# Automating the Development of Metabolic Network Models

# Automating the development of metabolic network models

Robert Rozanski[1]*, Stefano Bragaglia[2], Oliver Ray[2], Ross King[1]

[1] School of Computer Science, University of Manchester, M13 9PL, UK
[2] Department of Computer Science, University of Bristol, BS8 1TH, UK

**Abstract.** Although substantial progress has been made in the automation of many areas of systems biology, from data processing and model building to experimentation, comparatively little work has been done on integrated systems that combine all of these aspects. This paper presents an active learning system, "Huginn", that integrates experiment design and model revision in order to automate scientific reasoning about Metabolic Network Models. We have validated our approach in a simulated environment using substantial test cases derived from a state-of-the-art model of yeast metabolism. We demonstrate that Huginn can not only improve metabolic models, but that it is able to both solve a wider range of biochemical problems than previous methods, and to utilise a wider range of experiment types. Also, we show how design of extended crucial experiments can be automated using Abductive Logic Programming for the first time.

## 1 Introduction

Biological systems are extremely complicated. Even the model cellular systems of *Escherichia coli* and *Saccharomyces cerevisiae* consist of thousands of genes, proteins, small molecules, *etc.*, all interacting in complicated spatiotemporal ways. In addition, as biological systems have evolved through Darwinian evolution, Ockham's razor is not as effective as it is in the physical sciences.

Currently, although many computational tools are used to build systems biology models, the evaluation and analysis of these models is still mostly done by humans, who identify conflicting results, suspicious or low-confidence elements of models, ask specific questions to test the models, and run manual experiments. However, humans can only investigate small parts or aspects of models, because of their typical size and complexity. This bottleneck could be overcome by automating model development, *i.e.* the process of asking specific questions, running tailored experiments to answer them, and revising models if needed.

* corresponding author: rozanskr@cs.man.ac.uk
Huginn is an open-source software, available at: github.com/robaki/huginnCMSB2015
All figures included in this paper are in public domain; files can be downloaded from: github.com/robaki/huginnCMSB2015

## 1.1 Adam, a robot scientist

King *et al.* [10] created an automated system that investigated the problem of orphan enzymes in metabolic models of yeast. The system, "Adam", was able to propose initial hypothetical models, and then design two-factor growth experiments to test them. The experiments were run using automated laboratory equipment. The data were then analysed to determine which models to refute. Adam, although successful, has multiple limitations. Its methods of proposing hypotheses were specific to the problem of orphan enzymes. Its experiment design and hypothesis testing algorithms were limited to only one type of experiment, and could not be easily extended. It also lacked general revision capabilities. These limitations make Adam unsuitable candidate for a general-purpose metabolic model development system.

## 1.2 Huginn

We have developed Huginn[1], to overcome some of the limitations of Adam. In doing this we have drawn from Machamer's, Darden's and Craver's (MDC) theory of discovering mechanisms. We have adopted MDC concept of mechanism to represent Metabolic Network Models (MNM) in a way suitable for automated system. We have also used their characterisation of the final stage of the model development process as a guide to the design of Huginn. We used Logic Programming, and Abductive Logic Programming (ALP) (Gringo [9], Clasp [8] and XHAIL [15]) to automate model construction and revision, as well as testing consistency of models with experiments. We have also used them to automate experiment design in a novel way.

## 1.3 Metabolic networks as biological mechanisms

A significant amount of research in biology is concerned with development of models of mechanisms (*e.g.* of DNA replication). By representing what is happening in biological systems these models provide a way to predict and explain their behaviour in a way understandable to humans. Recently the notion of mechanisms in biology has attracted the attention of philosophers of science, who have tried to specify what these mechanisms are, and how they are discovered. [2,5,6]

In this study we have adopted the notion of mechanism proposed by MDC [14]. They characterise mechanisms as collections of entities and activities organised in such ways that they can produce regular changes from setup to termination conditions. For example, a model of cellular respiration would show how cells produce ATP from glucose through a series of chemical reactions and transport processes.

The core qualitative information about metabolism are the chemical reactions and other processes that can occur in an organism, as well as chemical substances involved in them. MNM represents these processes in a form of hypergraphs. MNMs typically abstract away not only the concentration and dynamics

[1] From the Norse mythology – one of two ravens scouting the world for Odin.

of the system, but also some of the conditions, *e.g.* certain enzymes are expressed only under specific conditions. MNMs can be understood as MDC-type descriptions of mechanism. MNM show how certain chemicals are produced from other chemicals by representing continuous chemical paths from the former to the latter. Initial and termination conditions are the presence of specific species (*e.g.* metabolites) and genes in specific compartments (*e.g.* cytosol). Activities like chemical reactions, transport, gene expression and complex formation connect these conditions through intermediate steps.

## 2 Methods

### 2.1 Discovery of mechanisms

The MDC concept of biological mechanism was developed to better understand the discovery of mechanisms. Discovery should not be understood here as an event, but as an extended iterative process of exploration, specification, building, testing and revision. According to MDC [4, 6], the process starts with exploring and characterising the phenomenon of interest, *i.e.* one that is to be explained by description of mechanism. Then, incomplete and often abstract sketches of mechanisms are formulated, taking into account clues such as the nature of the phenomenon, its context (*e.g.* evolutionary), its spatial and time characteristics. These sketches show how the phenomenon could possibly be produced. Through specification and initial evaluation sketches are turned into schemata: these still may be to some extend incomplete or abstract, but contain enough information to allow production of fully specified models. Then, through further instantiation (if required) and searching for direct experimental evidence, final descriptions of mechanisms are produced. The transition between each of these stages involves construction, evaluation and anomaly resolution (revision), which is guided by specific strategies.

In this paper we focus on the latter stages of the discovery process, where phenomenon is fully characterised and models of mechanisms are composed entirely of non-abstract elements, *i.e.* they are constructed from specific reactions, proteins, metabolites and not from place-holder elements. We implement a number of strategies proposed by MDC in the design of Huginn, specifically:

- continuity and productivity are taken into account in construction, consistency testing, and revision
- generation and elimination of rival hypotheses (using crucial experiments)
- searching for direct evidence for hypotheses by *in vivo* and *in vitro* experiments:
  - entity and activity detection
  - characterising entities *in vitro* (enzymes' properties and complex formation)
  - disrupting mechanisms, and studying changes (gene deletions and changes in medium composition)

The model development process used in Huginn (see fig. 1) is initialised in a number of steps. First, initial models and experiment results are recorded. Then models are checked for consistency with the results, as well as other criteria, like ability to produce termination conditions (*i.e.* synthesize final compounds) and presence of disconnected activities (*e.g.* reactions which substrates are not present in the model). Models that failed are revised. If the pool of initial models is smaller than user-specified threshold, then additional models are produced to fill that gap and the system is ready to enter its proper development cycle.

The first step in the development cycle is to design an experiment to test current working models. Then, the experiment is executed (simulated) and results used to test working models. Refuted models are then revised. If there is no way to make a model logically consistent with the results, then one or more of them will be ignored. This ability to ignore results is important for dealing with limitations of the Knowledge Representation method, as well as factors such as experimental noise, and the open world problem. The quality scores of models are then recalculated based on the number of covered and ignored results.

Huginn stops development process if at least one of three conditions is true. The first condition is lack of progress. If there were new models produced recently or if the best (highest quality score) model has recently changed, then development continues. The second condition is running out of experiments to execute, which happens when working models become empirically equivalent. In this case Huginn tries to redesign models at random, but if it fails 10 times, it stops. The last condition is running out of time or exceeding maximum number of cycles: both values are specified by the user.

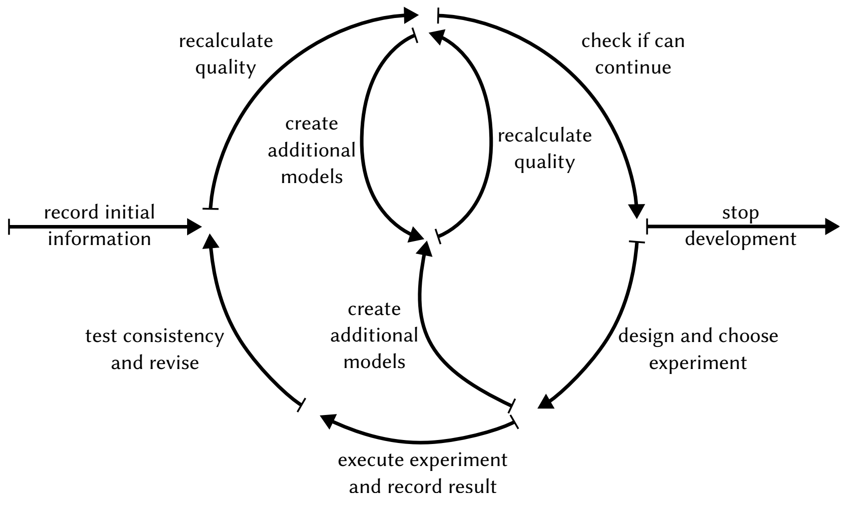


**Fig. 1.** Model development process

## 2.2 Abductive logic programming

The development process relies on four core operations: consistency checking, revision, production of additional models and experiment design. We use ALP for these operations. Abductive inference is typically understood as inference to the best explanation. In ALP abduction is defined as constructing a hypothesis $H$, that together with background knowledge $B$, entails a set of examples $E$:

$$B \cup H \models E$$

Unlike deduction, abduction is a defeasible form of inference, *i.e.* given true background knowledge and examples (observations), it may produce false hypotheses. However, it has the advantage of being able to produce novel knowledge. ALP tools have been used previously for completion [3, 11] and revision of metabolic networks [16]. Thanks to optimisation capabilities of existing tools one can generate theories that not only satisfy hard logical constraints, but are also optimal with respect to user-specified criteria.

## 2.3 Representing models using logic

MNM can be formalised and translated into datalog-style logic programs. Entities are defined by their type, identifier and version. Huginn currently supports four types of entities: metabolite, protein, complex or gene. Versions enable one to represent uncertainty regarding an entity's properties. Two currently supported properties are *catalyses* and *transports*.

Huginn supports five types of activities: chemical reaction, complex formation, expression, transport or growth. Substrate and product predicates are used for all types of activities, and these specify not only what entities are required and produced, but also in what compartments. Apart from substrates, chemical reactions and transport may need catalysts or transporters respectively.

Models are defined by specifying which setup conditions and activities they contain.

All these facts describe the elements involved in the MNM. In order to determine which metabolites are synthesizable, simulation rules are added to this description. A group of rules marks as *active* activities which all substrates are either initially present or synthesizable (in appropriate compartment) and which catalyst/transporter requirements are met. An additional rule marks all products of active reactions as synthesizable.

## 2.4 Experiment types and predictions

Model descriptions need to be supplemented with prediction and consistency rules to support the use of empirical information. Predictions describe what outcome models predict w.r.t. description of experiment. Outcome is binary: true or false. In addition, model can be indifferent w.r.t. experiment (it does not predict any outcome). Model is inconsistent with a result of experiment if outcome of the experiment is different from the predicted one. Prediction rules determine predicted outcomes of experiments. There are seven types of experiments currently used in Huginn and each of them has its separate set of prediction rules:

*Entity Detection*: detection of metabolites, proteins or complexes.
*Entity Localisation*: as above, but in a specified compartment.
*Activity Detection*: used for detecting growth.
*Activity Reconstruction*: checks if activities can be reconstructed without enzymes or transporters.
*Reconstruction Enzymatic Reaction*: checks whether given entity can catalyse specific reaction.
*Reconstruction Transporter Required*: as above, but for transporters.
*Two Factor Growth Experiment*: used previously to test candidate parent genes of orphan enzymes [10]. It tests whether decreased growth rate after gene deletion can be offset by addition of a particular metabolite.

Some types of experiments can include interventions: addition or substraction of a specific entity from specific compartment. In our study we have restricted interventions to manipulation of the growth medium (addition/substraction of nutrients) and gene deletions. The way the interventions are handled differs depending on the nature of the task (revision, experiment design, *etc.*).

### 2.5 Automating crucial tasks

As mentioned above, the four essential tasks in the model development cycle are: consistency check, revision, construction of additional models and experiment design. All of these tasks were automated using Logic Programming (LP) techniques.

**consistency check:** This step consist of checking whether models are consistent with all known results as well as additional structural criteria. Specifically, models must synthesize all compounds specified in the termination conditions, they must not contain any activities that are missing substrates, and they cannot contain two versions of the same entity (that situation would be equivalent to having inconsistent beliefs about the entity's properties).

**revision:** The models are revised by supplementing requirements from consistency check with mode declarations that specify what activities can be added and removed. XHAIL then tries to minimise a number of changes to the model. In cases where more than one optimal solution is found, one of them is chosen at random to keep the population of working models at a constant size.

For example, metabolite *met_8* was detected in cells with deleted gene *g*26. This outcome is in conflict with predictions of model (a) (fig. 2). In that model *met_8* can be synthesized from input metabolites *met_7* and *met_11* (marked

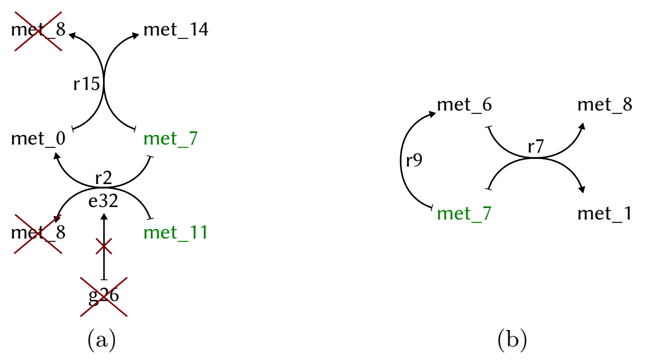


**Fig. 2.** Revision example: (a) deletion of $g26$ disrupts reactions $r2$ (lack of enzyme) and $r15$ (lack of substrate: $met\_0$) and thus prevents the model from producing $met\_8$, contrary to experimental results. Consistency with the results can be restored by adding two additional reactions (b) which can produce $met\_8$ independently from $g26$.

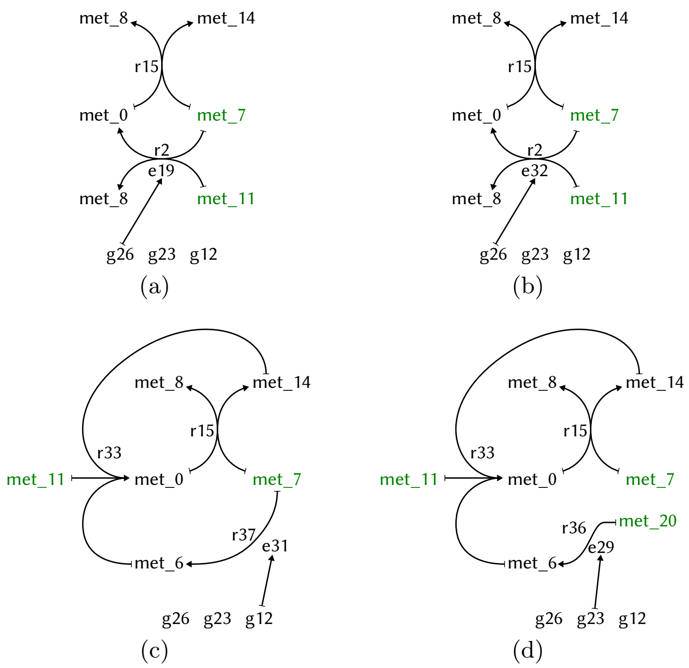


**Fig. 3.** Experiment design example: models (a) and (b) rely on gene $g26$ to produce $met\_0$ and $met\_14$, while models (c) and (d) rely on genes $g12$ and $g23$ respectively. Thus experiment consisting in deleting $g26$ and detecting either $met\_0$ or $met\_14$ will split these models into two groups: one predicting that the metabolite will be synthesised despite deletion, the other that it will not be.

green) in reaction $r2$, which requires enzyme coded by $g26$. Alternatively, it can be synthesised in $r15$, but that requires some source of substrate $met\_0$. Since the only source of $met\_0$ is $r2$, deletion of $g26$ disrupts both reactions and $met\_8$ is not produced. Consistency with the experimental result can be restored by adding reaction(s) that can synthesise $met\_8$ independently from $g26$, *e.g.* reactions $r9$ and $r7$ (fig. 2(b)).

**construction of additional models:** Additional models are constructed using almost the same approach as in revision, but with the addition of a requirement that the resulting models must be different (contain different set of activities) from any of the working models.

**experiment design:** The idea behind our approach to experiment design is to design experiment that will split the working models into two groups of equal size: one predicting that outcome of experiment is true, the other that it is false. This can be understood as an extension of the concept of crucial experiment. The same principle was used before as a strategy for choosing experiments from pre-generated sets [11].

For example, lets consider four models from fig. 3. The input metabolites are $met\_7$, $met\_11$ and $met\_20$ (marked green, only shown where relevant), and the output metabolite is $met\_14$. All models synthesize $met\_14$ in $r15$, but differ in ways they produce required substrate for this reaction: $met\_0$. Models (a) and (b) rely on $r2$ and gene $g26$, while models (c) and (d) use $r37$ (needs gene $g12$) and $r36$ (needs $g23$) respectively. Therefore, if $g26$ is deleted models (a) and (b) will predict that $met\_14$ is not produced, while (c) and (d) will predict that it is produced. One of plausible experiments for this group of models is then a *detection entity* experiment, detecting $met\_14$ and involving one gene deletion (of $g26$).

Since some models may be considered to be better and therefore more probably correct in a subjective sense, we split not raw numbers of models, but rather their total quality score. Since designing experiment that will split scores into equal groups is not always possible, this task was implemented as optimisation problem. The system tries to minimise total penalty, which is calculated as follows:

$$P = |0.5 * \sum_{m} q(m) - \sum_{m \in T} q(m)| + |0.5 * \sum_{m} q(m) - \sum_{m \in F} q(m)| + \sum_{m \in I} q(m)$$

where $m$ is model, $q(m)$ is model's quality, $T$, $F$ and $I$ are sets of models that predict that the outcome is true, false or indifferent respectively. Due to the complicated nature of this task it was implemented using Gringo/Clasp directly, not through XHAIL.

## 3 Results and Conclusions

The goal of our study was to evaluate whether the proposed system can be used in model development. To answer this question we supplied Huginn with initial models containing errors and run the development process to see whether the models would be improved. At this initial stage of evaluation the use of real biochemical experiments is not necessary, and would not be cost effective. Instead we ran simulations using reference models, which are fragments of the yeast consensus metabolic model 7.11 [1], between 14 and 54 activities in size. The knowledge bases containing the activities and entities for model development were created by mixing elements from a given reference model with additional, erroneous elements the role of which is to make the development process harder. The initial models were created by randomly selecting a set of activities from these knowledge bases.

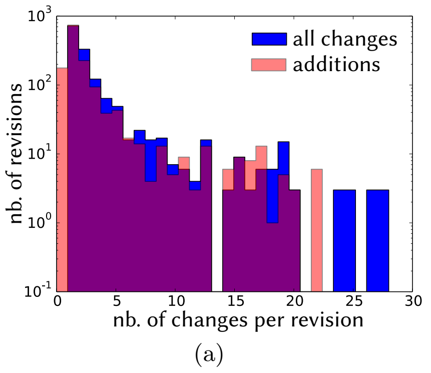
The improvement of models consisted of removing and adding activities so that working models resemble the reference model. To quantify the difference between a model and the reference model we use the symmetric difference between the sets of activities involved in the models.

The results of our simulations show that Huginn can successfully improve initial models, with an average reduction in initial error of 76%. For the smaller test-cases the working models tended to quickly become empirically equivalent and attempts to recover from it through generation of randomised models would fail. For the larger test-cases, Huginn tended to continue development until time-out. The final unsuccessful attempt to construct new models was not associated with any quality change[2] (p=0.13). However, since in many cases constructing random models allowed Huginn to recover and continue development process, including this ability was beneficial[2] (p=0.0003).

One of the significant differences between Adam and Huginn is Huginn's ability to use more types of experiments. To test whether this difference is beneficial we ran additional simulations while limiting available experiment types to only *two-factor growth* experiments. The results show that using more experiments is associated with larger improvements[3] (p=0.047). The main experiment types used by Huginn were *two-factor growth* (48% of experiments) and *entity detection* (51%). A portion of the latter involved multiple interventions: gene deletions and medium manipulations (19%). In the most extreme case an experiment would use 4 gene deletions on top of manipulating the medium composition. While execution of such experiment in practice would be challenging at best, it shows that ALP techniques used in Huginn can successfully cope with complex experiment design problems. The rest of the experiments used in development were *entity localisation* experiments.

Another significant difference between Adam and Huginn is in their revision abilities. Adam can only add individual missing expression activities. While, thanks to XHAIL, Huginn can handle a wider range of activities, also remove

[2] Tested using pair-wise comparison of improvement and then a binomial test
[3] Tested using paired, one-tailed t-test



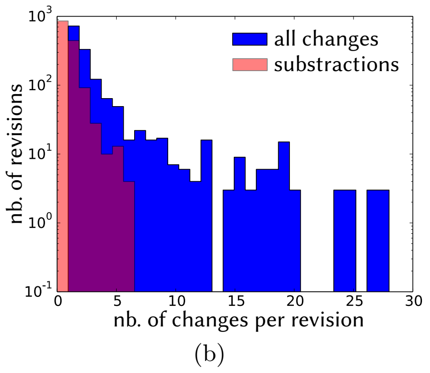


**Fig. 4.** Size of model revisions: each model revision may consist of multiple changes (additions or substractions of activities). The histograms compare distributions of additions (a) and substractions (b) with all changes. Note the log scale on y axis.

them, and introduce multiple changes in one revision. Therefore, it should be able to make more substantial changes to MNM structures. Our simulations show that it is indeed the case: 50% of the revisions involved more than one change (addition/substraction), while the largest involved as many as 28 changes (fig. 4). Many of these revisions combined the addition and substraction of activities (29%). The majority of revisions (60%) involved changing elements other than expression activities. These results show that Huginn takes advantage of its enhanced revision abilities to introduce larger changes to the models, and is therefore capable of solving wider range of biochemical problems than Adam – not only the problem of orphan enzymes, but also other structural problems in the metabolic networks.

We conclude that Huginn qualitatively improves on Adam by using more types of experiment, and a more versatile revision method, and that these improvements translate into an increased ability to correct models. We also conclude that the presented experiment design solution can not only design useful experiments, but also handle complicated tasks that require multiple interventions. More extensive *in silico* tests are still needed to test Huginn's performance in different configurations and under different circumstances. For example, we have not yet tested Huginn's ability to handle inconsistencies in results (*e.g.* introduced by experimental errors).

## 4 Related work

Thagard demonstrated the use of various types of abduction in hypothesis formation using an AI system called PI. [18] Here, we used "simple abduction" to revise refuted models.

Substantial advancements have been done in the field of computational discovery. Langley *et al.* [12] describes BACON, DALTON, GLAUBER, and STAHL

– seminal systems designed to model historical discoveries of quantitative and qualitative laws.

Džeroski and Todorovski [7] described QMN and LAGRANGE – systems for discovering quantitative and qualitative laws governing dynamical systems. Schmidt and Lipson [17] developed a system for discovering non-trivial conservation laws from experimental data. Todorovski *et al.* [19] developed HIPM, a system for developing complex hierarchical models of dynamical systems using induction, while taking advantage of expert knowledge. Compared to these studies we have focussed on qualitative aspects of scientific discovery, which can provide necessary insight into functioning of biological systems in terms of mechanistic explanations. However, methods for developing quantitative models are likely to be useful in further steps of building biological models.

Valdés-Pérez created MECHEM, a system for proposing possible intermediate steps of chemical transformations. The system uses information about chemicals' composition and structure to constrain the search-space as well as divide-and-conquer and Ockham's razor heuristics to make the search more efficient. An interesting feature of the system is its ability to propose new reactants. [20] Compared to MECHEM, Huginn focuses on developing larger models of metabolism from pre-defined reactions and on using biological experiments to gradually constraint the search-space.

Langley [13] summarises the lessons learned from their experience with developing computational tools for scientific discovery. They advise one to use the scientists' representations and their knowledge; tools should not just summarise, but provide explanations. Our approach follows these lessons. Representation of metabolism used by Huginn is taken from biochemistry, ensuring that it is easily understandable by biologists. Huginn records all produced models and results so checking why particular models were produced is possible.

## Acknowledgment

This work is supported by an EPSRC-EU Doctoral Training Award and the Faculty Engineering and Physical Sciences of the University of Manchester.

## References

[1] H. W. Aung, S. A. Henry, and L. P. Walker. Revising the representation of fatty acid, glycerolipid, and glycerophospholipid metabolism in the consensus model of yeast metabolism. *Industrial Biotechnology*, 9(4):215–228, 2013.

[2] W. Bechtel and R. C. Richardson. *Discovering complexity: Decomposition and localization as strategies in scientific research.* MIT Press, 2010.

[3] G. Collet, D. Eveillard, M. Gebser, S. Prigent, T. Schaub, A. Siegel, and S. Thiele. Extending the metabolic network of ectocarpus siliculosus using answer set programming. In *Logic Programming and Nonmonotonic Reasoning*, pages 245–256. Springer, 2013.

[4] C. Craver and L. Darden. Discovering mechanisms in neurobiology. *Theory and method in the neurosciences*, pages 112–137, 2001.

[5] C. F. Craver and L. Darden. *In search of mechanisms: Discoveries across the life sciences.* University of Chicago Press, 2013.

[6] L. Darden. *Reasoning in biological discoveries.* Cambridge University Press, 2006.

[7] S. Džeroski and L. Todorovski. Discovering dynamics: from inductive logic programming to machine discovery. *Journal of Intelligent Information Systems*, 4(1):89–108, 1995.

[8] M. Gebser, B. Kaufmann, A. Neumann, and T. Schaub. clasp: A conflict-driven answer set solver. In *Logic Programming and Nonmonotonic Reasoning*, pages 260–265. Springer, 2007.

[9] M. Gebser, T. Schaub, and S. Thiele. Gringo: A new grounder for answer set programming. In *Logic Programming and Nonmonotonic Reasoning*, pages 266–271. Springer, 2007.

[10] R. King, J. Rowland, S. Oliver, M. Young, W. Aubrey, E. Byrne, M. Liakata, M. Markham, P. Pir, L. Soldatova, et al. The automation of science. *Science*, 324(5923):85–89, 2009.

[11] R. D. King, K. E. Whelan, F. M. Jones, P. G. Reiser, C. H. Bryant, S. H. Muggleton, D. B. Kell, and S. G. Oliver. Functional genomic hypothesis generation and experimentation by a robot scientist. *Nature*, 427(6971):247–252, 2004.

[12] P. Langley. *Scientific discovery: Computational explorations of the creative processes.* MIT press, 1987.

[13] P. Langley. Lessons for the computational discovery of scientific knowledge. In *Proceedings of First International Workshop on Data Mining Lessons Learned*, pages 9–12. University of New South Wales, 2002.

[14] P. Machamer, L. Darden, and C. F. Craver. Thinking about mechanisms. *Philosophy of science*, pages 1–25, 2000.

[15] O. Ray. Nonmonotonic abductive inductive learning. *Journal of Applied Logic*, 7(3):329–340, 2009.

[16] O. Ray, K. Whelan, and R. King. Automatic revision of metabolic networks through logical analysis of experimental data. In *Inductive Logic Programming*, pages 194–201. Springer, 2010.

[17] M. Schmidt and H. Lipson. Distilling free-form natural laws from experimental data. *Science*, 324(5923):81–85, 2009.

[18] P. Thagard. *Computational philosophy of science.* MIT press, 1993.

[19] L. Todorovski, W. Bridewell, O. Shiran, and P. Langley. Inducing hierarchical process models in dynamic domains. In *Proceedings of The National Conference on Artificial Intelligence*, volume 20, page 892. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999, 2005.

[20] R. E. Valdés-Pérez. Machine discovery in chemistry: New results. *Artificial Intelligence*, 74(1):191–201, 1995.