Mach Learn (2012) 86:3–23
DOI 10.1007/s10994-011-5259-2

ILP turns 20

Biography and future challenges

Stephen Muggleton · Luc De Raedt · David Poole · Ivan Bratko · Peter Flach · Katsumi Inoue · Ashwin Srinivasan

Received: 8 May 2011 / Accepted: 29 June 2011 / Published online: 5 September 2011

Abstract Inductive Logic Programming (ILP) is an area of Machine Learning which has now reached its twentieth year. Using the analogy of a human biography this paper recalls the development of the subject from its infancy through childhood and teenage years. We show how in each phase ILP has been characterised by an attempt to extend theory and implementations in tandem with the development of novel and challenging real-world applications. Lastly, by projection we suggest directions for research which will help the subject coming of age.

Editors: Paolo Frasconi and Francesca Lisi.

S. Muggleton (✉)
Imperial College London, London, UK
e-mail: s.muggleton@imperial.ac.uk

L. De Raedt
Katholieke Universiteit Leuven, Leuven, Belgium
e-mail: Luc.DeRaedt@cs.kuleuven.be

D. Poole
University of British Columbia, Vancouver, Canada
url: http://www.cs.ubc.ca/~poole/

I. Bratko
University of Ljubljana, Ljubljana, Slovenia
e-mail: bratko@fri.uni-lj.si

P. Flach
University of Bristol, Bristol, UK
e-mail: Peter.Flach@bristol.ac.uk

K. Inoue
National Institute of Informatics, Tokyo, Japan
e-mail: ki@nii.ac.jp

A. Srinivasan
South Asian University, New Delhi, India
e-mail: ashwin.srinivasan@wolfson.oxon.org

Keywords Inductive Logic Programming · (Statistical) relational learning · Structured data in Machine Learning

1 Introduction

The present paper was authored by members of a discussion panel at the twentieth International Conference on Inductive Logic Programming. The topic of the panel was a discussion of the achievements of the field to date and possible new directions for the future. Discussions by email following the panel led to the present paper, in which an attempt has been made to reach consensus among the sometimes disparate views of the panelists.

The first workshop on Inductive Logic Programming was held twenty years ago, and brought together a diverse group of researchers working at the intersection of Machine Learning and Logic Programming. This paper reflects on the achievements of the subject to date and the challenges ahead. The subject has progressed through a number of distinct phases which we describe in terms of its infancy, childhood and teenage years. In each phase ILP has been characterised by an attempt to extend theory and implementations in tandem with the development of novel and challenging real-world applications. The paper suggests directions for future research which are aimed at helping the subject coming of age into adulthood.

This paper is organised as follows. Section 2 describes some of the motivations of the field by situating it within the broader context of Machine Learning. This is exemplified by various problems which benefit from a relational representation. Section 3.1 describes several phases of the twenty year history of the field starting from the year of the first conference and founding paper in 1991 and progressing through to the present day. Various possible future directions for the field are then described in Sect. 4. We then conclude the paper in Sect. 5 and provide a few pointers for further reading in Sect. 6. Lastly we summarise the author contributions to the paper in Sect. 6.

2 Motivation of ILP

The field of Inductive Logic Programming essentially combines Machine Learning with logical knowledge representation. To understand the needs for such a combination, consider learning from the two datasets in Fig. 1 (from Poole and Mackworth 2010). Dataset (a) is the sort used in traditional supervised and unsupervised learning. Standard textbook supervised learning algorithms can learn a decision tree, a neural network, or a support vector machine to predict *User Action*. A belief network learning algorithm can be used to learn a representation of the distribution over the features.

Dataset (b), from which we may want to predict what Joe likes, is different. Many of the values in the table are meaningless names that cannot be used directly in supervised learning. Instead, it is the relationship among the individuals in the world that provides the generalizations from which to learn. Learning from such datasets is the core task of Inductive Logic Programming (ILP) (Muggleton and De Raedt 1994; Lavrač and Džeroski 1993) mainly because logic programs provide a good representation for the generalizations required to make predictions. Logic programs are also more expressive than alternative representations that are sometimes used today to cope with such data (such as network and graph-based representations).

Lastly, a key goal of ILP is the generation of human-interpretable explanations. It is usually stratightforward to translate logic programs into a series of easily understandable

Example	*Author*	*Thread*	*Length*	*Where Read*	*User Action*
e_1	*known*	*new*	*long*	*home*	*skips*
e_2	*unknown*	*new*	*short*	*work*	*reads*
e_3	*unknown*	*follow up*	*long*	*work*	*skips*
e_4	*known*	*follow up*	*long*	*home*	*skips*
...	...	...	...	...	...

(a)

Individual	Property	Value
joe	*likes*	*resort_14*
joe	*dislikes*	*resort_35*
...	...	...
resort_14	*type*	*resort*
resort_14	*near*	*beach_18*
beach_18	*type*	*beach*
beach_18	*covered_in*	*ws*
ws	*type*	*sand*
ws	*color*	*white*
...	...	...

(b)

Fig. 1 Two datasets. Dataset (**a**) represents input to a traditional feature-based learner in which each example can be viewed as a vector of features describing a single object. By contrast dataset (**b**) represents input to a relational learner, in which each example describes the relationship between a group of named objects

sentences which can be understood by a domain expert. This is not the case in many other approaches within Machine Learning.

3 Overview and lessons learned

3.1 An overview (1991–2010)

3.1.1 Prehistory (1969–1990) and infancy (1991–1994)—prodigious beginnings

The prehistory of ILP can be found in seminal research by Plotkin (1969, 1971a, 1971b), Vere (1975), Shapiro (1983) and Sammut and Banerji (1986). The subject was initiated in 1991 with the publication of the founding paper (Muggleton 1991) and the launch of the first international workshop on ILP. A reference textbook covering the various approaches followed (Muggleton 1992). Much of the strength of the subject was related to its ability to draw on and extend the existing literature of its parent subjects of Machine Learning and Logic Programming.

The early phase of the subject was associated with the introduction of a number of foundational theoretical concepts. These included Inverse Resolution (Muggleton and Buntine 1988) and the related concepts of Saturation (Rouveirol and Puget 1989) and Predicate Invention (Muggleton and Buntine 1988; Stahl 1992; Muggleton 1991; Muggleton and De Raedt 1994). These theoretical developments were accompanied by initial results on PAC-learnability of various relevant classes of hypotheses including ij-determinate logic programs (Džeroski et al. 1993) and k-local logic programs (Cohen 1993), as well as negative results on PAC-learning of arbitrary logic programs (Kietz 1993).

Alongside these theoretical results a number of ILP systems were developed, in many cases linked to the theoretical developments above. These included widely deployed systems such as FOIL (Quinlan 1990), Golem (Muggleton and Feng 1990), CLINT (De Raedt and Bruynooghe 1991) and LINUS (Lavrač et al. 1991).

3.1.2 Childhood (1995–2001)—logical development

During this period the theoretical framework for ILP described in Muggleton and De Raedt (1994) was radically extended in the publication of the first textbook on the theoretical foundations of ILP (Nienhuys-Cheng and de Wolf 1997). This text expounded the refinement-graph theory of search used in many existing ILP systems. In addition, further significant results on the learnability of logic programs continued to be developed (Khardon 1998). Towards the end of the period a book was published describing the new field of Relational Data Mining (Džeroski and Lavrač 2001), a field in which ILP techniques were applied to the mutliple tables in a database setting.

In this same period the widely-used ILP system Progol (Muggleton 1995) introduced a new logically-based approach to refinement graph search of the hypothesis space based on inverting the entailment relation. Although Muggleton (1995) indicated the way in which abduction can be treated as a special case of inverting entailment, an implementation of Progol which supported this approach to abduction was delayed until later (Progol5.0, Muggleton and Bryant 2000). In particular, Progol5.0 avoided the assumption that the hypothesised predicate was identical to the predicate expressing the examples. Meanwhile the ILP system TILDE (Blockeel and De Raedt 1997) demonstrated the efficiency which could be gained by upgrading decision-tree learning algorithms to first-order logic. The upgrading methodology was soon extended towards other machine learning problems; cf. Sect. 3.5.1.

3.1.3 Contemporary ILP—teenage years (2002–2010)—indecision

The last ten years of ILP have been dominated by the development of methods for learning probabilistic logic representations, along with applications related to transfer learning (Mihalkova and Mooney 2009; Torrey and Shavlik 2010; Davis and Domingo 2009). A large variety of general probabilistic representations have emerged including Stochastic Logic Programs (SLPs) (Muggleton 1996), Bayesian Logic Programs (BLPs) (Kersting and De Raedt 2001), PRISM (Sato 2005), Independent Choice Logic (ICL) (Poole 2000), Markov Logic Networks (Domingos et al. 2006), CLP(BN) (Costa et al. 2003) and ProbLog (De Raedt et al. 2007).

A general framework for Probabilistic ILP (PILP) was introduced (De Raedt and Kersting 2004) alongside a multi-author reference book covering the various approaches being pursued (De Raedt et al. 2008). PILP systems tend to separate the learning of the underlying logic program from the estimation of probabilistic parameters associated with individual clauses. For instance, the PRISM system (Sato 2005) assumes the underlying logic program is provided by the modeller but provides EM-based parameter estimation techniques for learning the probabilistic labels. By contrast, in SLPs the underlying logic program is learned using a general ILP system, followed by a second phase in which the parameters are learned, often by a variant of the EM algorithm such as FAM (Cussens 2001). It has been argued (Muggleton 2002) that the structure and parameters of probabilistic logic representations should be learned simultaneously though this has turned out to be hard to achieve in practice.

3.2 Applications

Even in its infancy (see Sect. 3.1.1) ILP algorithms were successfully applied to hard real-world problems including finite element mesh design (Dolsak and Muggleton 1992), protein structure prediction (Muggleton et al. 1992), development of temporal rules for satellite diagnosis (Feng 1992) and learning qualitative simulation models (Bratko et al. 1991).

As part of the childhood phase (see Sect. 3.1.2) the development of new systems helped extend the class of applications to which ILP could be effectively applied. For instance, Progol's ability to learn non-determinate predicates allowed it to be successfully applied to an important biochemical discovery task involving mutagens (King et al. 1996), resulting in the publication of novel mutagens in one of the world's top scientific journals.

During the teenage years (see Sect. 3.1.3) ILP systems continued to be stretched by applications to hard scientific discovery problems. Most notably in this respect Progol5.0 was extended to support active selection of experiments within the Robot Scientist project. This project involved discovery of novel gene functions in yeast with results published in the journal Nature (King et al. 2004) with follow-up work in Science (King et al. 2009). PILP was also applied successfully to scientific discovery problems such as protein fold prediction (Chen et al. 2008).

3.3 The role of logic programming

In order to understand the development of ILP as a research area, it is important to reflect on its origins in and relationship with Logic Programming, which emerged as a declarative programming paradigm in the 1970s and became influential in the 1980s. Although it was customary to describe ILP as an area at the intersection of Machine Learning and Logic Programming, it is probably fair to say that the emphasis of much work fell more on the Logic Programming side. The idea of inductive synthesis of logic programs (Muggleton and Feng 1992; Quinlan and Cameron-Jones 1993), and more specifically Prolog programs, was appealing and was the driver of much initial research on predicate invention (Muggleton and Buntine 1988; Stahl 1992), multiple predicate learning (De Raedt and Lavrač 1996), and other forms of knowledge-intensive learning.

However, the computation embedded in a typical logic program is considerably more complex than that of logic-based classification rules. Inductive program synthesis needs to reconstruct that computation from examples and is therefore, in its generality, a much harder problem requiring a strong inductive bias. Furthermore, declaratively there is no difference between the answers computed by an efficient program—say, quicksort—and an inefficient one—say, permutation-sort. Expressing a preference for efficient programs requires a bias towards shorter proofs, which is easily done, but also a bias towards programs with smaller search spaces in which those proofs are found, which is much harder. So, gradually, the emphasis shifted to learning (typically non-recursive) first-order classification rules.

Eventually, the idea took hold that what defines and unifies much of ILP is the relational form of the input, rather than the first-order logic form of the output. This has led to a broadening of the field to relational learning, and an exploration of the natural connection with relational databases. While data and models were tightly connected in the original view of ILP, expressed as they were in the same language of first-order logic, nowadays the connection is established through the ability to construct and use first-order features encapsulating the relevant structure of the data. This change of perspective has helped to establish relational learning as a key area within Machine Learning.

However, certain artefacts from the Logic Programming origins can be observed to this day. For example, there is still a strong emphasis on a concept learning setting, in which one learns a model for only one of two classes. This is natural if one represents one of the classes by a predicate, and interprets the absence of a proof for that predicate as a proof for the opposite class (negation as failure). However, if ILP had been rooted in a functional language instead, one would have expected a multi-class setting to have been embraced from the start. Furthermore, there is a propensity to employ existential quantification to

aggregate over local variables in first-order features, just as Prolog does for variables that occur in the body of a clause but not in the head. The realisation that other aggregators—e.g., counts or averages—might be preferable in certain domains, and hence should be first-class citizens in the hypothesis language, took a long time to sink in (Krogel and Wrobel 2001; Knobbe et al. 2002; Vens et al. 2006). Thirdly, while the proliferation of metadata as a means to move towards semantics and away from syntax is visible all around us, there is still a certain reluctance to employ anything more than simple mode declarations in relational learning. If Prolog had been a strongly typed language things might have again been very different today.

There is no doubt that Prolog, as an executable language for rich first-order knowledge, has made an important contribution to Machine Learning. It has also been a key factor in establishing ILP as a coherent and active research area and community. Declarative programming has moved on considerably from the early Logic Programming days, and much can be gained from incorporating those advances in ILP and relational learning.

3.4 ILP's contribution to logical foundations of inductive reasoning

One of the important achievements of ILP is the development of *logical foundations* of inductive reasoning. Many important theoretical concepts in ILP that had appeared by early 1990s were quite original, and were situated in the intersection of Machine Learning, Logic Programming and AI. These theoretical studies were soon followed by works on implementations and applications. However, since then, research interests have gradually shifted to extensions of previously proposed frameworks, often by putting more emphasis on the standpoint of Machine Learning in terms of (multi)relational learning and probabilistic logic learning.

Early important logical theories of induction have been collected in such as Muggleton (1992), Nienhuys-Cheng and de Wolf (1997). Notably, the notion of *generalization* has been used in AI, but properties and many variants have been investigated in ILP, and a number of techniques for top-down and bottom-up *refinement* have been developed based on those generalization relations. Another salient feature of ILP is the use of prior or *background theories* for learning. *Covering* relations are precisely defined as logical entailment or subsumption, and their inverse relations have been devised for computing hypotheses in the presence of background theories (Muggleton 1995; Inoue 2004; Yamamoto et al. 2010).

The use of background theories enables us to extend the capability of inductive reasoning from classical learning tasks such as concept learning and classification to *theory completion* (Muggleton and Bryant 2000), the importance of which has been recently recognized in scientific discovery. On the other hand, *abduction* has played essential roles in knowledge discovery and development in AI and philosophy of science. Hence, integration of abduction and induction (Flach and Kakas 2000) has been discussed since 2000 in such diverse aspects as implementation of ILP systems using abductive methods (Muggleton and Bryant 2000; Inoue 2004; Ray et al. 2003; Inoue et al. 2010; Corapi et al. 2010) and "closing the loop" methodologies in scientific discovery (Flach and Kakas 2000; King et al. 2004; Synnaeve et al. 2011).

Among many applications, advances in those fundamental ILP techniques are well-suited to learning and discovering knowledge in *systems biology* (King et al. 2004; Tamaddoni-Nezhad et al. 2006; Synnaeve et al. 2011). Using logic modeling of biological systems, relations between elements can be declaratively described as constraints or *networks*, and background theories contain prior biological knowledge and databases of

gene/protein/metabolites and their interrelations. Then, theory completion can find missing links in incomplete networks, and those found hypotheses enable scientists to experiment with focused cases. Probability can be combined with logical inference, offering tools for modeling biological processes and for ranking logical hypotheses.

3.5 Lessons learnt

ILP applications in the 90s largely focused on the discovery of new and interpretable knowledge from structured data, often in the form of rules. Since then the range of tasks to which ILP techniques has been applied has significantly broadened and now covers almost all machine learning problems. The groundwork laid in these last twenty years by ILP has provided some valuable lessons, summarised by the following aphorisms:[1]

1. Downgrading ILP is possible, and interesting.
2. ILP operators and orderings can be reused in other settings.
3. Select the right learning setting.
4. Logical formulae are important in hybrid representations.
5. Use background knowledge liberally.
6. "Propositionalize" if you can.

3.5.1 From upgrading to downgrading

Most of the work so far in ILP has upgraded various propositional learning tasks, representations and systems towards the use of logical and relational representations. The upgrading approach has been very productive and resulted in many interesting new systems and representations, such as Quinlan's FOIL for rule learning (Quinlan 1990), Blockeel and De Raedt's Tilde (1997) for decision tree induction, Dehaspe's Warmr for relational association rule learning (Dehaspe and Toivonen 2001), Emde and Wettschereck's RIBL (Emde and Wettschereck 1996) for relational instance based learning, and Getoor and Koller's probabilistic relational models for Bayesian networks (Getoor et al. 2001). The resulting frameworks are very expressive and typically allow one to emulate the original setting and system. For instance, Bayesian nets are an instance of probabilistic relational models and Quinlan's well-known decision tree learner C4.5 (Quinlan 1987) is a special case of TILDE. At the same time, because of their expressiveness, they can work at different and intermediate levels of representation. For instance, graphs and networks can easily be represented using relational logic, and hence, ILP systems are applicable to graph and network based representations. Expressiveness also comes at a computational cost, which explains why the typical ILP systems are less efficient than and do not scale so well as more specialized systems. There are, however, new roles for ILP. First, such systems can and should be used as a base line for evaluating more specific approaches. Second, such systems can be downgraded, that is, specialized and optimized for working with some more specific representation, and there are many opportunities for doing so. One productive line of research that arguably downgrades ILP is that on mining and learning in graphs.

3.5.2 Operators and generality

The theory of ILP has contributed a rich variety of frameworks for reasoning about the generality of hypotheses. When using hypotheses in the form of logical formulae, the generality

[1]This section is based on Chap. 11 of the textbook on *Logical and Relational Learning* (De Raedt 2008).

relation coincides with that of logical entailment. Typically, a hypothesis G is said to be more general than a hypothesis S if G entails S, that is, if $G \models S$. Applying a deductive inference operator leads to specializations, and applying, inverted deductive ones, that is, inductive ones, leads to generalizations. A multitude of operators for generalization and specialization has been devised and are theoretically wellunderstood (Nienhuys-Cheng and de Wolf 1997; De Raedt 2008). Different operators exist that depend on the form of the hypotheses (single clause, multiple clause), the presence or absence of a background theory, the type of (deductive or inductive) inference rule applied, and the search strategy applied (heuristic or complete search). Many of the frameworks for generality can also be downgraded for use with more specialized representations, such as for instance graphs. The two most important frameworks for deciding whether one clause is more general than another one, are Plotkin's θ-subsumption (Plotkin 1969) and Malerba et al.'s OI-subsumption (Esposito et al. 1996). Specialized to graphs, these definitions correspond to the well-known notions of subgraph-isomorphism and -homeomorphism. As a consequence, it is easy (not to say straightforward) to adapt many of the results and operators of the subsumption frameworks to those of the graphmorphisms one. This can be used to obtain methods and algorithms for enumerating graphs with different properties. At the same time, some variants of the subsumption framework that take into account background knowledge in the form of sets of clauses or rules, might be adapted towards the graph mining setting, potentially leading to a new class of graph mining systems.

3.5.3 Three learning settings

The distinction between the model-theoretic and proof-theoretic perspective in logic has been used to define three settings for ILP that are applicable to different types of data. In the first setting, learning from interpretations (Blockeel et al. 1999), an example is a logical interpretation I, that is, a state-description or possible, world, and an example is covered by a hypothesis H (that is, a logical formula) if I is a model for H. In the second setting, learning from entailment (Muggleton and De Raedt 1994), an example corresponds to an observation about the truth or falsity of a formula F. A hypothesis H then covers the formula F if F is entailed by the hypothesis, that is, $H \models F$. In the final setting, learning from proofs (Passerini et al. 2006), an example is a proof (or a trace) and an example P is covered by a hypothesis H if P is a possible proof in the hypothesis H. Interpretations are the natural type of examples used in, for instance, Bayesian networks and item-set mining; observations in the form of true and false formulae are typical of scientific knowledge discovery problems, and proofs and traces are very natural when learning tree-bank grammars and Markov models. The settings provide different types of clues about the underlying target theory, and can be ordered according to difficulty. Proofs carry the most information, as they directly encode (instantiated) rules of the unknown target theory; interpretations provide full information about a specific example; whereas formulae summarize or aggregate information about multiple states (or interpretations). Therefore, learning from proofs is easier than learning from interpretations, which in turn is easier than learning from entailment (De Raedt 1997).

3.5.4 Unification and variables

A key difference between propositional logic and relational and first order logic lies in the use of variables and unification. This is a very powerful tool for Machine Learning and Data Mining in at least two respects. First, logical expressions that contain variables can

be used as general templates that make abstraction of specific instances. Knowledge based model construction applies this idea to generate graphical models. Consider, for instance, Domingos et al.'s Markov Logic (Domingos et al. 2006), in which a set S of weighted logical formulae of the form $w : f$ is used to construct a Markov network. This is realized by generating from each ground instance f of a formula $w : f$ some local fragment of the Markov network. On the one hand, the templates provide a general and compact description that allows one to deal with multiple extensions, and on the other, it also allows parameter tying which facilitates the learning. Second, variables and unification can also be used to propagate information. This is not only useful when performing deduction in logic, but also in the above sketched knowledge based model construction approach. For instance, in a logical Markov model context, abstract state transitions such as $p(X) \rightarrow q(X)$, where $p(X)$ is an abstract state, can be instantiated to grounded states, e.g., $p(c1)$. The abstract state transition can then be instantiated to $p(c1) \rightarrow q(c1)$ denoting that a transition to state $q(c1)$ occurs (perhaps with a particular probability). The value c1 is propagated from one state to the next and realizes a kind of memory in the Markov model. This mechanism by itself is important as it adds a lot of expressive power as shown for instance in Logical HMMs (Kersting et al. 2006). Even though the logical Markov model and Markov Logic use elements of logic, they go way beyond a purely logical representation as they merge graphical models and logic. As a consequence, logic is no longer used as a target language but rather as a means for realizing interesting intermediate representations.

3.5.5 Background knowledge

ILP has always stressed the importance of incorporating background knowledge in the learning process. It has enabled this by exploiting the underlying representation formalism, which typically supports the definition of intensional or view predicates or relations, which provide additional information about the domain of discourse. These predicates can then be used as any other predicate or relation in the learning process. This is a simple but powerful mechanism because (when using logic programs) basically any "programmable" form of background theory can be specified. In this way, even learning algorithms have been encoded as background predicates (cf. Craven and Slattery's WebKB 2001). Furthermore, because hypotheses are encoded in the same language, already learned hypotheses can be added to the background theory as well.

3.5.6 Propositionalization and aggregation

ILP has contributed several so-called propositionalization techniques (Kramer et al. 2001) that transform structured Machine Learning and Data Mining problems into a simpler format, typically a feature-vector or an attribute-value representation (Kramer et al. 2001; De Raedt 2008). The resulting problems can be directly input into (more) standard Machine Learning and Data Mining algorithms. Two types of techniques ca be distinguished: static propositionalization, which first maps the ILP problem into the simpler format and then invokes learners on the simpler representations, and dynamic approaches, which incrementally construct a set of good features by coupling the propositionalization step with the learning one (Landwehr et al. 2007). Aggregation often plays an important role in propositionalization, and at the same time, as aggregation results in a loss of information, so does propositionalization (Krogel and Wrobel 2001).

4 Some future directions

In Sect. 3.1 we saw how, over the last two decades, ILP has developed from a prodigious brainchild to a relatively mature area with applications published in the world's top scientific journals. Here, we present a somewhat eclectic collection of future directions for ILP.

4.1 ILP and probabilistic logical models as foundations of AI?

The multidimensional design space of intelligent agents proposed by Poole and Mackworth (2010) shows how ILP and probabilistic relational models can form part of the foundations of AI. This is useful from a historical perspective to understand existing research and as a vision for future research.

One dimension is that of *representation scheme*, which is about exploiting compactness for computational gains. One can represent the world explicitly in terms of states, or in terms of features. The features can be explicitly represented or be described in terms of individuals and relations. Another dimension is the *learning* dimension that specifies whether the agent's knowledge is given or learned from experience. While most Machine Learning is in terms of features, a definition of inductive Logic Programming could be learning in terms of individuals and relations.

Other dimensions have to do with uncertainty, whether actions are deterministic or stochastic and whether the world is fully observable or partially observable. Most probabilistic models are built on features (or so-called "random variables"). Uncertainty can be combined with individuals and relations giving languages for probabilistic relational models. Let us illustrate this idea on an example.

Statistical relational models are typically defined in terms of parametrized random variables (Poole 2003) which are often drawn in terms of plates (Buntine 1994). A parametrized random variable corresponds to a predicate or a function symbol in logic. It can include logical variables (which form the parameters). In the following examples, we will write logical variables (which denote individuals) in upper case, and constants, function and predicate symbols in lower case. We assume that the logical variables are typed, where the domain of the type, the set of individuals of the type, is called the population.

Parametrized random variables are best described in terms of an example. Consider the case of diagnosing students' performance in adding multi-digit numbers of the form

$$\begin{array}{rr} & x_1\ x_0 \\ + & y_1\ y_0 \\ \hline & z_2\ z_1\ z_0 \end{array}$$

A student, given values for the x's and the y's, provides values for the z's.

Whether a student gets the correct answer for z_i depends on x_i, y_i, the value carried in and whether she knows addition. Whether a student gets the correct carry depends on the previous x, y and carry, and whether she knowns how to carry. This dependency can be seen in Fig. 2.

Here $x(D, P)$ is a parametrized random variable. There is a random variable for each digit D and each problem P. A ground instance, such as $x(d_3, problem_{57})$, is a random variable that may represent the third digit of problem 57. Similarly, there is a z-variable for each digit D, problem P, student S, and time T. The plate notation can be read as duplicating the random variable for each tuple of individual the plate is parametrized by.

The basic principle used by all methods is that of *parameter sharing*: the instances of the parametrized random created by substituting constants for logical variables share the

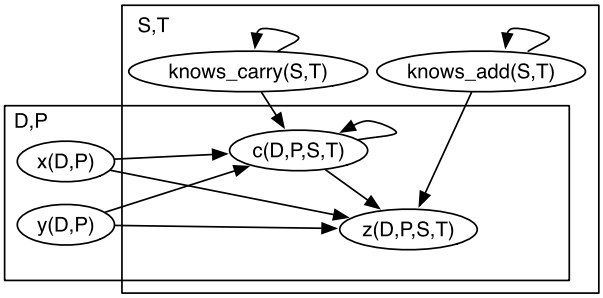

Fig. 2 Belief network with plates for multidigit addition

same probabilistic parameters. The various languages differ in how to specify the conditional probabilities of the variables variable given its parents, or the other parameters of the probabilistic model.

The first such languages (e.g., Horsch and Poole 1990), described the conditional probabilities directly in term of tables, and require a combination function (such as noisy-and or noisy-or) when there is a random variable parametrized by a logical variable that is a parent of a random variable that is not parametrized by the logical variable. Tables with combination functions turn out to be not a very flexible representation as they cannot represent the subtleties involved in how one random variable can depend on others.

In the above example, $c(D, P, S, T)$ depends, in part, on $c(D-1, P, S, T)$, that is, on the carry from the previous digit (and there is some other case for the first digit). A more complex example is to determine the probability that two authors are collaborators, which depends on whether they have written papers in common, or even whether they have written papers apart from each other.

To represent such examples, it is useful to be able to specify how the logical variables interact, as is done in logic programs. The independent choice logic (ICL) (Poole 1997, 2008) (originally called probabilistic Horn abduction Poole 1991, 1993) allows for arbitrary (acyclic) logic programs (including negation as failure) to be used to represent the dependency. The conditional probability tables are represented as independent probabilistic inputs to the logic program. A logic program that represents the above example is in Chap. 14 of (Poole and Mackworth 2010). This idea also forms the foundation for PRISM (Sato and Kameya 1997, 2008), which has concentrated on learning, and for ProbLog (De Raedt et al. 2007), a project to build an efficient and flexible language.

There is also work on undirected models, exemplified by Markov logic networks (Richardson and Domingos 2006), which have a similar notion of parametrized random variables, but the probabilities are derived from weights on first-order clauses. Such models have the advantage that they can represent cyclic dependencies, but there is no local interpretation of the parameters, and probabilistic inference relies on a global normalization. Avoiding a global normalization allows directed models to prune large parts of the model (or equivalently only consider part that are related to a query).

The work on statistical relational AI can be defined as the combination of probabilistic models, learning, reasoning with individuals and relations. However, there are many other dimensions that need to be incorporated to build an intelligent agent, including hierarchical representations, utilities, ongoing processes, multiple agents and bounded rationality. While there are many examples of the low-dimensional combinations (e.g., DTProbLog Van den Broeck et al. 2010 incorporates utilities in fully-observable models, and the original ICL (Poole 1997) incorporates utilities and partial observability but did not have efficient inference or learning), most of the design space is largely unexplored. It should be noted that

the same points in design space can be reached from different directions. For example, reinforcement learning (Sutton and Barto 1998), which has evolved from reasoning with explicit states to reasoning with features, has recently been expanded to reason with relations (van Otterlo 2009; Džeroski et al. 2001).

There are a number of connections with AI whose time has come to be explored more thoroughly including:

- Dealing with multiple heterogeneous data sets. We need ontologies to deal with the semantic interoperability, and incorporate uncertainty. There has been surprisingly little work on probabilistic reasoning and learning with datasets described using formal ontologies.
- Basing predictions on data that are not superficially relevant, but give important information. For example, in predicting the effects of global warming, not only are data about global temperatures relevant, but also laboratory experiments that show the effect of CO_2 on radiating heat are also relevant. We do not have learning methods that can use the other information effectively.
- Exploiting the potential efficiencies that are available from relational representations. Because relational representations generalize over individuals, there should be ways to exploit that for computational gain. There have been limited advances for lifted inference (Milch et al. 2008), but much remains to be done.

4.2 Whatever happened to predicate invention?

Predicate invention in ILP means automatic introduction of new, hopefully useful predicates during the process of learning from examples. Such new predicates can then be used as part of background knowledge in finding a definition of the target predicate. In the early years of ILP, in late 1980s and early 1990s, there was much excitement about predicate invention. It was viewed as one of the most exciting and challenging issues of ILP research; challenging because of its search complexity. A considerable proportion of ILP research in those years was devoted to this problem, with some interesting results.

A good review of that work was written by Stahl (1996). Relevant works from that period include the Duce system (Muggleton 1987), inverse resolution for first order predicate invention (Muggleton and Buntine 1988), and others (Bain and Muggleton 1991; Flach 1993; Morik et al. 1993; Wrobel 1994). Why was the motivation for research in predicate invention so strong, and still is? Automatically introducing new useful predicates means expanding the learner's hypothesis language. This is important as it enables more compact formulation of relevant hypotheses, or induction of successful hypotheses which would not otherwise be possible at all, for example in some cases of recursive formulation.

From the application point of view, predicate invention is so attractive because it is a most natural form of automated discovery. Predicate invention is a way of finding new theoretical terms, or abstract new concepts that are not directly observable in the measured data. Russell and Norvig (2010) comment: "Some of the deepest revolutions in science come from the invention of new predicates and functions—for example, Galileo's invention of acceleration, or Joule's invention of thermal energy. Once these terms are available, the discovery of new laws becomes (relatively) easy."

A recent European project (XPERO, www.xpero.org; for a simplified demo see www.ailab.si/xpero) demonstrates predicate invention for discovery of abstract concepts by an autonomous robot. In experiments with manipulating objects, the robot discovered the (never directly observed in measured data) notion of a movable object (Bratko et al. 2008). In another experiment which involved several robots of different strength, the notion of 'object X

is movable by robot R' was invented. When the robot was planning paths, the concept of an obstacle was invented. When planning to move boxes to their goal positions, the concept of a tool was automatically introduced (Bratko 2010).

Unfortunately, the research in the past 15 years did not keep pace with the level of activity of the enthusiastic period around 1990. It seems that there is a general agreement among researchers, also expressed by Dietterich et al. (2008), that (a) predicate invention is a key problem in ML, and (b) it is exceptionally hard due to its high combinatorial complexity, maybe too hard. The number of alternatives in which new predicates can be introduced is extremely explosive, and recognizing their (potential) benefits early in the search among all these alternatives is very hard.

However, given the importance of predicate invention and the early successes, it is still surprising that not more research is being devoted to this problem. More work on this topic should be encouraged in the future of ILP research. If breakthroughs will be difficult to achieve, they will be very rewarding.

4.3 Next-generation ILP applications and theory?

Within ILP demanding applications have often led to the development of new implementations and associated theory. One cutting-edge application class is found in Synthetic Biology. This area aims at modifying micro-organisms in order to achieve new functions. For instance, organisms can be redesigned to consume toxic waste or to transform domestic waste into fuels. For ILP to provide support for the design tasks in areas like Synthetic Biology would require advances on a number of fronts. A successful ILP system would require not only an ability to provide accurate biochemical models, but also an experimental framework which would support the ability to suggest design modifications which alter the existing network to achieve new functions. This would require conceptual advances in least the following areas.

Higher-order and functional extensions. John Lloyd (2003) has argued the case for ILP to use higher-order logic as an underlying representation for learning. Lloyd's arguments largely relates to the advantages of a functional representation. However, higher-order representations could also provide more flexible ways of representing background knowledge which could support, for example, general definitions of the properties of symmetric relations or tail-recursive definitions. Higher-order background knowledge also has potential for direct reasoning about meta-logical processes such as predicate invention and probabilistic reasoning.

Learning actions and strategies. The majority of ILP systems and applications still represent variants of concept learning. However, many important classes of relational real-world problems exist which require action-based formalisms. Examples of such application classes include planning, scheduling, design and robotics. Research has started in the ILP, UAI and planning communities to learn about actions, plans and employ decision theory (Moyle and Muggleton 1997; Otero 2005; Sanner and Kersting 2010; van Otterlo 2009; Džeroski et al. 2001).

Theory of experimentation. It is clear from the Robot Scientist project that ILP has important applications in areas of experimental science. Given this, it is vital to develop a comprehensive experimental framework for ILP which incorporates probabilities of experimental outcomes and costs of experiments. One of the limitations of the Robot Scientist approach is the assumption of a fixed and finite set of experiments. The details of the experimental protocol and instrument description lay outside of the background knowledge used. In science, these details and choices are often critical.

Agency. Reasoning about agents is important in many real-world applications. Such applications include ecological modelling, analysis of social networks and economic modelling.

4.4 Wanted: nuts, bolts & co.

4.4.1 The engineering of ILP systems: why?

Over a decade ago, C.A.R. Hoare gave a public lecture, in which he posed two questions: "Is computing Science?" and "Is software Engineering?" Positive answers depended on the extent to which well-understood scientific principles of abstraction, mathematical modelling and logical deduction were employed in establishing the foundations of the field; and engineering design principles of correctness, flexibility, decomposition and re-use were employed in the development of software. These questions just as easily apply to ILP. That is: "Is ILP Science?" and "Is ILP software Engineering?" Over the past two decades, a steadfast committment to addressing conceptual questions using mathematics and logic has done much to provide a positive answer to the first question. But it is to the second—whether ILP systems are sufficiently well-engineered—that is the concern of this section.

Why is this question important? Let us consider a related question. Despite mathematical clarity and any number of demonstrations of its applicability to complex modelling tasks, few would disagree that ILP remains a connoisseur's choice of modelling techniques. Why is this so? Current ILP systems do have shortcomings that hinder their use in routine data analysis:

Formalism. The principal concerns here are twofold: (a) the use of logic (usually Prolog) programs as a representation language; and (b) the need to provide adequate background knowledge. Difficulties from the former arise with the lack of familiarity of logic programs amongst non-ILP specialists. How many data analysts, and domain experts can be expected to understand the inputs and output of an ILP system? With background knowledge, problems arise because an ILP system's performance is heavily dependent on the presence of adequate background knowledge. How much and what kind of background knowledge can be termed "adequate" for a particular problem?

Use. Again, there are two main concerns: (a) efficiency, arising from the size of the space of possible models, testing models and the size of datasets; and (b) the lack of a set of well-engineered tools that allow a data analyst experimentation with and testing of scientific advances, visualisation of results, and modifications to existing functionality.

Some steady progress is being made in addressing all of these concerns—except the last. Here, there has been little to no work done: while over 100 ILP systems have been constructed since 1991, less than a handful can even begin to be used meaningfully by ILP practitioners other than the original developers. System descriptions, operating characteristics, guidelines on use and sensitivity analyses are practically non-existent. The professional ILP data analyst is thus a rarity, and the general data analyst capable of using ILP tools perhaps non-existent. An additional unfortunate feature is that the implementation effort of the last two decades has not been cumulative. As a result, each of the 100 or more systems have re-implemented many parts that exist in other systems. The need for well-engineered, reusable tools with clearly understood specifications and properties perhaps does not greatly matter at the early stages of a field of science, but will increasingly play an important role if the field is to move beyond the realm of theory to one of applications. So it is with ILP.

4.4.2 The engineering of ILP systems: how?

We are at a point where ILP now has some substantial scientific momentum: what is needed is a commitment to establish a corresponding engineering momentum. This requires the establishment of a world-wide ILP development initiative—perhaps as a co-ordinated set of graduate-level projects—that has as its goal the development of re-usable software components that are flexible enough to build software useful for the ILP trade, and by the ILP hobbyist for DIY ILP. We enviage something like the model-building tool-set Meccano™. Here is the introduction to this from http://en.wikipedia.org/wiki/Meccano:

> Meccano is a model construction system comprising re-usable metal strips, plates, angle girders, wheels, axles and gears, with nuts and bolts to connect the pieces. It enables the building of working models and mechanical devices.

Imagine what we could do with an ILP Meccano Set! At the very least, such a set will need to contain software components that are fundamental to all, or at least most, ILP systems. These components would have to be: understandable to ILP system builders; useful, in the sense of being reusable; and have a functionality that is clearly specified. In addition, we would like the set of components to be reasonably complete, in order that interesting systems could be built from them. In software terms, this means the development of components for hypothesis construction, combinatorial search, experimentation, visualisation *etc*.; with built-in mechanisms for parallelism, and connection to statistical packages, external databases and so on. In software terms, this means the development of components for hypothesis construction, combinatorial search, experimentation, visualisation and so on.

At this point, the reader may be concerned that nothing said here appears to be directed at making ILP more usable by domain-experts (like biologists, for example). This is indeed so: the ILP Meccano Set is for ILP practitioners and data analysts. But this is a start, and the onus rests squarely on the ILP community to construct well-engineered, professional ILP systems. To paraphrase the Rabbi Hillel: "If not us, Who? If not now, When?"

5 Conclusion: ILP's next 20 years: back to AI and logic?

The past twenty years of ILP has not only provided firm mathematical foundations for the future, but also many new insights into the potential of logic-based representations for machine learning. However, it is also clear that many challenging open problems still confront the field, both in terms of basic representation, theory and efficient implementation of systems. Further developments in these areas promise increasing breadth of application for ILP systems in the future.

An example of the ongoing integrated development of theory and application in ILP can be found in the development of ILP systems for scientific discovery. Applications in Systems Biology and other domains in the last few years has demanded basic theories of learning about causality and learning in network structures. *Reasoning about action* has been a classical topic in AI, but its connection to ILP has not yet been fully explored. There are some basic studies on learning theories in situation calculus, event calculus (Moyle and Muggleton 1997) or action theories (Otero 2005), but more development is necessary to deal with practical applications. In particular, *learning dynamics of systems* is a challenging topic, in which we need to infer the past and future status of a system and the physical laws that govern the system with possible feedback loops from dynamically changing observations. We also need to have *hybrid systems* dealing with quantitative information. There are

several ways to combine symbolic and numerical approaches, based on such as qualitative abstraction, discretization, interval-based constraints, probability and fuzzy logic, but ILP must contribute more to a basic theory in this direction.

Taking the systems' viewpoint, our goal should become closer to that in AI. However, many existing ILP systems have been based on classical syntax and semantics of Logic Programming, yet the field of Logic Programming has changed in these 20 years, shifting from Horn logic programs computed with SLD(NF) resolution toward *constraint* and *answer set programming*. This change reflects incompatible demands for more expressive *knowledge representation* and for *scalability*. We can even take further steps forward on knowledge representation. In fact, logic programs and related structural representations are not necessarily used for symbolic methods. There has been much work on powerful or tractable languages of description logic (Lisi and Malerba 2003), action theories and constraints. Thus, covering relations and refinement techniques for extended classes of logic programs should be soon explored. As for applications to systems biology and dynamic domains, negative effects like inhibition and deletion frequently appear in causal networks, whose effects are similar to nonmonotonic negation.

As for the scalability issue, we also need to take advantage of those techniques developed in other fields of computer science. For example, implementation based on *SAT* technologies could be more explored in ILP. Related to SAT, *SMT* (satisfiability modulo theories) is also suitable for more complex domains. Learning in *parallel* and *distributed* environments (Muggleton et al. 2002; Graham et al. 2003) is also necessary to solve huge problems as well as physically distributed problems.

Another important goal of systems biology is the revision of biological networks (Tamaddoni-Nezhad et al. 2006, 2007). This requires not only building theories but *updating* them. Knowledge-based updating has been formally explored in logic-based AI, but has not been connected to logic learning in ILP. *Predicate invention* (Muggleton and Buntine 1988) was introduced in the early days of ILP but has not been well addressed recently, and is still one of the hard ILP tasks. Understanding predicate invention from the viewpoint of semantics of theory update is thus important now. A method to discover unknown relations from incomplete networks has been recently introduced in Inoue et al. (2010) based on *meta-level abduction*, which infer missing rules, missing facts, and unknown causes that involve predicate invention in the form of existentially quantified hypotheses. This invention is also a realization of *hidden object invention* (Dietterich et al. 2008), but lifting meta-level abduction to higher-order is also a promising method in this direction.

6 Further reading

In order to find out more about the field of Inductive Logic Programming, we would like to point the reader to a number of books such as Muggleton (1992), a historical collection of early ILP papers (Lavrač and Džeroski 1993), an early text book on ILP (Nienhuys-Cheng and de Wolf 1997), providing a theoretical perspective on ILP, and Džeroski and Lavrač (2001), a collection of papers on ILP giving a data mining perspective. More recent collections of papers on the probabilistic perspective in ILP are contained in De Raedt et al. (2008), Getoor and Taskar (2007). De Raedt (2008) is an up-to-date textbook on logical and relational learning; shorter introductory background on many of the topics discussed can be found in the Encyclopedia of Machine Learning (Sammut and Webb 2010).

Author contributions

SHM wrote Sects. 1, 3.1, 3.2 and 4.3. PF wrote Sect. 3.3; LDR wrote Sect. 3.5 based on Chap. 11 of the book (De Raedt 2008) and Sect. 6; KI wrote Sects. 3.4 and 5; DP wrote Sects. 2, 4.1; IB wrote Sect. 4.2; and AS wrote Sect. 4.4.

Acknowledgements SHM would like to thank the Royal Academy of Engineering and Microsoft Research for their support of his research chair. SM also acknowledges the support of Syngenta Ltd in the University Innovation Centre at Imperial College and the BBSRC for their funding of the Centre for Integrative Systems Biology at Imperial College. AS is a Ramanujan Fellow of the Government of India; and for some period of this work was at the School of Computational and Integrative Sciences, Jawaharlal Nehru University, New Delhi. AS is also a Visiting Professor at the Computing Laboratory, University of Oxford; and an Adjunct Professor at the School of CSE, University of New South Wales, Sydney.

References

Bain, M., & Muggleton, S. H. (1991). Non-monotonic learning. In D. Michie (Ed.), *Machine intelligence* (Vol. 12, pp. 105–120). London: Oxford University Press.

Blockeel, H., & De Raedt, L. (1997). Lookahead and discretisation in ILP. In N. Lavrač & S. Džeroski (Eds.), *LNAI: Vol. 1297. Proceedings of the seventh international workshop on inductive logic programming* (pp. 77–84). Berlin: Springer.

Blockeel, H., De Raedt, L., Jacobs, N., & Demoen, B. (1999). Scaling up inductive logic programming by learning from interpretations. *Data Mining and Knowledge Discovery*, *3*(1), 59–93.

Bratko, I. (2010). Discovery of abstract concepts by a robot. In *LNAI: Vol. 6332. Proceedings of discovery science 2010* (pp. 372–379). Berlin: Springer.

Bratko, I., Muggleton, S. H., & Varsek, A. (1991). Learning qualitative models of dynamic systems. In *Proceedings of the eighth international machine learning workshop*, San Mateo, CA. San Mateo: Morgan-Kaufmann.

Bratko, I., Leban, G., & Žabkar, J. (2008). An experiment in robot discovery with ilp. In *Proceedings of the 18th international conference on inductive logic programming (ILP 2008)*. Berlin: Springer.

Buntine, W. L. (1994). Operations for learning with graphical models. *Journal of Artificial Intelligence Research*, *2*, 159–225.

Chen, J., Muggleton, S. H., & Santos, J. (2008). Learning probabilistic logic models from probabilistic examples. *Machine Learning*, *73*(1), 55–85. doi:10.1007/s10994-008-5076-4.

Cohen, W. (1993). PAC-learning a restricted class of logic programs. In S. Muggleton (Ed.), *Proceedings of the 3rd international workshop on inductive logic programming* (pp. 41–72).

Corapi, D., Russo, A., & Lupu, E. (2010). Inductive logic programming as abductive search. In *Technical communications of ICLP'10* (pp. 54–63).

Craven, M., & Slattery, S. (2001). Relational learning with statistical predicate invention: Better models for hypertext. *Machine Learning*, *43*(1/2), 97–119.

Cussens, J. (2001). Parameter estimation in stochastic logic programs. *Machine Learning*, *44*(3), 245–271.

Davis, J., & Domingo, P. (2009). Deep transfer via second-order markov logic. In *Proceedings of the twenty-sixth international workshop on machine learning* (pp. 217–224). San Mateo: Morgan Kaufmann.

De Raedt, L. (1997). Logical settings for concept-learning. *Artificial Intelligence*, *95*(1), 197–201.

De Raedt, L. (2008). *Logical and relational learning*. Berlin: Springer.

De Raedt, L., & Bruynooghe, M. (1991). Clint: a multistrategy interactive concept-learner and theory revision system. In *Proceedings of the 1st international workshop on multistrategy learning* (pp. 175–191). San Mateo: Morgan Kaufmann.

De Raedt, L., & Kersting, K. (2004). Probabilistic inductive logic programming. In S. Ben-David, J. Case, & A. Maruoka (Eds.), *Lecture notes in computer science: Vol. 3244. Proceedings of the 15th international conference on algorithmic learning theory*. Berlin: Springer.

De Raedt, L., & Lavrač, N. (1996). Multiple predicate learning in two inductive logic programming settings. *Journal on Pure and Applied Logic*, *4*(2), 227–254.

De Raedt, L., Kimmig, A., & Toivonen, H. (2007). ProbLog: a probabilistic Prolog and its application in link discovery. In R. Lopez de Mantaras & M.M. Veloso (Eds.), *Proceedings of the 20th international joint conference on artificial intelligence (IJCAI-2007)* (pp. 2462–2467).

De Raedt, L., Frasconi, P., Kersting, K., & Muggleton, S. H. (Eds.) (2008). *LNAI: Vol. 4911. Probabilistic inductive logic programming*. Berlin: Springer.

Dehaspe, L., & Toivonen, H. (2001). Discovery of relational association rules. In Džeroski, S., & Lavrač, N. (Eds.), *Relational data mining* (pp. 189–212). Berlin: Springer.

Dietterich, T., Domingos, P., Getoor, L., Muggleton, S. H., & Tadepalli, P. (2008). Structured machine learning: the next ten years. *Machine Learning*, *73*(1), 3–23. doi:10.1007/s10994-008-5079-1.

Dolsak, B., & Muggleton, S. H. (1992). The application of Inductive Logic Programming to finite element mesh design. In S. H. Muggleton (Ed.), *Inductive logic programming* (pp. 453–472). London: Academic Press.

Domingos, P. S., Kok, S., Poon, H., Richardson, M., & Singla, P. (2006). Unifying logical and statistical ai. In *Proceedings of the twenty-first national conference on artificial intelligence, AAAI06* (pp. 2–7). Menlo Park/Cambridge: AAAI Press/MIT Press.

Džeroski, S., & Lavrač, N. (Eds.) (2001). *Relational data mining*. Berlin: Springer.

Džeroski, S., Muggleton, S. H., & Russell, S. (1993). Learnability of constrained logic programs. In *Proceedings of the European conference on machine learning* (pp. 342–347). London: Springer.

Džeroski, S., De Raedt, L., & Driessens, K. (2001). Relational reinforcement learning. *Machine Learning*, *43*(1/2), 5–52.

Emde, W., & Wettschereck, D. (1996). Relational instance-based learning. In *Proceedings of the 13th international machine learning conference* (pp. 122–130).

Esposito, F., Laterza, A., Malerba, D., & Semeraro, G. (1996). Refinement of Datalog programs. In *Proceedings of the MLnet familiarization workshop on data mining with inductive logic programming* (pp. 73–94).

Feng, C. (1992). Inducing temporal fault diagnostic rules from a qualitative model. In S. H. Muggleton (Ed.), *Inductive logic programming*. London: Academic Press.

Flach, P. (1993). Predicate invention in inductive data engineering. In P. B. Brazdil (Ed.), *Lecture notes in artificial intelligence: Vol. 667. Machine learning: ECML-93* (pp. 83–94). Berlin: Springer.

Flach, P. A., & Kakas, A. C. (Eds.) (2000). *Abduction and induction: essays on their relation and integration*. Dordrecht: Kluwer Academic.

Getoor, L., & Taskar, B. (Eds.) (2007). *An introduction to statistical relational learning*. Cambridge: MIT Press.

Getoor, L., Friedman, N., Koller, D., & Pfeffer, A. (2001). Learning probabilistic relational models. In Džeroski, S., & Lavrač, N. (Eds.), *Relational data mining* (pp. 307–335). Berlin: Springer.

Graham, J. H., Page, C. D., & Kamal, A. H. (2003). Accelerating the drug design process through parallel inductive logic programming data mining. In *Proceedings of the IEEE computer society bioinformatics conference—CSB* (pp. 400–402). New York: IEEE Press.

Horsch, M., & Poole, D. L. (1990). A dynamic approach to probabilistic inference using Bayesian networks. In *Proc. sixth conference on uncertainty in AI*, Boston, July 1990 (pp. 155–161).

Inoue, K. (2004). Induction as consequence finding. *Machine Learning*, *55*, 109–135.

Inoue, K., Furukawa, K., Kobayashiand, I., & Nabeshima, H. (2010). Discovering rules by meta-level abduction. In L. De Raedt (Ed.), *LNAI: Vol. 5989. Proceedings of the nineteenth international conference on inductive logic programming (ILP09)* (pp. 49–64). Berlin: Springer.

Kersting, K., & De Raedt, L. (2001). Towards combining inductive logic programming with bayesian networks. In *LNAI: Vol. 2157. Proceedings of the eleventh international conference on inductive logic programming* (pp. 118–131). Berlin: Springer.

Kersting, K., De Raedt, L., & Raiko, T. (2006). *Logical Hidden Markov Models*, *25*, 425–456.

Khardon, R. (1998). Learning first order universal Horn expressions. In *Proceedings of the eleventh annual ACM conference on computational learning theory* (pp. 154–165). New York: ACM.

Kietz, J. U. (1993). Some lower bounds on the computational complexity of inductive logic programming. In P. Brazdil (Ed.), *Lecture notes in artificial intelligence: Vol. 667. Proceedings of the 6th European conference on machine learning* (pp. 115–123). Berlin: Springer.

King, R. D., Muggleton, S. H., Srinivasan, A., & Sternberg, M. J. E. (1996). 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.

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

King, R. D., Rowland, J., Oliver, S. G., Young, M., Aubrey, W., Byrne, E., Liakata, M., Markham, M., Pir, P., Soldatova, L. N., Aparkes, A., Whelan, K. E., & Clare, A. (2009). The automation of science. *Science*, *324*(5923), 85–89.

Knobbe, A. J., Siebes, A., & Marseille, B. (2002). Involving aggregate functions in multi-relational search. In *Proceedings of the 6th European conference on data mining principles and practice of knowledge discovery in databases* (p. 1).

Kramer, S., Lavrač, N., & Flach, P. (2001). Propositionalization approaches to relational data mining. In S. Džeroski & N. Lavrač (Eds.), *Relational data mining* (pp. 262–291). Berlin: Springer.

Krogel, M.-A., & Wrobel, S. (2001). Transformation-based learning using multirelational aggregation. In *LNCS: Vol. 2157. Inductive logic programming* (pp. 142–155).

Landwehr, N., Kersting, K., & De Raedt, L. (2007). Integrating naive Bayes and Foil. *Journal of Machine Learning Research*, *8*, 481–507.

Lavrač, N., & Džeroski, S. (1993). *Inductive logic programming: techniques and applications*. Chichester: Ellis Horwood.

Lavrač, N., Džeroski, S., & Grobelnik, M. (1991). Learning non-recursive definitions of relations with LINUS. In Y. Kodratoff (Ed.), *Lecture notes in artificial intelligence: Vol. 482. Proceedings of the 5th European working session on learning*. Berlin: Springer.

Lisi, F. A., & Malerba, D. (2003). Bridging the gap between horn clausal logic and description logics in inductive learning. In *LNCS: Vol. 2829. AI*IA 2003: Advances in artificial intelligence*. Berlin: Springer.

Lloyd, J. W. (2003). *Logic for learning*. Berlin: Springer.

Mihalkova, L., & Mooney, R. J. (2009). Transfer learning from minimal target data by mapping across relational domains. In *IJCAI-09: Proceedings of the twentieth international joint conference on artificial intelligence* (pp. 1163–1168). San Mateo: Morgan-Kaufmann.

Milch, B., Zettlemoyer, L. S., Kersting, K., Haimes, M., & Kaelbling, L. P. (2008). Lifted probabilistic inference with counting formulas. In *Proceedings of the twenty third conference on artificial intelligence (AAAI)*.

Morik, K., Wrobel, S., Kietz, J., & Emde, W. (1993). *Knowledge acquisition and machine learning: theory, methods and applications*. London: Academic Press.

Moyle, S., & Muggleton, S. H. (1997). Learning programs in the event calculus. In N. Lavrač & S. Džeroski (Eds.), *LNAI: Vol. 1297. Proceedings of the seventh inductive logic programming workshop (ILP97)* (pp. 205–212). Berlin: Springer.

Muggleton, S. H. (1987). Duce, an oracle based approach to constructive induction. In *IJCAI-87* (pp. 287–292). Los Altos: Kaufmann.

Muggleton, S. H. (1991). Inductive logic programming. *New Generation Computing*, *8*(4), 295–318.

Muggleton, S. H. (Ed.) (1992). *Inductive logic programming*. San Diego: Academic Press.

Muggleton, S. H. (1995). Inverse entailment and Progol. *New Generation Computing*, *13*, 245–286.

Muggleton, S. H. (1996). Stochastic logic programs. In L. de Raedt (Ed.), *Advances in inductive logic programming* (pp. 254–264). Amsterdam: IOS Press.

Muggleton, S. H. (2002). Learning structure and parameters of stochastic logic programs. In *Proceedings of the 12th international conference on inductive logic programming* (pp. 198–206). Berlin: Springer.

Muggleton, S. H., & Bryant, C. H. (2000). Theory completion using inverse entailment. In *Proc. of the 10th international workshop on inductive logic programming (ILP-00)* (pp. 130–146). Berlin: Springer.

Muggleton, S. H., & Buntine, W. (1988). Machine invention of first-order predicates by inverting resolution. In *Proceedings of the 5th international conference on machine learning* (pp. 339–352). Los Altos: Kaufmann.

Muggleton, S. H., & De Raedt, L. (1994). Inductive logic programming: Theory and methods. *Journal of Logic Programming*, *19–20*, 629–679.

Muggleton, S. H., & Feng, C. (1990). Efficient induction of logic programs. In *Proceedings of the first conference on algorithmic learning theory* (pp. 368–381). Tokyo: Ohmsha.

Muggleton, S. H., & Feng, C. (1992). Efficient induction of logic programs. In S. H. Muggleton (Ed.), *Inductive logic programming* (pp. 281–298). London: Academic Press.

Muggleton, S. H., King, R. D., & Sternberg, M. J. E. (1992). Protein secondary structure prediction using logic-based machine learning. *Protein Engineering*, *5*(7), 647–657.

Muggleton, S. H., Fidjeland, A., & Luk, W. (2002). Scalable acceleration of inductive logic programs. In *IEEE international conference on field-programmable technology* (pp. 252–259). New York: IEEE Press.

Nienhuys-Cheng, S.-H., & de Wolf, R. (1997). *LNAI: Vol. 1228. Foundations of inductive logic programming*. Berlin: Springer.

Otero, R. (2005). Induction of the indirect effects of actions by monotonic methods. In *Proceedings of the fifteenth international conference on inductive logic programming (ILP05)* (Vol. 3625, pp. 279–294). Berlin: Springer.

Passerini, A., Frasconi, P., & De Raedt, L. (2006). Kernels on Prolog proof trees: statistical learning in the ILP setting. *Journal of Machine Learning Research*, *7*, 307–342.

Plotkin, G. D. (1969). A note on inductive generalisation. In B. Meltzer & D. Michie (Eds.), *Machine intelligence* (Vol. 5, pp. 153–163). Edinburgh: Edinburgh University Press.

Plotkin, G. D. (1971a). *Automatic methods of inductive inference*. Ph.D. thesis, Edinburgh University, August 1971.

Plotkin, G. D. (1971b). A further note on inductive generalization. In *Machine intelligence* (Vol. 6). Edinburgh: Edinburgh University Press.

Poole, D. L. (1991). Representing diagnostic knowledge for probabilistic Horn abduction (pp. 1129–1135). Sydney.

Poole, D. L. (1993). Probabilistic Horn abduction and Bayesian networks. *Artificial Intelligence*, *64*(1), 81–129.

Poole, D. L. (1997). The independent choice logic for modelling multiple agents under uncertainty. *Artificial Intelligence*, *94*, 7–56. Special issue on economic principles of multi-agent systems.

Poole, D. L. (2000). Abducing through negation as failure: stable models within the independent choice logic. *Journal of Logic Programming*, *44*(1–3), 5–35.

Poole, D. L. (2003). First-order probabilistic inference. In *Proc. eighteenth international joint conference on artificial intelligence (IJCAI-03)*, Acapulco, Mexico (pp. 985–991).

Poole, D. L. (2008). The independent choice logic and beyond. In L. De Raedt, P. Frasconi, K. Kersting, & S. Muggleton (Eds.), *LNCS: Vol. 4911. Probabilistic inductive logic programming: theory and application*. Berlin: Springer.

Poole, D. L., & Mackworth, A. K. (2010). *Artificial intelligence: foundations of computational agents*. New York: Cambridge University Press.

Quinlan, J. R. (1987). Generating production rules from decision trees. In *Proceedings of the tenth international conference on artificial intelligence* (pp. 304–307). Los Altos: Kaufmann.

Quinlan, J. R. (1990). Learning logical definitions from relations. *Machine Learning*, *5*, 239–266.

Quinlan, J. R., & Cameron-Jones, R.M. (1993). FOIL: a midterm report. In P. Brazdil (Ed.), *Lecture notes in artificial intelligence: Vol. 667. Proceedings of the 6th European conference on machine learning* (pp. 3–20). Berlin: Springer.

Ray, O., Broda, K., & Russo, A. (2003). Hybrid abductive inductive learning: a generalisation of Progol. In *Lecture notes in artificial intelligence: Vol. 2835. Proceedings of the 13th international conference on inductive logic programming (ILP'03)* (pp. 311–328). Berlin: Springer.

Richardson, M., & Domingos, P. (2006). Markov logic networks. *Machine Learning*, *62*, 107–136.

Rouveirol, C., & Puget, J.-F. (1989). A simple and general solution for inverting resolution. In *EWSL-89* (pp. 201–210). London: Pitman.

Russell, S. J., & Norvig, P. (2010). *Artificial intelligence: a modern approach* (3rd ed.). New Jersey: Pearson.

Sammut, C., & Banerji, R.B. (1986). Learning concepts by asking questions. In R. Michalski, J. Carbonnel, & T. Mitchell (Eds.), *Machine learning: an artificial intelligence approach* (Vol. *2*, pp. 167–192). Los Altos: Kaufmann.

Sammut, C., & Webb, G. (Eds.) (2010). *Encyclopedia of machine learning*. Berlin: Springer.

Sanner, S., & Kersting, K. (2010). Symbolic dynamic programming. In C. Sammut & G. Webb (Eds.), *Encyclopedia of machine learning*. Berlin: Springer.

Santos Costa, V., Page, D., Qazi, M., & Cussens, J. (2003). CLP(BN): Constraint logic programming for probabilistic knowledge. In *Proceedings of the 19th conference on uncertainty in artificial intelligence* (pp. 517–524).

Sato, T. (2005). Generative modeling with failure in prism. In *International joint conference on artificial intelligence* (pp. 847–852). San Mateo: Morgan Kaufmann.

Sato, T., & Kameya, Y. (1997). PRISM: a symbolic-statistical modeling language. In *Proceedings of the 15th international joint conference on artificial intelligence (IJCAI-97)* (pp. 1330–1335).

Sato, T., & Kameya, Y. (2008). New advances in logic-based probabilistic modeling by PRISM. In L. De Raedt, P. Frasconi, K. Kersting, & S. Muggleton (Eds.), *LNCS: Vol. 4911. Probabilistic inductive logic programming* (pp. 118–155). Berlin: Springer.

Shapiro, E. Y. (1983). *Algorithmic program debugging*. Cambridge: MIT Press.

Stahl, I. (1992). *Constructive induction in inductive logic programming: an overview* (Technical report). Fakultat Informatik, Universitat Stuttgart.

Stahl, I. (1996). Predicate invention in inductive logic programming. In L. De Raedt (Ed.), *Advances in inductive logic programming* (pp. 34–47). Amsterdam: IOS Press.

Sutton, R. S., & Barto, A. G. (1998). *Reinforcement learning: an introduction*. Cambridge: MIT Press.

Synnaeve, G., Inoue, K., Doncescu, A., Kameya, Y., Sato, T., Ishihata, M., & Nabeshima, H. (2011). Kinetic models and qualitative abstraction for relational learning in systems biology. In *Proceedings of the international conference on bioinformatics models, methods and algorithms*.

Tamaddoni-Nezhad, A., Chaleil, R., Kakas, A., & Muggleton, S. H. (2006). Application of abductive ILP to learning metabolic network inhibition from temporal data. *Machine Learning*, *64*, 209–230. doi:10.1007/s10994-006-8988-x.

Tamaddoni-Nezhad, A., Chaleil, R., Kakas, A., Sternberg, M. J. E., Nicholson, J., & Muggleton, S. H. (2007). Modeling the effects of toxins in metabolic networks. *IEEE Engineering in Medicine and Biology*, *26*, 37–46. doi:10.1109/MEMB.2007.335590.

Torrey, L., & Shavlik, J. W. (2010). Policy transfer via Markov logic networks. In L. De Raedt (Ed.), *LNAI: Vol. 5989. Proceedings of the nineteenth international conference on inductive logic programming (ILP09)* (pp. 234–248). Berlin: Springer.

Van den Broeck, G., Thon, I., van Otterlo, M., & De Raedt, L. (2010). DTProbLog: A decision-theoretic probabilistic prolog. In *Proceedings of the AAAI conference on artificial intelligence (AAAI 2010)*.

van Otterlo, M. (2009). *The logic of adaptive behavior—knowledge representation and algorithms for adaptive sequential decision making under uncertainty in first-order and relational domains*. Amsterdam: IOS Press.

Vens, C., Ramon, J., & Blockeel, H. (2006). Refining aggregate conditions in relational learning. In J. Fürnkranz, T. Scheffer, & M. Spiliopoulou (Eds.), *Lecture notes in computer science: Vol. 4213. Proceedings of the 10th European conference on principles and practice of knowledge discovery in databases* (pp. 383–394). Berlin: Springer.

Vere, S. A. (1975). Induction of concepts in the predicate calculus. In *Proceedings of the 4th international joint conference on artificial intelligence* (pp. 282–287). San Mateo: Morgan Kaufmann.

Wrobel, S. (1994). Concept formation during iterative theory revision. *Machine Learning*, *14*, 169–191.

Yamamoto, Y., Inoue, K., & Iwanuma, K. (2010). From inverese entailment to inverese subsumption. In *Proceedings of the 20th international conference on inductive logic programming (ILP'10)*.