Linköping Electronic Articles in
Computer and Information Science
Vol. 6(2001): nr 12

Combining Inductive Logic Programming, Active Learning and Robotics to Discover the Function of Genes

C.H. Bryant

S.H. Muggleton

S.G. Oliver

D.B. Kell

P. Reiser

R.D. King

Linköping University Electronic Press
Linköping, Sweden

http://www.ep.liu.se/ea/cis/2001/012/

Published on November, 2001 by
Linköping University Electronic Press
581 83 Linköping, Sweden

Linköping Electronic Articles in
Computer and Information Science
ISSN 1401-9841
Series editor: Erik Sandewall

Typeset by the authors using LaTeX
Formatted using étendu *style*

Recommended citation:
<Authors>. <Title>. Linköping Electronic Articles in
Computer and Information Science, Vol. 6(2001): nr 12.
http://www.ep.liu.se/ea/cis/2001/012/. *November, 2001.*

This URL will also contain a link to the authors' home pages.

Affiliations

C.H. Bryant[1] S.H. Muggleton[2]
Department of Computer Science, University of York, Heslington, York, YO10 5DD, UK.

S.G. Oliver
University of Manchester, Oxford Road, Manchester, M13 9PT, UK.

D.B. Kell
Institute of Biological Sciences, The University of Wales Aberystwyth, Penglais, Aberystwyth, Ceredigion SY23 3DB, Wales, UK.

P. Reiser R.D. King
Department of Computer Science, The University of Wales Aberystwyth, Penglais, Aberystwyth, Ceredigion SY23 3DB, Wales, UK.

[1]Current address: School of Computing, St Andrew St, The Robert Gordon University, Aberdeen, AB25 1HG, Scotland, UK.

[2]Current address: Department of Computing, Imperial College of Science, Technology and Medicine, 180 Queen's Gate, London SW7 2BZ.

Abstract

The paper is addressed to AI workers with an interest in biomolecular genetics and also to biomolecular geneticists interested in what AI tools may do for them. The authors are engaged in a collaborative enterprise aimed at partially automating some aspects of scientific work. These aspects include the processes of forming hypotheses, devising trials to discriminate between these competing hypotheses, physically performing these trials and then using the results of these trials to converge upon an accurate hypothesis. As a potential component of the reasoning carried out by an "artificial scientist" this paper describes ASE-Progol, an Active Learning system which uses Inductive Logic Programming to construct hypothesised first-order theories and uses a CART-like algorithm to select trials for eliminating ILP derived hypotheses. In simulated yeast growth tests ASE-Progol was used to rediscover how genes participate in the aromatic amino acid pathway of Saccharomyces cerevisiae. The cost of the chemicals consumed in converging upon a hypothesis with an accuracy of around 88% was reduced by five orders of magnitude when trials were selected by ASE-Progol rather than being sampled at random. While the naive strategy of always choosing the cheapest trial from the set of candidate trials led to lower cumulative costs than ASE-Progol, both the naive strategy and the random strategy took significantly longer to converge upon a final hypothesis than ASE-Progol. For example to reach an accuracy of 80%, ASE-Progol required 4 days while random sampling required 6 days and the naive strategy required 10 days.

1 Introduction

As stated in the abstract this paper is addressed both to AI workers and to biomolecular geneticists.

One way in which scientists work is that they form hypotheses which could explain some phenomenon of interest. They devise and perform trials to discriminate between these competing hypotheses and then they use the results of these trials to converge upon an accurate hypothesis. Our long term project aims to partially automate these aspects of scientific knowledge discovery by combining Inductive Logic Programming (ILP) [36], Active Learning [10, 29, 20] and Robotics. In this paper one particular piece of the process is studied in computer simulation.

Typically Machine Learning systems that produce human-comprehensible hypotheses from data assume that the data exists *a priori* and are not concerned with the collection of that data. In contrast, Active Learning systems do actively select trials to discriminate between contending hypotheses but most do not produce human-comprehensible hypotheses. We have developed ASE-Progol (**A**ctive **S**election of **E**xperiments with Progol). ASE-Progol is an Active Learning system which uses ILP to construct hypothesised first-order theories, uses a CART-like algorithm to select trials for eliminating ILP derived hypotheses and instructs a robot to physically perform these trials. The following is an example of the kind of hypothesis which derives in this study.

> The Open Reading Frame (ORF) `'YDR254W'` codes for an enzyme which catalyses a reaction generating the product `'C00074'` from the reactant `'C00631'` or vice versa.

Eventually a robot will automatically be instructed to perform these trials.

This paper takes a step towards answering the following question.

> Is the cost of converging upon an accurate hypothesis significantly reduced if trials are selected by ASE-Progol rather than if they are:
>
> 1. sampled at random (without replacement);
> 2. selected using the naive strategy of always choosing the cheapest trial from the set of remaining candidate trials.

We do not compare the cost of converging upon an accurate hypothesis when trials are selected by ASE-Progol with the cost of the optimal choice of trials because there is no guarantee that an optimal solution can be found efficiently (see Section 2.1). It would be interesting to compare the performance of ASE-Progol with that of domain experts. However the pathways of yeast amino acid metabolism which we use for our trials were intensively studied around fifty years ago. This makes them a well-characterised area for carrying out our investigation, though not one in which we have available detailed records of expert choice of trials. Hence the only practical options are comparison with 1) random sampling 2) the naive strategy of always choosing the cheapest trial from the set of candidate trials. This paper defines "cheapness" as the cost of chemicals required to conduct a given trial, disregarding differential labour costs.

The design and implementation of ASE-Progol is described in Section 2. Section 3 describes how we propose to record the cost of converging upon an accurate hypothesis. Section 4 introduces an application domain, Functional Genomics, together with an approach to it known as auxotrophic mutant trials, and justifies our choice of micro-organism, *Saccharomyces cerevisiae.* Section 5 describes the model we use to simulate auxotrophic mutant trials. Section 6 takes a step towards answering the question posed above by applying ASE-Progol to this model. Section 7 is the Discussion, Section 8 is the Conclusion and future work is proposed in Section 9. Appendix A justifies one aspect of the design of ASE-Progol. Appendix B lists a mapping of ORFs to reactions. Appendix C describes how auxotrophic mutant trials will be performed automatically in vitro by a standard laboratory robot. Appendix D describes related work.

2 ASE-Progol

Definition 1 (Trial) *A trial makes a binary partition of the candidate hypotheses and has an associated cost which is incurred when the trial is performed.*

Definition 2 (Experiment) *An experiment is a series of trials.*

2.1 Active Learning

The ASE-Progol approach is related to Active Learning [10, 29, 20] and Robotics. Active Learning is described in [10] as follows.

> .. active learning studies the closed-loop phenomenon of a learner selecting actions or making queries that influence what data are added to its training set. Examples include selecting joint angles or torques to learn the kinematics or dynamics of a robot arm, selecting locations for sensor measurements to identify and locate buried hazardous wastes, or querying a human expert to classify an unknown word in a natural language understanding problem.

In the present paper we consider a form of Active Learning in which a series of trials is chosen to minimise the expected cost of eliminating all but one of the hypotheses.

2.1.1 An Optimal but Computationally Expensive Solution

In the early 1970's Fedorov [15] introduced a theory which provides an optimal solution to this problem. (Note that Fedorov's theory of optimal experiments aims at directing choices for sequences of experiments. This should not be confused, for instance, with Sequential Analysis [55], which is aimed at minimising the number of randomly chosen experiments required to distinguish hypotheses.) However, Fedorov's approach is NP-hard in the general setting of minimising the expected experimental cost associated with discriminating among hypotheses. This is illustrated by the table and associated decision tree in Fig.1. Here the problem of finding the minimum cost sequence of trials $e_1, .., e_m$ which discriminates among the hypotheses $H_1, .., H_n$ is translated to the problem of finding a binary decision tree with minimum expected cost. It is known that this problem is NP-hard; for a formal proof see [21]. Although particular instances of NP-hard problems are computationally tractable (e.g. the travelling salesman solution for a particular small graph G), there is no guarantee that an arbitrarily chosen instance will be (e.g. for a million node graph G').

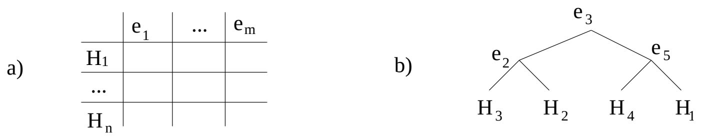

Figure 1: a) is a binary matrix relating trials $(e_1, .., e_m)$ to hypotheses $(H_1, .., H_n)$. Cell $\langle e_i, H_j \rangle$ has value 1 if the outcome of e_i is logically consistent with H_j and 0 otherwise. b) is an associated binary decision tree illustrating how hypotheses might be discriminated by the outcomes of trials.

2.1.2 A Near Optimal Solution

However, according to results by Muggleton and Page [35], within a Bayesian setting, the CART algorithm [3] gives a near optimal solution for this binary decision tree construction problem in time polynomial in m, n. Thus while CART is usually used in Machine Learning for constructing hypothesised propositional theories based on example data, ASE-Progol uses ILP to construct hypothesised first-order theories and uses a CART-like algorithm to select trials for eliminating ILP derived hypotheses.

2.2 How ASE-Progol Selects Trials

To understand how ASE-Progol selects trials, the reader must first comprehend 1) how the outcome of a trial is used to partition the set of hypotheses and 2) how ASE-Progol calculates the prior probability that a hypothesis is correct.

2.2.1 Partitioning the Set of Hypotheses

Let H be the set of candidate hypotheses and T be the set of candidate trials. Let t be the candidate trial selected by ASE-Progol. The outcome of the trial t is used to make a binary discrimination between competing hypotheses. Those hypotheses which are consistent with the outcome of the trial in question[3] form a subset, $H_{[t]}$, referred to as the *consistent hypotheses* while those which are inconsistent form a subset, $H_{[\bar{t}]}$, referred to as the *inconsistent hypotheses*. $\{H_{[t]}, H_{[\bar{t}]}\}$ is a partition of H.

2.2.2 The Prior Probability that a Hypothesis is Correct

Every candidate hypothesis has a prior probability of being the correct hypothesis. For any $h \in H$, let $p(h)$ be the prior probability that h is correct. ASE-Progol calculates a value for $p(h)$ from the compression of a hypothesis. Compression can be defined as follows [33].

$$Compression(h|E) = \log_2 p(h|E) - \log_2 p(h_E|E) = \log_2 \frac{p(h|E)}{p(h_E|E)}$$

where E is the set of examples from which the hypotheses were generated and h_E is the hypothesis equivalent to returning the data ungeneralised. Rearranging gives:

$$p(h|E) = 2^{Compression(h|E)} \times p(h_E|E)$$

[3]Consistency is determined by ASE-Progol from the metabolic pathway description.

As the set of candidate hypotheses computed by ASE-Progol is not complete, the probabilities are normalised. The normalised probability of a hypothesis, h_i, is

$$p\prime(h_i|E) = \frac{2^{Compression(h_i|E)}}{\sum_{h \in H} 2^{Compression(h|E)}}$$

2.2.3 The CProgol f measure of compression.

ASE-Progol takes the compression of a hypothesis to be the rounded value of the CProgol [33] f measure of compression. This is related to compression as follows.

$$Compression = size(E) - MDL_{h,E}$$

where E is the set of examples from which the hypotheses were generated and MDL is the Minimal Description Length [44].

$$MDL_{H,E} = min_{h \in H} size(h) + size(E|h)$$

The CProgol [33] f measure of compression is

$$f = |E^+| - \frac{|E^+|}{p}(l_1 + l_2 + n)$$

where $|E^+|$ is the number of examples in E which are positive, p is the number of positive examples covered by h, n is the number of negative examples covered by h, l_1 is the number of literals in the body of h and l_2 is an estimate of the number of further literals needed to complete h. l_2 is calculated by inspecting the output variables in h and determining whether they have been defined.

2.2.4 The Expected Cost of Experimentation

Let $EC(H, T)$ denote the minimum expected cost of experimentation given the set of candidate hypotheses H and the set of candidate trials T.

Definition 3 (EC)

$$EC(\emptyset, T) = 0$$

$$EC(\{h\}, T) = 0$$

$$EC(H, T) = min_{t \in T}\left[C_t + p(t)EC(H_{[t]}, T - t) + (1 - p(t))EC(H_{[\bar{t}]}, T - t)\right]$$

where C_t is the cost of the trial t and $p(t)$ is the probability that the outcome of the trial t is positive.

$p(t)$ can be computed as the sum of the probabilities of the hypotheses which are consistent with a positive outcome of t.

2.2.5 Entropy and The Expected Cost of Experimentation

The ASE-Progol approach to minimising the expected cost of experimentation relies on a basic result from information theory. Consider the problem of designing a code to transmit messages drawn at random, where the probability of encountering message i is $p(i)$. Shannon and Weaver [7] showed that in an optimal coding scheme, the length of the binary code for i is $-\log_2(p(i))$. If the bits of the code are interpreted as the outcomes of binary trials then the number of trials needed to eliminate all hypotheses except h is at most $\lceil -\log_2(p(h))\rceil$. Rounding is used because the number of trials must be a natural number. When the number of hypotheses is not a power of two, $-\log_2(p(h))$ will not be a natural number and the number of trials required to eliminate all but the correct hypothesis will depend upon which of the hypotheses is correct. The ceiling is used because the correct hypothesis may be one of those requiring the most trials to eliminate all the other candidate hypotheses.

The Shannon-Weaver Law states that, given symbols $a, b, c \ldots n$ which do not necessarily have an equal probability of occurrence, the average information, I_n, in bits per symbol is given by [22]:

$$I_n = -\sum_{i=1}^{n} p(i) \log_2(p(i))$$

In Information Theory the average information, I_n, is called entropy, by analogy with the entropy of physics and thermodynamics, which takes a similar mathematical form.

Theorem 1 (EC Bound) *Given hypothesis space H and trial set T, the value of $EC(H,T)$ is bounded above by*

$$\begin{aligned}(max_{t\in T}C_t) \quad &+ \quad p(t)(max_{t\prime\in(T-t)}C_{t\prime})\lceil\log_2|H_{[t]}|\rceil \\ &+ \quad (1-p(t))(max_{t\prime\in(T-t)}C_{t\prime})\lceil\log_2|H_{[\bar{t}]}|\rceil\end{aligned}$$

Proof
The minimal solution of the recurrence defined can be considered as represented by a binary tree with trial costs labeling the arcs. According to Shannon & Weaver the worst case occurs when the probabilities assigned to hypotheses in H are uniform. In this case, assume that for all $h \in H, p(h) = \frac{1}{|H|}$, and the tree has a maximum depth m. Clearly $m \le \lceil\log_2|H|\rceil$. Considering Definition 3 on page 4 we have:

$$\begin{aligned}EC(H,T) \quad &\le \quad \sum_{h\in H}\frac{(max_{t\in T}C_t)m}{|H|} \\ &= \frac{(max_{t\in T}C_t)\sum_{h\in H}\lceil\log_2|H|\rceil}{|H|} \\ &= (max_{t\in T}C_t)\lceil\log_2|H|\rceil\end{aligned}$$

Substituting this into the recurrence in Definition 3 on page 4 gives:

$$\begin{aligned}EC(H,T) \quad \le \quad &(max_{t\in T}C_t) + p(t)(max_{t\prime\in(T-t)}C_{t\prime})\lceil\log_2|H_{[t]}|\rceil \\ &+(1-p(t))(max_{t\prime\in(T-t)}C_{t\prime})\lceil\log_2|H_{[\bar{t}]}|\rceil\end{aligned}$$

□

As a heuristic function we use the following approximation.

$$\begin{aligned}EC(H,T) \approx min_{t\in T}[C_t \quad &+ \quad p(t)(mean_{t\prime\in(T-t)}C_{t\prime})J_{H_{[t]}} \\ &+ \quad (1-p(t))(mean_{t\prime\in(T-t)}C_{t\prime})J_{H_{[\bar{t}]}}]\end{aligned}$$

where

$$J_H = -\sum_{h\in H} p(h)\lfloor\log_2(p(h))\rfloor$$

Given that:

$$C_t = (max_{t' \in T} C_{t'}) - x, \qquad \text{where} \quad x \geq 0$$

$$mean_{t \in (T-t)} C_t = (max_{t' \in (T-t)} C_{t'}) - y, \qquad \text{where} \quad y \geq 0$$

$$J_H = \lceil \log_2 |H| \rceil - z, \qquad \text{where} \quad z \geq 0$$

it follows that this heuristic function has the upper bound given in Theorem 1 on page 5.

2.3 ASE-Progol Implementation

ASE-Progol has been implemented as a unix shell script which calls CProgol [33] at different points in its execution in order to perform deduction, induction and, when required, abduction.

During a single execution, ASE-Progol completes one experiment. Trials are represented by ground unit clauses. Examples are positive and negative instances of the predicate representing a trial. Hypotheses are represented by single Prolog clauses.

ASE-Progol may be supplied with domain specific knowledge encoded as a Prolog program. That part of this program which does not change during an execution of ASE-Progol will be referred to as *static knowledge* during the remainder of this paper. (This is to distinguish it from the dynamic hypothesis-set and example-set.) The static knowledge must allow the cost of any given trial to be determined (see Section 5.3).

2.3.1 ASE-Progol Modes and Options

ASE-Progol can operate in one of three modes, namely Unaided, Head-start or Fast-forward. Head-start mode allows ASE-Progol to utilise a set of laboratory results obtained without the use of ASE-Progol. In Unaided mode there is no such set of results. Fast-forward mode allows ASE-Progol to utilise the results from a previous execution. In fast-forward mode execution recommences at a specified iteration of the learning cycle. ASE-Progol enters the loop at the point where a new example has just been created. Fast-forward mode requires a trace of the examples and hypotheses generated during a previous execution.

ASE-Progol can be instructed to select trials 1) at random or 2) naively by choosing the cheapest trial from the set of candidate trials. Normally ASE-Progol will instruct the robot to perform the trial which minimises the expected cost of experimentation (see Section 2.2). The random and naive options can be used to obtain baselines against which the normal performance of ASE-Progol can be measured.

ASE-Progol allows the User to specify a limit on both the experimental resources which may be consumed by ASE-Progol and the number of trials which may be performed.

2.3.2 ASE-Progol Termination Conditions

The execution of ASE-Progol terminates when either the experimental resources have been exhausted, the number of trials performed exceeds a limit specified by the User or ASE-Progol is unable to suggest a novel trial.

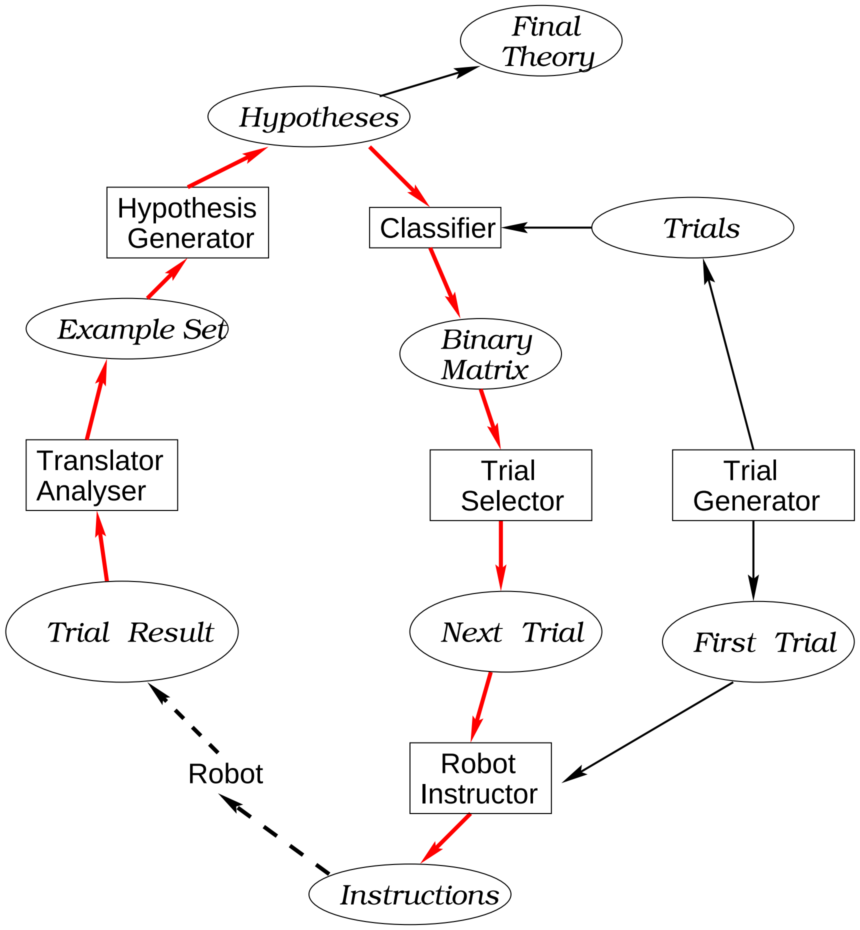

Figure 2: The boxes represent components of ASE-Progol and the ellipses represent information which is passed between these components. Joined lines indicate flow of information within ASE-Progol. Dotted lines show how information flows between ASE-Progol and the laboratory. To date, all of the components have been implemented except for the Robot Instructor and the Translator and Analyser: we have yet to attempt to physically perform trials using robotics.

2.3.3 Components of ASE-Progol

Fig.2 shows the components of ASE-Progol and the information which is passed between them. These components are described below.

Trial Generator This component generates the trials which appear in a classification matrix of the form shown in Figure 1a on page 3.

The Trial Generator requires a Prolog program which can generate a set of candidate trials. The Prolog program must be provided by the User; this has the advantage that it may include encoded domain-specific static knowledge.

In Unaided mode, the execution of ASE-Progol begins with the Trial Generator selecting the first trial by generating a random sample[4] of 20 trials from the space of possible trials using the Prolog program and then selecting the cheapest trial in this sample.

On each subsequent iteration of the loop, the Trial Generator uses the Prolog program to generate a set of 20 candidate trials. It then removes from this list of 20 trials, any duplicate trials or trials which have already been performed by the robot. This leaves a set of candidate novel trials. Thus the maximum number of candidate trials that may appear in the classification matrix is 20.

If the Hypothesis Generator component has generated more than one hypothesis during the current iteration of the learning cycle then the Trial Generator passes the set of candidate novel trials to the Classifier component of ASE-Progol. If there are less than two candidate hypotheses, then the cheapest of candidate novel trials is chosen as the next trial to be performed. This trial is passed directly to the Robot Instructor component, bypassing the Classifier and Trial Selector components: it does not make sense to search for the trial which will best discriminate between hypotheses when there are less than two candidate hypotheses.

We justify our upper-bound on the number of candidate trials that may appear in a classification matrix as follows. The execution times of the Classifier component increases as the product of the number of candidate trials and the number of candidate hypotheses in the classification matrix increases. As the number of candidate hypotheses is not known prior to the execution of ASE-Progol, an upper bound must the placed on the number of candidate trials sampled using the program. However this upper bound must be sufficiently high to ensure that there is a low probability that the sample will not contain a novel trial. Appendix A proves that ASE-Progol's upper bound of 20 on the number of candidate trials ensures that this holds.

Classifier This component determines (and records) whether the outcome of each trial, together with the static knowledge, is consistent with each hypothesis and thus generates a matrix of the form shown in Fig.1a on page 3.

Trial Selector This computes the expected cost of experimentation for each candidate trial as described in Section 2.2. The trial which minimises the expected cost of experimentation is selected as the next to be performed by the robot.

[4]Appendix A explains why 20 candidate trials are generated

Hypothesis Generator A fresh set of candidate hypotheses is generated on each iteration of the loop. CProgol is passed a file in batch mode which includes the usual Progol mode and type declarations, integrity constraints etc. and which consults the current set of examples and the static knowledge. The set of candidate hypotheses comprises all of the compressive hypotheses which CProgol [33] generates.

Using CProgol to generate hypotheses is advantageous because it:- a) performs a complete search of a User-defined hypothesis space, and is capable of returning all hypotheses within the space that meet User-defined criteria; b) attaches a posterior probability estimation to each hypothesis, given the data; in fact, Progol's search is driven by these probability estimates.

Robot Instructor This will instruct the robot to perform the next trial.

Translator and Analyser This will create an example from the result of the trial and add it to the current example-set. Thus the cardinality of the example-set is incremented on each iteration of the loop.

To date, all of the above components have been implemented except for the Robot Instructor and the Translator and Analyser: we have yet to attempt to physically perform trials using robotics. Therefore when the current version of ASE-Progol seeks the outcome of an trial, it consults an oracle which contains the outcome for every possible trial in a given domain.

3 Recording Cost Differences

To the best of our knowledge, nearly all of the performance curves which have been used to present previous results in Active Learning have been plots of predictive accuracy against the number of training examples [47, 12, 6, 9, 10, 8]. Often two curves are plotted on the same graph; one for Active Learning and one for random sampling. Such plots allow the difference in the number of examples required to reach a particular level of performance to be seen.

However one drawback of such plots is that they ignore any variation in the cost of obtaining individual examples. When such variation does exist and the aim is to compare the cost of attaining particular levels of performance, these plots are potentially misleading.

To overcome this drawback, we use a novel form of performance curve, namely a plot of the cumulative cost of the training examples against performance. (This is related to the form of the performance curves used by Fox et al. in which performance is plotted against a particular resource, namely time [17].)

Performance should be measured as the average performance of the hypotheses generated during a particular iteration of the learning cycle. This is the correct performance measure because it rewards learners which discriminate between competing hypotheses. Another obvious option would be to the select the hypothesis with the highest compression from the set of hypotheses generated during a particular iteration of the learning cycle and to measure the accuracy of this hypothesis. However such an approach would fail to reward learners which eliminate incorrect hypotheses.

4 The Application Domain: Functional Genomics

The focus of genome research has moved to the problem of identifying the biological functions of genes. This is known as *Functional Genomics*. The problem is important because nothing is known about the function of between 30-60% of all new genes identified from sequencing [13, 18, 40]. Functional Genomics is recognised as central to a deeper understanding of biology, and the future exploitation of biology in medicine, agriculture, and biotechnology in general. Functional Genomics is an appropriate application to test ASE-Progol because: a) relational representations are appropriate for the problem; b) Progol has been successful in related types of problems previously [34, 48, 49, 26, 46, 25, 16, 52, 53] ; c) the problem is important; d) automation is essential to the application; e) robotic technology is already commonly used in genomics [45, 51, 2, 42].

4.1 The Function of a Gene

The term 'function of a gene' does not have a widely-accepted precise definition because of the multilevel and multifactorial interactions that accompany changes in the expression of virtually *any* gene. Rieger *et al.* [43] state that the notion of the function of a gene is of necessity ambiguous and that it should be defined across various levels of analysis: physiological role, participation in cellular processes and biochemical pathways, underlying molecular mechanics etc. In the next section, we describe our approach to Functional Genomics. Within the confines of this approach, it is sufficient to define the function of a gene to be how a gene participates in a biochemical pathway.

4.2 Metabolism and Growth Trials

One approach to Functional Genomics involves auxotrophic mutant experiments. These involve feeding a micro-organism different mixtures of nutrients known as *growth media* and measuring the resulting growth. Cells of the micro-organism import molecules from a growth medium and convert them to molecules which are essential for growth via pathways of chemical reactions known as *metabolic pathways*. Each chemical reaction is catalysed by an enzyme. Some of these enzymes are known and others are not. The genes which code for the latter are genes whose function is not known.

To find out what the function of a gene is, mutants of a particular micro-organism are grown which do not contain the gene in question. The process of removing a single gene from a micro-organism is referred to as *single-gene-deletion*. The effect of not having the gene can then be compared with a control i.e. the wild form of the organism which does not have the mutation. To make the effect of the mutation observable it is necessary to feed samples of the mutant with different growth media and compare the resulting growth rates with that of the wild strain. A 'chemically defined medium' is a medium whose precise chemical composition is known. Such media are prepared by adding precise amounts of highly purified inorganic or organic metabolites to distilled water [4].

4.3 Brewers' and Bakers' Yeast

Ultimately, ASE-Progol will instruct a robot to perform auxotrophic mutant experiments on a model organism, namely *Saccharomyces cerevisiae* i.e. brewers' and bakers' yeast. Yeast has many features that "commend it as a system with which to pioneer a systematic approach to screening for the function of novel genes" [40]. The investigator can control its physiology to some extent because it is unicellular and can grow on chemically defined media. Most importantly, excellent tools exist for its genetic manipulation, permitting the precise deletion of genes under investigation.

The genome of yeast has many homologies to human genes. Thus the systematic sequencing of yeast has opened the door to the identification of the basic biological mechanisms common to all eukaryotes, including humans, which are not accessible through classic approaches. The function of the genes in yeast will provide a unique tool with which to search for drugs to treat not only infectious diseases but also major killers such as cancer [40].

Yeast is being used as a test-bed for the development of a variety of functional genomics technologies which work at various levels of analysis including the genome and metabolic pathways. (A review of these is beyond the scope of this paper; for a recent review see [11].) One argument for tackling functional genomics at the level of the metabolic pathways, is that the number of metabolites is an order of magnitude lower than the number of genes. The completion of the sequencing of the yeast genome revealed that there are 6000 protein encoding genes [18]. There are some 563 low-molecular weight intermediates on the metabolic map that are relevant to yeast metabolism [5].

5 Model of the Aromatic Amino Acid Pathway of Yeast

As we have yet to physically perform trials using robotics, in the experiment described in this paper (see Section 6) we simulate single-gene-deletion growth trials using a model of the aromatic amino acid pathway of yeast. For an introduction to this pathway see a standard text on biochemistry such as [30].

The aromatic amino acids are essential amino acids i.e. amino acids which humans must obtain from dietary sources. However microorganisms, such as yeast, and plants can synthesise them using the aromatic amino acid pathway which is never found in animals. The pathway is therefore an important target for herbicides, antibiotics and live vaccines [27].

The model uses unique identifiers from the literature to refer to metabolites, enzymes and Open Reading Frames (ORFs). ORFs are putative genes: some ORFs may not code for anything. Metabolites are referred to using their accession numbers in the LIGAND database in the Kyoto Encyclopaedia of Genes and Genomes (KEGG) [19]; enzymes by their Enzyme Commission classification numbers [38]; and ORFs by the systematic name used by the Munich Information Center for Protein Sequences (MIPS-GSF) [31].

A complete representation of all of the known steps in this pathway is shown in Figure 3. The model assumes that the enzymes catalyse reactions in both directions. Note that there are differences between our model and the corresponding KEGG map (see
`http://www.genome.ad.jp/dbget-bin/get_pathway?org_name=sce&mapno=00400`)

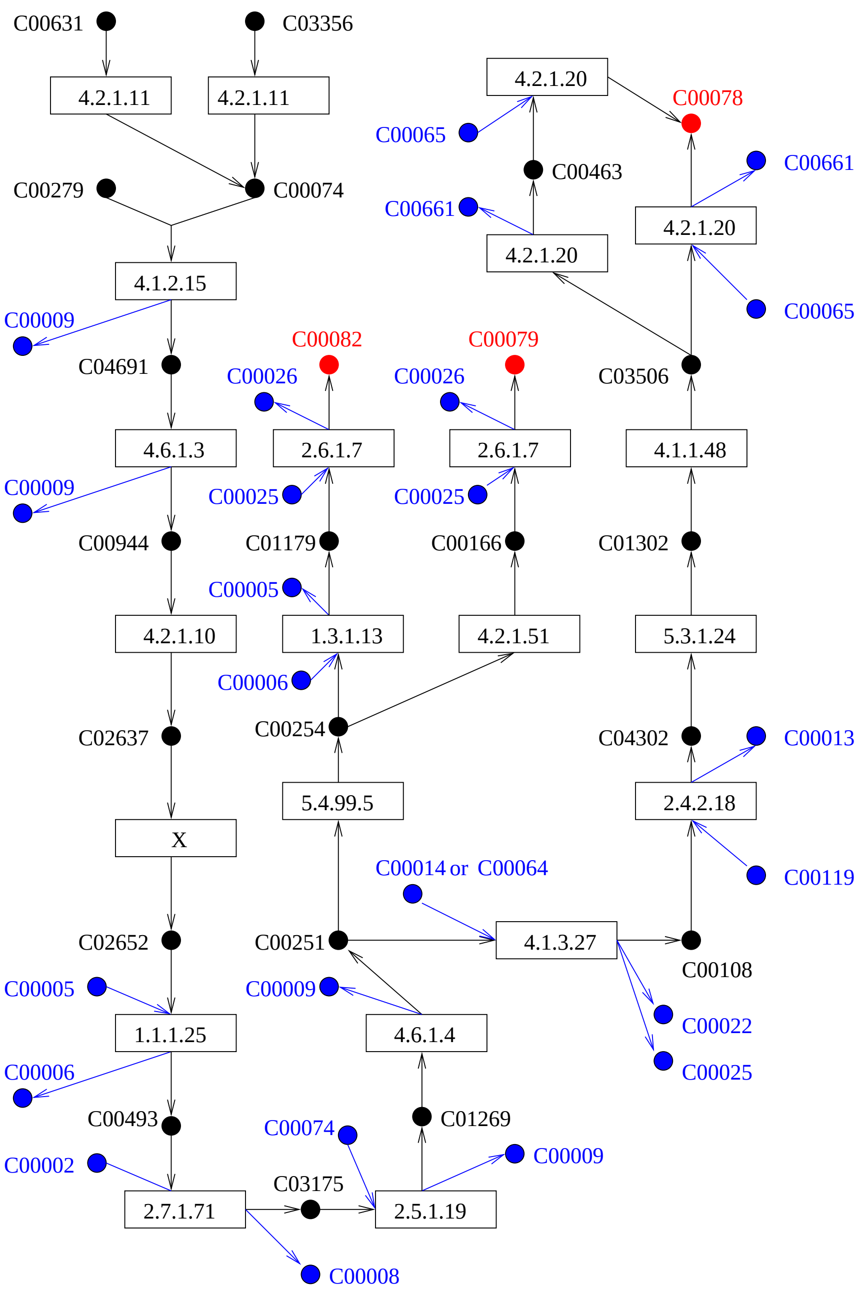

Figure 3: The aromatic amino acid pathway of yeast. The circles represent metabolites, referred to by their KEGG [19] accession numbers. The pathway pertains to the biosynthesis of the aromatic amino acids phenylalanine, tyrosine and tryptophan which are shown in red. The main reaction paths are shown in black. Molecules which are involved in the reactions but which do not lie on the main pathway are shown in blue. Each rectangle, together with those metabolites which are linked to it, represents a chemical reaction. Reactants are shown entering on one side of each rectangle and products leaving on the other. Each rectangle is labelled by the EC-number [38] of the enzyme which catalyses the reaction. For clarity arrows are included pointing to intermediate and end products.

because there were errors and missing data in the KEGG map.

Although Figure 3 uses arrows to indicate intermediate and end products, our logical model assumes that the enzymes catalyse reactions in both directions. The gene deletions in our experiments can result, in some cases, in local reversal of the normal direction of reactions. It is necessary for the logical model to capture this.

The enzyme which connects metabolites C02652 and C02637 is labelled as 'X' rather than by an Enzyme Commission classification number. These metabolites are stereoisomers i.e. molecules which have the same molecular and structural formulae but have different configurations [54]. We do not know what class of enzyme would cause such a conformational change and thus we label this reaction 'X'.

Appendix B lists the many-to-many mapping of ORFs to reactions in our model. Some ORFs are associated with more than one reaction.

5.1 Metabolites

We place metabolites in the model in the following mutually exclusive categories:-

Optional These are the metabolites which may, if desired, be added to a growth medium and, if they are added, be expected to enter the cell. Not all metabolites can be added to a growth medium. Some cannot be used because they are not chemically stable. Other are not available in a given laboratory or from the suppliers of that laboratory. Some metabolites can be added to a growth medium but can never have an effect because the cell is impermeable to them. When the model was prepared[5] (January 2001), the metabolites which could be added to a growth medium were those listed in Table 1.

Ubiquitous - These are metabolites which are never included in growth media as they are assumed to be ubiquitous in the cell. The metabolites in the model which were assumed to be ubiquitous are: 'C00002', 'C00005', 'C00006', 'C00014', 'C00025', 'C00064', 'C00065', 'C00074', 'C00119', 'C00279', 'C00631' and 'C03356'. These are the reactants for:

- the first step of the pathway;
- each step which are not provided by the previous steps in the pathway.

We can assume that compounds which are not provided by the previous steps in the pathway are ubiquitous because, in our proposed experimental procedure for automating growth trials using robotics (see Appendix C.1), all growth media will contain all the compounds needed for growth in the wild strain of yeast due to the presence of SGD.

Other All the metabolites which are not categorised as either Optional or Ubiquitous are placed in this category.

The model assumes that:-

[5] In reality membership of this category can change over time as stocks of particular metabolites either become available or are exhausted. The model does not represent this temporal characteristic of the data.

1. prior to metabolism, the only metabolites which exist in the cell are those in the category Ubiquitous and those in the growth medium;
2. a reaction will only proceed if all the reactants exist within the cell beforehand;
3. the only result of a reaction is that all of the products are added to the molecules which exist in the cell.[6]

5.2 Growth media

In general a growth medium is a combination of growth nutrients. If there are n optional nutrients and no upper bound is placed on the number of nutrients which may be added to a medium then the number of possible media is equal to the size of the power set of the optimal nutrients i.e. 2^n. Thus there are $2^{13} = 8192$ combinations of the 13 optional nutrients.

However, to make our preliminary experiments with the robot tractable, we plan not to add more than a few extra metabolites to a basal growth medium. Therefore our model includes an upper bound of three on the number of nutrients which may be added to the basal medium. If there are 13 optional nutrients and the number of optional nutrients which may be added to a basal medium is in the range 0 - 3 then the number of possible media = $\sum_{i=0}^{3} \binom{13}{i} = 378$.

5.3 The cost of growth trials

The cost of a trial is taken to be the sum of the relative costs of the metabolites used in that trial. The relative cost of a metabolite is the cost of that metabolite relative to the cheapest metabolite rounded to the nearest integer. Some metabolites can be purchased in packs, some individually by weight and some by either method. The cost of those metabolites which are sold individually was taken to be the cost per gramme. The cost of a metabolite sold in a pack was taken to be the cost of the pack divided by the number of metabolites included in the pack. For those metabolites which can be purchased by more than one method, the lowest price for each metabolite was used. The data was taken from the SIGMA catalogue (`http://www.sigmaaldrich.com/`) during January 2001. The costs of the optional metabolites are listed in Table 1.

The model assumes that:-

1. both the wild and mutant strains are freely available.[7]
2. the presence of SGD (see Appendix C.1) in the growth medium does not incur a cost because the stock of each ORF is stored in SGD.
3. the cost of the energy consumed by the laboratory equipment during a trial is negligible.
4. the cost of the labour required to perform the trial is negligible: our ultimate aim is to automate auxotrophic mutant experiments as far as is possible.

[6]The law of conservation of mass is not contravened: the model assumes that the reactants are constantly replenished by further metabolism.

[7]A large European research network called EUROFAN [39] was established. The first aim of EUROFAN and parallel activities in Canada, Japan and the USA was the production of a complete set of 6000 single-gene-deletion mutants covering all the ORFs revealed by the complete yeast genome sequence [41].

Table 1: Cost of metabolites. Three of the metabolites (C00079, C00078, C00082) were only available as part of a pack of 21 metabolites. The cost of these metabolites was taken to be the cost of the pack, £88.25, divided by the number of metabolites included in the pack. None of the other ten in the table were were available as part of a pack.

Metabolite	Price (£)	Mass	Cost (£)/gramme	Cost Relative
C00108	20.56	25g	0.82	10
C00251	223.36	0.25mg	893400.00	11168000
C00013	39.95	500g	0.08	1
C00463	15.16	1g	15.16	190
C00079	only available in a pack		4.20	53
C00078	only available in a pack		4.20	53
C00082	only available in a pack		4.20	53
C00026	23.85	100g	0.24	3
C00166	23.73	10g	2.37	30
C01179	15.63	1g	15.63	195
C00254	174.02	50mg	34800.00	435050
C00022	9.16	5g	1.83	23
C00493	50.64	1g	50.64	633

6 Experiment

6.1 Aim of the Experiment

The aim of the experiment is to take a step towards determining whether the cost of converging upon an accurate hypothesis is significantly reduced if trials are selected so as to minimise the cost of experimentation rather than if they are sampled at random (without replacement) or selected using the naive strategy of always choosing the cheapest trial from the set of novel candidate trials. The experiment uses the model described in Section 5 as a test-bed. Note that during this experiment we assume that growth is recorded as a binary "growth/no growth" variable.

6.2 Materials

6.2.1 Representation of Trials and Hypotheses

Hypotheses are represented by `codes(orf, enzyme_class, reactions, products)` facts. For example `codes('YDR254W','4.2.1.11',['C00631'],['C00074'])` represents the conjecture that the ORF `'YDR254W'` codes for an enzyme which catalyses the reaction which generates the product `'C00074'` from the reactant `'C00631'` or vice versa.[8]

Trials are represented by `phenotypic_effect(orf, growth_medium)` facts, where `orf` is the ORF whose function is to be learnt during the current execution of ASE-Progol and `growth_medium` is a growth medium as defined in Section 5.2. For example `phenotypic_effect('YBR166C',`

[8] The model assumes that the enzymes catalyse reactions in both directions.

Table 2: Algorithm to predict whether a phenotypic effect will be observed or not. (Note that during this experiment we assume that growth is recorded as a binary "growth/no growth" variable.)

---

1. Determine whether the three aromatic amino acids can be metabolised by the wild strain of yeast.

2. IF the wild strain cannot metabolise the aromatic amino acids THEN
   Go to step 3.
   ELSE
   REPEAT
   (a) Simulate the creation of a mutant strain of yeast by removing a reaction from the model.
   (b) Determine whether the three aromatic amino acids can be metabolised by the mutant strain of yeast.
   (c) IF the mutant strain cannot metabolise the aromatic amino acids THEN
   (i) Predict that a phenotypic effect will be observed.
   (ii) Go to step 4.
   ELSE
   Reintroduce the missing reaction.
   ENDIF

   UNTIL all reactions have been tried.

3. Predict that a phenotypic effect will *not* be observed.

4. EXIT

`['C00022'])` represents the trial in which the mutant strain of yeast created by removing the ORF `'YBR166C'` is fed the growth medium formed by adding the nutrient `'C00022'` to the basal medium (see Appendix C.1).

6.2.2 Static Knowledge

Table 3 lists part of the logic program which represents the static knowledge. This program uses the algorithm listed in Table 2. The program also contains:

- one `enzyme(enzyme_class, reactants, products)` fact for each reaction shown in Fig.3, where `reactants` and `products` are lists of metabolites;
- a definition of the predicate `cost(phenotypic_effect(ORF, Growth_medium), Cost)`

Table 3: Part of the logic program which represents the static knowledge. (Note that during this experiment we assume that growth is recorded as a binary "growth/no growth" variable.)

```
phenotypic_effect(ORF, Growth_medium):-
    generated_by_other_pathways(Ubiquitous_metabolites),
    union(Ubiquitous_metabolites, Growth_medium, Starts),
    connected(Starts, Wild_products),
    ends(Ends),
    subset(Wild_products, Ends),
    enz(Enzyme, Reactants, Products),
    encodes(ORF, Enzyme, Reactants, Products),
    connected_without_this_step(Starts, Mutant_products,
                                Enzyme, Reactants, Products),
    not(subset(Mutant_products, Ends)).

enz(Ec, R2, R1) :- enzyme(Ec, R1, R2).
enz(Ec, R1, R2) :- enzyme(Ec, R1, R2).

encodes(ORF, Enzyme, Reactants, Products):-
    codes(ORF, Enzyme, Reactants, Products).

encodes(ORF, Enzyme, Reactants, Products):-
    codes(ORF, Enzyme, Products, Reactants).

generated_by_other_pathways(['C00002', 'C00005', 'C00006',
'C00014', 'C00025', 'C00064', 'C00065', 'C00074', 'C00119',
'C00279', 'C00631', 'C03356']).

ends(['C00078', 'C00079', 'C00082']).
```

which reflects the part of the model described in Section 5.3.

6.2.3 Oracle

The static knowledge, together with `codes(orf, enzyme_class, reactants, products)` facts representing the complete and correct mapping of genes to reactions, were used as an Oracle which ASE-Progol consulted when it sought the outcome of a trial.

6.2.4 Generation of Trials

Recall from Section 2.3 that the Trial Generator component of ASE-Progol requires a Prolog program which can generate a set of candidate trials. In the experiment described here the Trial Generator calls a Logic Program which:-

1. generates a random sample of growth media from the population of those subsets of the set of optional nutrients whose size has an upper bound of three.
2. from this sample, creates a list of `phenotypic_effect(orf, growth_medium)` facts where `orf` is the ORF whose function is to be learnt during the current execution of ASE-Progol and `growth_medium` is a member of the sample.

6.2.5 The Example Set

Examples were represented by positive or negative instances of the predicate `phenotypic_effect(orf, growth_medium)`. For example `phenotypic_effect('YBR166C', ['C00022'])` represents the fact that a phenotypic effect is observed if a mutant strain of yeast created by removing the ORF `'YBR166C'` is fed the growth medium formed by adding the nutrient `'C00022'` to the basal medium.

There are 378 examples (Section 5.2) for each ORF in the model. The class of each example (positive or negative) was deduced from the complete model. Table 4 shows the distribution of positives and negatives for each ORF. There are no positive examples of the `phenotypic_effect/2` predicate for the ORFs `YDR254W`, `YER090W`, `YGL026C`, `YHR174W`, and `YMR323W`. Hence phenotypic effects can be observed for just 12 of the 17 ORFs in the model. Thus when conducting rediscovery experiments with ASE-Progol, only these 12 ORFs are used.

The example-set was used during training and testing as described below.

6.3 Method

Parts of the model were removed and the ability of ASE-Progol to efficiently recover the performance of the model was measured. This section is divided into subsections which describe:-

- how and why an ILP approach known as *Theory Completion Using Inverse Entailment* [37] was used to attempt to recover the removed parts of the model;
- the training and testing method;

Table 4: Distributions of positive and negative examples

ORF	Number of examples	
	Positive	Negative
YBR166C	232	146
YBR249C	70	308
YDR007W	232	146
YDR035W	70	308
YDR127W	104	274
YDR254W	0	378
YDR354W	232	146
YER090W	0	378
YGL026C	0	378
YGL148W	104	274
YGL202W	366	12
YHR137W	366	12
YHR174W	0	378
YKL211C	232	146
YMR323W	0	378
YNL316C	232	146
YPR060C	151	22

- how a comparison was made between the cost of converging upon an accurate hypothesis when ASE-Progol selects trials so as to minimise the cost of experimentation and the cost when 1) it samples them at random; 2) it adopts the naive strategy of always choosing the cheapest trial.

6.3.1 Theory Completion Using Inverse Entailment

The main real-world applications of ILP to date involve the "Observation Predicate Learning" (OPL) assumption [37], in which both the examples and hypotheses define the same predicate. In this experiment hypotheses (`codes/4` facts) are removed and then an attempt is made to recover them using a training set of examples of the observable predicate `phenotypic_effect/2`. The OPL assumption cannot be made when trying to regenerate the `codes/4` facts from the training data because `codes/4` is below `phenotypic_effect/2` in the calling diagram. Therefore an approach to non-OPL called Theory Completion using Inverse Entailment (TCIE) was used. TCIE is based on inverse entailment [33] and is implemented within Progol5.0[9]. Progol5.0 was instructed to complete the model using TCIE [37] as follows.

```
:- modeh(1, codes(#orf, #enzyme, #list_of_metabolites,
#list_of_metabolites))?
:- observable(phenotypic_effect/2)?
```

During training, Progol5.0 was prevented from generating pairs of `codes/4` facts whose only difference was that the values of the third and fourth were exchanged by placing the following constraint in the learning file.

[9]The C source code for Progol5.0 can be down-loaded from `ftp://ftp.cs.york.ac.uk/pub/ML_GROUP/progol5.0`.

```
:- hypothesis(codes(_, _, Reactants, Products), true, _),
   Reactants @< Products.
```

Such pairings are not needed due to the presence of the `encodes/4` predicate (see Table 3).

6.3.2 Training and Testing

Both training and testing were done for one ORF at a time. The test-set for any given ORF was the 378 examples of the observable predicate `phenotypic_effect/2` for the ORF in question. The performance measure used was the average of the predictive accuracies of the hypotheses generated during a particular iteration of the learning cycle.

The recovery in performance was recorded by plotting two types of curves:

1. the cost of the chemicals consumed by ASE-Progol during training versus performance during testing;
2. the duration of the training versus performance during testing.

The efficiency of the recovery was gauged by comparing the recovery when trials were selected so as to minimise the cost of experimentation (see Section 2.2) with the recovery when trials were:

- sampled at random (without replacement);
- selected using the naive strategy of always choosing the cheapest trial from the set of novel candidate trials.

Tables 5 and 6 summarise the experimental method. Training and testing were repeated ten times to allow subsequent calculation of the mean accuracy and standard error. During every execution of ASE-Progol the limit on the consumption of experimental resources was set to 100000000. The limit on the number of trials which may be performed was set to 10 during every execution except those where the naive strategy of always choosing the cheapest trial was used. In the latter case, the limit was increased from 10 to 50 because this led to an increase in the range of cumulative cost over which the three learning strategies could be compared: after 10 trials, the cost incurred by the naive strategy was only a small fraction of that incurred by the other two strategies.

The performance curve which shows the consumption of chemicals gives an estimate of the cost of the chemicals in £. The conversion from relative costs to actual cost of a trial in £ was based on the following:

1. the assumption that all nutrients would be used at a concentration of 50mg/litre;
2. the total volume in each well of the microtitre plate would be 750E-6 litres (see Appendix C);
3. three wells would be required for each growth trial because each trial would be performed in triplicate by the robot;
4. the cost of the cheapest nutrient in the set of optional nutrients was £0.08/gramme (see Section 5.3).

Hence actual costs in £ were obtained by multiplying relative costs by $3 \times \frac{50}{1000} \times \frac{750}{10^6} \times 0.08 = 9E-6$.

Table 5: Part A of experimental method.

```
remove all of the codes/2 facts from the model;
for each one of the ORFs to be learnt do
  for i in 1 to 10 do
    generate codes/2 facts using the following 3 strategies:
    1 ASE – selecting trials which minimise the expected cost of
    experimentation;
    2 Random – sampling trials at random (without replace-
    ment);
    3 Naive – selecting trials using the naive strategy of always
    choosing the cheapest trial from the set of novel candidate
    trials.
    for each learning strategy do
      determine, for each iteration of the learning cycle, the aver-
      age accuracy attained and the cumulative cost (CC) of the
      relative costs of the chemicals consumed.
    end
    for each of (ASE Random) do
      for j in (0 1 1.5 2 2.5 3 3.5 4 5 6 7 8) do
        estimate accuracy^a when CC = 10^j
      end
    end
    for (Naive) do
      for j in (0 1 1.5 1.75 2 2.25 2.5 2.75 3 3.25 3.5 4) do
        estimate accuracy^a when CC = 10^j
      end
    end
  end
end
```

[a] by finding the points which correspond to the two CCs which are closest to j, fining the line $y = mx + c$ which goes through these points and then calculating y when $x = j$. When this leads to extrapolation above or below the valid range $(0 - 100)$ the estimate is taken to be 100 or 0 respectively.

Table 6: Part B of Experimental method.

for each of (*ASE Random*) **do**
  for j **in** (0 1 1.5 2 2.5 3 3.5 4 5 6 7 8) **do**
    calculate mean and standard error of accuracy when CC = 10^j
  end
end
for (*Naive*) **do**
  for j **in** (0 1 1.5 1.75 2 2.25 2.5 2.75 3 3.25 3.5 4) **do**
    calculate mean and standard error of accuracy when CC = 10^j
  end
end
for each learning strategy **do**
  plot $log_{10}(P)$ versus accuracy with horizontal error bars, where P = cumulative cost of the chemicals consumed expressed in £.
  for each iteration of the learning cycle **do**
    estimate mean and standard error of the accuracy
  end
  plot duration of the experiment in days[a] versus accuracy with horizontal error bars.
end

[a]The duration of the experiment in days is taken to be equal to the number of trials performed: the duration of each trial is 24 hours (see Appendix C). The number of trials performed is equal to the number of iterations of the learning cycle.

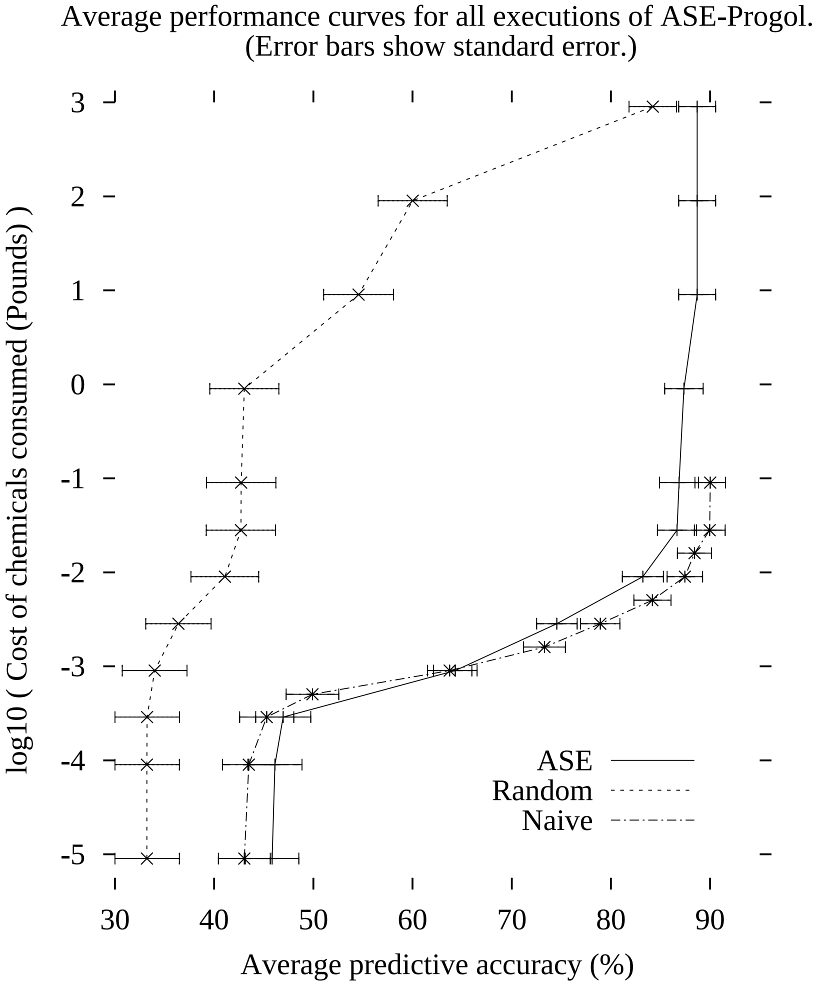

Figure 4: Results. Performance curves which show the cost of the chemicals consumed during training versus performance during testing

6.4 Results and Analysis

Figure 4 shows that the cost of the chemicals consumed in converging upon a hypothesis with an accuracy in the range $46 - 88\%$ is reduced if trials are selected so as to minimise the cost of experimentation rather than if they are sampled at random (without replacement). To reach an accuracy in the range $46-80\%$, selecting trials so as to minimise the cost of experimentation incurs five orders of magnitude less experimental costs. For example to reach an accuracy of 70%, sampling at random costs $\pounds 2.4E2$ while minimising the cost of experimentation costs $\pounds 1.8E - 3$.

Figure 4 also shows that the cost of the chemicals consumed is not reduced if trials are selected so as to minimise the cost of experimentation rather than if they selected using the naive strategy of always choosing the cheapest trial from the set of novel candidate trials. The error bars on the learning curve indicate that while less than $\pounds 1E - 3$ worth of chemicals have been consumed, there is no significant difference between these two

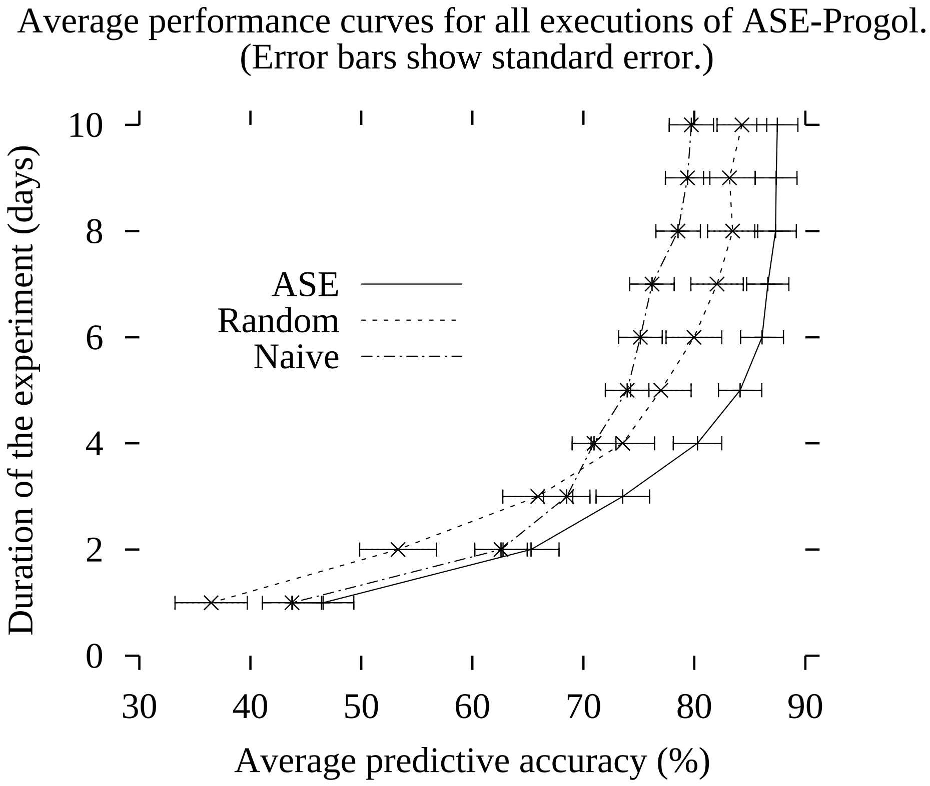

Figure 5: Results. Performance curves which show the duration of the experiment versus performance during testing.

strategies. When the cost of the chemicals consumed rises above £$1E-3$, higher accuracies are achieved by the naive strategy.

Figure 5 shows that the time required to converge upon a hypothesis with an accuracy in the range $74-87\%$ is shortest if trials are selected so as to minimise the cost of experimentation rather than if they are sampled at random (without replacement) or selected using the naive strategy. The results suggest that it takes at least twice as long to reach an accuracy in this range if trials are selected using the naive strategy rather than if they selected so as to minimise the cost of experimentation. For example to reach an accuracy of 80%, minimising the cost of experimentation requires 4 days while random sampling (without replacement) requires 6 days and the naive strategy requires 10 days.

7 Discussion

ASE-Progol has been designed on the basis of minimising the expected cost of a chosen series of experiments. While this appeared reasonable during the ASE-Progol design phase, the results in Section 6.4 seem to indicate that what really needs to be minimised is a function combining expected cost and expected number of trials. In terms of expected material costs Figure 4 shows that ASE-Progol is outperformed by the naive strategy of simply performing the cheapest trial. However Figure 5 indicates that ASE-Progol performs best in terms of the number of trials required. In this case, as might be expected from general results in Computational Learning Theory (eg [14]), random queries produce reasonably rapid convergence of the space of consistent hypotheses. One reasonable decision theoretic approach to the combination of time and cost would be to use maximum information gain per unit cost.

8 Conclusion

Our aim is to partially automate some aspects of scientific work, namely the processes of forming hypotheses, devising trials to discriminate between these competing hypotheses, performing these trials and then using the results of these trials to converge upon an accurate hypothesis.

To the best of our knowledge, there has only been one application of a Scientific Discovery system which has resulted in convergence upon an accurate hypothesis by physically performing trials in a laboratory [23]. The system used in this application, FAHRENHEIT, is limited to inducing numerical laws. Although the discovery of numerical laws is an important aspect of Scientific Discovery, we do not believe that this is the type of inference that is required to solve many contemporary problems in Bioinformatics. For example, we have proposed that a form of abduction (see [37]) should be used for Functional Genomics.

We have developed ASE-Progol (**A**ctive **S**election of **E**xperiments with Progol). ASE-Progol is an Active Learning system which uses ILP to construct hypothesised first-order theories, uses a CART-like algorithm to select trials for eliminating ILP derived hypotheses and instructs a robot to physically perform these trials.

We have used a novel form of performance curve, namely a plot of the cumulative cost of the training examples against performance. We have demonstrated how, in contrast to the form of performance curve normally used in Active Learning, our novel form of performance curve allows one to compare the costs incurred by different leaning strategies.

We have applied ASE-Progol to a discovery task in Functional Genomics, the domain in which we aim to physically perform trials using robotics. We have already published results elsewhere which support our proposed method of inferring the function of genes from the results of single-gene-deletion growth trials [37]. The new results reported here take a first step towards answering the question of whether the cost of converging upon an accurate hypothesis is significantly reduced if trials are selected by ASE-Progol rather than if they are sampled at random (without replacement) or selected using the naive strategy of always choosing the cheapest trial from the set of novel candidate trials. The cost of the chemicals consumed in converging upon a hypothesis with an accuracy in the range $46 - 88\%$ was reduced if trials were selected by ASE-Progol rather than if they were sampled at random. To reach an accuracy in the range $46 - 80\%$, ASE-Progol incurred five orders of magnitude less experimental costs than random sampling (without replacement). ASE-Progol required less time to converge upon a hypothesis with an accuracy in the range $74 - 87\%$ than if trials were sampled at random or selected using the naive strategy. For example to reach an accuracy of 80%, ASE-Progol required 4 days while random sampling required 6 days and the naive strategy required 10 days.

9 Future Work

9.1 Using Inhibitors to Change a Phenotype

It is only possible to rediscover the function of 12 of the 17 ORFs in the model, given the set of optional nutrients i.e. those metabolites which could, if desired, be added to a growth medium and, if they were added, be expected to enter the cell (see Section 5.1). For the other five ORFs, none of the growth media gives rise to a positive example (see Section 6.2.5). Future work will investigate whether the function of these five ORFs can be discovered by allowing inhibitors to be added to a growth medium.

Inhibition is a reduction in the rate of a catalysed reaction by substances called *inhibitors*. In biochemical reactions, in which the catalysts are enzymes, if the inhibitor molecules resemble the reactants they may bind to the active site of the enzyme, so preventing normal enzymatic activity. Alternatively they may form a complex with the substrate-enzyme intermediate or irreversibly destroy the enzyme configuration and active-site properties. If inhibitors are added to growth media in auxotrophic mutant experiments then they can affect whether phenotypic effects are observed because they can affect one or more reactions in the pathway by inhibiting the enzymes which catalyse those reactions.

When attempting to learn the function of a given ORF using the model described in Section 5, there is only one variable in the auxotrophic mutant experiments which ASE-Progol can control, namely the choice of nutrients to be added to the growth medium. We plan to introduce a second variable by allowing inhibitors to be added to a growth medium. The static knowledge will be extended to include knowledge of the affect of various inhibitors on the aromatic amino acid pathway of yeast. We will then investigate whether this will enable ASE-Progol to rediscover the function of those five ORFs whose function cannot be found by only adding nutrients to growth media.

Acknowledgements

We are very grateful to both D.Michie and D.Page for their considerable help with aspects of this work. We would also like to thank J.Cussens for his comments on an early draft of this work and T.Dietterich, T.Mitchell and S.Thrun for helpful answers to our questions regarding related work.

This work was supported by the EPSRC grant GR/M56067 and BBSRC grant 2/E11552.

A Upper Bound on the Number of Candidate Trials

The Trial Generator component of ASE-Progol requires a Prolog program which can generate a set of candidate trials. This section explains why sampling 20 candidate trials ensures that there is a low probability that the sample will not contain a novel trial.

Theorem 2 ($|T|$ Bound) *Let p be the probability of choosing a trial which is not novel, T be a set of candidate trials sampled independently and ran-*

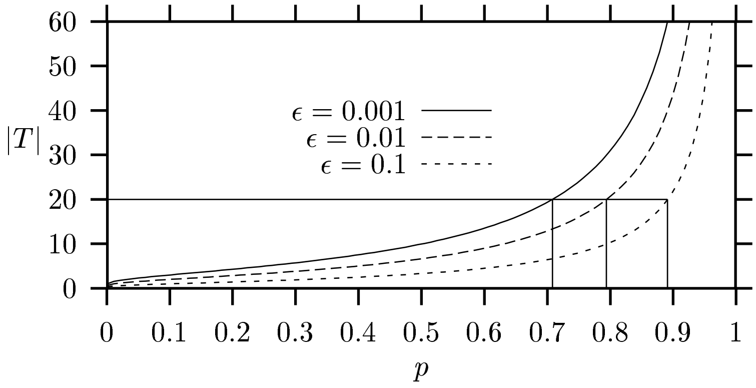

Figure 6: Proof that ASE-Progol's upper bound of 20 on the number of candidate trials that may appear in a classification matrix is sufficiently high to ensure that there is a low probability that a sample of candidate trials generated by the program will not contain a novel trial. $|T|, p$ and ϵ are defined by Thereom 4

domly and ϵ be a bound on q, the probability that T does not contain a novel trial. $|T|$ must be of size at least $\frac{log\ \epsilon}{log\ p}$.

Proof

$$\epsilon \geq q = p^{|T|}$$

$$|T| log\ p = log\ \epsilon$$

$$|T| = \frac{log\ \epsilon}{log\ p} \tag{1}$$

$$|T| \geq \frac{log\ \epsilon}{log\ p}$$

□

Fig.6 shows how the value of $|T|$ varies with the value of p when ϵ has the values $0.001, 0.01$ and 0.1. The intercepts on the x axis corresponding to $|T| = 20$ show that if the Trial Generator uses the program to sample 20 trials from the space of possible trials then the probability that none of these 20 trials will be novel is:

- below 0.001 while less than 71% of the population of trials have been performed.
- below 0.1 while less than 90% of the population of trials have been performed.

Table 7: A Mapping of ORFs to Reactions in our Model.

```
codes('YDR254W','4.2.1.11',['C00631'],['C00074']).
codes('YDR254W','4.2.1.11',['C03356'],['C00074']).
codes('YHR174W','4.2.1.11',['C00631'],['C00074']).
codes('YHR174W','4.2.1.11',['C03356'],['C00074']).
codes('YMR323W','4.2.1.11',['C00631'],['C00074']).
codes('YMR323W','4.2.1.11',['C03356'],['C00074']).
codes('YBR249C','4.1.2.15',['C00009','C04691'],['C00074','C00279']).
codes('YDR035W','4.1.2.15',['C00009','C04691'],['C00074','C00279']).
codes('YDR127W','4.6.1.3',['C04691'],['C00009','C00944']).
codes('YDR127W','4.2.1.10',['C00944'],['C02637']).
codes('YDR127W','X',['C02652'],['C02637']).
codes('YDR127W','1.1.1.25',['C00006','C00493'],['C00005','C02652']).
codes('YDR127W','2.7.1.71',['C00002','C00493'],['C00008','C03175']).
codes('YDR127W','2.5.1.19',['C00074','C03175'],['C00009','C01269']).
codes('YGL148W','4.6.1.4',['C01269'],['C00009','C00251']).
codes('YER090W','4.1.3.27',['C00014','C00251'],
      ['C00022','C00025','C00108']).
codes('YER090W','4.1.3.27',['C00064','C00251'],
      ['C00022','C00025','C00108']).
codes('YKL211C','4.1.3.27',['C00064','C00251'],
      ['C00022','C00025','C00108']).
codes('YDR354W','2.4.2.18',['C00013','C04302'],['C00108','C00119']).
codes('YDR007W','5.3.1.24',['C04302'],['C01302']).
codes('YKL211C','4.1.1.48',['C01302'],['C03506']).
codes('YGL026C','4.2.1.20',['C00065','C03506'],['C00078','C00661']).
codes('YGL026C','4.2.1.20',['C00065','C00463'],['C00078']).
codes('YGL026C','4.2.1.20',['C03506'],['C00463','C00661']).
codes('YPR060C','5.4.99.5',['C00251'],['C00254']).
codes('YNL316C','4.2.1.51',['C00254'],['C00166']).
codes('YGL202W','2.6.1.7',['C00025','C00166'],['C00026','C00079']).
codes('YGL202W','2.6.1.7',['C00025','C01179'],['C00026','C00082']).
codes('YHR137W','2.6.1.7',['C00025','C00166'],['C00026','C00079']).
codes('YHR137W','2.6.1.7',['C00025','C01179'],['C00026','C00082']).
codes('YBR166C','1.3.1.13',['C00006','C00254'],['C00005','C01179']).
```

B Mapping of ORFs to Reactions

The many-to-many mapping of ORFs to reactions in our model is represented by the `codes(orf, enzyme_class, reactions, products)` facts shown in Table 7. For example `codes('YDR254W','4.2.1.11',['C00631'],['C00074'])` represents the fact that the ORF `'YDR254W'` codes for an enzyme which catalyses the reaction which generates the product `'C00074'` from the reactant `'C00631'` or vice versa.[10]

[10] The model assumes that the enzymes catalyse reactions in both directions.

Figure 7: Beckman/Coulter Biomek 2000. A standard laboratory robot designed for liquid handling tasks.

C Automating Growth Trials using Robotics

The robotics needed to automate growth trials are straightforward: the system only has to specify what the robot is to do within a tightly constrained task type, rather than how it is to be done. A standard laboratory robot designed for these sorts of liquid handling tasks, namely the Beckman/Coulter Biomek 2000 (see Figure 7), will be used. The input to the Biomek 2000 will be a list of samples to be made from specified quantities of a given set of mutants and growth media. The Biomek 2000 will prepare samples on 96-well microtiter plates.[11] Each well of the microtiter plate will contain a total volume of 750μL. The content of each well will then be mixed thoroughly, sealed with a gas-permeable film (to avoid evaporation problems) and incubated at 30C for 24 hours. The Biomek 2000 will subsequently transfer the microtiter plates to a machine which will read the absorbance of a coloured product in each of the wells of the plate. From these readings, the growth of the culture in each well will be derived.

C.1 Synthetic Glucose (Dextrose) Minimal Medium (SGD)

Every growth media will contain all the compounds needed for growth in the wild strain of yeast. This will be achieved by adding Synthetic glucose (dextrose) minimal medium (SGD) [28, 1] to every medium. SGD is a synthetic medium containing 2% (20 grammes/litre) of glucose and 0.67% of *Bacto yeast nitrogen without amino acids*, a complex of different compounds which provides the requirements of nitrogen, phosphorous, sulphur, minerals, trace elements and vitamins for growth. Growth media will differ from one another in which mutants or additional nutrients or metabolites are added to the SGD.

[11] Microtiter plates are small, transparent plastic trays containing a number of wells. The standard number of wells is 96.

D Related Work

Section 2.1 referred to related work on Active Learning. This section discusses other related lines of research. To the best of our knowledge, there has only been one application of a Scientific Discovery system which has resulted in convergence upon an accurate hypothesis by physically performing trials in a laboratory (see Appendix D.1). The system used in this application, FAHRENHEIT, is limited to inducing numerical laws.

D.1 The BACON Family of Systems

The longest-standing line of research which is related to ASE-Progol is the work on the BACON family of systems. These systems are limited to generating relatively simple numerical equations. Perhaps the most relevant such work is the extension of FAHRENHEIT [23] which enabled it to generate hypotheses on-line and subsequently change the experimentation pattern to concentrate new trials on a particular area that best contributed to the growth of the emerging theory. This was demonstrated in the domain of electrochemistry where trials involved computer operated sensors and actuators. One of the reasons that this domain was chosen was that concepts could be represented by relatively simple equations.

D.2 Robotic Discovery and Reinforcement Learning

The careful choice of trials by ASE-Progol is related to *directed exploration* [50] in robotic discovery and reinforcement learning. Directed exploration involves an important tradeoff between exploration (analogous to experimentation) and exploitation (using previously learned knowledge) in the context of robot navigation and neural network learning, a tradeoff that doesn't appear within ASE-Progol. This tradeoff is studied in more detail by Moore and Atkeson, who propose *prioritised sweeping*, a successful heuristic approach to balancing exploration and exploitation [32]. Kearns and Singh present a related algorithm E^3 (the Explicit Explore or Exploit Algorithm) that provably produces a near-optimal *return* with a number of actions and total computation time bounded by a polynomial in the relevant task-specific parameters [24].

The key distinction between ASE-Progol and reinforcement learning is that the eventual payoff in reinforcement learning comes from exploitation of learned knowledge, whereas learning itself is the end goal in ASE-Progol. Hence ASE-Progol does not exhibit the tradeoff between exploration (experimentation) and exploitation. It would do if one sought to maximise gain of information per unit cost.

The work in robotic discovery that is perhaps most closely related to ASE-Progol is by Fox, Burgard, and Thrun on *active localisation* [17]. Localisation is the problem of a robot estimating its present position from sensor data. In their approach to localisation, the robot "controls its various effectors so as to most efficiently localise itself." More specifically, they employed a greedy approach analogous to the one proposed by Mitchell and described in this paper. In this approach, "actions are generated by maximising the expected decrease in uncertainty, measured in entropy." Their results speak well for this approach. They note that although the localisation problem in general is NP-hard, in practice their robot always was able to localise itself using this greedy strategy.

References

[1] A.Adam, D.E.Gottschling, C.A.Kaiser, and T.Stearns. *Methods in Yeast Genetics: A Cold Spring Harbour Laboratory Course Manual*. Cold Spring Harbour Laboratory Press, Appendix A 1997.

[2] B.R.Bochner, P.Gadzinski, and E.Panomitros. Phenotype microarrays for high-throughput phenotypic testing and assay of gene function. *Genome Research*, 11:1246–1255, 2001.

[3] L. Breiman, J.H. Friedman, R.A. Olshen, and C.J. Stone. *Classification and Regression Trees*. Wadsworth, Belmont, 1984.

[4] T.D. Brock, M.T. Madigan, J.M. Martinko, and J. Parker. *Biology of Microorganisms*. Prentice Hall, London, 1994.

[5] P.R. Butler. Personal communication with S.G. Oliver.

[6] C.A.Thompson, M.E.Califf, and R.J.Mooney. Active learning for natural language parsing and information extraction. In I.Bratko and S.Dzeroski, editors, *Proceedings of the Sixteenth International Machine Learning Conference(ICML-99)*, pages 406–414, Bled, Slovenia, 1999. Morgan Kaufmann.

[7] C.E.Shannon and W.Weaver. *The mathematical theory of communication*. University of Illinois Press, Urbana, 1949.

[8] D. Cohn, L. Atlas, and R. Ladner. Improving generalization with active learning. *Machine Learning*, 15:201–221, 1994.

[9] D. A. Cohn. Neural network exploration using optimal experimental design. *Neural Networks*, 9(6):1017–1083, 1996.

[10] D. A. Cohn, Z.Ghabhramani, and M. I. Jordan. Active learning with statistical models. *Journal of Artificial Intelligence Research*, 4:129–145, 1996.

[11] D.Delneri, F.L.Brancia, and S.G.Oliver. Towards a truly integrative biology through the functional genomics of yeast. *Current Opinion in Biotechnology*, 12:87–91, February 2001.

[12] D.D.Lewis and W.A. Gale. A sequential algorithm for training text classifiers. In W.B.Croft and C.J.van Rijsbergen, editors, *SIGIR 94: Proceedings of the Seventeenth Annual International ACM-SIGIR Conference on Research and Development in Information Retrieval*, pages 3–12, London, 1994. Springer-Verlag.

[13] B. Dujon. The yeast genome project - what did we learn? *Trends in Genetics*, 12:263–270, 1996.

[14] A. Ehrenfeucht, D. Haussler, M. Kearns, and L. Valiant. A general lower bound on the number of examples needed for learning. In *COLT 88: Proceedings of the Conference on Learning*, pages 110–120, Los Altos, CA, 1988. Morgan-Kaufmann.

[15] V. V. Fedorov. *Theory of Optimal Experiments*. Academic Press, London, 1972.

[16] P. Finn, S. Muggleton, D. Page, and A. Srinivasan. Pharmacophore discovery using the inductive logic programming system Progol. *Machine Learning*, 30:241–271, 1998.

[17] D. Fox, W. Burgard, and S. Thrun. Active Markov localization for mobile robots. *Robotics and Autonomous Systems*, 25(3-4):195–207, 1998.

[18] Goffeau, A. *et multi al.* Life with 6000 genes. *Science*, 274:546–567, 1996.

[19] S. Goto, T. Nishioka, and M. Kanehisa. LIGAND: chemical database of enzyme reactions. *Nucleic Acids Research*, 28:380–382, 2000.

[20] M. Hasenjäger and H. Ritter. Active learning with local models. *Neural Processing Letters*, 7:107 – 117, 1998.

[21] L. Hyafil and R. Rivest. Constructing optimal binary decision trees is NP-complete. *Information Processing Letters*, 5(1):15–17, 1976.

[22] J.F.Young. *Information Theory.* Butterworth, London, 1971.

[23] J.M.Zytkow, J.Zhu, and A.Hussam. Automated discovery in a chemistry laboratory. In *Proceedings of the Eighth National Conference on Artificial Intelligence*, pages 889–894. The AAAI Press, 1990.

[24] M. Kearns and S. Singh. Near-optimal reinforcement learning in polynomial time. In Jude Shavlik, editor, *Proceedings of the Fifteenth International Conference on Ma chine Learning*, San Mateo, CA, 1998. Morgan Kaufmann.

[25] R. King, S. Muggleton, R. Lewis, and M. Sternberg. Drug design by machine learning: The use of inductive logic programming to model the structure-activity relationships of trimethoprim analogues binding to dihydrofolate reductase. *Proceedings of the National Academy of Sciences*, 89(23):11322–11326, 1992.

[26] R. King, S. Muggleton, A. Srinivasan, and M. Sternberg. Structure-activity relationships derived by machine learning: the use of atoms and their bond connectives to predict mutagenicity by inductive logic programming. *Proceedings of the National Academy of Sciences*, 93:438–442, 1996.

[27] K.M.Herrmann and L.M.Weaver. The shikimate pathway. *Annual Reviews of Plant Physiology and Plant Molecular Biology*, 50:473–503, 1999.

[28] Difco Laboratories. *The Difco Manual.* Detroit Michigan, USA, tenth edition, 1984.

[29] F-R. Lin and M. J. Shaw. Active training of backpropagation neural networks using the learning by experimentation methodology. *Annals of Operations Research*, 75:129–145, 1997.

[30] L.Stryer. *Biochemistry.* Freeman, New York, 1995.

[31] H.W. Mewes, K. Heumann, A. Kaps, K. Mayer, F. Pfeiffer, S. Stocker, and D. Frishman. MIPS: a database for protein sequences and complete genomes. *Nucleic Acids Research*, 27:44–48, 1999.

[32] Andrew W. Moore and Christopher G. Atkeson. Prioritized sweeping: Reinforcement learning with less data and less time. *Machine Learning*, 13:103–130, 1993.

[33] S. Muggleton. Inverse entailment and Progol. *New Generation Computing*, 13:245–286, 1995.

[34] S. Muggleton, R. King, and M. Sternberg. Protein secondary structure prediction using logic-based machine learning. *Protein Engineering*, 5(7):647–657, 1992.

[35] S. Muggleton and D. Page. A learnability model for universal representations and its application to top-down induction of decision trees. In K. Furukawa, D. Michie, and S. Muggleton, editors, *Machine Intelligence 15*. Oxford University Press, 1999.

[36] S. Muggleton and L. De Raedt. Inductive logic programming: Theory and methods. *Journal of Logic Programming*, 19,20:629–679, 1994.

[37] S.H. Muggleton and C.H. Bryant. Theory completion using inverse entailment. In J.Cussens and A.Frisch, editors, *Proceedings of the Tenth International Conference on Inductive Logic Programming*, number 1866 in Lecture Notes in Artificial Intelligence, pages 130–146, London, UK, 2000. ©Springer Verlag http://www.springer.de/comp/lncs/index.html.

[38] International Union of Biochemistry and Molecular Biology. *Enzyme Nomenclature: Recommendations (1992) of the Nomenclature Committee of the International Union of Biochemistry and Molecular Biology*. Academic Press, New York, 1992.

[39] SG Oliver. *Trends in Genetics*, 12:241–242, 1996.

[40] S.G. Oliver. From DNA sequence to biological function. *Nature*, 379:597–600, 1996.

[41] SG Oliver, MK Winson, DB Kell, and F Baganz. Systematic functional analysis of the yeast genome. *Trends in Biotechnology*, 16(9):373–378, September 1998.

[42] K.J. Rieger, M. El-Alama, G. Stein, C. Bradshaw, P.P. Slonimski, and K. Maundrell. Chemotyping of yeast mutants using robotics. *Yeast*, 15:973–986, 1999.

[43] K.J. Rieger, G. Orlowska, A. Kaniak, J.Y. Coppee, G. Aljinovic, and P.P.Slonimski. Large-scale phenotypic analysis in microtitre plates of mutants with deleted open reading frames from yeast chromosome III: Key-step between genomic sequencing and protein function. In A.G.Craig and J.D.Hoheisel, editors, *Automation*, volume 28 of *Methods in Microbiology*, pages 205–227. Academic Press, 1999.

[44] J. Rissanen. A universal prior for integers and estimation by Minimum Description Length. *Annals of Statistics*, 11:416–431, 1982.

[45] M. Schena, D. Shalon, R.W. Davis, and P.O. Brown. Quantitative monitoring of gene-expression patterns with a complementary-DNA microarray. *Science*, 270:467–470, 1995.

[46] A. Srinivasan, S. Muggleton, R. King, and M. Sternberg. Theories for mutagenicity: a study of first-order and feature based induction. *Artificial Intelligence*, 85(1,2):277–299, 1996.

[47] S.Soderland. Learning information extraction rules for semi-structured and free text. *Machine Learning*, 34:233–272, February 1999.

[48] M. Sternberg, R. King, R. Lewis, and S. Muggleton. Application of machine learning to structural molecular biology. *Philosophical Transactions of the Royal Society B*, 344:365–371, 1994.

[49] M. Sternberg, R. Lewis, R. King, and S. Muggleton. Modelling the structure and function of enzymes by machine learning. *Proceedings of the Royal Society of Chemistry: Faraday Discussions*, 93:269–280, 1992.

[50] S. Thrun. The role of exploration in learning control. In D. White and D. Sofgre, editors, *Handbook of Intelligent Control: Neural, Fuzzy and Adaptive Approaches*. Van Nostrand Reinhold, Florence, KY, 1992.

[51] J.M. Trent, M.H. Bittner, J. Zhang, R. Wiltshire, M. Ray, Y. Su, E. Gracia, P. Meltzer, J. DeRisi, L. Penland, and P. Brown. Use of microgenomic technology for analysis of alterations in DNA copy number and gene expression in malignant melanoma. *Clinical and Experimental Immunology*, 107(S1):33–40, 1997.

[52] M. Turcotte, S.H. Muggleton, and M.J.E. Sternberg. Automated discovery of structural signatures of protein fold and function. *Journal of Molecular Biology*, 2000. Accepted subject to revision.

[53] M. Turcotte, S.H. Muggleton, and M.J.E. Sternberg. The effect of relational background knowledge on learning of protein three-dimensional fold signatures. *Machine Learning*, 2000. In press.

[54] P. Uzzel. *Aspects of Isomerism.* Methuen, 1971.

[55] A. Wald. *Sequential Analysis.* John Wiley, New York, 1947.