. We sequentially amplify on each point in the sections that follow.

1. The identification of the "object problem" as a fundamental limit in applying the dynamical systems methodology to the biological sciences and beyond (section 1).

2. The claim that overcoming that limit requires a theory of object construc-

tion and, thus, necessarily involves a substantive overlap with the foundations of the computational sciences.

3. The position that a concept of "organization" derives from placing a theory of objects in a suitably constrained many-body dynamical setting (section 2.1.2). The conventional settings of either dynamics alone or syntactical manipulation alone are insufficient; "organization" derives (or, if one prefers, self-organizes) from their combination in a *constructive dynamical system.*

4. Finally, the "organizations" resulting from a constructive dynamical setting have the potential to address problems that have stubbornly resisted solution.

3.1 ... to the "object problem"

The identification of the "object problem" as a fundamental limit in applying the dynamical systems methodology to the biological sciences and beyond (section 1).

Whenever a particular level of analysis of Nature is populated with objects whose internal structure engenders specific action capable of changing or creating other objects, the dynamical systems methodology encounters a fundamental limit. The reason is that the formal machinery of dynamical systems is geared to handle changes in quantities, but not changes in object structure. We believe that a formal understanding of such a level of Nature requires a theory that *combines* variables that can take objects as values with the more familiar variables that hold quantitative values (such as concentrations). To convey an intuitive flavor of this, think of action as "parametrized" by structure, and imagine a "derivative" in object space giving information about the change of object action resulting from a change in object structure. Clearly, being able to meaningfully define such a thing puts stringent conditions on how structure is coupled to action. To even start thinking about this requires a powerful formal machinery capable of expressing the coupling of structure to action for the objects pertinent to a particular domain of application. This is the "object problem".

The "object problem" is nowhere seen more crisply than in chemistry. Chemical reactions are events in which both concentrations (i.e., quantities) and objects (i.e., structures) change. The projection of a chemical reaction involving large numbers of molecules on a phase-space of concentrations is known as reaction kinetics. To set up a chemical reaction as a dynamical system in concentration space, one only requires knowledge of the proper couplings among the concentrations of reactants and products. It is sufficient if these are known as empirical facts; knowledge of the chemical identity and properties of reactants and products is not necessary. Cranking the tools of, say, infintesimal calculus yields the time

evolution of reactant and product concentrations. Remove kinetics for a moment by considering just the information conveyed by a chemical reaction when it is notated on paper. We are left with a reaction arrow, "$\rightarrow$", expressing a *relation* among molecular structures. A general method capable of describing the time evolution of the contents of a reaction vessel for an *arbitray* initial mixture of molecular species would require nothing less than a formal system implicitly representing the space of molecular objects and the relation "$\rightarrow$" over them. That is a formal theory of chemistry.

3.2 ... to the foundations of mathematics

The claim that overcoming that limit requires a theory of object construction and, thus, necessarily involves a substantive overlap with the foundations of the computational sciences.

Where do we get a formal theory of chemistry? The answer is crucial to our approach. To date we do not have a formal, axiomatized theory of chemistry that is useful in everyday practice, despite the fact that quantum mechanics successfully grounds chemistry in the behavior of electrons and nuclei. The problem is one of choosing the "right" level of description. With respect to both molecular biology and industrial metabolisms alike, quantum mechanics is far too fine grained, and, aside from issues of feasibility, does not convey a satisfactory understanding of "what chemistry is actually doing"; it is too close to the trees to see the forest. To put it provokingly, our understanding of life will derive in large measure from *how* we understand chemistry. It is clear, then, that identifying a coarser grain of analysis capable of hosting a formal theory of chemistry would be of tremendous practical import and by no means limited to the foundations of theoretical biology.

Chemistry and computation

The stance we took in prior work [26, 27], and further elaborated here, is based on the intuition that at some level of description the reactive processes of chemistry are *analogous* to manipulations (rewrites) of syntactical objects. This puts us right into the domain of the computational sciences, section 2.1.1. In the present context, the reader is well advised to detach from an all-too narrow notion of computation as "number crunching". Much effort in the computational sciences goes into devising formal systems of syntactical constructs (we call "objects") that are interrelated by operations of transformation defined on them. It is in this sense that "computation" is the science of the construction of abstract objects with structure-specific "behavior".

What is crucial in the present context is *how* object behavior is synthesized from basic elements, as it is here that insights into fundamental mechanisms of "construction" or "interaction" are revealed, and can be compared with empirical facts. For what is desired in a theory of objects is not just a formalism and the theorems that accompany it, but a transparency of interpretation in the intended application domain. In a chemical application, one wants to capture *at least* the twin facts that (i) product molecules are lawfully constructed from substitution of parts of reactant molecules and (ii) that the same product can be produced by a diversity of different reactants. One wants, therefore, a theory of *combinational structures* and *substitution* together with the resultant theory of *equality*. Indeed, this is what drew us originally to λ-calculus. A great variety of alternative formal systems - Turing machines, Petri nets, Post systems, cellular automata, to mention but a few - allow expression of the same set of functions on the natural numbers, and, thus, may be regarded as being equivalent *in that respect*. It should be clear by now, however, that what is needed here is not merely a member of this universality class, but a member whose features are germane to the chemical problem at hand.

Identifying λ-calculus as a plausible candidate for a chemical interpretation hardly qualifies λ-expressions as anything but the most metaphorical of molecules. It is merely a foothold. A crucial one, however. It enabled our painstaking progression from MC0 to MC2, showing how a formal representation can be preserved, while progressively refining the chemical interpretation of the operators of the formalism. This would have hardly been possible with, say, Turing machines, Boolean operators, or assembler code. In that respect we differ markedly from pertinent work by Kauffman [47, 49], Rasmussen [76], McCaskill [62], Thürk [85] and Ikegami [46].

The issue here is one of grounding and formalizing an ontology, not just of capturing a phenomenology. Yet, this does not require a one-to-one mapping between some formalism expressing computational processes and real-world molecules with their chemical reactions. The map should not be confused with the territory; we do not want to "simulate" chemistry. We take the computational perspective as one enabling a different - logical - level of description of chemistry which is distinct from one that accounts for its actual physical implementation. The latter is the domain of quantum physics. For example, whether the actual protein folding process belongs to the class of computable functions is entirely irrelevant to us. For all we need to capture is the *logic* of the connection between structure and action specificity, not the physical process by which this connection is implemented. Herein lies the point of a specification language for chemistry.

What is gained by a computational theory of objects?

What is gained is best seen in comparison with prior approaches to modelling chemical collectives [3, 24, 48, 70, 79], following Kauffman's [47, 48] original casting of the problem. Particularly germane here are the efforts of Bagley, Farmer, Kauffman and Packard [3, 24]. Their work is an ideal contrast, in that the objective is identical to our own, but the methodology used to achieve the end differs. In its essence their model consists of strings over some alphabet, for example 0's and 1's, meant to represent polymers that recognize each other by "complementarity" (0 pairs with 1). A string can act as the docking place for others, thereby catalysing specific concatenation and splicing reactions. This leads to the assembly of reaction networks capable of maintaining themselves on the basis of a monomer or string flow through the system. Such a model shares with our own the appearance of self-maintaining collectives (that they call autocatalytic).

The crucial difference between our approaches is that the core of our model is a *theory* of object construction - rather than the imitation of particular chemicals. This is what gives us the capacity to specify what the "organization" of an emerging collective of objects is in terms of a mathematical formalism. Three broad consequences follow.

1. The abstraction of molecules in MC0 and MC1 as symbolic functions allows organization to be detected as a closure of interaction, manifested by invariant syntactical regularities and invariant algebraic laws characterizing the action of those objects maintained in the collective. It cannot be overemphasized that this characterization can be made by an observer of the system who is ignorant of λ-calculus. Indeed, an organization can be specified as an algebraic rewrite system that is independent from λ-calculus, and, thus, the process by which it originated. With the theory of objects comes whole-cloth a quite different theory of the collective.

2. A formal theory of objects makes transparent which features of the collective organization derive from the underlying theory of objects, and which features are curiosities derived from particular initial conditions, parameter settings, or a particular chemical stance. The distinction is between what we have elsewhere called "digital naturalism" [29] and the claim for a theory of self-maintenance.

3. As emphasized at the outset, section 1, an abstract theory of objects plays a role analogous to that of differential equations. The analysis of a dynamical system cast in terms of differential equations yields the characterization of manifolds in phase space that govern the set of its *possible* trajectories. Consider our MC0 implementation and imagine we seeded our reactor with

one λ-expression known to be a basis for all λ-calculus. Imagine further that the container is itself infinite in size and the reactor would then be capable of holding all possible (normal form) expressions. When we impose a dynamics on the objects (which, in our case, is a scheme coupled to the reactor size), we sieve particular "trajectories" in object space. Recall that our kinetic scheme is designed to favor the maintenance of objects that are constructed by the extant population of objects. As a consequence "trajectory" in object space "converges" to an "attractor" - a self-maintaining organization.

This casting emphasizes the need for a theory of the "motion" in "object space" induced by the object constructors (here, functional application or logical inference) under the continously updating kinetics imposed by the extant network of objects. How might such a motion differ under dynamics different from our own (see for example [46])? Or with equivalent dynamics and different object-constructors? Is there a meaningful formal concept of a "trajectory" in "object space"? What is "continuity"? Is there a useful definition of "distance" between "attractors" (in our case, algebraic structures in λ-space)? Questions like these require a theory of objects, not only to be answered, but even to be asked.

The value is apparent. Consider just one instance. A methodological imperative of the dynamical systems approach is the *a priori* choice of the pertinent entities and their functional couplings defining the system. The fact that the choice must be made *a priori* has the consequence that *the dynamical systems methodology can never be used to address the origin of that same system*, a profound limitation to which we have referred elsewhere [26] as the existence problem. It is apparent in MC0 that our reactor in settling upon a self-maintaining set of objects has settled upon a fixed system of variables and functional couplings between them; thus, *particular dynamical systems appear as limiting cases (such as "fixed points") of constructive dynamical systems*. Are constructive dynamical systems "generators" of dynamical systems? If so, a formalization of the motion in such a space holds promise as a methodology to address scientific questions which include the phrase "the origin of...".

Grounding and unifying other's ideas

Attractive intellectual constructs previously lacking a formal interpretation are rendered accessible to conventional modes of scientific investigation in our setting. Maturana and Varela's concept of "autopoiesis" [56, 57, 58, 87, 88] is particularly close, indeed arguably indistinguishable, from our concept of organization. It shares the key ingredient that the system is composed of "components" which engage in a network of interactions that enable the continuous regeneration of these same "components". Thus the autopoietic system is, at essence, a matter

of constructive relationships closed upon interaction; this Varela labels autonomy (we say, self-maintenance). Autopoietic systems share with our λ-organizations a number of other features, including regenerative abilities, accessibility only to inputs that influence "component" interrelations, and capacity for hierarchical coupling.

Rosen's (M,R)-system (for metabolism-repair system) [20, 80] resembles an autopoietic system, but Rosen's "components" are pure abstract functionalities. Rosen packs a whole metabolism into a single functional letter (a "metabolic function"), or speaks of a "repair function" and a "replication function", the three of which entail each other in a circular fashion by mutually acting on their domains and ranges. The point that we find concordant here is Rosen's emphasis on the causal circularity inherent to functional organization. Note, however, that Rosen's (M,R)-systems lack any notion of object construction[9].

Thus both contructs - autopoietic systems and (M,R)-systems - share with ours the essential notion of closed relations of construction between the parts of the system. Our work departs from both, however, in providing a concrete theory of the conditions necessary to realize a universe of such systems and to characterize their features in a standard formal setting.

We suspect that the autopoietic concept differs *only* as a consequence of Maturana and Varela and subsequent investigators [71] having come to it without the benefits of viewing organization as the consequence of joining dynamics and construction. The only claim of Maturana and Varela that is not instantiated in our organizations is their requirement that the system be spatially bounded. This is essential for them, for it is the only device by which their "components" may be isolated from the "rest-of-the-world". The seeming need of a membrane laid out in space is, in our view, only required because the characterization of autopoietic systems is not built upon a theory of its components. Our organization are indeed bounded, but bounded syntactically (i.e., λ-organizations are special invariant subspaces of λ-space). A bounding is indeed a necessary feature of organizations, but the space need not be 3-space. Perhaps it is not surprising that several disciplines which have found the concept of autopoiesis of utility (e.g., notably law and social psychiatry) find the requirement of spatial bounding dispensible (see review by [71]). At minimum, then, our work has converged to a notion similar to that of autopoiesis from an independent angle; quite plausibly, though, we have unwittingly generated a formal interpretation of a heretofore frustratingly elusive notion of considerable importance.

[9]Rosen claims that (M,R)-systems are inherently unformalizable. Casti [11, chapter 7] approaches some of their aspects by means of dynamical systems.

3.3 ... to concurrency and self-organization

The position that a concept of "organization" derives from placing a theory of objects in a suitably constrained many-body dynamical setting (section 2.1.2). The conventional settings of either dynamics alone or syntactical manipulation alone are insufficient; "organization" derives (or, if one prefers, self-organizes) from their combination in a constructive dynamical system.

It bears emphasis that "objects", as we frame them here, are defined by structure-action relationships *where each action is a mapping from structures to structures*. In a many-body setting this generates a *constructive feed-back loop* (in analogy to dynamical feed-backs) which causes the emergence of "organization". The so-defined constructive feed-back is absent in genetic algorithms [44, 39], genetic programming [53], classifier systems [43], and models of evolutionary optimization [1, 28]. While these systems deal with objects whose structure entails action, the action does not participate in object construction. This is exactly what puts our concept of organization outside their scope.

We hestitate to attribute to constructive dynamical systems claims of "emergence" or "self-organization", in that these terms are increasingly used with quite different attributions. Our organizations, however, do emerge in the sense that an organization possesses (*ex post*) a level of description that is independent of the abstract chemical universe within which it originated. Similarly, the core-objects (constructors) of the organization self-organized in the restricted (but meaningful) sense that the constructive dynamical system converged to them by an endogenous motion in object space.

Our use of the words self-organization and emergence differ sharply from the frequent use of these terms as meaning "a phenomenon displayed by a collective and unexpected by the investigator." The distinction is, again, one between a theory of the collective grounded in a formalism to which an interpretation (a meaning) is given to the operators of the formalism and the observation of an instance of collective phenomena for which no underlying theory guides interpretation or guarentees generality (i.e., "digital naturalism", *sensu* [29]). In drawing this distinction, we intend no disrespect for the value of such "complex systems" studies. In at least two cases a lack of formal grounding of components of the collective is appropriate. First, it is appropriate whenever the underlying suite of behaviors of the components are themselves empirically established to be disassociable from the features of the system left unmodelled. Examples include much of individual-based modelling in behavioral and community ecology (i.e., there is no need for a theory of molecules-as-proofs to study the consequences of odor trails on patterns of ant dispersion). Second, it is often desirable to leave uninterpreted the nature of the objects when one is seeking to implement a system whose objective is a search. Holland's genetic algorithms [44] and classifier systems [43] need

not be faithful chromosomes or genotype-to-phenotype representations when the intended objective is an efficient search engine.

A more appropriate embedding of constructive dynamical systems lies in the domain of computer science addressing concurrency. The ground here is not so much infertile as it it poorly prepared for sowing. Susan Oyama's underground classic, *The Ontogeny of Information* [73], documents the lack of rudimentary intellectual hygiene in the free use of metaphor from computer science to describe and interpret biological observations. The attributions are ubiquitous: the genes as "code", the genome as "algorithm", the cell as "massively parallel computer", and the like. Indeed, the metaphors are not merely passive inaccuracies in the service of rhetorical aims; they actively frame our thinking (see, e.g., [74] on representations of the *ras* pathway). We will not tread lightly here. *The use of the computational metaphor is crucially vacuous without a formal translation between a chemical syntax and a syntax of computation.* Here we reawaken the problem of interpretation; our as-yet-incomplete march from MC0 to MCn has achieving this formal interpretation as its intent.

It is, indeed, our stance that the formalisms *natural* to biology derive not from physics (as a discipline), but from imposing those of physics on those of computer science. Organisms are coherent chemical collectives, molecules are constructed from molecules, and computation is a science of representing possible constructions. The control of timing the interaction of physical objects and the representation of the same objects as performing constructions (computations) lies at the heart of both realizing a genuinely parallel computer and our attempt to develop a meaningful representation of a chemical parallelism (viz our discussion of reaction classes in MC2, section 2.3.2). Disarticulating the physical from the computational is what makes parallelism a hard problem in computer science.

The relation is sufficiently subtle to remind the reader of the manner in which our organizations were realized. The strategy we adopted was two-pronged: we first have projected chemical objects onto abstract logical entities, using λ-calculus as a formalization of their constructive interactions. Then we have dipped these entities into a cocktail of chosen physical modalities: (i) many objects *coexist* in the same system (reactor), (ii) an object-species has a *concentration* (objects can occur in multiple instances), (iii) objects interact by *random collision*, (iv) objects have a *finite lifetime* (constrained flow reactor).

Point (i) enables a constructive feed-back loop by providing a *context* from which the interaction partners of an object are drawn, and to which the product of a reaction is returned. Points (ii) and (iii) provide a simple (pseudo-chemical) kinetics that biases interactions to occur among object species with the highest concentration (*relevance*). Point (iv) implements the overall effect of removal, such as loss, decay, or inactivation. In conjunction with points (ii) and (iii) this

removal reinforces precisely those reaction pathways whose constituents on the reactant side also appear on the product side somewhere along the path. The result are self-maintaining collectives of objects, with each object *being simultaneously interpreted as a physical object and a function.*

We are drawn to this view of parallelism by physical intuition – flow reactors and chemicals being common laboratory objects. The computer scientist comes from a quite different intuitional base. Physicality is foreign ground, the strength of computation by syntactical manipulation lies in its leaving open the interpretation of the syntax. We find it remarkable, then, to what degree our work abrades with that of theoretical computer science. The body of work known as *communication and concurrency* [42, 64, 65, 67], aims at a formalization of the behavior of systems consisting of many coexisting and independently interacting heterogenous computational agents. Individual agents are referred to as "processes". Processes "communicate" to influence each other's behavior (i.e., the ability to communicate). Two examples serve to illustrate how short the intellectual distance is separating the issues in concurrency and our approach to organization. In π-calculus [68, 69], a popular concurrent formalism (Appendix A.3), a single expression is equivalent to the entire content of our reactor at a given point in time. The evolution of the reactor appears as a series of "reductions" of this large expression. Even more striking is the correspondence between our system and a device used to choose among the multiple communication channels available to a given concurrent process. To address this issue, Berry and Boudol [5, 6] introduced the notion of a Chemical Abstract Machine as a possible execution machine for the π-calculus. Berry and Boudol's insight - which predates our work - was to use physical aspects of chemistry (such as randomly colliding objects) to implement a concurrent computation, while we independently originated the reverse interpretation, i.e., to use computational objects as a proxy for molecules and to join them together into a concurrent setting. Despite the distance in intended application, then, both approaches share the underlying challenge of imposing a physics upon computational construction. Computer scientists seeking to *implement* concurrency no longer have the luxury of ignoring physical modalities - their processes must communicate in time to other processes with "real-world" positions and properties. This is, again, little different from our own problem in reverse; we start from a dynamical systems setting and need to add to it a formalism of object construction. We both face the challenge of confronting time and properties that characterize objects as physical entities, while simultaneously endowing the same entities with the power of construction.

Despite this correspondence, our efforts depart sharply from work in concurrency and communication in one important respect. Our organizations self-organize from random communications in a many-body, flow-reactor setting. Self-organization is anathema to computer engineers - indeed, such lack of rigid control

over communication is precisely what they seek a formalism to avoid. Control over communication is deemed essential; one need only imagine the task of a systems administrator attempting to communicate with an operating system after a series of system calls have unexpectedly self-organized. Yet, as our organizations indicate, the product of constructive dynamical systems is not a lack of coherent behavior, but the creation of collectives with predictable features and properties. Perhaps the treatment of concurrency would benefit from exploring the feature in which our work departs from their own. Surely - at the moment - the prospect of e-mail messages spontaneously combining into a coherent manuscript strikes the authors as an acceptable price to pay for an occasional unexpected core dump!

3.4 ... to biology and beyond

Finally, the "organizations" resulting from a constructive dynamical setting have the potential to address problems that have stubbornly resisted solution.

Biology has only two claims to theories unto itself - Mendel's theory of transmission and Darwin's theory of natural selection. The intellectual history of the first half of this century is a story of continuing debate over whether the two theories were in conflict. Fisher, Haldane, and Wright demonstrated that no conflict existed and the same fields are filled today by a self-perpetuating army of investigators using the *same plows* (powered now by computers much as farmers today use tractors). The talk of plows and tractors is not intended as idle ridicule, for the tools are the issue here; the limits of the tools define the "barrier" we address. The tools employed by Fisher to show that the great theories of biology were concordant required casting the problem in a fashion that threw out the constructive aspects of biology, rendering the problem tractable as one in dynamical systems. Throwing out construction meant throwing out the organism; trying to put the organism back in is fair epitome of the intellectual history ever since.

The claim is that biology requires a trinity of theories; we have two of them; we lack only "a theory of the organism". The claim we make is that the self-maintaining organizations we derive hold promise as that missing theory. Indeed, such a theory need not await a global solution to the specification language for chemistry. The fact that our organizations can be described in a formalism distinct from that in which they were generated (i.e., as abstract rewrite systems rather than λ-expressions) leaves their terms (like those of any syntax) open to interpretations other than chemical. From this realization flows a diversity of potential applications.

Given a universe of self-maintaining abstract rewrite systems (ARS), the uses are limited solely by the properties of the particular ARS and the interpretation given to its terms. An interpretation of the terms as engineering functions in a machine

might be route to a self-repair mechanism, an interpretation as a semiotic unit as a device for natural language interpretation, and interpretation of terms-as-molecules as germane to a blueprint for the design of a self-maintaining chemical manufacturing process [7], as it is to a metabolic cycle in a cell or a system of cell-cell communications defining an organ. The origin of such specifications from a research program in artifical chemistry or in experimental λ-calculus is irrelevant. It cannot be overemphasized that herein lies the significance of the characterization of our organizations in an alternative formalism.

To the extent that one accepts that the missing "theory of the organism" is recognizable as a general specification procedure for self-maintaining systems of constructive relations, MC0 suffices. Application to the biological issues left wanting for a generation - indeed to domains distinct from biology - are limited solely by the interpretation given to the terms of the abstract rewrite system and the extent to which its properties are germane to the question at hand. While work-in-progress portends considerable promise in applications to evolutionary biology in particular, it is neither feasible nor appropriate to address them here. After all, the editors asked us to identify a "barrier to knowledge", they did not ask us to lift it.

The claim for relevance here is large indeed. Hence, we conclude with a warning. The optimism and the passion with which we assess the potential of constructive dynamical systems has all the characteristics of a bullish investor at the eve of the market's collapse. The reader would do well to heed the admonition

...beware of the boa constructor.
Erwin Panofsky

Acknowledgements: We thank Harald Freund for numerous discussions on the limits and potentials of MC0. This is paper #41 from the Center for Computational Ecology at Yale.

References

[1] C. Amitrano, L. Peliti, and M. Saber. Population dynamics in a spin-glass model of chemical evolution. *J. molec. Evol.*, 29:513–525, 1990.

[2] A. Asperti. Causal dependencies in multiplicative linear logic with MIX. *Math. Struct. in Comp. Sci.*, 11:1–31, 1993.

[3] R. J. Bagley and J. D. Farmer. Spontaneous emergence of a metabolism. In C. G. Langton, C. Taylor, J. D. Farmer, and S. Rasmussen, editors, *Artificial Life II*, Santa Fe Institute Studies in the Sciences of Complexity, pages 93–141, Redwood City, 1992. Addison-Wesley.

[4] H. G. Barendregt. *The Lambda Calculus: Its Syntax and Semantics*. Studies in Logic and the Foundations of Mathematics. North-Holland, Amsterdam, second revised edition, 1984.

[5] G. Berry and G. Boudol. The chemical abstract machine. In *17th ACM Annual Symposium on Principles of Programming Languages*, pages 81–94, New York, 1990. ACM Press.

[6] G. Berry and G. Boudol. The chemical abstract machine. *Theoretical Computer Science*, 96:217–248, 1992.

[7] P. Bro. Artificial life in real chemical reaction systems. Book manuscript in preparation (contact address: P. Bro, Santa Fe Institute, 1399 Hyde Park Road, Santa Fe NM 87501), 1995.

[8] L. W. Buss. *The Evolution of Individuality*. Princeton University Press, Princeton, 1987.

[9] L. Cardelli. Type systems. Chapter in a forthcoming CRC Handbook of Computer Science and Engineering, available at http://www.research.digital.com/SRC/personal/Luca_Cardelli/Papers.html, 1996.

[10] L. Cardelli and P. Wegner. On understanding types, data abstraction, and polymorphism. *Computing Surveys*, 17:471–522, 1985.

[11] J. Casti. *Reality Rules (vol.2)*. Wiley, New York, 1992.

[12] A. Church. A set of postulates for the foundation of logic. *Annals of Math. (2)*, 33:346–366, 1932.

[13] A. Church. A set of postulates for the foundation of logic (erratum). *Annals of Math. (2)*, 34:839–864, 1932.

[14] A. Church. *The Calculi of Lambda Conversion*. Princeton University Press, Princeton, 1941.

[15] L. Damas and R. Milner. Principal type-schemes for functional programs. In *Proceedings of the 9th Annual Symposium on Principles of Programming Languages*, pages 207–212, New York, 1982. ACM.

[16] V. Danos and L. Regnier. The structure of the multiplicatives. *Arch. Math. Logic*, 28:181–203, 1989.

[17] V. Danos and L. Regnier. Proof-nets and the Hilbert space. In J.-Y. Girard, Y. Lafont, and L. Regnier, editors, *Advances in Linear Logic*, London Mathematical Society Lecture Note Series, pages 307–328, Cambridge, 1995. Cambridge University Press.

[18] N. G. de Bruijn. A survey of the project Automath. In J. R. Hindley and J. P. Seldin, editors, *To H. B. Curry: Essays on Combinatory Logic, Lambda-Calculus and Formalism*, pages 579–607. Academic Press, New York, 1980.

[19] M. E. Szabo (ed.). *The Collected Papers of Gerhard Gentzen*. North Holland, Amsterdam, 1969.

[20] R. Rosen (ed.). *Foundations of Mathematical Biology (vol.2)*. Academic Press, New York, 1972.

[21] M. Eigen. Self-organization of matter and the evolution of biological macromolecules. *Naturwissenschaften*, 58:465–526, 1971.

[22] M. Eigen, J. S. McCaskill, and P. Schuster. The molecular quasi-species. *Advances in Chem. Phys.*, 75:149–263, 1989.

[23] M. Eigen and P. Schuster. *The Hypercycle*. Springer Verlag, Berlin, 1979.

[24] J. D. Farmer, S. A. Kauffman, and N. H. Packard. Autocatalytic replication of polymers. *Physica D*, 22:50–67, 1982.

[25] A. Fleury and C. Retoré. The mix rule. *Math. Struct. in Comp. Sci.*, 4:273–285, 1994.

[26] W. Fontana and L. W. Buss. 'The arrival of the fittest': Toward a theory of biological organization. *Bull. Math. Biol.*, 56:1–64, 1994.

[27] W. Fontana and L. W. Buss. What would be conserved 'if the tape were played twice'. *Proc. Natl. Acad. Sci. USA*, 91:757–761, 1994.

[28] W. Fontana, W. Schnabl, and P. Schuster. Physical aspects of evolutionary optimization and adaptation. *Phys. Rev. A*, 40:3301–3321, 1989.

[29] W. Fontana, G. Wagner, and L. W. Buss. Beyond digital naturalism. *Artificial Life*, 1:211–227, 1994.

[30] G. Frege. *Begriffsschrift, eine der arithmetischen nachgebildete Formelsprache des reinen Denkens*. Louis Nebert, Halle, 1879.

[31] H. Freund. Self-maintaining λ-organizations and analysis via rewrite systems. Poster presented at the 3rd European Conference on Artificial Life (ECAL 95) in Granada, Spain, June 1995.

[32] G. Gentzen. Untersuchungen über das logische Schließen. *Mathematische Zeitschrift*, 39:176–210, 405–431, 1935.

[33] J.-Y. Girard. Linear logic. *Theoretical Computer Science*, 50:1–102, 1987.

[34] J.-Y. Girard. Towards a geometry of interaction. In J. W. Gray and A. Scedrov, editors, *Categories in Computer Science and Logic*, pages 69–108. American Mathematical Society, 1989. Proceedings of the AMS-IMS-SIAM Joint Summer Research Conference, June 14–20, 1987, Boulder, Colorado; Contemporary Mathematics Volume 92.

[35] J.-Y. Girard. Geometry of interaction II: Deadlock-free algorithms. In P. Martin-Löf and G. Mints, editors, *COLOG-88*, pages 76–93. Springer-Verlag LNCS 417, 1990.

[36] J.-Y. Girard. La logique linéaire. *Pour La Science, Edition Française de 'Scientific American'*, 150:74–85, April 1990.

[37] J.-Y. Girard. Linear logic: Its syntax and semantics. In J.-Y. Girard, Y. Lafont, and L. Regnier, editors, *Advances in Linear Logic*, pages 1–42. Cambridge University Press, 1995. Proceedings of the Workshop on Linear Logic, Ithaca, New York, June 1993.

[38] J.-Y. Girard, Y. Lafont, and P. Taylor. *Proofs and Types*. Cambridge Tracts in Theoretical Computer Science 7. Cambridge University Press, 1988.

[39] D. E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley, Reading, Mass., 1989.

[40] C. Hankin. *Lambda Calculi. A Guide for Computer Scientists*. Clarendon Press, Oxford, 1994.

[41] M. Hennessy. *Algebraic Theory of Processes*. The MIT Press, Cambridge, Mass., 1988.

[42] C. A. R. Hoare. *Communicating Sequential Processes*. Prentice Hall, Englewood Cliffs, 1985.

[43] J. H. Holland. Escaping brittleness: The possibilities of general purpose machine learning algorithms applied to parallel rule-based systems. In R. S. Michalski, J. G. Carbonell, and T. M. Mitchell, editors, *Machine learning: An artificial intelligence approach*. Kaufmann, Los Altos, CA, 1986.

[44] J. H. Holland. *Adaptation in Natural and Artificial Systems*. Bradford Books, The MIT Press, Cambridge, Mass., 1992. reprint edition.

[45] W. A. Howard. The formulae-as-types notion of construction. In J. R. Hindley and J. P. Seldin, editors, *To H. B. Curry: Essays on Combinatory Logic, Lambda-Calculus and Formalism*, pages 479–490. Academic Press, 1980.

[46] T. Ikegami and T. Hashimoto. Coevolution of machines and tapes. In F. Morán, A. Moreno, J. J. Merelo, and P. Chacón, editors, *Advances in Artificial Life*, Third European Conference on Artificial Life, Granada, Spain, pages 234–245, Berlin, 1995. Springer Verlag.

[47] S. A. Kauffman. Cellular homeostasis, epigenesis and replication in randomly aggregated macromolecular systems. *J. Cybernetics*, 1:71–96, 1971.

[48] S. A. Kauffman. Autocatalytic sets of proteins. *J. Theor. Biol.*, 119:1–24, 1986.

[49] S. A. Kauffman. *The origins of order*. Oxford University Press, New York., 1993.

[50] J. W. Klop. Term rewriting systems. In S. Abramsky, D. M. Gabbay, and T. S. E. Maibaum, editors, *Handbook of Logic in Computer Science*, volume 2, pages 1–116. Clarendon Press, Oxford, 1992.

[51] J. W. Klop and A. Middeldorp. An introduction to Knuth-Bendix completion. *CWI Quarterly*, 1, 1988.

[52] D. E. Knuth and P. E. Bendix. Simple word problems in universal algebra. In J. Leech, editor, *Computational Problems in Abstract Algebra*. Pergamon Press, New York, 1970.

[53] J. R. Koza. *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. The MIT Press, Cambridge, Mass., 1992.

[54] Y. Lafont. From proof-nets to interaction nets. In J.-Y. Girard, Y. Lafont, and L. Regnier, editors, *Advances in Linear Logic*, London Mathematical Society Lecture Note Series, pages 225–247, Cambridge, 1995. Cambridge University Press.

[55] R. Lalement. *Computation as Logic.* Prentice Hall, Englewood Cliffs, 1993.

[56] P.-L. Luisi. Defining the transition to life: self-replicating bounded structures and chemical autopoiesis. In W. Stein and F. J. Varela, editors, *Thinking About Biology*, Santa Fe Institute Studies in the Sciences of Complexity, pages 3–23, Redwood City, 1993. Addison-Wesley.

[57] H. Maturana and F. J. Varela. *De Máquinas y Seres Vivos: Una teoría de la organizacíon biológica.* Editorial Universitaria, Santiago de Chile, 1973. Reprinted in: H. Maturana and F. J. Varela, *Autopoiesis and Cognition: The Realization of the Living*, 1980.

[58] H. Maturana and F. J. Varela. *Autopoiesis and Cognition: The Realization of the Living.* D. Reidel, Boston, 1980.

[59] M. Mauny. Functional programming using Caml Light. User's manual available from ftp.inria.fr by anonymous ftp, January 1995.

[60] J. Maynard-Smith. Natural selection and the concept of a protein space. *Nature*, 255:563–564, 1970.

[61] J. Maynard-Smith and E. Szathmáry. *The major transitions in evolution.* W. H. Freeman, Oxford, 1995.

[62] J. S. McCaskill. Polymer chemistry on tape: a computational model for emergent genetics. unpublished manuscript, MPI für biophysikalische Chemie, Göttingen, 1988.

[63] R. Milner. A theory of type polymorphism in programming. *Journal of Computer and System Sciences*, 17:348–375, 1978.

[64] R. Milner. *A Calculus of Communicating Systems.* Lecture Notes in Computer Science, Vol 92. Springer-Verlag, Berlin, 1980.

[65] R. Milner. *Communication and Concurrency.* Prentice Hall, New York, 1989.

[66] R. Milner. The polyadic π-calculus: a tutorial. Report ECS-LFCS-91-180, University of Edinburgh, 1991.

[67] R. Milner. Elements of interaction. *Comm. ACM*, 36:78–89, 1993.

[68] R. Milner, J. Parrow, and D. Walker. A calculus of mobile processes, I. *Information and Computation*, 100:1–40, 1992.

[69] R. Milner, J. Parrow, and D. Walker. A calculus of mobile processes, II. *Information and Computation*, 100:41–77, 1992.

[70] E. Minch. *Representation of Hierarchical Structure in Evolving Networks.* PhD dissertation, State University of New York at Binghamton, 1988.

[71] J. Mingers. *Self-Producing Systems - Implications and Applications of Autopoiesis.* Plenum Press, New York, 1995.

[72] H. J. Morowitz. *Beginnings of Cellular Life: Metabolism Recapitulates Biogenesis.* Yale University Press, New Haven, 1992.

[73] S. Oyama. *The Ontogeny of Information.* Cambridge University Press, Cambridge., 1985.

[74] S. Oyama. The accidental chordate: contingency in developmental systems. In B. H. Smith and A. Plotnitsky, editors, *Mathematics, Science, and Postclassical Theory (South Atlantic Quart., Vol. 94)*, pages 509–526. Duke University Press, Durham, NC, 1995.

[75] V. R. Pratt. The duality of time and information. In W. Cleaveland, editor, *Proceedings of the Third International Conference on Concurrent Theory*, pages 237–253, New York, 1992. Springer-Verlag.

[76] S. Rasmussen, C. Knudsen, R. Feldberg, and M. Hindsholm. The coreworld: Emergence and evolution of cooperative structures in a computational chemistry. *Physica D*, 42:111–134, 1990.

[77] C. Reade. *Elements of Functional Programming.* Addison-Wesley, Reading, Mass., 1989.

[78] C. Rétoré. *Réseaux et Séquents Ordonnés.* PhD dissertation, University of Paris VII, 1993.

[79] D. S. Rokshar, P. W. Anderson, and D. L. Stein. Self-organization in prebiological systems: Simulation of a model for the origin of genetic information. *J. Mol. Evol.*, 23:110, 1986.

[80] R. Rosen. *Life Itself: A Comprehensive Inquiry into the Nature, Origin, and Fabrication of Life.* Columbia University Press, New York, 1991.

[81] D. Sangiorgi. *Expressing Mobility in Process Algebras: First-Order and Higher-Order Paradigms.* PhD dissertation, University of Edinburgh, 1992.

[82] M. Schönfinkel. Über die Bausteine der mathematischen Logik. *Math. Annalen*, 92:305–316, 1924.

[83] B. C. Smith. *On the Origin of Objects.* Bradford Books, The MIT Press., 1996.

[84] E. Szathmáry. A classification of replicators and lambda-calculus models of biological organization. *Proc. R. Soc. Lond. B*, 260:279–286, 1995.

[85] M. Thürk. *Ein Modell zur Selbstorganisation von Automatenalgorithmen zum Studium molekularer Evolution.* PhD dissertation, Universität Jena, Germany, 1993.

[86] A. S. Troelstra. *Lectures on Linear Logic.* CSLI Lecture Notes 29, Center for the Study of Language and Information, Stanford, California, 1992.

[87] F. J. Varela. *Principles of Biological Autonomy.* North-Holland, New York, 1979.

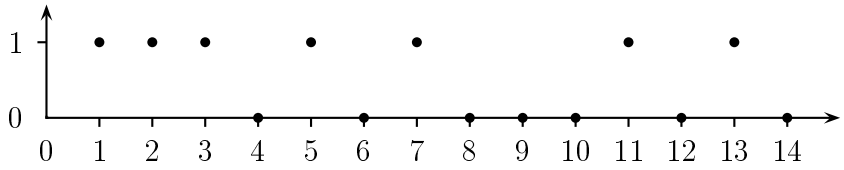
[88] F. J. Varela, H. R. Maturana, and R. Uribe. Autopoiesis: the organization of living systems, its characterization and a model. *BioSystems*, 5:187–196, 1974.

Appendix

A λ-calculus for tourists

A.1 Conceptual

The modern view of "function" is that of an arbitrary *set* of pairs - (argument/value) - whose first element is unique. The entire graph of the function is taken to be available at once, as a given, with no "cost" for its generation. For example, you are given the following graph, in which a prime is paired with 1 and a non-prime with 0, as fully completed to the infinite right.



In contrast, an older view emphasizes a function as a *rule of computation*, i.e., as a process of symbolic manipulation that produces a value when *applied* to an argument.

> Given a number n try dividing it by 2 and by each odd integer up to the biggest integer which is smaller or equal to the square root of n. If none of the trial divisors divides n, return 1, otherwise return 0.

The point is that one trades the instant random access to a look-up table which is so big as not to fit into the universe, with a procedure which fits into your pocket, but at the "cost" that it must be carried out. Procedures have to be expressed in some formal language. The requirement of a language, in turn, entails a refinement of the world into "*behavior*" and "*that which behaves.*" The former remains the still "ethereal" graph, while the latter is an "object", that is, a symbolic structure shaped by some sort of syntax that can be subject to new kinds of manipulation. This procedural or computational paradigm lies at the base of our project. It takes seriously the fact that in the physical world one never manipulates behavior, only the objects that behave.

Of course, the above example is no more than a joke; we would have to be equally explicit about what we mean by "divide", "2", "each", "integer", "biggest", "smaller", "or", "equal", "square root". It is no joke, however, that the λ-calculus invented by A. Church in the 1930's [12, 13] (following a trail pioneered by M. Schönfinkel [82]) does just that.

A.2 Instant Syntax and Semantics

The universe of λ-objects consists of *terms* with a particular structure. This structure is defined inductively, starting from "atoms".

TERMS

λ-calculus	Informal Interpretation	Tourist Notation
x	x is an atomic name taken from some available name space.	x
$\lambda x.A$	Make the λ-term A into a function of x. This is done by turning x in A from being a literal "x" into something replaceable, a variable. Since x now holds a place, it's particular name has lost significance. All that matters is the link to its corresponding λ marker, indicated by writing λx. One says, x has been *abstracted*.	Given the expression $x^2 + 4x + y$ we turn it into a function in x by declaring x to be a variable: $x \rightarrow x^2 + 4x + y$.
$(A)B$	If A and B are λ-terms, then their juxtaposition $(A)B$ denotes the *application* of A to B	Given the function $f : x \rightarrow x^2 + 4x + y$ and given 6, we can speak of $f(6)$.

So far we have only terms. Let's have some action.

REDUCTION

λ-calculus	Informal Interpretation
$(\lambda x.A)B \rightarrow A[x := B]$	When applying a function $\lambda x.A$ to an argument B, we proceed by literally *substituting* for the placeholder x the argument B. The fact that the place(s) held by x has (have) been filled is documented by removing the placeholder declaration λx. In the above example, $f(6) \rightarrow 60 + y$.

That's all there is.

Some details on reduction

There are two technical details one should be aware of. (i) The abstractor λ has a *scope* (like an ordinary integral sign), i.e., in $(\lambda x.A)B$ the binding influence of λ stops at A, and does not continue into B. (ii) The idea behind substitution is to replace equals by equals, meaning that the behavior of $(\lambda x.A)B$ should be the same as that of $A[x := B]$. Unbound literals must, therefore, never get bound during substitution. This one, for example, is illegal: $(\lambda x.\lambda y.(x)y)\lambda z.\boxed{\text{y}} \rightarrow \lambda y.(\lambda z.\boxed{\text{y}})y$. The boxed y has been captured by a λy. To perform the substitution safely, one has to rename the bound y into, say, w. We skip the formalization of these statements.

In a reduction step an "application" annihilates an "abstraction". **Normalization** is the process in which all the reductions that are possible within a term are carried out. At that point a term is said to be in **normal form**. The normal form is unique (if it exists - see below). This property of the reduction relation on λ-terms is called **confluence**. The reflexive, symmetric and transitive closure of the reduction relation is an **equivalence** relation on terms, i.e., two λ-terms are equivalent, if they have the same normal form.

Within the scope of our chemical metaphor a normal form is the analogue of a stable molecular form. The application of one (abstraction) term to another is analogous to a reactive encounter. Such a configuration is (usually) not a normal form, and is "stabilized" by normalization - the λ-analogue of a "reaction path". In this view of chemistry, "(free) energy" is that which causes molecular transition states to stabilize into products and is captured by our *requirement* that terms be in normal form. Other aspects of energy, such as differential rate constants, are not captured in Minimal Chemistry Zero (but see section 2.3.3 for MC2).

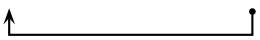
Two examples: $A \stackrel{\text{def}}{=\!=} \lambda x.((x)\lambda y.y)x$ is in normal form; so is $B \stackrel{\text{def}}{=\!=} \lambda u.(u)\lambda v.v$, but not $(A)B \stackrel{\text{def}}{=\!=} (\lambda x.((x)\lambda y.y)x)\lambda u.(u)\lambda v.v$. We normalize (underlining the subterms being reduced at each step):

$$\underline{(\lambda x.((x)\lambda y.y)x)\lambda u.(u)\lambda v.v} \rightarrow (\underline{(\lambda u.(u)\lambda v.v)\lambda y.y})\lambda u.(u)\lambda v.v \rightarrow$$
$$(\underline{(\lambda y.y)\lambda v.v})\lambda u.(u)\lambda v.v \rightarrow \underline{(\lambda v.v)\lambda u.(u)\lambda v.v} \rightarrow \lambda u.(u)\lambda v.v$$

In this case B is a fixed point of A.

Not every term has a normal form. For example, here's a term which is not normalizable:

$$(\lambda x.(x)x)\lambda x.(x)x \rightarrow (\lambda x.(x)x)\lambda x.(x)x$$



A calculus in which every term has a normal form is called **strongly normalizing**. Our model utilizes only terms with a normal form. Indeed, we even discard those terms which fail to normalize within some specified limits.

It is worth pointing out that in this version of λ-calculus *every term can be applied to every term*. Thus, any term can be filled into the place held by a variable. Or,

by means of slogan, there's no syntactical distinction between function and data. *Everything in λ is, in some sense, a function.* This is a very powerful concept. You may have wondered where the "numbers" are, or where the familiar "addition" has gone. No such operations are given. Everything has to be constructed just with what we've got, i.e., variables, abstraction and application. (It can be done.) The intriguing feature of λ-calculus is in forcing one to realize that something is, for example, a numeral (a representation of a number), if it *behaves* - via application and reduction - like a number. Something is a numeral when it is a member of a sequence of distinct terms that have a successor function, and there exists a test for a distinguished element "zero". Likewise, something is an "addition", if it *behaves* like an addition *in relation to some system of numerals.* Change the system of numerals and the object that behaved like an addition doesn't anymore. Here's an example of a numeral system:

$$\begin{array}{rcl}
\lambda f.\lambda x.x & \text{corresponds to} & 0 \\
\lambda f.\lambda x.(f)x & \text{corresponds to} & 1 \\
\vdots & \vdots & \vdots \\
\lambda f.\lambda x.\underbrace{(f)...(f)}_{n \text{ times}}x & \text{corresponds to} & n \\
\vdots & \vdots & \vdots
\end{array}$$

Relative to it, the addition operation becomes $+ \stackrel{\text{def}}{=} \lambda m.\lambda n.\lambda f.\lambda x.((m)f)((n)f)x$. The normalization of $3 + 2$ is displayed in table A1. It looks slightly frightening, but most of the 54 intermediate steps are just necessary rearrangements in preparation for reductions.

Note then the "relativity" of the system – one defines *what behaves* and generates behaviors. The power of the system derives from just this flexibility. "Behavior," in the example above, was treated as a device which sends numerals into numerals. This frame need not be maintained. In fact, nothing prevents us from taking the addition function, $+$, and apply it to something else than a numeral - to itself, say. So here is $(+)+$:

$$\begin{array}{rl}
 & (\lambda m.\lambda n.\lambda f.\lambda x.((m)f)((n)f)x)\lambda m.\lambda n.\lambda f.\lambda x.((m)f)((n)f)x \rightarrow \cdots \\
\cdots \rightarrow & \lambda n.\lambda f.\lambda x.\lambda u.\lambda v.((f)u)((((n)f)x)u)v
\end{array}$$

(Where the variable names u and v come from renaming during normalization.)

Can we still meaningfully say that a "function" has been computed? No. In full λ-calculus the notion of a "function" is better replaced by the more vague notion of an "operator". This point is crucial for our usage of λ-calculus. Indeed, *the individual interactions in our reactor by and large don't compute anything, they solely rearrange symbolic structures.* The interpretation of their "behavior"

is framed by the algebraic and kinetic properties of the organization that their actions participate in maintaining. The same can be said of chemistry.

$$(\underbrace{(\lambda m.\lambda n.\lambda f.\lambda x.((m)f)((n)f)x)}_{+}\ \underbrace{\lambda f.\lambda x.(f)(f)(f)x)}_{3}\ \underbrace{\lambda f.\lambda x.(f)(f)x}_{2}$$

1 $(\lambda n.(\lambda m.\lambda f.\lambda x.((m)f)((n)f)x)\lambda f.\lambda x.(f)(f)(f)x)\lambda f.\lambda x.(f)(f)x$
2 $(\lambda n.\lambda f.(\lambda m.\lambda x.((m)f)((n)f)x)\lambda f.\lambda x.(f)(f)(f)x)\lambda f.\lambda x.(f)(f)x$
3 $(\lambda n.\lambda f.\lambda x.(\lambda m.((m)f)((n)f)x)\lambda f.\lambda x.(f)(f)(f)x)\lambda f.\lambda x.(f)(f)x$
4 $(\lambda n.\lambda f.\lambda x.((\lambda m.(m)f)\lambda f.\lambda x.(f)(f)(f)x)(\lambda m.((n)f)x)\lambda f.\lambda x.(f)(f)(f)x)\lambda f.\lambda x.(f)(f)x$
5 $\lambda f.(\lambda n.\lambda x.((\lambda m.(m)f)\lambda f.\lambda x.(f)(f)(f)x)(\lambda m.((n)f)x)\lambda f.\lambda x.(f)(f)(f)x)\lambda f.\lambda x.(f)(f)x$
6 $\lambda f.\lambda x.(\lambda n.((\lambda m.(m)f)\lambda f.\lambda x.(f)(f)(f)x)(\lambda m.((n)f)x)\lambda f.\lambda x.(f)(f)(f)x)\lambda f.\lambda x.(f)(f)x$
7 $\lambda f.\lambda x.((\lambda m.(m)f)\lambda f.\lambda x.(f)(f)(f)x)(\lambda n.(\lambda m.((n)f)x)\lambda f.\lambda x.(f)(f)(f)x)\lambda f.\lambda x.(f)(f)x$
8 $\lambda f.\lambda x.(((\lambda m.m)\lambda f.\lambda x.(f)(f)(f)x)(\lambda m.f)\lambda f.\lambda x.(f)(f)(f)x)(\lambda n.(\lambda m.((n)f)x)\lambda f.\lambda x.(f)(f)(f)x)\lambda f.\lambda x.(f)(f)x$
9 $\lambda f.\lambda x.((\lambda f.\lambda x.(f)(f)(f)x)(\lambda m.f)\lambda f.\lambda x.(f)(f)(f)x)(\lambda n.(\lambda m.((n)f)x)\lambda f.\lambda x.(f)(f)(f)x)\lambda f.\lambda x.(f)(f)x$
10 $\lambda f.\lambda x.(\lambda x.(\lambda f.(f)(f)(f)x)(\lambda m.f)\lambda f.\lambda x.(f)(f)(f)x)(\lambda n.(\lambda m.((n)f)x)\lambda f.\lambda x.(f)(f)(f)x)\lambda f.\lambda x.(f)(f)x$
⋮ ⋮
25 $\lambda f.\lambda x.(f)(f)(\lambda y_1.(\lambda x.(y_1)x)(\lambda n.(\lambda m.((n)f)x)\lambda f.\lambda x.(f)(f)(f)x)\lambda f.\lambda x.(f)(f)x)f$
26 $\lambda f.\lambda x.(f)(f)((\lambda y_1.\lambda x.(y_1)x)f)(\lambda y_1.(\lambda n.(\lambda m.((n)f)x)\lambda f.\lambda x.(f)(f)(f)x)\lambda f.\lambda x.(f)(f)x)f$
27 $\lambda f.\lambda x.(f)(f)(\lambda x.(\lambda y_1.(y_1)x)f)(\lambda y_1.(\lambda n.(\lambda m.((n)f)x)\lambda f.\lambda x.(f)(f)(f)x)\lambda f.\lambda x.(f)(f)x)f$
28 $\lambda f.\lambda x.(f)(f)(\lambda x.((\lambda y_1.y_1)f)(\lambda y_1.x)f)(\lambda y_1.(\lambda n.(\lambda m.((n)f)x)\lambda f.\lambda x.(f)(f)(f)x)\lambda f.\lambda x.(f)(f)x)f$
29 $\lambda f.\lambda x.(f)(f)((\lambda y_1.y_1)f)(\lambda x.(\lambda y_1.x)f)(\lambda y_1.(\lambda n.(\lambda m.((n)f)x)\lambda f.\lambda x.(f)(f)(f)x)\lambda f.\lambda x.(f)(f)x)f$
30 $\lambda f.\lambda x.(f)(f)(f)(\lambda x.(\lambda y_1.x)f)(\lambda y_1.(\lambda n.(\lambda m.((n)f)x)\lambda f.\lambda x.(f)(f)(f)x)\lambda f.\lambda x.(f)(f)x)f$
⋮ ⋮
43 $\lambda f.\lambda x.(f)(f)(f)(\lambda x.((\lambda f.f)(\lambda n.f)\lambda f.\lambda x.(f)(f)x)(\lambda f.(f)x)(\lambda n.f)\lambda f.\lambda x.(f)(f)x)(\lambda n.x)\lambda f.\lambda x.(f)(f)x$
44 $\lambda f.\lambda x.(f)(f)(f)((\lambda f.f)(\lambda n.f)\lambda f.\lambda x.(f)(f)x)(\lambda x.(\lambda f.(f)x)(\lambda n.f)\lambda f.\lambda x.(f)(f)x)(\lambda n.x)\lambda f.\lambda x.(f)(f)x$
45 $\lambda f.\lambda x.(f)(f)(f)((\lambda n.f)\lambda f.\lambda x.(f)(f)x)(\lambda x.(\lambda f.(f)x)(\lambda n.f)\lambda f.\lambda x.(f)(f)x)(\lambda n.x)\lambda f.\lambda x.(f)(f)x$
46 $\lambda f.\lambda x.(f)(f)(f)(f)(\lambda x.(\lambda f.(f)x)f)(\lambda n.x)\lambda f.\lambda x.(f)(f)x$
47 $\lambda f.\lambda x.(f)(f)(f)(f)((\lambda x.\lambda f.(f)x)(\lambda n.x)\lambda f.\lambda x.(f)(f)x)(\lambda x.f)(\lambda n.x)\lambda f.\lambda x.(f)(f)x$
48 $\lambda f.\lambda x.(f)(f)(f)(f)(\lambda f.(\lambda x.(f)x)(\lambda n.x)\lambda f.\lambda x.(f)(f)x)(\lambda x.f)(\lambda n.x)\lambda f.\lambda x.(f)(f)x$
49 $\lambda f.\lambda x.(f)(f)(f)(f)(\lambda f.(f)(\lambda x.x)(\lambda n.x)\lambda f.\lambda x.(f)(f)x)(\lambda x.f)(\lambda n.x)\lambda f.\lambda x.(f)(f)x$
50 $\lambda f.\lambda x.(f)(f)(f)(f)((\lambda f.f)(\lambda x.f)(\lambda n.x)\lambda f.\lambda x.(f)(f)x)(\lambda f.(\lambda x.x)(\lambda n.x)\lambda f.\lambda x.(f)(f)x)(\lambda x.f)(\lambda n.x)\lambda f.\lambda x.(f)(f)x$
51 $\lambda f.\lambda x.(f)(f)(f)(f)((\lambda x.f)(\lambda n.x)\lambda f.\lambda x.(f)(f)x)(\lambda f.(\lambda x.x)(\lambda n.x)\lambda f.\lambda x.(f)(f)x)(\lambda x.f)(\lambda n.x)\lambda f.\lambda x.(f)(f)x$
52 $\lambda f.\lambda x.(f)(f)(f)(f)(f)(\lambda f.(\lambda x.x)(\lambda n.x)\lambda f.\lambda x.(f)(f)x)f$
53 $\lambda f.\lambda x.(f)(f)(f)(f)(f)(\lambda x.x)(\lambda n.x)\lambda f.\lambda x.(f)(f)x$
54 $\lambda f.\lambda x.(f)(f)(f)(f)(f)(\lambda n.x)\lambda f.\lambda x.(f)(f)x$

$$\overbrace{\lambda f.\lambda x.(f)(f)(f)(f)(f)x}^{5}$$

Table A1: $3 + 2 = 5$ in λ-calculus.

λ-calculus is a theory of equality based on substitution. In a more specialized sense, λ-calculus is a general theory of functions. Having served as a template for LISP, it inspired the "functional style" of programming. Moreover, as detailed in text, λ-calculus has served as a tool in constructive proof-theory and, accordingly, in mechanizing parts of logic (see section C.1). This makes λ-calculus an almost obligate check point when introducing new paradigms of computation to which properties such as termination, confluence, normalization, or substitution

are central. *In the computational sciences these ingredients of λ-calculus play a role comparable to that of the fundamental principles of physics* [55]. Good introductions are Hankin [40] or Lalement [55], an encyclopedic treatment for aficionados with a good pair of shoes is Barendregt [4].

A.3 Beyond λ

The limits of λ-calculus are found in its inherently sequential paradigm. This is reflected by its non-commutative basic mode of interaction, application. What would be a commutative analogue? This question leads one to parallelism, or more precisely, *concurrency.* In contrast to sequentiality, the concurrent paradigm of computation considers a system of many heterogenously behaving independent entities that "interact" with one another (usually asynchronously). The proper technical word for interaction in such a setting is *communication* and the entities are called *processes.* The theory of communication and concurrency is among the most exciting and challenging frontiers in today's computational sciences. This is obviously not the place for a tutorial in concurrency; some places to start are [42, 65, 67]. Here we paint with a broad brush just some of the issues at stake so as to situate our work relative to it.

The transition from the concept of "function" to that of a "process" is illustrated by means of an example due to Robin Milner [67]. Consider the behavior of the following program, A, where "`:=`" means an assignment:

```
1: x := 2;          (assign 2 to x)
2: y := x+3;        (assign to y the content of x plus three)
3: print y;
```

Clearly, the program A will print `5`. Suppose now that there is a further concurrently running program B that has access to the memory location referred to by `x` in A. Such a program can alter the value at `x` after A has executed its first statement and before it executes the second. As a consequence, the observation of A does not yield a specific result anymore; it may print anything, depending on the behavior of B. This is the kind of situation that the theme of "communication and concurrency" is roughly about. The concept of "process" emphasizes behavior primarily as the ability to communicate at various points in time [41] rather than a computational activity. The issue, as emphasized in [75], is one of an apparent duality of time and information.

What is being communicated? The simplest kind of communication is a synchronization between two processes, i.e. a "handshake". One process pauses until it receives a signal from another upon which it resumes its behavior [64]. The

next order of communication involves the sending and receiving of port names themselves. This yields a system where "pointers" are passed around, thereby changing the communication topology of the system over time [68, 69]. At the next order whole processes rather than their address can be communicated [81]. This in turn raises the issues of "access" and "privacy". On the formal side the challenge is to find "calculi" which enable to reason about various notions of processes and their equivalence, in analogy to what λ-calculus does for the sequential realm.

Communication occurs between *ports* of processes. Ports are named, and each name has a complement. For example: a and $\overline{a}$, where a may stand for an input port and $\overline{a}$ for an output port. Communication can only occur between ports that bear complementary names. This ensures commutativity of communication by definition.

π-calculus

One foundational attempt at mobile processes (systems with changing communication topology) is the π-calculus of Milner, Parrow and Walker [68, 69]. Just to give a glimpse of it for the purpose of comparison with λ-calculus, here's a π-expression:

$$\underbrace{\bar{x}y.A}_{1} \mid \underbrace{x(u).\bar{u}v.B}_{2} \mid \underbrace{y(z).C}_{3}$$

It denotes a soup of three processes, $1, 2, 3$, that co-exist independently. This concurrence is expressed by the operator $|$. The processes in the example are only partially specified, since A, B and C stand for further structure which we disregard. $\bar{x}y$ means "output the name y along channel x", while $x(u)$ means "receive a name along channel x and substitute that name for u in the remaining process" (this input prefix binds u much like a λ). In the above example a communication can occur between process 1 and process 2 along channel x, yielding:

$$\underbrace{A}_{1'} \mid \underbrace{\bar{y}v.B[u := y]}_{2'} \mid \underbrace{y(z).C}_{3}$$

The point is that, as a result of this event, process 2 (now $2'$) has obtained a port name that enables it to communicate with process 3:

$$\underbrace{A}_{1'} \mid \underbrace{B[u := y]}_{2''} \mid \underbrace{C[z := v]}_{3'}$$

Concurrent processes can occur embedded within a process, such as in $\bar{x}y.(A|B)$. Furthermore, there is a scoping operator ν which restricts the use of x to process A in $(\nu x)A$, and there is a choice operator $+$ behaving so that in $A + B$ a communication with A destroys B and vice versa. Finally, there is a replication operator $!A$ which permits process A to spin off further copies of itself allowing for recursion.

"|" might be a way to notate the concurrency of the λ-particles in our flow-reactor. As in λ-calculus, a "type"-discipline for π-calculus can be defined. For

the further development of our model, we are inclined towards the logic path to concurrency rather than π-calculus, as developed in the text. Readers wishing to further explore π-calculus are referred to [66].

The world of functions and the world of processes emphasize the halting problem differently. While termination is a *desideratum* for functions or algorithms, the opposite is typically true for processes. There one looks for conditions under which a community of processes is guaranteed never to dead-lock, as there are many situations where ongoing communication or interactivity is required. Examples include operating systems, whether in air traffic control systems, computer systems, mobile telephone networks, or...living and cognizing systems. *The focus on the absence of dead-lock shifts the attention from computation to organization.* This clearly locates concurrency very close to our project.

B Types for tourists

B.1 The chemistry of types

Types are a high-level statement about the *behavior* of objects. A conventional addition function, for example, has type $N \times N \to N$, meaning that it accepts pairs of integers, and returns integers. This is not sufficient to distinguish it from a subtraction function, but is enough to distinguish it from a function that adds "carriage returns" to a string of characters. The definition of a *type system* decides on how much about the actual behavior of an object is conveyed by its type. A type system also provides a procedure to infer the type of a compound object from the types of its components.

We first need a way to express types. The notation is inductive like the syntax of λ-calculus. We start by defining a set of atomic types, called *simple (or ground) types*, say $\mathcal{T} = \{\mathtt{a}, \mathtt{b}, \mathtt{c}, \ldots\}$. From simple types we construct compound types with the help of *type constructors*. This is analogous to λ-calculus where compound terms are built from two *term constructors* - abstraction and application (see Appendix A). The choice of type constructors reflects the kind of actions one seeks to capture. For the sake of simplicity we consider here only one type constructor: the *function type* "$\to$". A type, then, is either

- a simple type: $\mathtt{s} \in \mathcal{T}$, or
- a function type: $\mathtt{s} \to \mathtt{t}$, where $\mathtt{s}$ and $\mathtt{t}$ are types.

The function type $\mathtt{s} \to \mathtt{t}$ denotes a mapping which accepts objects of type $\mathtt{s}$ and

returns objects of type `t`. Think of it as one kind of chemical bond ("$\rightarrow$"). It links together the *action(s)* of atoms (or groups of atoms).

The type system is coupled to the λ-calculus by means of *inference rules* based on the structure of λ-terms. Let's proceed intuitively at first: if a variable x in λ-calculus has type `s`, notated $x : \mathtt{s}$, and if the expression E has type `t`, notated $E : \mathtt{t}$, then the term $\lambda x.E$ has type $\mathtt{s} \rightarrow \mathtt{t}$, notated $\lambda x.E : \mathtt{s} \rightarrow \mathtt{t}$. What we have just made is a bond connecting action `s` with action `t`. The corresponding term - the abstraction $\lambda x.E$ - *is* the physical bond - as opposed to its type ($\rightarrow$) which indicates it's potential chemical activity. Indeed, a bond that can be made, can be broken. This holds for types as well. When the object $\lambda x.E$ whose action is $\mathtt{s} \rightarrow \mathtt{t}$ is brought into contact with an object $F : \mathtt{s}$, the bond is broken, and the action `t` is recovered: $(\lambda x.E)F : \mathtt{t}$. The corresponding normalized object can be shown to have the same type. The fact that the type doesn't change upon normalization indicates that *types do not compute results*; the computation is done by the λ-calculus mechanics. Types are just a statement about the possible reactions and results.

Summing up, "abstraction" makes bonds, "application" breaks bonds (of the "$\rightarrow$" kind). A bond works here like an "if-then" relation, since $\mathtt{s} \rightarrow \mathtt{t}$ specifies the conditions that have to be met to break and to release the "then" portion (in this case simply to encounter an object of type `s`).

B.2 Polymorphism

A function of type $\mathtt{s} \rightarrow \mathtt{t}$, where `s` and `t` are simple types, is *monomorphic*, because - in terms of our metaphor - it only has one shape: it recognizes only things of the shape `s`, and it returns things of the particular shape `t`. Seen this way there are infinitely many identity operations, $\lambda x.x$, with different types - such as: $\mathtt{a} \rightarrow \mathtt{a}$, $\mathtt{b} \rightarrow \mathtt{b}$, $(\mathtt{a} \rightarrow \mathtt{b}) \rightarrow (\mathtt{a} \rightarrow \mathtt{b})$, etc. - yet all do the same thing. In fact, these different types are but specific instances of a generic type $\alpha \rightarrow \alpha$, where α can be anything. Because it can be anything,we can treat it as a variable - a *type variable*. This is expressed as $\forall\alpha.\alpha \rightarrow \alpha$, also known as a *type scheme*. The introduction of type variables yields the concept of a *polymorphic type*. In contrast to a monomorphic operator, a polymorphic one can act on a variety of things with different shapes. For example, a function with type $\forall\alpha.(\alpha \rightarrow \mathtt{a}) \rightarrow (\mathtt{b} \rightarrow \alpha)$ can operate on any object which is an *instance* of the type $\forall\alpha.\alpha \rightarrow \mathtt{a}$, such as $\mathtt{c} \rightarrow \mathtt{a}$ or $\forall\beta.(\mathtt{c} \rightarrow \beta) \rightarrow \mathtt{a}$, but it cannot act on instances of $\forall\beta.\beta \rightarrow \mathtt{c}$. In the context of the chemical metaphor, polymorphism means that our abstract molecules can have different degrees of "specificity". Some could be "rigid" (monomorphic), others could be completely unspecific, and still others could cover the spectrum of specificity in between.

B.3 Type inference

Given the structure of a λ-term how do we infer its type? To begin with we assign type schemes to certain variables initially. This initial assignment, A, has the status of a boundary condition. It specifies our "chemistry". We will write the derivation of type σ for the expression P under the assumptions A as an inference: $A \vdash P : \sigma$ (read: "from A derive P of type σ"). Here's the complete set of rules for this game [55, 63], which we explain intuitively below:

Tautology $\quad x : \sigma \in A \vdash x : \sigma$

Instantiation $\quad \dfrac{A \vdash e : \sigma}{A \vdash e : \sigma'} \ (\sigma > \sigma')$

Generalization $\quad \dfrac{A \vdash e : \sigma}{A \vdash e : \forall\alpha.\sigma} \ (\alpha \text{ not free in } A)$

Application $\quad \dfrac{A \vdash e : \tau' \to \tau \qquad A \vdash e' : \tau'}{A \vdash (e)e' : \tau}$

Abstraction $\quad \dfrac{A_x \cup \{x : \tau'\} \vdash e : \tau}{A \vdash \lambda x.e : \tau' \to \tau}$

Let $\quad \dfrac{A \vdash e : \sigma \qquad A_x \cup \{x : \sigma\} \vdash e' : \tau}{A \vdash (\text{let } x = e \text{ in } e') : \tau}$

The meaning of the **Taut**-rule is simply that a free variable has the type assigned to it in the boundary condition. If there is no assignment the expression x is not typable, and is barred from the universe.

The meaning of rule **App** is also clear. It is useful, however, to know how the rule is implemented, since it introduces an important concept. Suppose that we have an object e whose type has been established to be σ, and that we want to apply it to an object e' of type τ'. For this to be possible e must have a type of the form $\tau' \to \tau$ where τ stands for a generic unknown type of $(e)e'$ that needs to be determined. Hence, for the interaction $(e)e'$ to be possible e's established type σ and the required type $\tau' \to \tau$ must be made equal. This may be possible, since σ and τ' may contain type variables which can be made more specific in order to satisfy the equality. This means we must look for some type substitution T of the free variables in σ and in $\tau' \to \tau$ such that $T\sigma = T(\tau' \to \tau)$. T is called a *unifier*, and the procedure for finding T is called *unification*. It boils down to solving a set of equations. For details about how this procedure is carried out the reader is referred to any standard textbook on type theory. The point is that a

successful unification will end up with a particular τ, the desired type of $(e)e'$. If unification is not successful, then e cannot be applied to e', i.e., the interaction term $(e)e'$ does not exist.

It is clear now that a type system poses constraints on permissible λ-terms. For example, $\lambda x.(x)x$ is not any longer an element of the universe of objects, for it has no type. To type the subterm $(x)x$ we would have to first assume the generic type α for x, and then use rule **App** which requests that x be of type $\alpha \to \alpha$. But the equation $\alpha = \alpha \to \alpha$ is recursive and has no solution (in this type system).

Recall that we model a chemical reaction by the application of a function: $(\lambda x.E)F$. In **Let** the argument, F, is typed first, and then its type is assigned to the variable x when proceeding in the type synthesis of E. The **Let** rule allows for more general interactions than are otherwise permitted by **App**. We use **Let** to model a reaction.

Finally, the **Abs**-rule is used like this. If the variable x has a type assigned in the boundary condition A, τ' say, then we must use τ' in the derivation of the type for the function body e. If e is determined to have type τ, then the whole expression is $\tau' \to \tau$. On the other hand, if x has no assignment, then we are free to temporarily assume one. We assume a generic α, and proceed to derive the type for e. During this process the assumed type α may need to be specialized into $\tau'(< \alpha)$ to meet type constraints (viz unification). The resulting type for the overall expression is $\tau' \to \tau$, and the boundary condition A is left unchanged.

The other two rules, **Gen** and **Inst**, are used to generalize and to instantiate (specialize) a type in a particular way. Their explanation is not crucial at this level of discussion, and we skip it.

C Logic background

C.1 The Curry-Howard isomorphism

The Curry-Howard isomorphism [45] provides a rigorous link between the computational sciences and logic.

Recall the two rules for typing abstraction and application in λ-calculus (Appendix B):

$$\textbf{Abs} \quad \frac{A \cup \{x : \tau'\} \vdash e : \tau}{A \vdash \lambda x.e : \tau' \to \tau}$$

$$\textbf{App} \quad \frac{A \vdash e : \tau' \to \tau \qquad A \vdash e' : \tau'}{A \vdash (e)e' : \tau}$$

The notation is understood as a rule which links the two hypotheses (above the horizontal line) with a conclusion (below the line).

Take for instance the **App**lication rule and consider what remains when everything but the type information is erased:

$$\frac{\tau' \to \tau \qquad \tau'}{\tau}$$

Now read τ' and τ as *logical propositions*, and interpret the function arrow " $\to$" to mean logical *implication*. Then, *if* we know that τ' `implies` τ, *and if* we know that τ' actually holds, *then* we can conclude that τ holds. This logical inference is known as *modus ponens*. For example, empiricists routinely use this inference, reasoning that if an event τ is known (say, by prior experiment) to be contingent upon an event τ', and τ' is an empirical observation in a current experiment, then we observe τ in the current experiment.

In logic, "to know that a proposition holds" means to *prove* it, i.e. to stepwise assemble the proposition with the help of a "scaffold" (the proof). Made of special building blocks (rules of inference), a proof is a syntactical object just like a λ-term. Let us symbolize the text documenting the proof of a hypothesis with vertical dots (meaning, "insert the formal steps of proof here"):

$$\frac{\begin{array}{cc}\vdots & \vdots \\ \text{a proof} & \text{another proof} \\ \vdots & \vdots \\ \tau' \to \tau & \tau'\end{array}}{\tau}$$

Now, this could be seen as a proof of the proposition τ by combining a proof of $\tau' \to \tau$ and one of τ'. Indeed, *modus ponens* is a step in the construction of proofs. We could use the following scheme to name the steps in the proof:

$$\left.\begin{array}{c}\dfrac{\begin{array}{cc}\vdots & \vdots \\ e & e' \\ \vdots & \vdots \\ \tau' \to \tau & \tau'\end{array}}{\tau}\end{array}\right\} (e)e' \tag{10}$$

This, however, is precisely the rule for *typing* an "application" (here, $(e)e'$) in λ-calculus (**App**). From this point of view the rule for typing an **Abs**traction is interpreted as:

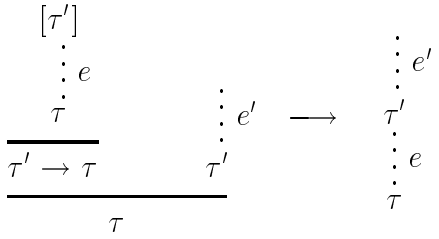
$$\left.\begin{array}{c}\dfrac{\begin{array}{c}[\tau'] \\ \vdots \\ e \\ \vdots \\ \tau\end{array}}{\tau' \to \tau}\end{array}\right\} \lambda x.e \tag{11}$$

This is meant to illustrate that a proof of τ, using the assumption τ', is a proof of $\tau' \to \tau$ where τ' has been removed from the list of assumptions. One says that τ' has been *discharged*[10]. In typed λ-calculus a free (unbound) x is, therefore, seen to stand for an assumption made (of type τ'). "Abstraction" - i.e., the binding of x as a variable - discharges that assumption, yielding a logical implication. Indeed, the object $\lambda x.e$ is a function. The function takes a (proof of the) proposition τ' and returns a (proof of the) proposition τ; hence it proves $\tau' \to \tau$.

The two rules (10) and (11) together define what "$\to$" means by stating how to eliminate and how to introduce it, respectively, from a logical formula. The implication connective is *introduced* by shuffling an assumption from the proof (the meta-language) into the logical formula (the object-language) which now keeps track of it[11]. The implication connective is *eliminated* by supplying a proof for the assumption expressed in the implication. Analogous rules of introduction and elimination exist for disjunction ($\vee$), conjunction ($\wedge$), and for the existential ($\exists$) and universal ($\forall$) quantifiers in the predicate case. This style of proof-presentation is called "natural deduction". We have introduced it here not for its direct utility in the chemical metaphor, but because it provides the simplest and most gentle connection between a logic and the typing of λ-terms.

Proof-normalization

How is the λ-calculus reduction process reflected in proof-theory? Reduction in λ-calculus is triggered by the application of a λ-expression to another: $(\lambda x.e)e'$ which becomes $e[x := e']$. The expression $(\lambda x.e)e'$ corresponds to a proof where the introduction of an implication (abstraction) is immediately followed by its elimination (application). Consider the case sketched below.



Here a proof is "normalized" by replacing copies of the derivation ending in τ' (i.e., right branch) for every discharged assumption τ' in the derivation on the

[10]This fact, however, must be recorded, for example by wrapping τ' into square brackets. This introduces a "non-local" action, i.e., an annotation at a location in the proof tree that is removed from the horizontal bar in (11) where things are currently happening. This will be avoided in another syntactical system introduced in Appendix C.2.

[11]Stated differently, if we can prove β from assumption α (i.e. $\alpha \vdash \beta$), then we can prove $\alpha \to \beta$ from no assumption (i.e. $\vdash \alpha \to \beta$). This is the "deduction property" of logical consequence ($\vdash$).

left branch (i.e., top-most segment of left branch).

C.2 Sequent calculus

Whenever a theory has "objects" as its subject, notation becomes of paramount importance. The reason is that to a good extent the theory *is* the notation. Major perspectives on logic are, therefore, characterized by differring notational systems.

One of them is Gentzen's sequent calculus, introduced here and elsewhere [32, 19], as a natural bridge between the natural deduction systems/types (discussed above) and linear logic (a discussion of which follows). The judgements derived in sequent calculus are not individual formulae like in the previous case, but rather ensembles of formulae. These judgements are called "sequents", and are of the form $\Gamma \vdash \Delta$, where the turnstile indicates "logical consequence" and Δ and Γ are *multisets* of formulae, i.e. sets where some formula may occur more than once. The connection between sequent calculus and natural deduction is direct in the case where the right side of the turnstile contains a single formula: the proof of the judgement $\Gamma \vdash \psi$ corresponds to the deduction of ψ under the hypotheses Γ. The sequent notation is a device to keep track of all assumptions made and all formulae derived up to any point in the proof tree (collecting assumptions on the left and conclusions on the right). In contrast to "natural deduction" (Appendix C.1, footnote 10), this makes the proof tree construction entirely local; what can be done at any stage depends solely on the end point of the tree.

The rules of the calculus - which we do not explain in detail here - taken together define the exact meaning of a sequent. The sequent

$$\phi_1, \ldots, \phi_n \vdash \psi_1, \ldots, \psi_m$$

means that the *conjunction* of the "antecedents" (ϕ_1 and ϕ_2 and ... and ϕ_n) *implies* the *disjunction* of the "succedents" (ψ_1 or ψ_2 or ... or ψ_m), i.e. $\phi_1 \wedge \ldots \wedge \phi_n \rightarrow \psi_1 \vee \ldots \vee \psi_m$ Stated in terms of a Boolean valuation $\mathcal{B}$ the sequent (12) says that "if all ϕ_i are true (under $\mathcal{B}$), then at least one ψ_i is true (under $\mathcal{B}$)".

Sequent calculus makes the symmetries and the algebraic properties of the logical connectives visible. In sequent calculus, like in natural deduction, proofs of judgements are built inductively by linking together other judgments through specific rules all the way up to assumptions or axioms. To provide the reader with the flavor of the system, we show the rules for implication ($\rightarrow$). Capital letters denote multisets of formulae, lower case letters denote individual formulae.

$$\text{left}\ \frac{\Gamma \vdash \phi, \Delta \qquad \Gamma', \psi \vdash \Delta'}{\Gamma, \Gamma', \phi \rightarrow \psi \vdash \Delta, \Delta'} \qquad\qquad \text{right}\ \frac{\Gamma, \phi \vdash \psi, \Delta}{\Gamma \vdash \phi \rightarrow \psi, \Delta} \tag{12}$$

For the purpose of an intuitive explanation, let us suppose that each judgement contains only one succedent, i.e., $\Delta = \emptyset$ and $\Delta' = \delta$. The left rule then means: (i) we know that under the stated conditions ϕ holds (left branch above the horizontal bar), and (ii) we know that under the stated conditions the assumption of ψ gives us δ (right branch above the bar). Clearly, if we can show that ϕ (what we have) implies ψ (what we lack), then δ would follow (under the stated conditions). This is tantamount to saying that we can derive δ from *assuming* the formula $\phi \rightarrow \psi$ in that context, and that is what appears below the bar.

Conversely, the right rule says that if - in a given context Γ - we can derive ψ by assuming ϕ, then we can derive from the context Γ alone that $\phi \rightarrow \psi$. This transfers the assumption from the meta-language of the proof to the object-language of the logical formula.

In sequent calculus logical connectives are introduced on the right and the left side of a judgement corresponding to introduction and elimination rules, respectively, in natural deduction (see Appendix C.1 and the rules for implication (12)). Similar symmetric schemes hold for the other logical connectives. Sequent calculus needs no axioms beyond the rules of proof, since it allows the use of arbitrary identities at any time:

$$\frac{}{\phi \vdash \phi}$$

An important feature of the calculus is the existence of three "structural" rules to manipulate proofs. They do not introduce logical connectives on either side:

$$\frac{\Gamma \vdash \Delta}{\Gamma, \phi \vdash \Delta} \qquad \frac{\Gamma \vdash \Delta}{\Gamma \vdash \phi, \Delta} \qquad \text{weakening}$$

$$\frac{\Gamma, \phi, \phi \vdash \Delta}{\Gamma, \phi \vdash \Delta} \qquad \frac{\Gamma \vdash \phi, \phi, \Delta}{\Gamma \vdash \phi, \Delta} \qquad \text{contraction}$$

The weakening rule "weakens" a proof by introducing antecedents or succedents that are unnecessary. If the weakening of the succedent strikes you as peculiar, remember that the succedent of a judgement is the disjunction of its formulae. *In λ-calculus weakening corresponds to the declaration of a variable which never occurs in the body of the function*, e.g., $\lambda x.\lambda y.y$. Operationally it means "discarding an input", since the argument supplied for x evaporates. The contraction rule means that one can use as many copies of a formula as one wishes. In other words: there is no resource accounting in classical logic. *In λ-calculus this corresponds to "nonlinearity", i.e., to the multiple occurrences of the same variable within the body of a function*, e.g., $\lambda x.(x)x$.

The third structural rule is the so-called **cut**-rule:

$$\frac{\Gamma \vdash \phi, \Delta \qquad \Gamma', \phi \vdash \Delta'}{\Gamma, \Gamma' \vdash \Delta, \Delta'} \qquad \textbf{cut} \tag{13}$$

The cut rule achieves a result in two steps: (i) by using a particular assumption ϕ (right branch) and by (ii) proving that assumption (left branch). This corresponds to a proof which uses "lemma" ϕ. Notice that in the proven sequent (below the bar) ϕ has been annihilated.

Cut and modus ponens

The cut rule is just another way of stating *modus ponens* (m.p.) of natural deduction. The sequent version of *modus ponens* is:

$$\frac{\Gamma \vdash \phi \to \psi \qquad \Gamma \vdash \phi}{\Gamma \vdash \psi}$$

In

$$\dfrac{\dfrac{\boxed{\Gamma \vdash \phi \to \psi}}{\Gamma, \Delta \vdash \phi \to \psi} \qquad \dfrac{\boxed{\Delta \vdash \phi}}{\Delta, \Gamma \vdash \phi}}{\boxed{\Delta, \Gamma \vdash \psi}}\ \text{m.p.}$$

we have used weakening and m.p. to derive a generalized m.p. (boxed sequents) with unequal contexts in the assumptions. From this generalized m.p. one derives cut:

$$\dfrac{\boxed{\Gamma \vdash \phi} \qquad \dfrac{\boxed{\phi, \Delta \vdash \psi}}{\Delta \vdash \phi \to \psi}\ \text{right} \to}{\boxed{\Delta, \Gamma \vdash \psi}}\ \text{m.p.}$$

Cut-elimination

The cut-rule deserves a special place in the order of things, and we explain why this is so at some length. The cut-rule (9) enables the combination of two proofs into a single proof, provided they can - metaphorically speaking - "trade" on a formula ϕ. The necessity to "trade" arises if one proof needs $\phi^{\perp}$ (it books ϕ as an *assumption*), while the other provides ϕ (it books ϕ as a *conclusion*).

The key point about sequent calculus is the famous Hauptsatz of Gentzen, which says that cut is not needed, meaning that the sequent calculus with cut can prove as much as the one without it. The main message comes *from how this is*

achieved. The theorem is proven by exhibiting a procedure through which the cut-rule can be eliminated from a proof without affecting the overall conclusion. The crucial step consists of replacing the occurrence of a cut by one or more cuts on formulae with smaller complexity. In this way the cut(s) bubble toward the leaves of the original proof-structure until they encounter an identity and disappear, i.e., cutting $\Gamma \vdash \psi, \Delta$ with $\psi \vdash \psi$ leaves $\Gamma \vdash \psi, \Delta$.

Cut-elimination

As an example consider the following cut on an implication introduced by the left and right rules (12):

$$\dfrac{\dfrac{\vdots}{\Gamma, \phi \vdash \Delta, \psi}}{\Gamma \vdash \Delta, \phi \to \psi} \qquad \dfrac{\dfrac{\vdots}{\Pi \vdash \Omega, \phi} \qquad \dfrac{\vdots}{\psi, \Xi \vdash \Lambda}}{\Pi, \Xi, \phi \to \psi \vdash \Omega, \Lambda}}{\Pi, \Xi, \Gamma \vdash \Delta, \Omega, \Lambda} \text{ cut on } \phi \to \psi$$

$$\vdots$$

In the process of cut-elimination this proof-segment is replaced by:

$$\dfrac{\dfrac{\dfrac{\vdots}{\Pi \vdash \Omega, \phi} \qquad \dfrac{\vdots}{\Gamma, \phi \vdash \Delta, \psi}}{\Pi, \Gamma \vdash \Omega, \Delta, \psi} \text{ cut on } \phi \qquad \dfrac{\vdots}{\psi, \Xi \vdash \Lambda}}{\Pi, \Xi, \Gamma \vdash \Delta, \Omega, \Lambda} \text{ cut on } \psi$$

$$\vdots$$

The two cuts which replace the previous one occur on less complex formulae, and since formulae are finite, this process will bottom out when a cut is finally made on an identity at the leaf of the proof tree. Similar procedures can be carried out for all connectives, independently of whether they are introduced left and right at the same level.

Cut-elimination does something analogous (but not formally identical) to reduction in λ-calculus. It replaces each occurrence of the assumption ϕ with a proof of it. Proofs that use cut are much easier to understand, since they are "modular" in the sense of using generic packages, which can be specialized "on demand" in different ways; for example, the package $\phi \vdash \Gamma$ can be specialized by means of $\Psi \vdash \phi$ to give $\Psi \vdash \Gamma$, or with $\Omega \vdash \phi$ to give $\Omega \vdash \Gamma$, etc. More specifically, a proof may first derive the theorem $(a+b)^2 = a^2 + 2ab + b^2$ and then use it via cut twice, once with $a = 5$ and once with $a = 2$. In the cut-free proof the $(a+b)^2$-theorem would be derived once specifically as $(5+b)^2 = 25 + 10 \cdot b + b^2$ and once specifically as $(2+b)^2 = 4 + 4 \cdot b + b^2$ without exploiting the fact that these are two instances of the same generic structure.

What happens is similar to the application of a function to a particular argument. In fact, for certain versions of the logic (e.g., if sequents are limited to only one formula on their right side) **cut** is exactly analogous to functional application, and the process of cut-elimination corresponds to the evaluation of the function, i.e. it represents the computation. In that case a cut-free proof basically corresponds to a normalized proof in natural deduction; it is a canonical proof [38].

C.3 Linear logic for tourists

In 1986 Jean-Yves Girard introduced *linear logic*. The system may be regarded as a theory about the control of the contraction and weakening rules (see Appendix C.2) of classical logic. In linear logic a formula ψ stands by default for a single occurrence which must be used exactly once. A formula may be, nonetheless, explicitly marked as potentially available in any number of copies ($!\psi$, read as "of course ψ"). Such a supply, however, may be accessed only by another modifier ($?\psi$, read as "why not ψ"). A naked ψ (not under the scope of ! or ? modalities) means exactly one copy of the formula ψ. In this spirit the logical connectives become descriptions of actions in which formulae are consumed.

To avoid confusion with the classical meanings, linear logic has its own notation. Linear implication is written as $\psi \multimap \phi$, and means that ψ is used up when giving rise to ϕ. Linear implication is, therefore, a *causal* relation. The symbol $\otimes$ (read: "cross") denotes a linear conjunction, for example, $\psi \otimes \psi$ indicates the cumulation of two instances of ψ obtained from disjoint resources. The resource sensitivity of linear implication does not permit, for example, $\psi \multimap (\psi \otimes \psi)$. This is in marked contrast to the classical case where $\psi \rightarrow (\psi \wedge \psi)$ is a provable formula.

To appreciate the meaning of this discipline, suppose that atomic formulae stand for real-world tokens. To use a textbook example, consider a vending machine which distributes soda cans and chocolate bars for one dollar each. We could use linear logic to characterize its behavior. Actions like `dollar` $\multimap$ `soda` or `dollar` $\multimap$ `chocolate` are possible, but not `dollar` $\multimap$ (`soda` $\otimes$ `chocolate`). However, we surely have (`dollar` $\otimes$ `dollar`) $\multimap$ (`soda` $\otimes$ `chocolate`). Note that `dollar` $\multimap$ `soda` is - like `dollar` - an action resource that can be used only once; it specifies the conversion of a *particular* `dollar` into a *particular* `soda`. To express the idea that this vending machine always converts `dollar`s into `soda`s or into `chocolate`s, we write !(`dollar` $\multimap$ `soda`) and !(`dollar` $\multimap$ `chocolate`).

The control over weakening and contraction has the consequence of requiring us to distinguish between different flavors of the classical connectives according to the way resources are being used. Classical conjunction, $\wedge$, splits into two linear connectives $\otimes$ ("cross") and & ("with"). The formal reason is shown in the detail-box below. The difference can be roughly summarized as follows. $\otimes$

acts as an accumulator of resources. In $\phi \otimes \psi$ both ϕ and ψ have been obtained from disjoint resources, then glued together into a pair which we must use as a unit. In $\phi \,\&\, \psi$ both ϕ and ψ arise from the *same* resource, and, therefore, we cannot have them both, but must choose one of them. By projecting out ϕ from $\phi \,\&\, \psi$ we lose ψ and vice versa. This difference is reflected by our vending machine which doesn't have an action $\texttt{dollar} \multimap (\texttt{soda} \otimes \texttt{chocolate})$, but does behave like $\texttt{dollar} \multimap (\texttt{soda} \,\&\, \texttt{chocolate})$. Note that choice is in the hands of the consumer, hence & is also called an "internal choice".

The splitting of classical conjunction [86]

Consider the case of classical conjunction, $\wedge$, in sequent calculus. We could use the following right $\wedge$-introduction rule:

$$\mathrm{R}_\wedge\colon\ \frac{\Gamma_0 \vdash \phi, \Delta_0 \qquad \Gamma_1 \vdash \psi, \Delta_1}{\Gamma_0, \Gamma_1 \vdash \phi \wedge \psi, \Delta_0, \Delta_1}$$

However, we could equally well use:

$$\mathrm{R}^*_\wedge\colon\ \frac{\Gamma \vdash \phi, \Delta \qquad \Gamma \vdash \psi, \Delta}{\Gamma \vdash \phi \wedge \psi, \Delta}$$

In fact, by virtue of weakening and contraction both rules are equivalent. We can derive $\mathrm{R}^*_\wedge$ from $\mathrm{R}_\wedge$:

$$\dfrac{\dfrac{\boxed{\Gamma \vdash \phi, \Delta} \qquad \dfrac{\dfrac{\boxed{\Gamma \vdash \psi, \Delta}}{\Gamma, \Gamma \vdash \psi, \Delta}\,\text{weakening}}{\Gamma, \Gamma \vdash \psi, \Delta, \Delta}\,\text{weakening}}{\Gamma, \Gamma, \Gamma \vdash \phi \wedge \psi, \Delta, \Delta, \Delta}\,\mathrm{R}_\wedge}{\boxed{\Gamma \vdash \phi \wedge \psi, \Delta}}$$

where the double bar indicates the use of multiple contractions. And we can derive $\mathrm{R}_\wedge$ from $\mathrm{R}^*_\wedge$:

$$\dfrac{\dfrac{\dfrac{\boxed{\Gamma_0 \vdash \phi, \Delta_0}}{\Gamma_0, \Gamma_1 \vdash \phi, \Delta_0}}{\Gamma_0, \Gamma_1 \vdash \phi, \Delta_0, \Delta_1} \qquad \dfrac{\dfrac{\boxed{\Gamma_1 \vdash \psi, \Delta_1}}{\Gamma_0, \Gamma_1 \vdash \psi, \Delta_1}}{\Gamma_0, \Gamma_1 \vdash \psi, \Delta_0, \Delta_1}}{\Gamma_0, \Gamma_1, \Gamma_0, \Gamma_1 \vdash \phi \wedge \psi, \Delta_0, \Delta_1, \Delta_0, \Delta_1}\,\mathrm{R}^*_\wedge}{\boxed{\Gamma_0, \Gamma_1 \vdash \phi \wedge \psi, \Delta_0, \Delta_1}}$$

Similarly, the classical left $\wedge$-introduction rule can be written as:

$$\mathrm{L}_\wedge\colon\ \frac{\Gamma, \phi, \psi \vdash \Delta}{\Gamma, \phi \wedge \psi \vdash \Delta}$$

But equally well one could use:

$$\mathrm{L}^{1*}_{\wedge}\colon\ \frac{\Gamma, \phi \vdash \Delta}{\Gamma, \phi \wedge \psi \vdash \Delta} \qquad\qquad \mathrm{L}^{2*}_{\wedge}\colon\ \frac{\Gamma, \psi \vdash \Delta}{\Gamma, \phi \wedge \psi \vdash \Delta}$$

The classical equivalence of $\mathrm{L}^{\{1,2\}*}_{\wedge}$ with $\mathrm{L}_{\wedge}$ is again easily established. From $\mathrm{L}^{\{1,2\}*}_{\wedge}$ to $\mathrm{L}_{\wedge}$ by means of contraction:

$$\dfrac{\dfrac{\dfrac{\boxed{\Gamma, \phi, \psi \vdash \Delta}}{\Gamma, \phi \wedge \psi, \psi \vdash \Delta}}{\Gamma, \phi \wedge \psi, \phi \wedge \psi \vdash \Delta}}{\boxed{\Gamma, \phi \wedge \psi \vdash \Delta}}$$

We skip the other direction, from $\mathrm{L}_{\wedge}$ to $\mathrm{L}^{\{1,2\}*}_{\wedge}$, where weakening is used.

When contraction and weakening are absent, as in linear logic, the two sets of left/right introduction rules are no longer equivalent. Consequently, the connectives they introduce must be distinguished. Let us denote by $\otimes$ ("cross") the conjunctive connective obtained by the previously unstarred L/R pair:

$$\mathrm{R}_{\otimes}\colon\ \frac{\Gamma_0 \vdash \phi, \Delta_0 \qquad \Gamma_1 \vdash \psi, \Delta_1}{\Gamma_0, \Gamma_1 \vdash \phi \otimes \psi, \Delta_0, \Delta_1} \qquad\qquad \mathrm{L}_{\otimes}\colon\ \frac{\Gamma, \phi, \psi \vdash \Delta}{\Gamma, \phi \otimes \psi \vdash \Delta} \tag{14}$$

As can be seen from the rules, $\otimes$ is a "context-free" or "multiplicative" version of conjunction, in the sense that there is no restraint on the contexts for the $\mathrm{R}_{\otimes}$-rule (Γ_0, Γ_1). With the connective $\otimes$, the side formulae of each premises are accumulated in the conclusion. It is in this sense that that $\otimes$ acts as an accumulator of resources.

The other conjunctive connective, & ("with"), is defined by the previously starred $\mathrm{L}^*/\mathrm{R}^*$ pair:

$$\mathrm{R}_{\&}\colon\ \frac{\Gamma \vdash \phi, \Delta \qquad \Gamma \vdash \psi, \Delta}{\Gamma \vdash \phi \,\&\, \psi, \Delta} \qquad \mathrm{L}^{1}_{\&}\colon\ \frac{\Gamma, \phi \vdash \Delta}{\Gamma, \phi \,\&\, \psi \vdash \Delta} \qquad \mathrm{L}^{2}_{\&}\colon\ \frac{\Gamma, \psi \vdash \Delta}{\Gamma, \phi \,\&\, \psi \vdash \Delta}$$

In contrast to $\otimes$, the contexts must be the *same* for the R-rule to be applicable here. This makes & "contextual" or "additive" in the sense of a superposition. Said differently: the side formulae in each premise *coincide* with the side formulae of the conclusion, and, hence, & is not accumulative.

It is worth pointing out that, in the absence of contraction and weakening, the split between the introduction rules must occur in the way just shown. There cannot be a conjunction introduced, for example, by the L^*/R pair of rules. Cut-elimination would fail otherwise (see [86]). (In the classical case - where contraction and weakening ensure equivalence between $\otimes$ and & - it is, however, customary to use the L^*/R pair of rules to introduce $\wedge$.)

Similar arguments hold for classical disjunction, $\vee$, which in the absence of contraction and weakening splits into two linear connectives: $\parr$ ("par") and $\oplus$ ("either"). Again, with respect to resources the former is "cumulative" and the latter

expresses "superposition". Like "with", $\oplus$ is a choice, but in contrast to "with" the choice is external to the action. For example, our vending machine may be defective at times and swallow your `dollar` returning `nothing`. This choice is not under the control of the customer, hence: $\texttt{dollar} \multimap ((\texttt{soda} \,\&\, \texttt{chocolate}) \oplus \texttt{nothing})$.

The connective $\parr$ in $\phi \parr \psi$ expresses a mutual dependency of ϕ and ψ, which can be stated through linear implication. $\phi \parr \psi$ is equivalent to both $\phi^\perp \multimap \psi$ or $\psi^\perp \multimap \phi$. The symbol $\perp$ denotes "linear negation", and is *defined* by formal fiat. First, one postulates (like in classical logic) equivalence between $\phi^{\perp\perp}$ and ϕ, i.e.

$$\phi^{\perp\perp} \stackrel{\text{def}}{=\!=} \phi$$

This property is also called "involutivity". Second, negation expresses *dualities* between the linear connectives, much like the deMorgan laws in classical logic (e.g., $\phi \vee \psi = \neg(\neg\phi \wedge \neg\psi)$, where $\neg$ is classical negation):

$$\begin{aligned}
(\phi \otimes \psi)^\perp &\stackrel{\text{def}}{=\!=} \phi^\perp \parr \psi^\perp \\
(\phi \parr \psi)^\perp &\stackrel{\text{def}}{=\!=} \phi^\perp \otimes \psi^\perp \\
(\phi \oplus \psi)^\perp &\stackrel{\text{def}}{=\!=} \phi^\perp \,\&\, \psi^\perp \\
(\phi \,\&\, \psi)^\perp &\stackrel{\text{def}}{=\!=} \phi^\perp \oplus \psi^\perp
\end{aligned}$$

The involutivity of negation allows to pass from two-sided sequents, $\Gamma \vdash \Delta$, to equivalent one-sided sequents, $\vdash \Gamma^\perp, \Delta$, where $\Gamma^\perp$ is the linear negation of all formulae in Γ. With respect to logical consequence, $\vdash$, linear negation behaves like a matrix transposition in linear algebra.

One-sided sequents

One-sided sequents utlize the dualities expressed by $\perp$ to halve the rules needed for defining the linear connectives. For example, take the defining rules for $\otimes$ (14), and pass to the one-sided version by linearly negating what's on the left:

$$\mathrm{R}_\otimes\colon \frac{\vdash \phi, \Gamma_0^\perp, \Delta_0 \qquad \vdash \psi, \Gamma_1^\perp, \Delta_1}{\vdash \phi \otimes \psi, \Gamma_0^\perp, \Gamma_1^\perp, \Delta_0, \Delta_1} \qquad\qquad \mathrm{L}_\otimes\colon \frac{\vdash \phi^\perp, \psi^\perp, \Gamma^\perp, \Delta}{\vdash \phi^\perp \parr \psi^\perp, \Gamma^\perp, \Delta}$$

Because one-sided sequents are right-sided, nothing much changes with respect to the right rules. The left rule of $\otimes$, however, becomes the right rule for $\parr$. A similar situation occurs with the two-sided rules for $\parr$, the right one stays, and the left one turns into the right one of $\otimes$.

This has a very important consequence. The cut rule becomes *symmetric*, in the sense that there is no distinction between a premise and a conclusion within a sequent:

$$\frac{\vdash \Delta, \phi \qquad \vdash \phi^\perp, \Gamma}{\vdash \Delta, \Gamma}$$

The asymmetry between "premise" and "conclusion" is mirrored in the computational arena by the role-asymmetry between function and data, despite their syntactic indistinguishability. A function sends a premise into a conclusion and the datum supplies the premise. Ultimately this asymmetry reflects the *sequential* paradigm of a functional calculus. Its removal makes linear logic one approach to concurrency (Appendix A.3).

Cut can occur between any formula and its dual (negation). The connective $\parr$, for example, is the dual of $\otimes$, and $\phi \parr \psi$ can be cut with $\phi^\perp \otimes \psi^\perp$. The situation has an especially elegant "geometric" interpretation in Girard's proof-net concept [33] (see Appendix C.3.2) which is a sequent-calculus stripped to its syntactical bare bones.

C.3.1 The rules of the game

For the purpose of reference we conclude with a table of the connectives and the standard set of rules for the multiplicative and additive fragment of linear logic (i.e., MALL, this the fragment without ! and ?).

The linear connectives

disjunction	conjunction	resource use
$\parr$	$\otimes$	cumulative
$\oplus$	&	superposition
$\longleftarrow$ duality $\longrightarrow$		

The rules for the linear connectives

(Although we have treated sequents as (multi)sets, i.e. $\vdash \phi, \phi, \psi$ is the same as $\vdash \phi, \psi, \phi$, the permutation rule is stated as an explicit reminder that sequents are modulo permutation.)

Identity and cut

$$\frac{}{\vdash \phi, \phi^\perp}\ \textit{identity} \qquad\qquad \frac{\vdash \Gamma, \phi \qquad \vdash \phi^\perp, \Delta}{\vdash \Gamma, \Delta}\ \textit{cut}$$

Permutation

$$\frac{\vdash \Gamma}{\vdash \Gamma'}\ \Gamma' \textit{ is a permutation of } \Gamma$$

Connectives

$$\boxed{\dfrac{\vdash \Gamma, \phi \qquad \vdash \psi, \Delta}{\vdash \phi \otimes \psi, \Gamma, \Delta}\ \mathit{times}} \qquad \boxed{\dfrac{\vdash \Gamma, \phi, \psi}{\vdash \phi \parr \psi, \Gamma}\ \mathit{par}}$$

$$\dfrac{\vdash \Gamma, \phi \qquad \vdash \psi, \Gamma}{\vdash \phi \,\&\, \psi, \Gamma}\ \mathit{with} \qquad \dfrac{\vdash \Gamma, \phi}{\vdash \phi \oplus \psi, \Gamma}\ \mathit{l\text{-}plus} \qquad \dfrac{\vdash \Gamma, \psi}{\vdash \phi \oplus \psi, \Gamma}\ \mathit{r\text{-}plus}$$

The system with only the boxed connectives is known as the multiplicative fragment of linear logic (MLL). We have not formally used the exponential modalities ! and ? in this appendix or the main text. We therefore skip their proof-theoretic definition. The reader is referred to [37] for details.

C.3.2 Proof-nets

Consider a proof of the sequent $\vdash (A \otimes B^\perp) \otimes C, (A^\perp \parr B) \parr (C^\perp \otimes D), D^\perp$ using the rules listed in the previous section:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\vdash A, A^\perp \qquad \vdash B, B^\perp}{\vdash A \otimes B^\perp, A^\perp, B}\ {\scriptstyle times}}{\vdash A \otimes B^\perp, A^\perp \parr B}\ {\scriptstyle par} \qquad \vdash C, C^\perp}{\vdash (A \otimes B^\perp) \otimes C, A^\perp \parr B, C^\perp}\ {\scriptstyle times} \qquad \vdash D, D^\perp}{\vdash (A \otimes B^\perp) \otimes C, A^\perp \parr B, C^\perp \otimes D, D^\perp}\ {\scriptstyle times}}{\vdash (A \otimes B^\perp) \otimes C, (A^\perp \parr B) \parr (C^\perp \otimes D), D^\perp}\ {\scriptstyle par}$$

The atoms A, B, C, D and their negations are introduced as identities at the leaves of the proof. All other occurrences of these atoms derive from their first introduced instance. Writing the connectives as labelled wires between formulae, and connecting with a straight wire a formulae and its negation (as they always are introduced together), we can draw a picture of the above proof where only the essential information is recorded. Every formulae occurs exactly as many times as it has been introduced through identities. Figure 8 shows the proof above in this more concise notation; it is called a *proof-net* [33]. The rules for building proof-nets within MLL are fairly straightforward. A formulae and its negation are always connected by a wire. They form the simplest proofnet. A

$\otimes$ connects two disconnected proof-nets, and a $\parr$ connects two parts within the same proofnet.

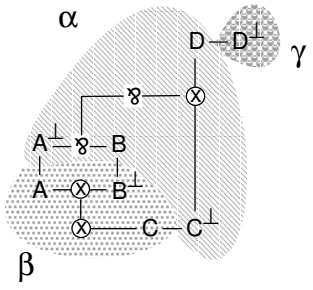


Figure 8: A proofnet.

The three shaded regions α, β, γ partition the proof-net in figure 8 exactly into the formulae of the conclusion sequent: $\beta \stackrel{\text{def}}{=\!=} (A \otimes B^{\perp}) \otimes C$, $\alpha \stackrel{\text{def}}{=\!=} (A^{\perp} \parr B) \parr (C^{\perp} \otimes D)$ and $\gamma \stackrel{\text{def}}{=\!=} D^{\perp}$. The boundaries of each region are always given by atomic links. The formula represented by the β-region can be cut with its dual $\beta^{\perp}$ $(= ((A \otimes B^{\perp}) \otimes C)^{\perp} = (A^{\perp} \parr B) \parr C)$ from another proof-net. The elimination of the cut amounts to disconnecting the cut-formulae at their atomic links and discarding the cut-formulae (but see figure 7, section 2.3.3). For an excellent introduction, see the appendix by Y. Lafont in [38].

The proof-net concept can be extended to full linear logic with the exponentials, ! and ?, as well as the additive connectives $\oplus$ and $\&$. The handling of proof-nets, however, becomes much trickier than in the simple case considered here.