Warning: easy to read in **over ~25min** or twice (tested), but text **may fry a brain in a hurry.**

Welcome to a **whirlwind** tour of modeling, encoding jams, and how compilers can assist in detecting surprise extinctions!

Compiling a Model to Forecast Fox and Rabbit Ecology with Evolvix

Laurence Loewe & **Seth Keel**
Wisconsin Institute for Discovery
University of Wisconsin-Madison

Problem: Forecasting requires models. They govern complex algorithms to simulate relevant time series. But manually mixing code for models and algorithms often chokes new modeling ideas in setup complexity. It can feel as if a new trip requires building a new car. **Solution**: A friendly compiler can simplify encoding, like Evolvix (Fig.1).

Fig.1: This modeling panorama shows **how Evolvix can accelerate and deepen biological thought** by moving busywork in modeling to a compiler. Experimental biologists often build mechanistic models in their minds. Evolvix assists by obtaining interpretable time series forecasts from a user-friendly syntax that is re-encoded (compiled) into two mathematical modeling formalisms shown below. Here a *hypothetical ecology* illustrates 8 steps in a modeling cycle. The textbook [1] result in step 7d (via 5d,6d) is extended in 7s (via 5s,6s) by recompiling to a formalism that treats individuals as indivisible, thus allowing extinctions to occur. Exploring different computational assumptions can be mission critical in biology.

Imagine, as if driving a car, biologists can rely on **a friendly compiler for translating** ideas about how a system works into simulations that predict observable time series. Imagine, such a compiler turns **readable statements** about a system's known biology, open questions, and remaining wiggle room **into** the **mathematics** required **for simulating** models. Such a compiler is rocket fuel for biology. It can explore many alternative models **to reduce guesswork in** planning **experiments** by calculating how much a potential measurement may contribute to testing some hypotheses of interest.

Biologists like modeling – if only encoding were easier! 100,000s of informal models published each year **can get a quantitative edge**, once modeling tools become easier to use and deal better with biouncertainty. So now, about **~99% of models built in biology are never simulated.** Example: SBML.org, one leading approach since 2003, lists 290 tools, has ~1650 published models in its BioModelsDB (rough avg.~5.7 models/tool/15yr or ~120/yr). Too many tools?

Why are models in biology encoded by hand in complicated ways if compilers can simplify encoding?

Evolvix aims to simplify accurate modeling of systems described in a stable extensible user-friendly language.

1 A rare Fox breed is to be saved

A rare Fox breed is to be saved on big grassy islands with Rabbit as food. Yet the Fox keeps going extinct. The big question is: **Why?** Can the model below predict Fox survival from initial counts? How is a 40-Fox-and-500-Rabbit-island best simulated?

Reality is inspiring and too much to fully grasp. So, we must model to cope.

2 Thinking on paper is key for modeling
Intuitive, quick, informal catching of ideas is key to focus draft models on essentials. Example:

Modeling is Art (focus, simplify, notation design) & Science (all else)
Not shown: compilers can simplify handling uncertain parameter values

Let's start with a simple predator-prey model [1] 2 with 40 foxes & 500 rabbits, interacting as follows:

1	Rabbit Breeds:	Rate: 0.5	per Month		
	1 Rabbit	⟶	2	Rabbit	
2	Fox Breeds:	Rate: 0.0002	per Month ...		
	1 Fox + 1 Rabbit	⟶	2	Fox	
3	Fox Dies:	Rate: 0.1	per Month		
	1 Fox	⟶	0	Fox	
4	Fox Feeds:	Rate: 0.0098	per Month ...		
	1 Fox + 1 Rabbit	⟶	1	Fox	

Now, how do we get time series of Fox counts?

Must encode to simulate! ... but **How?** 4

to select the best encodings is hard — to learn to encode can be hard — Solution: a user-friendly language to delegate most work to a compiler.

3 Model building is simplifying! Focus:
What Big Question do models need to answer?
Who is playing a relevant Part in the system?
List each key Part and all spaces it may exist in.
When is any Action changing any Part or its spaces?
List each Action, give its Rate / [TimeUnit·Opportunity]
Where and how to simplify more *or* less can be an art, but once decided, encoding could be as easy as above (4e).

4 Encoding is checking by hand if models can be described by following the rules of a chosen language. But user-friendly languages are rare and their formal rules difficult to design.

5 Compiling is saving time by automatically (re)encoding models using formal rules and checking for known errors.

Biology wins big time once imperfection and uncertainty tracking can be offloaded to a compiler for biodata.

Thanks to all Evolvix.org/thinkers

Hard — Encode by hand, but first **learn complicated rules** in **applied math** *or* **computational biology!** to 7d, to 7s

Hard — Encode by hand for stochastic individual-based Continuous-Time Markov Chain simulator 4s — less automatic compiling

Evolvix — Encode **once** by hand — **Easy** — **A few easy steps** translate model 2→4e — Evolvix.org/manual

Code below runs *as is* if entered as plain text into Evolvix.org prototype 0.3.1

4e
```
!L LineRemark: study key background steps 1-8 in ~30min
Evolvix Quest   Foxes_on_Island_with_Rabbits
( Question:   "How long will Foxes survive?"      )
Action    1 Rabbit_Breeds
( Rabbit      ---[Rate = 0.5     ]---> 2 Rabbit )
Action    2 Fox_Breeds
( Rabbit + Fox ---[Rate = 0.0002]---> 2 Fox     )
Action    3 Fox_Dies
(          Fox ---[Rate = 0.1     ]---> 0 Fox     )
Action    4 Fox_Feeds
( Rabbit + Fox ---[Rate = 0.0098]---> 1 Fox     )
Initial Amount of  Rabbit         = 500
Initial Amount of  Fox            =  40
  Simulate stochastically     until 200
!L Simulate deterministically until 200
!L all times in units of ["Month"], rates ["1/Month"]
```
5s=4s
5d=4d

Evolvix Quest Models describe **systems** with **Parts** randomly meeting for **Actions** at defined **Rates**; **TimeSeries Query** gets data from **Simulate Task**

Compiler re-encodes model using formal rules — more automatic

5 6s — **Parts kept whole** — **Proof of existence** in stochastic sample, to find next Action: roll many dice

5 6d — **Parts mashed up** — **Fast** deterministic prediction of mean; to find next Action: roll zero dice

Hard — Encode by hand, as below, into Ordinary Differential Equation (ODE) model for an ODE solver, 4d — less automatic compiling

Poster QQv10r0p2 2018-07-19

Encoding example: Equations like below are required for 6d (but useless for 6s). They show model 2 (same as [1] and Evolvix code 4e above) in the ODE language of applied math. Compilers like Evolvix can produce them automatically, cutting manual encoding costs.

4d = 5d

$$\frac{dn_R}{dt} = \underbrace{+\,r n_R(t)}_{\text{Action 1}} \underbrace{-\,\varepsilon c n_R(t) n_F(t)}_{\text{Action 2a}} \underbrace{-\,(a-\varepsilon) c n_R(t) n_F(t)}_{\text{Action 4}}$$

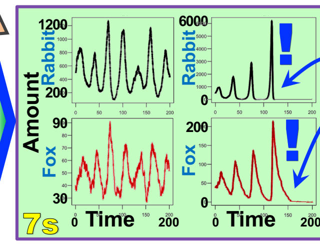
$$\frac{dn_F}{dt} = \underbrace{-\,\delta n_F(t)}_{\text{Action 3}} \underbrace{+\,\varepsilon c n_R(t) n_F(t)}_{\text{Action 2b}}$$

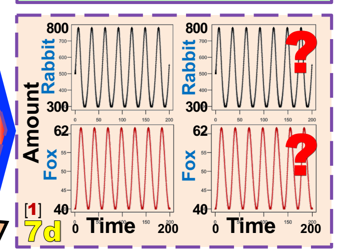
where: $c = 0.01,\ r = 0.5,\ \delta = 0.1,\ \varepsilon = 0.02,\ a = 1,$ Fox: $n_F = 40$, Rabbit: $n_R = 500$

Here each **Part** amount is changed by a differential equation that adds all **Action** fragments affecting the Part.

[1] ODE Model from Otto & Day 2007 'A biologist's guide to ... modeling..' p76 Eq3.18; values: p141 Fig4.17 has a max & min we recompute in 7d; we set 1 TimeUnit=1 Month.

8s A new stochastic run is unique and treats each individual Part as **indivisible ➔ Extinction possible!** Populations crash once too much Rabbit feeds too much Fox growth.





Hidden assumptions can affect **extinctions** in **Ecological Forecast** Time Series

8d Deterministic runs are identical assuming each individual Part stays **divisible ➔ Extinction impossible!** Populations will oscillate, pretending Part-mash (¼ Fox...) is alive (!) and keeps populations in perfect balance!

6 Simulating is testing *in silico.* For good models, simulations are like experiments in the wet-lab

7 Storing is deciding what matters about data and results.

8 Concluding is learning how to improve models, data collection, assumptions. Science is done by going back to 1 for the next round ...