A formal mathematical framework for physiological observations, experiments and analyses

Thomas A. Nielsen¹, Henrik Nilsson² and Tom Matheson^{1,*}

¹*Department of Biology, University of Leicester, University Road, Leicester LE1 7RH, UK*

²*School of Computer Science, University of Nottingham, Jubilee Campus, Nottingham NG8 1BB, UK*

Experiments can be complex and produce large volumes of heterogeneous data, which make their execution, analysis, independent replication and meta-analysis difficult. We propose a mathematical model for experimentation and analysis in physiology that addresses these problems. We show that experiments can be composed from time-dependent quantities, and be expressed as purely mathematical equations. Our structure for representing physiological observations can carry information of any type and therefore provides a precise ontology for a wide range of observations. Our framework is concise, allowing entire experiments to be defined unambiguously in a few equations. In order to demonstrate that our approach can be implemented, we show the equations that we have used to run and analyse two non-trivial experiments describing visually stimulated neuronal responses and dynamic clamp of vertebrate neurons. Our ideas could provide a theoretical basis for developing new standards of data acquisition, analysis and communication in neurophysiology.

Keywords: neuroinformatics; bio-ontologies; functional programming; insect vision, dynamic clamp

1. INTRODUCTION

Reproducibility and transparency are cornerstones of scientific methodology. As scientific experiments and analysis become increasingly complex, become reliant on computer code and produce larger volumes of data, the feasibility of independent verification and replication has—in practice—been undermined. Primary data sharing are now standard in applications of high throughput biology, but not in the many fields that produce heterogeneous observations [1,2]. Even when raw data and code used for experiments and analysis are fully disclosed, only a minority of findings can be reproduced without discrepancies [3–5]. It is often not realistic to verify that computer code used in analyses correctly implements an intended mathematical algorithm, yet errors can undermine a large body of work [6]. The combination of bias, human error and unverified software has led to the suggestion that many published research findings are flawed [7,8].

Some of these problems may be mitigated by developing explicit models of experimentation, evidence representation and analysis. A good model should simultaneously: (i) introduce a system of categorization directly relevant to the scientific field, such that scientists can define experiments and reason about observations in familiar terms, (ii) be machine executable; therefore

unambiguous and practically useful, and (iii) consist exclusively of terms that directly correspond to mathematical entities, enabling algebraic reasoning about, and manipulation of, procedures. Experiments are difficult to formalize in terms of relations between mathematical objects because they produce heterogeneous data [2], and because they interact with the physical world. In creating a mathematical framework for experiments, we take advantage of progress in embedding input and output [9,10] into programming languages where the only mechanism of computation is the evaluation of mathematical functions [11].

Here, we define a formal framework for physiology that satisfies the earlier-mentioned criteria. We show that there is a large conceptual overlap between physiological experimentation and functional reactive programming (FRP; [12,13]), a concise and purely functional formulation of time-dependent reactive computer programs. Consequently, physiological experiments can be concisely defined in the vocabulary of *signals* and *events* introduced by FRP. Such a language does not describe the physical components of biological organisms; it has no concept of networks, cells or proteins. Instead, it describes the observation and calculation of the mathematical objects that constitute physiological evidence ('observations').

Our framework provides:

— an explicitly defined ontology of physiological observations. Physiological databases have not been

*Author for correspondence (tm75@le.ac.uk).

Electronic supplementary material is available at <http://dx.doi.org/10.1098/rsif.2011.0403> or via <http://rsif.royalsocietypublishing.org>.

widely adopted [14,15] despite many candidates being available [16–19]. This contrasts with bioinformatics and neuroanatomy, where databases are routinely used [20,21]. We suggest that a flexible, concise and simple structure for physiological quantities can remedy some of the shortcomings [1,15] of existing databases and thus facilitate the sharing of physiological data and metadata [22];

- a concise language for describing complex experiments and analysis procedures in physiology using only mathematical equations. Experimental protocols can be communicated unambiguously, highlighting differences between studies and facilitating replication and meta-analysis. The provenance [23–25] of *any* observation can be extracted as a single equation that includes postacquisition processing and censoring. In addition, analysis procedures in languages with a clear mathematical denotation are verifiable as their implementation closely follows their specification [26];
- the theoretical basis for new tools that are practical, powerful and generalize to complex and multi-modal experiments. In order to demonstrate this, we have implemented our framework as a new programming language and used it for non-trivial neurophysiological experiments and data analyses. A strength of our approach is that its individual elements could, alternatively, be adopted separately or in different ways to suit different demands.

2. RESULTS

In order to introduce the calculus of physiological evidence (CoPE), we first define some terminology and basic concepts. We assume that *time* is global and is represented by a real number, as in classical physics. An *experiment* is an interaction between an observer and a number of organisms for a defined time period. An experiment consists of one or more *trials*: non-overlapping time periods during which the observer is running a *program*—instructions for manipulating the environment and for constructing mathematical objects, the *observations*. The *analyses* are further programs to be run during or after the experiment that construct other mathematical objects pertaining to the experiment. In the following sections, we give precise definitions of these concepts using terms from programming language theory and type theory, while providing an introduction to the terms for a general audience.

2.1. Type theory for physiological evidence

What kinds of mathematical objects can be used as physiological evidence? We answer this question within simple type theory [27,28], which introduces an intuitive classification of mathematical objects by assigning to every object exactly one *type*. These types include base types, such as integers $\mathbb{Z}$, text strings *String* and the Boolean type *Bool* with the two values *True* and *False*, as well as the real numbers $\mathbb{R}$ (which can be approximated in a programming language with *Float*). These base types are familiar to the users of

most programming languages. In addition, modern type systems, including simple type theory, allow types to be arbitrarily combined in several ways. For instance, if α and β are types, the type $\alpha \times \beta$ is the pair formed by one element of α and one of β ; $[\alpha]$ is a list of α s; $\alpha \rightarrow \beta$ is the type of functions that calculate a value in the type β from a value in α . The ability to write flexible type schemata and generic functions containing type variables ($\alpha, \beta, \dots$), which can later be substituted with any concrete type, is called ‘parametric polymorphism’ [27] (or ‘templates’ and ‘generics’ in the programming languages C++ and Java, respectively) and is essential to the simplicity and flexibility of CoPE.

We distinguish three type schemata in which physiological evidence can be values. These differ in the manner in which measurements appear in a temporal context, but all derive their flexibility from parametric polymorphism. *Signals* capture the notion of quantities that change in time. In physiology, observed time-varying quantities often represent scalar quantities, such as membrane voltages or muscle force, but there are also examples of non-scalar signals such as the two- or three-dimensional location of an animal or of a body part. Here, we generalize this notion such that for *any* type α , a signal of α is defined as a *function* from time to a value in α , written formally as

$$\text{Signal } \alpha = \text{Time} \rightarrow \alpha.$$

For instance, the output of a differential voltage amplifier might be captured in a *Signal Float*.

In order to model occurrences pertaining to specific instances in time, FRP defines *events* as a list of pairs of time points and values in a type α , called the ‘tags’:

$$\text{Event } \alpha = [\text{Time} \times \alpha].$$

For example, an event can be constructed from a number-valued signal that represents the time of the largest amplitude value of the signal, with that amplitude in the tag. Events that do not have a value of interest to associate with the time point at which it occurred can be tagged with the unit type $()$ that has only one element (i.e. no information). Events can therefore represent measurements where the principal information is *when* something happened, or measurements that concern *what* happened.

A third kind of information describes the properties of whole time periods. We define a *duration* of type α as a list of pairs, of which the first component is a pair denoting a start time and an end time. The last component is again a value of any type α :

$$\text{Duration } \alpha = [\text{Time} \times \text{Time} \times \alpha].$$

Durations are useful for manipulating information about a whole trial or a single annotation of an entire experiment, but could also be observations in their own right, such as open times of individual ion channels, or periods in which activity of a system exceeds a set threshold (e.g. during bursts of action potentials). Lastly, durations could be used for information that spans multiple trials—for instance, the presence or absence of a drug.

Table 1. Representation of physiological observations and quantities in the calculus of physiological evidence.

quantity	type
voltage across the cell membrane	<i>Signal Float</i>
ion concentration	<i>Signal Float</i>
animal location in two-dimensional space	<i>Signal</i> $(\text{Float} \times \text{Float})$
action potential	<i>Event ()</i>
action potential waveforms	<i>Event (Signal Float)</i>
spike detection threshold	<i>Duration Float</i>
spike interval	<i>Duration ()</i>
synaptic potential amplitude	<i>Event Float</i>
drug present	<i>Duration ()</i>
trial with parameter α	<i>Duration α</i>
visual stimulus	<i>Signal Shape</i>
laboratory notebook	<i>Event String</i>

As signals, events and durations can be instantiated for any type, they form a simple but flexible framework for representing many physiological quantities. We show a list of such examples primarily drawn from neurophysiology in table 1. A framework in any type system that does not support parametric polymorphism would have to represent these quantities fundamentally differently, thus removing the possibility of re-using common analysis procedures. Although parametric polymorphism is conceptually simple and the distinctions we are introducing are intuitive, common biomedical ontologies [29] *cannot* accommodate these definitions.

2.2. Calculating with signals and events

From direct observations, one often needs to process events and signals, create new events from signals, filter data and calculate statistics. Here, we formulate these transformations in terms of the lambda calculus [11], a family of formal languages for computation based solely on evaluating functions. These languages, unlike conventional programming languages, retain an important characteristic of mathematics: a term can freely be replaced by another term with identical meaning. This property (referential transparency; [30]) facilitates algebraic manipulation of and reasoning about programs [26]. The lambda calculus allows functions to be used as first-class entities: that is, they can be referenced by variables and passed as arguments to other functions. On the other hand, the lambda calculus disallows changing the value of variables or global states. These properties together mean that the lambda calculus combines verifiable correctness with a high level of abstraction, leading to programs that are in practice more concise [31] than those written in conventional programming languages. The lambda calculus or variants thereof has been used as a foundation for mathematics [32], classical [33] and quantum mechanics [34], evolutionary biochemistry [35], mechanized theorem provers [36,37] and functional programming languages [38].

In the lambda calculus, calculations are performed by function abstraction and application. $\lambda x \rightarrow e$ denotes the function with argument x and body e , and $f\ e$ the application of the function f to the

expression e (more conventionally written $f(e)$). For instance, the function $\text{add2} = \lambda x \rightarrow x + 2$, which can be written more conveniently as $\text{add2}\ x = x + 2$, adds two to its argument; hence, $\text{add2}\ 3 = (\lambda x \rightarrow x + 2)\ 3 = 3 + 2$ by substituting arguments in the function body.

We now present the concrete syntax of CoPE, in which we augment the lambda calculus with constructs to define and manipulate signals and events. This calculus borrows some concepts from earlier versions of FRP, but focuses exclusively on signals and events as mathematical objects and their relations. It does not have any control structures for describing sequences of system configurations, where a signal expression depends on the occurrence of events [12,13], although such constructs may be useful for simulations. As a result, CoPE is quite different from conventional FRP, which is also reflected in its implementation.

Let the construct $\{e\}$ denote a signal with the value of the expression e at every time point, and let the construct $\langle s \rangle$ denote the current value of the signal s in the temporal context created by the surrounding $\{\dots\}$ braces. For instance,

$$\{1\}$$

denotes the signal that always has the value 1; and the function *smap* defined as

$$\text{smap}\ f\ s = \{f\langle s \rangle\}$$

transforms, for any two types α and β , the signal s of α into a signal of β by applying the function f of type $\alpha \rightarrow \beta$ to the value of the signal at every time point.

The differential operator D differentiates a real-valued signal with respect to time, such that $D\ s$ denotes its first derivative and $D\ D\ s$ the second derivative of the signal s . When the differential operator appears on the left-hand side of a definition, it introduces a differential equation (see §2.5).

Events and durations can be manipulated as *lists*. Thus, a large number of transformations can be defined with simple recursive equations, including filters, folds and scans that are pivotal in functional programming languages [31]. In addition, we have added a special construct to detect events from existing signals. For instance, a threshold detector generates an occurrence of an event whenever the value of a signal crosses a specific level from below. Here, we generalize the threshold detector to an operator $??$ that takes a predicate (i.e. a function of type $\alpha \rightarrow \text{Bool}$), applies it to the instantaneous value of a signal and generates an event whenever the predicate *becomes* true. For instance,

$$(\lambda x \rightarrow x > 5)??\ s$$

denotes the event that occurs whenever the value of the signal s starts to satisfy the predicate $\lambda x \rightarrow x > 5$; that is, whenever it becomes greater than 5 after having been smaller. The expression $(\lambda x \rightarrow x > 5)??\ s$ thus defines a threshold detector restricted to threshold crossings with a positive slope.

Table S1 in the electronic supplementary material presents an informal overview of the syntax of CoPE; table S2 in the electronic supplementary material

details the types and names of some of the functions we have defined using these definitions.

2.3. Interacting with the physical world

In the previous examples, signals, events and durations exist as purely mathematical objects. In order to describe experiments, however, it must also be possible to observe particular values from real-world systems, and to create controlled stimuli to perturb these systems. For this purpose, we introduce *sources* and *sinks* that act as a bridge between purely mathematical equations and the physical world.

A source is an input port through which the value of some external quantity can be observed during the course of an experiment by binding it to a variable. If the quantity is time-varying, then the bound variable will denote a signal. For instance, binding a variable to source denoting a typical analogue-to-digital converter yields a signal of real numbers. However, a source may also refer to a time-invariant quantity.

The construct

identifier $\leftarrow^*$ *source*

binds the value or signal resulting from the observation of the *source* during the course of an experiment to the variable *identifier*. For a concrete example, the following code defines a simple experiment:

v $\leftarrow^*$ *ADC* 0 (kHz 20)

where kHz $x = 1000 \cdot x$. This describes the observation of the voltage signal on channel 0 of an analogue-to-digital converter at 20 kHz, binding the whole signal to the variable *v*. We have also used sources to sample values from probability distributions (see the electronic supplementary material).

In addition to making appropriate observations, an experiment may also involve a perturbation of the experimental preparation. For example, the manipulation could control the amount of electric current injected into a cell. Alternatively, non-numeric signals are used below to generate visual stimuli on a computer screen. Such manipulations require the opposite of a source, a *sink*: an output port connected to a physical device capable of effecting the desired perturbation. The value at the output at any point in time during an experiment is defined by connecting the corresponding sink to a signal. This is performed through the following construct, mirroring the source construct introduced earlier:

signal $\rightarrow^*$ *sink*.

As a concrete example, suppose we wish to output a sinusoidal stimulus. We first construct a time-varying signal describing the desired shape of the stimulus. In this case, we read a clock source that yields a signal counting the number of seconds since the experiment started:

seconds $\leftarrow^*$ *clock*

The sine wave can now be defined as

sineWave = { *A* · sin (*f* · (< seconds >) + *p*) }

where *A*, *f* and *p* are *Float*-valued constants specifying the amplitude, frequency and phase, respectively. We then write

sineWave $\rightarrow^*$ *DAC* 0 (kHz 20)

to send the *sineWave* signal to channel 0 of a digital-to-analogue converter at 20 kHz.

We have implemented this calculus as a new programming language and used this software to define and run two detailed experiments in neurophysiology.

2.4. Example 1

In locusts, the descending contralateral movement detector (DCMD) neuron signals the approach of looming objects to a distributed nervous system [39]. We have constructed several experiments in CoPE to record the response of DCMD to visual stimuli that simulate objects approaching with different velocities. In order to generate these stimuli, we augmented CoPE with primitive three-dimensional geometric shapes. Let the expression

cube *l*

denote a cube centred on the origin, with side length *l*,

translate (*x*, *y*, *z*) *s*

represent the shape that results from translating the shape *s* by the vector (*x*, *y*, *z*) and

colour (*r*, *g*, *b*) *s*

indicate the shape identical to *s* except with the colour intensity red *r*, green *g* and blue *b*. These primitives are sufficient for the experiments reported here. More complex stimuli can alternatively be defined in, and loaded from, external files; for instance, the source *texture* loads an image, such that

tx $\leftarrow^*$ *texture* 'image.tga'

loads the binary image in *image.tga* and creates a new function *tx*, which—when applied to a shape—returns a similar but appropriately textured shape. Thus,

tx (*cube* 1)

denotes a textured cube. Sources for loading complex polygons could be defined similarly.

As signals in CoPE are polymorphic, they can carry not just numeric values but also shapes; so we represent visual stimuli as values in *Signal Shape*. The looming stimulus consists of a cube of side length *l* approaching a locust with constant velocity *v*. The time-varying distance from the locust to the cube in real-world coordinates is a real-valued signal:

distance = { *v* · (< seconds >) − 5 } }

The *distance* signal is the basis of shape-valued signal *loomingSquare* representing the approaching

Figure 1. Diagram of an experiment to record the looming response from a locust descending contralateral movement detector (DCMD) neuron, showing the first five recorded trials from one animal. Experiment design: blue lines, simulated object size-to-approach speed ratio ($l/|v|$) for given approach trial, red lines, simulated object distance, red triangles, apparent collision time. Observed signal: black lines, recorded extracellular voltage. The largest amplitude deflections are DCMD spikes. Analysis: green dots, DCMD spikes, with randomly jittered vertical placement for display, thin black line, spike rate histogram with 50 ms bin size. The inter-trial interval of 4 min is not shown.

square:

```
loomingSquare =
  { colour (0, 0, 0)
    (translate (0, 0, (< distance >))
      (cube l)) }
```

loomingSquare differs from conventional protocols [40] for stimulating DCMD in that it describes an object that passes through the physical screen and the observer, and when displayed would thus disappear from the screen just before collision. In order not to evoke a large OFF response [41] at this point, the object is frozen in space as it reaches the plane of the surface onto which the animation is projected [42]. In order to achieve this effect, we define a new signal that has a lower bound of the distance from the eye to the visual display screen z_{screen}

$$distance' = \{ \max z_{screen} \langle : distance : \rangle \},$$

where $\max x y$ returns the larger of the two numbers x and y . *loomingSquare'* is identical to *loomingSquare* except for the use of *distance'*.

Finally, *loomingSquare'* is connected to a screen signal sink that represents a visual display unit capable of projecting three-dimensional shapes onto a two-

dimensional surface

```
loomingSquare' *> screen.
```

In our experiments, the extracellular voltage from the locust nerve (connective), in which the DCMD forms the largest amplitude spike, was amplified, filtered (see methods and listing 1 in the electronic supplementary material) and digitized:

$$voltage \leq ADC \ 0 \ (\text{kHz } 20)$$

loomingSquare' and *voltage* thus define a single object approach and the recording of the elicited response. This approach was repeated every 4 min, with different values of $1/|v|$. Figure 1 shows $1/|v|$ as values with type *Duration Float*, together with the *distance'* and *voltage* signals for the first five trials of one experiment on a common time scale.

The simplest method for detecting spikes from a raw voltage trace is to search for threshold crossings, which works well in practice for calculating DCMD activity from recordings of the locust connectives [40]. (We have also implemented in CoPE a spike identification algorithm based on template matching; see the electronic supplementary material.) If the threshold voltage for spike detection is v_{th} , then the event *spike*

Figure 2. (a) Spike rate histograms for approaches with $l/|v|$ of 0.01 (small-dashed line), 0.02 (large-dashed line) and 0.04 s (solid line), with 50 ms bin size, with collision time indicated by a black triangle. (b) Scatter plot of number of counted spikes against approach $l/|v|$ for individual trials. (c) Scatter plot of the maximum rate of spiking against $l/|v|$ for individual trials. $n = 1$ animal, 272 approaches.

can be calculated with

$$spike = tag() ((\lambda v \rightarrow v > v_{th}) ?? voltage),$$

where *tag* replaces every tag in some event with a fixed value, so that spike has type *Event* (). The spike event detected by threshold crossing is displayed on the common time scale in figure 1. The top row displays the spike rate histogram

$$H_{spike} = \{ \text{length}(\text{filter}(\text{between}(\langle \text{delay seconds} : \rangle \langle \text{seconds} : \rangle \circ fst) spikes)) \}$$

for each trial. This definition exploits the list semantics of events by using the generic list-processing function *filter* that takes as arguments predicate *p* and a list *xs*, and returns the list of elements in *xs* for which the predicate holds. Here, the predicate is *fst* (which returns the first element of a pair; here, the occurrence time) composed ($\circ$) with the function *between* = $\lambda x \rightarrow \lambda y \rightarrow \lambda z \rightarrow z > x \wedge z \leq y$, which tests whether the last of three numbers lies between the first two.

Figure 3. (a) Recorded intracellular voltage following conductance injections of a unitary simulated synaptic conductance, in the presence of A-type potassium conductances of increasing magnitude (values given are for the maximal conductance g_A ; 0, solid line; 10, large-dashed line; 40, small-dashed line; 100 nS; small-dashed line). (b) Similar to (a), but with a simulated presynaptic spike train with inter-spike intervals drawn from a Poisson distribution (here a mean of 120 s⁻¹; the spike trains used to test the different levels of A-type conductance are identical). (c) The postsynaptic spike rate plotted against the rate of simulated presynaptic inputs, with g_A as in (b).

We examined how the DCMD spike response varied with changes in $l/|v|$. The average of H_{spike} for three different values of $l/|v|$ is shown in figure 2a; whereas figure 2b,c shows the total number of spikes (*length spike*) and largest value of H_{spikes} for each approach, plotted against the value of $l/|v|$ [42]. The code that describes and executes these trials is given in the electronic supplementary material, listing 1. This code includes a description, captured in CoPE variables and with appropriate temporal context, of the experimental context that is not machine-executable. This description is based on proposed standards for minimal information about electrophysiological experiments [43]. Correspondences between these standards and CoPE variables is given in electronic supplementary material, table S3.

This experiment demonstrates that the calculus of physiological evidence can adequately and concisely describe visual stimuli, spike recording and relevant analyses for activation of a locust looming detection circuit. In order to demonstrate the versatility of this framework, we next show that it can be used to

implement dynamic clamp in an *in vivo* patch-clamp recording experiment.

2.5. Example 2

Dynamic clamp experiments [44,45] permit the observation of real neuronal responses to added simulated ionic conductances; for instance, a synaptic conductance or an additional Hodgkin–Huxley type voltage-sensitive membrane conductance. A dynamic clamp experiment requires that the current injected into a cell is calculated at every time point based on the recorded membrane potential. Here, we use CoPE to investigate the effect of an A-type potassium conductance [46] on the response of a zebrafish spinal motor neuron to synaptic excitation.

The output current i is calculated at each time-step from the simulated conductance g and the measured membrane voltage V_m :

$$\begin{aligned} V_m &<* ADC0 \text{ (kHz 20)} \\ i &= \{ (\langle V_m \rangle - E) \cdot \langle g \rangle \} \\ i &*> DAC0 \text{ (kHz 20)} \end{aligned}$$

The experiment is thus characterized by the conductance signal g (for clarity, here we omit the amplifier-dependent input and output gains).

In the simplest case, g is independent of V_m —for instance, when considering linear synaptic conductances [47]. We first consider the addition of a simulated fast excitatory synaptic conductance to a real neuron. Simple models of synapses approximate the conductance waveform with an alpha function [48]:

$$\begin{aligned} \text{alpha}_f \text{ amp } \tau &= \\ \{ \text{amp} \cdot \tau^2 \cdot \langle \text{seconds} \rangle \cdot \exp(-\langle \text{seconds} \rangle \cdot \tau) \}. \end{aligned}$$

In order to simulate a barrage of synaptic input to a cell, this waveform is convolved with a simulated pre-synaptic spike train. The spike train itself is first bound from a source representing a random probability distribution—in this case, series of recurrent events of type *Event* () for which the inter-occurrence interval is Poisson distributed. Our standard library contains a function *convolveSE*, which convolves an impulse response signal with a numerically tagged event such that the impulse response is multiplied by the tag before convolution.

$$\begin{aligned} \text{preSpike} &<* \text{poissonTrain rate} \\ g_{syn} &= \text{convolveSE}(\text{alpha}_f \text{ amp } \tau)(\text{tag1 preSpike}), \end{aligned}$$

where the signal g_{syn} could be used directly in a dynamic clamp experiment using the above template (i.e. $g = g_{syn}$). Here, we will examine other conductances that modulate the response of the cell to synaptic excitation.

Both the subthreshold properties of a cell and its spiking rate can be regulated by active ionic conductances, which can also be examined with the dynamic clamp. In the Hodgkin–Huxley formalism for ion channels, the conductance depends on one or more state variables, for which the forward and backward rate

constants depend on the membrane voltage. We show the equations for the activation gate of an A-type potassium current ([46]; following [49], but using International System of Units and absolute voltages). The equations for inactivation are analogous (see listing 2 in the electronic supplementary material).

We write the forward and backward rates as functions of the membrane voltage

$$\alpha_a v = \frac{k_{\alpha a1}(v + k_{\alpha a2})}{\exp((-k_{\alpha a2} - v)/k_{\alpha a3}) - 1},$$

and

$$\beta_a v = \frac{k_{\beta a1}(v + k_{\beta a2})}{\exp((v + k_{\beta a2})/k_{\beta a3}) - 1},$$

where with $k_{\alpha a1} = \text{inverseVolts} - 2.0 \times 10^5$, $k_{\alpha a2} = \text{volts} \ 0.0469$, $k_{\alpha a3} = k_{\beta a3} = \text{volts} \ 0.01$, $k_{\beta a1} = \text{inverseVolts} \ 1.75 \times 10^5$ and $k_{\beta a2} = \text{volts} \ 0.0199$, $\text{volts} = \text{inverseVolts} = \lambda x \rightarrow x$. The time-varying state of the activation gate is given by a differential equation. We use the notation $Dx = \{f(x, \langle \text{seconds} \rangle)\}$ to denote the ordinary differential equation that is conventionally written as $dx/dt = f(x, t)$, with starting conditions explicitly assigned to the variable x_0 . The differential equation for the activation variable a is

$$\begin{aligned} Da &= \{ \alpha_a \langle V_m \rangle \cdot (1 - \langle a \rangle) \\ &\quad - \beta_a \langle V_m \rangle \cdot \langle a \rangle \} \\ a_0 &= 0. \end{aligned}$$

The inactivation state signal b is defined similarly.

The current signal from this channel is calculated from Ohm's law:

$$i_A = \{ g_A \cdot \langle a \rangle \cdot \langle b \rangle \cdot (\langle V_m \rangle - E) \}.$$

This is added to the signal i defined above to give the output current, thus completing the definition of this experiment:

$$V_m \cdot g_{syn} + i_A *> DAC0 \text{ (kHz 20)}$$

Figure 3a,b shows the voltage response to a unitary synaptic conductance and a train of synaptic inputs, respectively, with g_A ranging from 0 to 100 nS. By varying the value of *rate*, we can examine the input–output relationship of the neuron by measuring the frequency of postsynaptic spikes. Spikes were detected from the first derivative of the V_m signal with

$$\text{spike} = \text{tag}((\lambda v \rightarrow v > v_{th}) ?? DV_m),$$

and the spike frequency calculated with the *frequencyDuring* function. This relationship between the postsynaptic spike frequency and the simulated synaptic input *rate* is plotted in figure 3c for four different values of g_A . The code for example 2 is given in the electronic supplementary material, listing 2.

3. DISCUSSION

We present a new approach to performing and communicating experimental science. Our use of typed, functional and reactive programming overcomes at

least two long-standing issues in bioinformatics: the need for a flexible ontology to share heterogeneous data from physiological experiments [15] and a language for describing experiments and data provenance unambiguously [23,50].

The types we have presented form a linguistic framework and an ontology for physiology. Thanks to the flexibility of parametric polymorphism, our ontology can form the basis for the interchange of physiological data and metadata without imposing unnecessary constraints on what can be shared. The ontology is non-hierarchical and would be difficult to formulate in the various existing semantic web ontology frameworks (web ontology language, [29], or resource description framework), which lack parametric polymorphism and functional abstraction. Nevertheless, by specifying the categories of mathematical objects that constitute evidence, it is an ontology in the classical sense of cataloguing the categories within a specific domain, and providing a vocabulary for that domain. We emphasize again that it is an ontology of *evidence*, not of the biological entities that give rise to this evidence. It is unusual as an ontology for scientific knowledge in being embedded in a *computational* framework, such that it can describe not only mathematical objects but also their transformations and observations. Recent work on metadata representation [51] has focused on delineating the information needed to repeat an experiment [43,52], but in practice it is often not clear *a priori* what aspects of an experiment could influence its outcome. With CoPE, our main goal was to describe unambiguously machine-executable aspects of the metadata of an experiment. Nevertheless, any information that can be captured in a type can be represented in the temporal contexts provided by CoPE, i.e. it can exist as signals, events or durations, as we have demonstrated in the code listings in the electronic supplementary material. Here, we make no fundamental distinction between the representation of data and metadata. All relevant information about an experiment is indexed by time and thus linked by overlap on a common time scale.

Parametric polymorphism and first-class functions are generally associated with research-oriented languages such as Haskell and ML rather than mainstream programming languages such as C++ or Java. It is likely that much of CoPE could be implemented in C++, where template metaprogramming implements a static form of parametric polymorphism, and template functors can be used to represent functions. These must all be resolved at compile-time, however; so it is difficult to use dynamically calculated or arbitrarily complex functions. The importance of true first-class functions and parametric polymorphism is becoming increasingly well recognized, and these features are now being implemented in the mainstream programming languages C++, C# and Java. We therefore expect that it will soon be possible to implement the formalism we are proposing in a wide range of programming languages.

Our mathematical definitions are unambiguous and concise, unlike typical definitions written in natural language, and are more powerful than those specified by graphical user interfaces or in formal languages

that lack a facility for defining abstractions. Our framework is not only a theoretical formalism, but we show that it can also be implemented as a very practical tool. This tool consists of a collection of computer programs for executing experiments and analyses, and for carrying out data management. By using programs that share data formats, experimentation and analysis can be controlled in a highly efficient manner, and many sources of human error eliminated. Existing tools use differential equations to define dynamic clamp experiments (model reference current injection; [53]) or simulations (X-Windows phase plane; [54]). Here, we show that a general (polymorphic) definition of signals and events, embedded in the lambda calculus, can define a much larger range of experiments and the evidence that they produce. Our experiment definitions have the further advantage that they *compose*; that is, more complex experiments can be formulated by joining together elementary building blocks.

Our full approach is particularly relevant to the execution of very complex and multi-modal experiments, which may need to be dynamically reconfigured based on previous observations, or to disambiguate difficult judgements about evidence [55]. Even if used separately, however, individual aspects of CoPE can make distinct contributions to scientific methodology. For instance, our ontology for physiological evidence can be used within more conventional programming languages or web applications that facilitate data sharing. In a similar way, the capabilities of CoPE for executing and analysing experiments could provide a robust core for innovative graphical user interfaces. We expect the formalism presented here to be applicable outside neurophysiology. Purely temporal information from other fields could be represented using signals, events and durations, or other kinds of temporal context formalized in type theory. In addition, the concepts of signals, events and durations could be generalized to allow not just temporal but also spatial or spatio-temporal contexts to be associated with specific values. Such a generalization is necessary for CoPE to accommodate data from the wider neuroscience community, including functional neuroanatomy and microscopy, and other scientific disciplines that observe and manipulate spatio-temporal data.

We have argued that observational data, experimental protocols and analyses formulated in CoPE (or similar frameworks) are less ambiguous and more transparent than those described using many current formulations. The structure of CoPE presents additional opportunities for mechanically excluding some types of procedural errors in drawing inferences from experiments. For instance, CoPE could incorporate an extension to simple type theory [56] that adds not only a consistency check for dimensional units, but also powerful aspects of dimensional analysis, such as the Buckingham's π -theorem. Furthermore, in our formulation, experimentally observed values exist as mathematical objects within a computational framework that can be used to define probability distributions. Hierarchical probabilistic notation [57] permits the construction of flexible statistical models for the directly observed data. This means that CoPE could

in principle be used to turn such probabilistic models into powerful data analysis tools accessible from within the CoPE formalism. Such data analysis procedures can largely be automated, for instance by calculating parameter estimates using statistical packages such as WinBUGS [58] or Mlwin [59]. Alternatively, it would be possible to compile descriptions of probabilistic models in CoPE to the specification format used by the AutoBayes [60] system, which would then be used to generate efficient code for statistical inference. When run, this code would return data to CoPE. This analysis workflow could be integrated seamlessly into an implementation of CoPE such that both the hierarchical model structure and the returned parameters would be defined by, and tagged with, the appropriate temporal contexts. This methodology could be used to quantify different aspects of uncertainty in the measurements, taking into account all available information, while largely avoiding ad hoc transformations of data. Integrating well-developed statistical tools with data acquisition and manipulation within CoPE would create a powerful platform for validating inferences drawn from physiological experiments.

4. EXPERIMENTAL PROCEDURES

4.1. Language implementation

We have used two different implementation strategies for reasons of rapid development and execution efficiency. For experimentation and simulation, we have implemented a prototype compiler that can execute some programs that contain signals and events defined by mutual recursion, as is necessary for the experiments in this study. The program is transformed by the compiler into a normal form that is translated to an imperative program that iteratively updates variables corresponding to signal values, with a time step that is set explicitly. The program is divided into a series of stages, where each stage consists of the signals and events defined by mutual recursion, subject to the constraints of input–output sources and sinks. This ensures that signal expressions can reference values of other signals at arbitrary time points (possibly in the future) as long as referenced signals are computed in an earlier stage.

In order to calculate a new value from existing observations after data acquisition, we have implemented the calculus of physiological evidence as a domain-specific language embedded in the purely functional programming language Haskell.

For hard real-time dynamic clamp experiments, we have built a compiler back-end targeting the LXRT (user-space) interface to the real-time application interface (RTAI; <http://rtai.org>) extensions of the Linux kernel, and the Comedi (<http://comedi.org>) interface to data acquisition hardware. Geometric shapes were rendered using OpenGL (<http://opengl.org>).

All the codes used for experiments, data analysis and generating figures are available at <http://github.com/glutamate/bugpan> under the GNU general public licence (<http://www.gnu.org/licenses/gpl.html>).

4.2. Locust experiments

Locusts were maintained at 1600 m^{-3} in $50 \times 50 \times 50\text{ cm}$ cages under a standard light and temperature regime of 12 h light at 36°C : 12 h dark at 25°C . They were fed ad libitum with fresh wheat seedlings and bran flakes. Recordings from locust DCMD neurons were performed as described previously [61]. Briefly, locusts were fixed in plasticine, with their ventral side facing upwards. The head was fixed with wax and the connectives were exposed through an incision in the soft tissue of the neck. A pair of silver wire hook electrodes were placed underneath a connective, and the electrodes and connective enclosed in petroleum jelly. The electrode signal was amplified $1000\times$ and bandpass filtered between 50 and 5000 Hz, before analogue-to-digital conversion at 18 bits and 20 kHz with a National Instruments PCI-6281 board. The locust was placed in front of a 22" CRT monitor running with a vertical refresh rate of 160 Hz. All aspects of the visual stimulus displayed on this monitor and of the analogue-to-digital conversion performed by the National Instruments board were controlled by programs written in CoPE running on a single computer.

The code for running the trials described in example 1, including relevant metadata, is given in the electronic supplementary material, listing 1.

4.3. Zebrafish experiments

Zebrafish were maintained according to established procedures [62] in approved tank facilities, in compliance with the Animals (Scientific Procedures) Act 1986 and according to University of Leicester guidelines. Intracellular patch-clamp recordings from motor neurons in the spinal cord of a 2-days old zebrafish embryo were performed as described previously [63]. We used a National Instruments PCI-6281 board to record the output from a BioLogic patch-clamp amplifier in current-clamp mode, filtered at 3 kHz and digitized at 10 kHz, with the output current calculated at the same rate by programs written in CoPE targeted to the RTAI back-end (see earlier text). The measured jitter for updating the output voltage was 6s and was similar to that measured with the RTAI latency test tool for the experiment control computer.

The code for the Zebrafish experiment trials, including relevant metadata, is given in the electronic supplementary material, listing 2.

We thank Jonathan McDermid for help with the Zebrafish recordings and Angus Silver, Guy Billings, Antonia Hamilton, Nick Hartell and Rodrigo Quian Quiroga for critical comments on the manuscript. This work was funded by a Human Frontier Science Project fellowship to T.N., a Biotechnology and Biological Sciences Research Council grant to T.M. and T.N., a BBSRC Research Development Fellowship to T.M., and Engineering and Physical Sciences Research Council grants to H.N. The author contributions were as follows: T.N. designed and implemented CoPE, carried out the experiments and data analyses, and wrote the draft of the paper. H.N. contributed to the language design, helped clarify the semantics and wrote several sections of the manuscript. T.M. contributed to the design of the experiments and the data analysis, and made

extensive comments on drafts of the manuscript. All authors obtained grant funding to support this project.

REFERENCES

- Gardner, D., Abato, M., Knuth, K. H. & Robert, A. 2005 Neuroinformatics for neurophysiology: the role, design, and use of databases. In *Databasing the brain: from data to knowledge* (eds S. H. Koslow & S. Subramaniam), p. 47–68. Hoboken, NJ: Wiley.
- Tukey, J. W. 1962 The future of data analysis. *Ann. Math. Stat.* **33**, 1–67. (doi:10.1214/aoms/1177704711)
- Ioannidis, J. P. et al. 2008 Repeatability of published microarray gene expression analyses. *Nat. Genet.* **41**, 149–155. (doi:10.1038/ng.295)
- Baggerly, K. A. & Coombes, K. R. 2009 Deriving chemosensitivity from cell lines: forensic bioinformatics and reproducible research in high-throughput biology. *Ann. Appl. Stat.* **3**, 1309–1334. (doi:10.1214/09-AOAS291)
- McCullough, B. D. 2007 Got replicability? *J. Money, Credit Bank Arch. Econ. J. Watch* **4**, 326–337.
- Chang, G., Roth, C. B., Reyes, C. L., Pornillos, O., Chen, Y. & Chen, A. P. 2006 Retraction. *Science* **314**, 1875. (doi:10.1126/science.314.5807.1875b)
- Ioannidis, J. P. 2005 Why most published research findings are false. *PLoS Med.* **2**, e124. (doi:10.1371/journal.pmed.0020124)
- Merali, Z. 2010 Computational science: error. *Nature* **467**, 775–777. (doi:10.1038/467775a)
- Peyton Jones, S. 2002 Tackling the awkward squad: monadic input–output, concurrency, exceptions, and foreign-language calls in Haskell. In *Engineering theories of software construction* (eds T. Hoare, M. Broy & R. Steinbruggen), p. 47–96. Amsterdam, The Netherlands: IOS Press.
- Wadler, P. 1995 Monads for functional programming. *Lect. Notes Comput. Sci.* **925**, 24–52.
- Church, A. 1941 *The calculi of lambda-conversion*. Princeton, NJ: Princeton University Press.
- Elliott, C. & Hudak, P. 1997 Functional reactive animation. *Int. Conf. Funct. Program.* **32**, 263–273.
- Nilsson, H., Courtney, A. & Peterson, J. 2002 Functional reactive programming, continued. In *Proc. 2002 ACM SIGPLAN workshop on Haskell, Pittsburgh, PA, USA, 3 October 2002*, pp. 51–64. New York, NY: ACM Press.
- Herz, A. V., Meier, R., Nawrot, M. P., Schiegel, W. & Zito, T. 2008 G-Node: an integrated tool-sharing platform to support cellular and systems neurophysiology in the age of global neuroinformatics. *Neural Netw.* **21**, 1070–1075. (doi:10.1016/j.neunet.2008.05.011)
- Amari, S. et al. 2002 Neuroinformatics: the integration of shared databases and tools towards integrative neuroscience. *J. Integr. Neurosci.* **1**, 117–128. (doi:10.1142/S0219635202000128)
- Jessop, M., Weeks, M. & Austin, J. 2010 CARMEN: a practical approach to metadata management. *Phil. Trans. R. Soc. A* **368**, 4147–4159. (doi:10.1098/rsta.2010.0147)
- Teeters, J. L., Harris, K. D., Millman, K. J., Olshausen, B. A. & Sommer, F. T. 2008 Data sharing for computational neuroscience. *Neuroinformatics* **6**, 47–55. (doi:10.1007/s12021-008-9009-y)
- Frishkoff, G., LePendu, P., Frank, R., Liu, H. & Dou, D. 2009 Development of neural electromagnetic ontologies (NEMO): ontology-based tools for representation and integration of event-related brain potentials. *Nat. Precedings*. (doi:10.1038/npre.2009.3458.1)
- Katz, P. S. et al. 2010 NeuronBank: a tool for cataloging neuronal circuitry. *Front. Syst. Neurosci.* **4**, 9. (doi:10.3389/fnsys.2010.00009)
- Rodriguez-Tome, P. 1996 The European Bioinformatics Institute (EBI) databases. *Nucleic Acids Res.* **24**, 6–12. (doi:10.1093/nar/24.1.6)
- Ascoli, G. A., Donohue, D. E. & Halavi, M. 2007 NeuroMorpho.Org: a central resource for neuronal morphologies. *J. Neurosci.* **27**, 9247–9251. (doi:10.1523/JNEUROSCI.2055-07.2007)
- Insel, T. R., Volkow, N. D., Li, T., Battey, J. F. & Landis, S. C. 2003 Neuroscience networks: data-sharing in an information age. *PLoS Biol.* **1**, E17. (doi:10.1371/journal.pbio.0000017)
- Pool, R. 2002 *Bioinformatics: converting data to knowledge, workshop summary*. Washington, DC: National Academy Press.
- Mackenzie-Graham, A. J., Van Horn, J. D., Woods, R. P., Crawford, K. L. & Toga, A. W. 2008 Provenance in neuroimaging. *NeuroImage* **42**, 178–195. (doi:10.1016/j.neuroimage.2008.04.186)
- Van Horn, J. D. & Toga, A. W. 2009 Is it time to re-prioritize neuroimaging databases and digital repositories? *NeuroImage* **47**, 1720–1734. (doi:10.1016/j.neuroimage.2009.03.086)
- Bird, R. & Moor, O. D. 1996 *The algebra of programming*. London, UK: Prentice Hall.
- Pierce, B. C. 2002 *Types and programming languages*. Cambridge, MA: MIT Press.
- Hindley, J. R. 2008 *Basic simple type theory*. Cambridge, UK: Cambridge University Press.
- Bechhofer, S. 2007 OWL web ontology language reference. W3C Recommendation. See <http://www.w3.org/TR/owl-ref/>
- Whitehead, A. N. & Russell, B. 1927 *Principia mathematica*. Cambridge, UK: Cambridge University Press.
- Hughes, J. 1989 Why functional programming matters? *Comp. J.* **32**, 98. (doi:10.1093/comjnl/32.2.98)
- Martin-Löf, P. 1985 *Intuitionistic type theory*. Naples, Italy: Prometheus Books.
- Sussman, G. J. & Wisdom, J. 2001 *Structure and interpretation of classical mechanics*. Cambridge, MA: MIT Press.
- Karczmarszuk, J. 2003 Structure and interpretation of quantum mechanics: a functional framework. In *Proc. 2003 ACM SIGPLAN workshop on Haskell, Uppsala, Sweden, 28 August 2003*, p. 50–61. New York, NY: ACM Press.
- Fontana, W. & Buss, L. W. 1994 What would be conserved if “the tape were played twice”? *Proc. Natl Acad. Sci. USA* **91**, 757–761. (doi:10.1073/pnas.91.2.757)
- de Bruijn, N. G. 1968 The mathematical language AUTOMATH, its usage and some of its extensions. *Lect. Notes Math.* **125**, 29–61.
- Harrison, J. 2009 *Handbook of practical logic and automated reasoning*. Cambridge, UK: Cambridge University Press.
- McCarthy, J. 1960 Recursive functions of symbolic expressions and their computation by machine, Part I. *Commun. ACM* **3**, 184–195. (doi:10.1145/367177.367199)
- Rind, F. C. & Simmons, P. J. 1992 Orthopteran DCMD neuron: a reevaluation of responses to moving objects. I. Selective responses to approaching objects. *J. Neurophysiol.* **68**, 1654–1666.
- Gabbiani, F., Mo, C. & Laurent, G. 2001 Invariance of angular threshold computation in a wide-field looming-sensitive neuron. *J. Neurosci.* **21**, 314–329.

- 41 O'Shea, M. & Rowell, C. H. 1976 The neuronal basis of a sensory analyser, the acridid movement detector system. II. Response decrement, convergence, and the nature of the excitatory afferents to the fan-like dendrites of the LGMD. *J. Exp. Biol.* **65**, 289–308.
- 42 Hatsopoulos, N., Gabbiani, F. & Laurent, G. 1995 Elementary computation of object approach by wide-field visual neuron. *Science* **270**, 1000–1003. (doi:10.1126/science.270.5238.1000)
- 43 Gibson, F. *et al.* 2008 Minimum information about a neuroscience investigation (MINI) electrophysiology. *Nat. Precedings*. (doi:10.1038/npre.2008.1720.1)
- 44 Robinson, H. P. & Kawai, N. 1993 Injection of digitally synthesized synaptic conductance transients to measure the integrative properties of neurons. *J. Neurosci. Methods* **49**, 157–165. (doi:10.1016/0165-0270(93)90119-C)
- 45 Sharp, A. A., O'Neil, M. B., Abbott, L. F. & Marder, E. 1993 Dynamic clamp: computer-generated conductances in real neurons. *J. Neurophysiol.* **69**, 992–995.
- 46 Connor, J. A. & Stevens, C. F. 1971 Voltage clamp studies of a transient outward membrane current in gastropod neural somata. *J. Physiol.* **213**, 21–30.
- 47 Mitchell, S. J. & Silver, R. A. 2003 Shunting inhibition modulates neuronal gain during synaptic excitation. *Neuron* **38**, 433–445. (doi:10.1016/S0896-6273(03)00200-9)
- 48 Carnevale, N. T. & Hines, M. L. 2006 *The NEURON book*. Cambridge, UK: Cambridge University Press.
- 49 Traub, R. D., Wong, R. K., Miles, R. & Michelson, H. 1991 A model of a CA3 hippocampal pyramidal neuron incorporating voltage-clamp data on intrinsic conductances. *J. Neurophysiol.* **66**, 635–650.
- 50 Murray-Rust, P. & Rzepa, H. S. 2002 Scientific publications in XML: towards a global knowledge base. *Data Sci.* **1**, p84–98. (doi.org/10.2481/dsj.1.84)
- 51 Bower, M. R., Stead, M., Brinkmann, B. H., Dufendach, K. & Worrell, G. A. 2009 Metadata and annotations for multi-scale electrophysiological data. *Conf. Proc. IEEE Eng. Med. Biol. Soc.* **2009**, 2811–2814. (doi:10.1109/IEMBS.2009.5333570)
- 52 Taylor, C. F. *et al.* 2007 The minimum information about a proteomics experiment (MIAPE). *Nat. Biotechnol.* **25**, 887–893. (doi:10.1038/nbt1329)
- 53 Raikov, I., Preyer, A. & Butera, R. J. 2004 MRCI: a flexible real-time dynamic clamp system for electrophysiology experiments. *J. Neurosci. Methods* **132**, 109–123. (doi.org/10.1016/j.jneumeth.2003.08.002)
- 54 Ermentrout, B. 1987 *Simulating, analyzing, and animating dynamical systems: a guide to xppaut for researchers and students*. Philadelphia, PA: Society for Industrial Mathematics.
- 55 Kriegeskorte, N., Simmons, W. K., Bellgowan, P. S. & Baker, C. I. 2009 Circular analysis in systems neuroscience: the dangers of double dipping. *Nat. Neurosci.* **12**, 535–540. (doi:10.1038/nn.2303)
- 56 Kennedy, A. J. 1997 Relational parametricity and units of measure. *Proc. Symp. Princ. Program. Lang.* **25**, 442–455.
- 57 Gelman, A. & Hill, J. 2006 *Data analysis using regression and multilevel/hierarchical models*. Cambridge, UK: Cambridge University Press.
- 58 Gilks, W. R., Thomas, A. & Spiegelhalter, D. J. 1994 A language and program for complex Bayesian modelling. *The Statistician* **43**, 169. (doi:10.2307/2348941)
- 59 Goldstein, H., Browne, W. & Rasbash, J. 2002 Multilevel modelling of medical data. *Stat. Med.* **21**, 3291–315. (doi:10.1002/sim.1264)
- 60 Fischer, B. & Schumann, J. 2003 AutoBayes: a system for generating data analysis programs from statistical models. *J. Funct. Program.* **13**, 483–508. (doi:10.1017/S0956796802004562)
- 61 Matheson, T., Rogers, S. M. & Krapp, H. G. 2004 Plasticity in the visual system is correlated with a change in lifestyle of solitary and gregarious locusts. *J. Neurophysiol.* **91**, 1–12. (doi:10.1152/jn.00795.2003)
- 62 Westerfield, M. 1994 *The zebrafish book: a guide for the laboratory use of zebrafish (Danio Rerio)*. Eugene, OR: University of Oregon Press.
- 63 McDearmid, J. R., Liao, M. & Drapeau, P. 2006 Glycine receptors regulate interneuron differentiation during spinal network development. *Proc. Natl Acad. Sci. USA* **103**, 9679–9684. (doi:10.1073/pnas.0504871103)