Checking Scientific Assumptions by Simulation

Joseph Phillips[1], Ronald Edwards[2], Raghuveer Kumarakrishnan[1]

[1]DePaul University, School of Computer Science, Telecommunications and Information Systems
243 S. Wabash Ave., Chicago, IL 60604
jphillips@cs.depaul.edu, krv4u@yahoo.com
http://facweb.cs.depaul.edu/jphillips
[2]DePaul University, Department of Biological Sciences
2325 N. Clifton Ave., Chicago, IL 60614
redwards@depaul.edu
http://condor.depaul.edu/~biology/edwards

Abstract

We describe extensions to a science querying system to make efficient simulators. Given a scientific model's details and assumptions it creates a C++ simulator to check for assumption consistency. We used it to test both Evolution and Intelligent Design.

1. Introduction

Computers can help scientists better when computers understand a scientific model's assumptions *and* its details. Assumptions can constrain model search by limiting the allowable model space, but in this paper we show how a model's details can be used to test the reasonableness of assumptions themselves. We harness the speed of computers and the search capabilities of artificial intelligence to run multiple simulations. We also avoid a computer's incomplete knowledge of our world by stochastically defining environments.

Using this technique we were able to explore differences between the assumptions made by Evolutionary Biologists and believers in Intelligent Design. Intelligent Design states that living things have more complexity than could have arisen by chance, so some kind of rational designer (*e.g.* perhaps a god, but not necessarily) must have helped. This required an interesting scientific analysis that quantified the effect of a mechanism other than maximal selection or mutation for increasing allelic frequencies (which might lead to speciation).

Sections 2 and 3 cover prior work and motivate our work. The section 4 details our approach and its application. Section 5 introduces our biological application. Section 6 discusses our contributions and 7 concludes.

2. Prior work

Computer simulation of scientific phenomena is, of course, a broad field. We consider the most relevant computer science work to be Computational Scientific Discovery approaches that use hierarchical, abstract, scientific processes as the main semantic unit. Our research also borrows heavily from 20th century philosophy of science to determine what the high level components of a scientific model should be.

Valdes-Perez introduced the idea of explicitly playing to the strengths of computers to the field of Computational Scientific Discovery by doing exhaustive model search. His algorithms were given constraints that define acceptability and were told to search the model space from the simplest possible to ones of increasing complexity to find the first several that fit all constraints. Mechem, for example, did this for chemical reaction mechanisms [13].

Langley and Nordhausen developed the idea of creating a hierarchy of processes that have associated equations [8]. Todorovski and Dzeroski built upon a long line of equation finders to create one that finds differential equations using a grammar as a heuristic guide [12]. These teams combined their efforts to create a system that fits time-varying data given abstract processes with generic equations [6]. The system uses both semantic constraints among the abstract processes and the data to find instantiated processes with specific equations that fit and explain some phenomenon.

We also draw upon the philosophy of science for ideas on how to represent scientific models. In the 1920s a group of philosophers now called the “Vienna Circle” started thinking of how to banish beliefs other than that

which could be built upon direct sensations or upon tautologies. This effort (first named Logical Positivism and later Logical Empiricism) dismissively called beliefs like cultural assumptions "metaphysics". Englishman A. J. Ayer and other positivists who left continental Europe in the 1930s popularized this idea in the English-speaking world [1]. There it became the dominant philosophy of science after World War II.

Logical Empiricism severely declined in the 1960s for several reasons including the realization that observing something required the *prior* existence of knowledge to interpret it. Even babies can interpret observations to some degree, so some of this knowledge must be pre-scientific.

Further, much of this knowledge is also cultural because our language determines the words we use to describe observations. Goodman elegantly pointed this out with the "grue" paradox [4]. Something is "grue" iff it appears green before some particular time "t" safely in the future and then appears blue thereafter. Therefore everything that people call "green" (like emeralds) might actually be "grue" and everything that people call "blue" (like the sky) might actually be "bleen". We will not know until time "t". Why then do we continue to use the word "green"? This might not correspond to reality.

Goodman's answer was embrace (but to try to be aware of) our cultural assumptions like the definitions of words. They've worked in the past; hopefully they'll work in the future.

We draw upon philosophers too. Thagard introduced Processes of Induction (or PI) to propose a computational scheme for scientific reasoning and discovery, but not as a working discovery tool [11]. PI represents models as having theories, laws and data. It evaluated scientific models by multiplying a simplicity metric by a data coverage metric.

3. Motivating Example Simulation

An Aristotelian and a Newtonian cannot agree on how objects move and would like to resolve their difference by simulation. The Aristotelian believes that their natural state is at rest. The Newtonian believes that once in motion they tend to stay in motion. However, they do agree on Galileo's (friction-free) equation for falling objects $height = kt^2$. (This equation came from experiment so it is hard for either to refute.) They also agree that figure 1 is relevant:

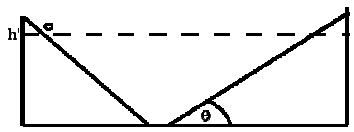

Fig. 1. Galilean experimental setup.

Balls roll down the left incline and up the right. When they do they reach the same height h' on the right incline that they were released at on the left incline, though it may take a different time to get there depending on the right incline's angle θ.

Their differing fundamental assumptions about what objects do in the long term go into their own *metaphysics*. For us, metaphysics is the part of the model that holds a scientist's cultural assumptions.

Simulations allow both scientists to see how well their beliefs agree with their common model by varying angle θ. The Aristotelian sees that (in the friction free limit) the ball will roll back and forth indefinitely, thus contradicting their assumption. The Newtonian sees agreement with ball's continual motion over a range on angles.

Of course the Galilean notion of inertia is implicit in the equation so the Aristotelian was bound to be disproved. And that is precisely the point: we have shown how simulations can vary free parameters of an abstract model and can argue for (or against) the fundamental assumptions that scientist holds. We can apply this technique to stochastic models where the disagreement is less obvious.

4. Assumptions and Their Application

Our models have five components: three empirical ones (Theory, Laws and Data) and two non-empirical ones (Metaphysics and Mathematics). (By "empirical" we mean "open to revision based upon observation".) Theory holds high level empirical knowledge (*e.g.* Newton's Laws of Motion plus Gravitation), Data holds low level measurements (*e.g.* Tycho Brahe's observations) and Laws holds knowledge that is supported by the Theory but closer to Data (*e.g.* Kepler's Laws of Planetary Motion). Metaphysics contains cultural assumptions made by a scientist (*e.g.* the ontology of domain objects, definitions of dimensions and units, and *a priori* limitations that a scientist may place on a model). Mathematics gives information on how to transform problems without adding empirical content, like the knowledge needed to change co-ordinate systems.

Our component architecture is consistent with our previous work [9] and is supported by models of science held by the modern philosophers. The notion of "Metaphysics" (even that word) comes straight from the philosophy of science. We follow Goodman [4] and Lakatos [5] by admitting that cultural assumptions impose constraints on observations and the other aspects of empirical content. We place them in the Metaphysics component where they are centralized and potentially open to critique. The Mathematics component comes from the Logical Empiricist idea of tautology. Finally, we follow Thagard by splitting our empirical knowledge into Theory, Law and Data. Our conceptualization of "Law" may not be universally shared. "Law" is structurally similar to "Theory" but is supported by (perhaps provable from) Theory, Metaphysics and Mathematics.

Having five components gives us the both ability to constrain search. Storing assumptions gives our system:

1. Limited ability to automatically check the consistency of the Theory, Laws and Data with the Metaphysics,
2. The ability to constrain the search for revisions to Theory, Laws and Data using metaphysical knowledge, and,
3. The usage of the Theory, Laws and Data to define a simulation to check the suitability of the Metaphysics (which is the subject of this paper).

Additionally, having five components gives us the ability to use one knowledge base to support several queries. The Metaphysics component is always checked first in response to any query to enforce conformity with basic assumptions. The Mathematics component is always checked last to recast a query into another form after all other components have failed. In between those two we may call the empirical components in any order. Thus we can compute in a variety of ways and emulate the types of thought exhibited by practicing scientists. Defined orderings include:

ab initio (Metaphysics, Theory, Mathematics). Computes the deductive closure of the theory and is meant to see if first principles alone are sufficient to derive some result.

read data (Metaphysics, Data, Mathematics). Computes the deductive closure of the data and is meant to see if some result naturally must follow from observation, regardless of the theory.

theorize (Metaphysics, Theory, Law, Data, Mathematics). Uses all components but seeks a more theory-driven answer first.

empiricize (Metaphysics, Theory, Law, Data, Mathematics). Uses all components but seeks a more observation-driven answer first.

given (Metaphysics, Mathematics). Ignores empirical components to ask for an answer that is the deductive closure of the assumptions. These answers can be used both (1) as a baseline to see what added power the empirical components actually provides, and (2) as a check to see how much empirical knowledge has crept into the assumptions.

All five components are implemented in Scilog [10]. Scilog extends Prolog for scientific knowledge in two ways. First it defines new predicates and functions to state and compute scientific properties. Second it gives precision and units (with their dimensions) to primitive values, and defines the domains and scales to which they belong.

Scilog is designed to answer <object, attribute, value> frame-style queries. It holds scientific knowledge by natively supporting Prolog-style facts; Prolog-style rules; equations and decision trees that pertain to a hierarchy of processes (*cf.* Physics: *equations of motion*); and equations and decision trees that do not pertain to any particular process (*cf* Physics: *equations of state*). Scilog also supports database access. Equations, decision trees and processes have preconditions associated with them that must hold for that assertion to be used. Like Langley *et al*. we describe scientific processes hierarchically. Our process classes interrelate domain objects that take part in the process' action. We call these objects *process entities*.

Scilog values have a primary value (a mean, median or mode), a certainty range, a domain, a scale and optional references to the object and attribute being described. Domains and scales specify the units, dimensions, legal values and other meta-knowledge about values. There are three numeric domain types (integer, float_pt and fixed_pt) and one non-numeric type (concept). All three numeric types have upper and lower bounds corresponding to (1) logically-defined limitations (*e.g*. latitudes can not exceed +90), (2) bounds on the system being measured, (3) bounds for values that can not be detected at all by the given instruments, (4) bounds imposed by the system that it empirically has never been seen to exceed, (5) bounds on which values can be *reliably* directed by the given instruments, and (6) bounds saying that no value has been observed outside

for some given system.

Besides being able to describe a Scilog value with one primary value, Scilog also supports defining values as Gaussians or sampled distributions.

Scilog extends the native Prolog predicates and functions **=**, **is**, **>**, **<**, *etc.* to check dimensions and convert between units of the same dimension. Additionally, the operators **+**, **-**, *etc.* both invent new domains as needed and compute certainty ranges given the certainty ranges of their parameters. If this results in a non-Gaussian probability distribution then the resulting value is a distribution of 32 weighted random samples.

For this work we have extended Scilog to support the new assertions. Process classes now can have constraints and procedures associated with them besides preconditions, equations and decision trees. Process constraints are boolean expressions that give expectations about their outcomes. Procedures are labeled bits of code that tell how the process can be simulated. Procedures are defined using the commands given in Table 1.

Table 1. Commands for defining simulations

Command:
noop: Does nothing.
code_block(*list*): Executes the items in *list*.
assignment(*id1*,*id2*): Assigns process entity *id2* to process entity *id1*.
define_entities_attr(*id*,*attr*): Assigns process entity *id*'s attribute *attr* to the value returned by an equation or decision tree associated with the process class that defines *attr* for entity *id*. *attr* may be an expression.
define_entities_attr (*id*,*attr*,*val*): Assigns process entity *id*'s attribute *attr* to have value *val*. *attr* may be an expression.
dotimes(*exp*,*cmd*): Does *cmd exp* times.
process_instance_simulate (*class*,*id*,*list*): Recursively calls process *class* to (optionally) create process entity *id* given the parameters of *list*. The members of *list* may be process entities, domain objects, or objects that tell how to choose between domain objects (*e.g.* weighted random choice among members of a set).

This is how our system creates a simulator. First, the user loads one particular model, including its metaphysics. Then in addition to the other predictive functionality that the system implements, the user may choose to investigate how well the metaphysics fits with the rest of the model. This option causes the system to consider serially all process classes given in the metaphysics. For each one, if it specifies at least one constraint and has a procedure that calls another process class, then the system loops through the called process class and its subclasses to consider if simulation can be made.

A simulation can be made when a metaphysical process class (P_M) makes claims about an empirical process class (P_E) and certain conditions are met. All process classes recursively called by P_E must have procedures and are incorporated into the simulator. When a procedure needs to compute an attribute the system does the following:

1. Sees if the value is the same for all objects that it could describe. If it is, then the system fills it in with a constant.
2. Sees if an equation or decision tree associated with the current process class can answer that query. If so, then that assertion is recursively incorporated into the simulation.
3. Sees if some other equation or decision tree can answer the query.
4. Sees if domain information exists for that attribute. If so, then it is assigned a random value from that domain at the beginning of the simulation.

If all of the steps fail, then it gives up that approach to making a simulator.

After the simulator is made it revisits the preconditions of P_M. Preconditions concerning the random values are incorporated into the simulator's random value choosing operation.

5. Experimental Domain: Population Genetics

We used the software detailed in the previous section to build a population simulator that used genetic knowledge. The simulator kept track of the frequencies of alleles in a population, where an allele is a form of a gene that partially codes for some organism's property. This simulator was used to test whether a series of improbable events is necessary to increase a non-maximally favored allele's frequency (as Intelligent Design states) or whether reasonably probable events can still increase a non-maximally favored allelic

frequency (as implied by Evolution).

This is an important question because the proponents of Intelligent Design state that only improbable selection can significantly shift allelic frequencies to potentially lead to speciation.

5.1 Evolutionary Biology and Intelligent Design

We explored shifts in an allele's frequency in a population over time, and their implication for the development of new traits and ultimately new species. When there are two traits with equal fitness in a sufficiently large, randomly-mating population we expect the underlying allelic frequencies (and thus percentage of individuals with either trait) to remain constant over successive generations by the Hardy-Weinberg equilibrium. In these simulations, however, we relaxed both the notion of "equal fitness" and of "sufficiently large population".

For most of the 20th Century evolution had been explained in terms of genetics by a paradigm called *New* (or *Modern*) *Synthesis*. In the 1960s this paradigm was extended by Mayr [7] by positing that speciation was driven by selection for new traits appearing in geographically distinct subpopulations.

Followers of Intelligent Design, however, believe this account of macroevolution (and hence Darwinian evolution as a whole) to be improbable. They agree that natural selection could lead to allelic frequency shift but state that it is unlikely. They interpret it as requiring too many conditions [2], namely geographical isolation, new trait mutation, and superior fitness. In the absence of any one of these Intelligent Design claims that increases in allelic frequency will be rare, and that the formation of distinct subpopulations and new species will be unlikely.

5.2 Knowledge

First we must see what both models have in common and how they differ. Their common components include:

1. **Logistic growth**. When the population in a fixed area has a size **N** that is small relative to how many organisms that area can support (the area's *carrying capacity*, or **K**) then the population grows exponentially. However, as it grows it levels off to approach **K** asymptotically. This is given in the differential equation:

   dN = r * N * (1 – N/K)

   where **r** is a growth rate that is characteristic to the species and its environment.
2. **Mendelian genetics**. We assume that genes will have two alleles: a dominant (*e.g.* **A**) and a recessive (*e.g.* **a**). If the organism has the allele pairing **AA** or **Aa** then it exhibits the dominant phenotype. Having pairing **aa** will make it exhibit the recessive phenotype.
3. **The remaining Hardy-Weinberg conditions**. We assumed that there is no mutation, no organisms entering or leaving the population, no inter-generational mating, and that individuals within the same generation are equally likely to mate with each other. These assumptions are standard and necessary to avoid the multidimensional parameter space of differing mating opportunities, fertility, brood sizes, age of sexual maturity, longevity, mutation rates, immigration rates, emigration rates and breeding season's timing and duration.

Though simple, these assumptions are powerful. They make few claims about "real-world" genetics and behavior and therefore can encompass a wide variety of cases. (The most unrealistic simplification may be that a gene uniquely "controls" a trait, but developmental genes may influence hundreds of others.) Another advantage of using relatively basic biology is that these ideas are not challenged by either Evolutionists or believers in Intelligent Design. This knowledge goes into the Theory component.

5.3 Methods

We then defined a "Best of a Bad Circumstance" model to see if an intermediate choice can still increase, even though it is not the best. The idea is that there are two genes: **gene_a** and **gene_b**. If an organism has at least one copy of **gene_a**'s dominant allele **A** then it has maximal fitness. If, however, it has two copies of the recessive allele **a** then it has one of two possible fitnesses. The "Best of a Bad Circumstance" fitness (which in general is worse than the maximal fitness) is available if it has at least one copy of **gene_b**'s dominant allele **B**. If, however, it has genotype **aabb** then it has the absolute worst fitness.

Intelligent Design states that "The three conditions (geographic isolation, new trait creation, and superior

fitness) are all simultaneously necessary for new allelic frequency increase, and this is improbable." Note that in our model there is geographic isolation but no mutation to create new traits. Also, **B** always encodes for a strategy that is never better, and most often worse, than **A**'s. The test we wish to conduct is "*Does **B**'s frequency ever increase higher than due to fitness alone?*" This is currently inexpressible, so instead we gave Intelligent Design's Metaphysical component the process constraint "*Does **B**'s frequency increase?*" We did the more detailed analysis by SAS afterward.

We translated our "Best of a Bad Circumstance" model to a Scilog program with five process classes:

Table 2. Theory Component knowledge

Process Class:
population_over_time: Runs *create_1st_generation* and then *create_subsequent_generation* 100 times
create_1st_generation: Calls *randomly_create_organism* the number of times given by the initial generation size.
randomly_create_organism: Stochastic decision trees initialize alleles for **gene_a** and **gene_b** according to the free parameter initial probabilities of **A** and **B**. A third decision tree then computes the organism's fitness.
create_subsequent_generation: It uses Logistic Growth to compute how many organisms to create. For each it randomly chooses parents (weighted by their fitness) and calls *randomly_mate_organisms*.
randomly_mate_organisms: Decision trees stochastically define alleles for **gene_a** and **gene_b** according to parents and Mendelian genetics. A third computes its fitness. The organism is also attached to its generation.

Next, the biologist among us defined the legal domains for all attributes. From this Scilog knows the legal values of the free parameters used in the simulations.

Table 3. Domains of free attributes

Parameter and Domain
Initial generation size: x, where x = floor(10^y + 0.5), y in {1.0, 1.0625, 1.125, .. 3.0}[1]
Carrying capacity: x, where x = floor(10^y + 0.5), y in {1.0, 1.0625, 1.125, .. 3.0}
Growth rate: {0,0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9,1.0}
generations: {100}
*p(**A**) in init. population:* {0,0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9,1.0}
*p(**B**) in init. population:* {0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9,1.0}
*fitness(**A???**):* {1.0}
*fitness(**aaB?**):* {0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9,1.0}
*fitness(**aabb**):* {0,0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9,1}[2]

Next we encoded the Intelligent Design assumption set in its Metaphysics as a process class *creationist_over_time* which calls *population_over_time*. It is also given the preconditions that the initial generation is not larger than the carrying capacity, and that *fitness(**aabb**)* is not greater then *fitness(**aaB?**)*. Finally, it has the constraint that the fraction of **B**'s in the initial generation is expected to be greater than or equal to the fraction of **B**'s in the final generation.

The "null hypothesis" (that these conditions *can* increase **B** allelic freq.) is the metaphysics for Evolution. This would result in the same simulator, with the expectation for **B** switched.

Finally, we told our system to create a simulation program. The result of this was a C++ simulator that randomly choose free parameters from their legal domains (re-choosing all parameters if the initial generation size was greater than the carrying capacity or if *fitness(**aabb**)* was greater than *fitness(**aaB?**)*) and did the simulation. It was called 100,000 times by a

1 Subject to the constraint that it is never greater than the *Carrying Capacity*

2 Subject to the constraint that it is never greater *than fitness(**aaB?**)*

wrapper program that re-seeded the simulator's random number generator with the wrapper's index.

5.4 Results

In addition to counting the number of cases meeting the test constraint, our system saved the values of random and test constraint variables to let users do additional analyses.

The most consistent outcome of the simulations is that the frequency of **A** always increased, while the frequency of **B** increased in 57,038 of the 100,000 trials. Although this contradicts the Intelligent Design's assertion that **B** should never increase, its proponents could say that this is just selection. (

However, a more detailed look at when **B** increases shows that selection is only one effect. Figure 3 plots the mean increase in **B**'s frequency as a function of carrying capacity and shows an increase for ***some*** small carrying capacities.

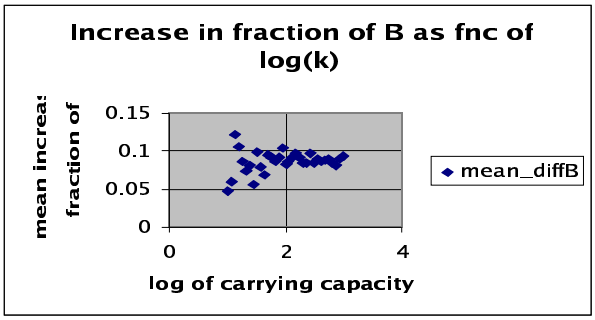

Fig. 2. Inc. in B as a fnc of the log of the K.

Moreover, principal component analysis by SAS gives us the following:

Table 4. Principal Component Analysis

	Eigenvalue	***Proportion***
V1	1.47065	29.41%
V2	1.16894	23.38%
V3	0.999898	20.00%
V4	0.830874	16.62%
V5	0.529637	10.59%

	Greatest absolute vect	***2nd greatest absolute vect***
r	v3: 0.999433	v2: -0.029435
K	v5: -0.707107	v1: 0.706781
n0	v5: 0.707053	v1: 0.706742
fit(aaB)-fit(aabb)	v4: 0.706908	v2: -0.706071
inc. B	v4: 0.707044	v2: 0.706863

All variance was accounted for by five eigenvectors, each of which had a high (> 0.5) loading on the following variables:

1. Increase in carrying capacity with the initial generation's size (29.41%)
2. Increase in the frequency of **B** without selection for B (23.38%)
3. Variation in birth rate (20.00%)
4. Increase in the frequency of **B** with an increase in selection for **B** (16.62%)
5. Increase in the initial generation's size without carrying capacity (10.59%)

This shows two independent trends concerning **B**. First, Vector 4 (17% of the variance) accounts for selection. **B** increases as *fitness(**aaB?**)* becomes greater than *fitness(**aabb**)*. This tendency is spread through-out figure 2 and is expected by Intelligent Design.

Second, however, is Vector 2 (with 23% of the variance) which shows an increase in **B** *without* a high selection. This is the *Founder Effect:* that random variations in the initial generation influence later generations. Figure 2 shows how this effect leads **B** to become disproportionately frequent in ***some*** smaller environments (*e.g.* small islands). Apparently below a certain size (about 20 in our simulations, which may not be a real-world value), the favorable effects of selection are enhanced by a tendency for the Founder Effect to reinforce the frequency and potential fixation of **B**. **B** was never selectively favored nor artificially increased by mutation. The only required (and reasonable) condition was the geographic isolation of a diverse subpopulation.

Our result (organisms in many smaller environments evolved a new trait *without* mutation *or* a high selection coefficient) agrees with post "New Synthesis" Biology. It supports Brook's idea of "evolution as entropy," in that the ability for change (even innovation) is the default condition of organisms [3]. Multiple improbable events (the right mutation, in the right population, at the

right time) are unnecessary.

6. Discussion

Both our framework's power and the originality of this research come from the extension of a specialized scientific reasoner to create automatically simulators to test base assumptions. We acknowledge that it would be straight-forward to hand-code this simulation in C++ or Java. Our contribution, however, is that of ***automatic simulator generation***. We can take a scientific model that was used to answer empirical questions and ***automatically*** generate a simulator to check its assumptions.

This work has neither "proved" nor "disproved" Intelligent Design because the vague statement "X, Y and Z make A too unlikely" is inherently difficult to prove or disprove. Instead, we have shown Intelligent Design is ***inconsistent*** with basic biological theory and attribute distributions that we think reasonable more than 50% of the time. Intelligent Design's proponents claim that it is a modern science on par with Evolution. However, they have not (to the best of our knowledge) discounted Logistic Growth (the simplest useful model of population growth in an nutrient-limited environment), Mendelian Genetics (the simplest useful model of inheritance) or the Hardy-Weinberg analysis (the simplest useful explanation why many frequencies empirically do not change).

7. Conclusion

In this work we have:

1. Addressed a specific genetic and population dynamic question about how allelic frequency can change.
2. Extended a system for specific question answering to do simulations.
3. Built a simulator with a general algorithm that used specific domain knowledge of process classes and their relationships.
4. Given scientists an opportunity to write and test some of their assumptions and those of others.

One may claim that examining one's basic assumptions is hard because one may not be conscious of them all. We agree – and therefore believe it best to build up libraries of assumption sets.

Future work will include implementing the other functionality that having Metaphysics allows (checking empirical component consistency and constraining search for empirical component modifications).

We gratefully acknowledge DePaul's University Research Council for support.

References

1. Ayer, A.J. (1952) *Language, Truth and Logic*. Dover Publications, NY.
2. Behe, Michael (1996) *Darwin's Black Box: The Biochemical Challenge to Evolution*, Free Press, NY.
3. Brooks, D.R., Wiley, E.O. (1988) *Evolution as Entropy: Towards a Unified Theory of Biology*. Univ. of Chicago Press, Chicago.
4. Goodman, N. (1966) The New Riddle of Induction, in: *Journal of Philosophy* 63: 281-331.
5. Lakatos, I. (1970) Falsification and the methodology of scientific research programmes. In Lakatos, I. and Musgrave, A. (ed.) *Criticism and the growth of knowledge*. Cambridge University Press: Cambridge.
6. Langley, Pat, Sanchez, Javier, Todorovski, Ljupco, Dzeroski, Saso (2002) "Inducing process models from continuous data" in *Proc. of 19th Int'l Conf. of Machine Learning*, Morgan Kaufmann, San Francisco, p 347-354.
7. Mayr, E. (1963) *Animal Species and Evolution*. Harvard Univ. Press, Cambridge, MA.
8. Nordhausen, B., Langley, P., (1990) An integrated approach to empirical discovery, in Shrager, J., and Langley, P. (ed.) *Computational Models of Scientific Discovery and Theory Formation*. Morgan Kaufmann. San Mateo.
9. Phillips, J. (2001) Towards a method of searching a diverse theory space for scientific discovery. *Discovery Sci.*, Springer-Verlag, Berlin, p 304-322.
10. Phillips, J. (2003) Scilog: A language for scientific processes and scales. *Discovery Sci.*, Springer-Verlag, Berlin, p 442-451.
11. Thagard, P. (1988) Computational Philosophy of Science, MIT Press, Cambridge MA.
12. Todorovski, L. Dzeroski, S. (1997) Declarative bias in equation discovery. *Proc. of the 17th Int'l Conf. of Machine Learning*, Morgan Kaufmann, San Francisco, p 376-384.
13. Valdes-Peres, Raul (1995) Machine discovery in chemistry: new results. *Artificial Intelligence*, 74 (1): 191-201.