

Deductive Database Integration and Knowledge Discovery

Carolina Ruiz
Department of Computer Science
Worcester Polytechnic Institute
Worcester, MA 01602 USA
ruiz@cs.wpi.edu
<http://www.cs.wpi.edu/~ruiz/>

Abstract

Knowledge Discovery in Databases (KDD) and Deductive Databases (DDBs) are two important areas of contemporary research in artificial intelligence and databases. In this paper, we explore connections between these research areas and describe potentially beneficial cooperations between them. We survey some approaches that relate KDD and DDBs and describe our contributions to knowledge discovery in deductive databases by means of deductive database integration. We also mention our work on the alternative rule formalism of association rule mining, and propose comparing the merits of logical rules relative to those of association rules.

1 Introduction

Deductive Databases (DDBs) is a subfield of Logic Programming (LP). A DDB is an assemblage of two components: A knowledge base, which is a collection of statements about the world being represented, and an inference system, which is a mechanism in charge of the extraction and derivation of information from the knowledge base. The statements in the knowledge base typically take the form of *facts* that are *true* in the world, and *rules* that describe general laws satisfied in the world. These facts and rules, as well as the inference mechanism, are commonly formalized in terms of mathematical logic.

Knowledge Discovery in Databases (KDD) has been defined as the “nontrivial process of identifying valid, novel, potentially useful, and ultimately understandable patterns in data” [8]. KDD techniques usually rely on the assumption that the data are stored in a relational database or in a relational-like table. There have been some investigation of knowledge discovery in richer databases, among those, object oriented databases and active databases [18, 33]. There seems to have been less work on exploring interconnections between knowledge discovery and deductive databases.

It is well-accepted in the KDD and in the machine learning communities that background knowledge plays

an important role in the discovery process. Deductive databases provide a powerful knowledge representation mechanism that make them suitable for representing and deploying background knowledge for and during the discovery process. Reciprocally, deductive databases can benefit from the wide variety of application domains that KDD is currently exploring and that are generating a great deal of interest in academic and industrial settings.

2 Relationships between Deductive Databases and the KDD Process

Fayyad et al. [8] characterize knowledge discovery in databases as an interactive, iterative, user-driven process. They have identified nine steps for this process. Here we group those nine steps into three rough categories:

- Data and Knowledge Collection and Integration: (1) Developing an understanding of application domain, relevant prior knowledge, and user’s goal; (2) Creating a target data set; (3) Data cleaning and preprocessing including removing noise, collecting necessary information, dealing with missing values; and (4) Data reduction and projection, that is, finding useful features to represent the data depending on the user’s goal.
- Data Mining: (1) Matching the goals to a particular data mining approach; (2) Selecting data mining technique and appropriate parameters; and (3) Application of specific algorithms for extracting patterns from the data.
- Evaluation: (1) Interpreting mined patterns, using for instance visualization tools; and (2) Using the knowledge by incorporating it into the same or other systems.

As mentioned in the Introduction, there has been some work on interconnecting the knowledge discovery process with deductive databases. In our view, there are several ways in which this interconnection can be achieved.

Using deductive databases to support the knowledge discovery process. Background knowledge as well as the data to be mined can both be represented in a deductive database. The deductive engine of the database can be used to aid the discovery process. Aronis, Ruiz, and Buchanan [2] deploy deductive methods to address the data and knowledge collection and integration stage of the discovery process. Mitchell [30] characterizes the inductive bias of a concept learning system as a minimal set of assertions needed so that the system's induced classification of an unseen instance follows *deductively* from the training data and the attributes of the instance. In [48], Simoudis, Livezey, and Kerber present their *Recon* system that integrates rule induction, deductive databases, and data visualization into the discovery process. Other KDD systems that use deductive database technology for hypothesis testing are IMACS [4] and KNOWLEDGE MINER [46]. Kero et. al. [19] present an overview of data mining techniques in the context of deductive–inductive inference and describe an architecture of their KNOWLEDGE MINER system.

Using KDD approaches to build deductive databases. In particular, (inductively) learning deductive rules from relational–like data. The Inductive Logic Programming (ILP) community has been actively pursuing this front. They focus on the combination of inductive learning and computational logic methods (see e.g. [39, 40]). Quinlan's FOIL system [38] is among the most successful ILP systems. For general overviews of ILP achievements see [5, 32]. Also, Flener and Yilmaz [9] provide an overview of inductive synthesis of recursive logic programs. Džeroski [7] surveys ILP techniques in the context of KDD.

Using KDD techniques to discover knowledge from deductive databases. At least three approaches for this purpose come to mind. In the first one, the deductive engine is used to generate all facts that can be inferred from the rules together with the data. Then a data mining method is employed to find patterns in those deduced facts. Goh, Tsukamoto, and Nishio [14] follow this approach. Their work is particularly useful when the set of facts deduced by the database system is large and so difficult to store and handle.

A second way of performing knowledge discovery in DDBs is to learn directly from the deductive rules together with the data. Lavrač, Gamberger and Jovanoski [20] provide a study of relevance for learning in deductive databases. Han, Cai, and Cercone [17] describe an attribute–oriented induction technique for discovering several types of knowledge rules from relational databases and discuss extensions of this techniques to deductive databases. Džeroski and Lavrač

[6] present an application of discovery to deductive databases.

A third, intrinsically more difficult approach is that of integrating separate deductive databases. We have developed techniques that address the data and knowledge integration stage of the KDD process in the context of deductive databases, and as a happy byproduct, these techniques implement also the data mining part of the process. This is because a successful integration of separate DDBs yields a unified collection of rules that can be regarded as patterns across all the original DDBs. We expand on our work on this approach in the following section.

3 Integration of Deductive Databases for Discovery

In this section, we describe our approach to the merging of deductive databases. Our work addresses a more general problem: integrating several logic programs (LPs) that may use different forms of negation. To the best of our knowledge, our work provides the first general framework to solve the aforementioned integration problem.

3.1 Negation in Logic Programming

The simplest logic programs contain only statements about what is *true* in the world (see e.g. [49, 13, 45]). Hence, a strategy or rule for *default negation* must be adopted, if one desires to infer from these systems what is *false* in the world. A rule for default negation establishes when it is sound to conclude that a statement is *false* in the world in the absence of information forcing the statement to be *true*. It is a heuristic way to model how humans use commonsense reasoning when they jump to conclusions based on incomplete information. Several default rules have been proposed. (See e.g. [24, 25, 31, 42].) Some concentrate only on the epistemological soundness of the conclusions they derive while others also address the issue of the computational complexity of the inference mechanism. Among the most widely used default rules in knowledge representation and reasoning are the Stable semantics [10] and the Well–Founded semantics [50].

An alternative recent approach to dealing with negated information is to allow logic programs to also store statements asserting explicitly that a given piece of information is *false* in the world. (See e.g. [34, 11, 37, 51].) In general, it would be too expensive, if not impossible, to store explicitly in a knowledge base all information that is *false* in the world. Hence, from a pragmatic point of view, explicit negation should be used for selected *false* statements only, and a default rule should be used to infer the remaining *false* statements.

It is widely accepted that most likely no single default rule will prove adequate for all problems. Despite this, the various notions of default negation have been used in the past as separate ways to interpret and to deduce information that is *false*. That is, each application has chosen only one of these rules and has applied it to every piece of data in the domain of the application. Expressive power is undoubtedly gained by allowing the interaction of different theories for negation in the same LP. In this way different pieces of information in the domain could be treated appropriately and separate LPs can be merged into a unified one.

3.2 Integrating Deductive Rules that Contain Different Forms of Negation

Our approach to deductive database integration builds upon a general formalism that we have introduced that allows for the interaction of several forms of negation in the same LP [27, 28, 43, 29, 44]. Some theoretical aspects of this research include the investigation of the expressive power of the new formalisms and the characterization of their semantics.

3.3 Problem Definition

The semantics of LPs with rules of the form

$$\bigvee_{i=0}^k a_i \leftarrow \bigwedge_{i=1}^l b_i, \bigwedge_{i=1}^m \text{not } c_i \quad (1)$$

where the a 's, b 's and c 's are atomic formulas has been extensively studied (see e.g. [22, 23]). These systems are usually called *normal disjunctive logic programs* (\mathcal{LP}_1^V programs for short). Many alternative meanings have been proposed for the default operator *not*. The purpose of our approach is to introduce and investigate the semantics of LPs in which rules are of the form

$$\bigvee_{i=0}^k \text{\textasciitilde} a_i \leftarrow \bigwedge_{i=1}^l \text{\textasciitilde} b_i, \bigwedge_{i=1}^{n_1} \text{not}_1 \text{\textasciitilde} c_i^1, \dots, \bigwedge_{i=1}^{n_m} \text{not}_m \text{\textasciitilde} c_i^m \quad (2)$$

where the a 's, b 's and c 's are atomic formulas that may be positive (+) or explicitly negated (\sim); $m \geq 0$; and the operators $\text{not}_1, \dots, \text{not}_m$ are interpreted using m alternative semantics for default negation. We denote the class of these LPs by $\mathcal{LP}_m^{V, \sim}$.

This investigation was undertaken in incremental steps as described below.

3.3.1 LPs Combining Default and Explicit Negations

As a first step, we investigated the semantics of *normal extended disjunctive logic programs* ($\mathcal{LP}_1^{V, \sim}$ programs for short) which are LPs with rules of the form (2) where $m \leq 1$. In this class of systems, a default negation *not* as well as explicit negation \sim appear in the

rules. The results of this investigation are summarized in [27, 28].

The main contribution of this research is the description of general techniques for extending model, fixpoint, and proof theoretic characterizations of an arbitrary \mathcal{LP}_1^V semantics to cover the class $\mathcal{LP}_1^{V, \sim}$.

We provide an improved way to transform $\mathcal{LP}_1^{V, \sim}$ programs into (explicit negation-free) \mathcal{LP}_1^V programs which capture naturally the intended meaning of the original programs. This approach builds on the ones presented in [34] and [12]. The transformation, called the *prime- \perp -transformation*, makes it possible to give simple and elegant model, fixpoint and proof theoretical characterizations of a semantics of $\mathcal{LP}_1^{V, \sim}$ and makes it easy to prove that these three characterizations are equivalent. Furthermore, inconsistencies in the semantics of an $\mathcal{LP}_1^{V, \sim}$ program are pinpointed by the transformation, making them easy to locate and to correct. Integrity constraints external to the program are not necessary to detect these inconsistencies. Illustrations of these techniques are given for stable [10, 36], disjunctive well-founded [3] and stationary semantics [37].

The complexity of computing a semantics of an $\mathcal{LP}_1^{V, \sim}$ program can be expressed in a straightforward manner in terms of the complexity of computing the meaning of the translation of the program. Similarly, complexity results known for the class \mathcal{LP}_1^V can be generalized easily to the extended case. In particular, we show that *credulous reasoning* and *determining the existence of models* for propositional $\mathcal{LP}_1^{V, \sim}$ programs under the stable semantics are NP-complete problems while *skeptical reasoning* is a co-NP-complete problem.

Since the techniques used are general enough to be applied to any semantics of \mathcal{LP}_1^V programs, they are described in terms of a generic such semantics which we call SEM. The extended version of SEM which is defined for the whole class $\mathcal{LP}_1^{V, \sim}$ is called XSEM. We prove that if the model, fixpoint, and proof theoretic characterizations of SEM are equivalent so are the extended model, fixpoint, and proof theoretic characterizations of XSEM.

We concentrate on the study of 3-valued $\mathcal{LP}_1^{V, \sim}$ semantics. In this way, the results in this study apply also to the case of reasoning with incomplete knowledge.

3.3.2 Integrating Definite LPs with Multiple Kinds of Negations

The next step in this research was the study of LPs consisting of rules of the form

$$\text{\textasciitilde} a \leftarrow \bigwedge_{i=1}^l \text{\textasciitilde} b_i, \bigwedge_{i=1}^{n_1} \text{not}_1 \text{\textasciitilde} c_i^1, \dots, \bigwedge_{i=1}^{n_m} \text{not}_m \text{\textasciitilde} c_i^m,$$

when $m = 2$. [29, 44] are devoted to this topic.

Since the (partial) stable semantics [10, 36] is commonly regarded as one of the most important semantics

for default negation, we study the meaning of LPs with two forms of default negation when not_1 is interpreted using the stable semantics (hence not_1 will be denoted by not_{STB}) and not_2 is interpreted using another arbitrary but fixed semantics for default negation SEM (not_2 will be denoted by not_{SEM}).

The model theoretic semantics of such LPs is defined as the collection of models M satisfying the fixpoint equation $M \in SEM(P^{M/not_{STB}})$, where $P^{M/not_{STB}}$ denotes the \mathcal{LP}_1^\sim program that results from evaluating the default negation not_{STB} in P with respect to M , using an extended version of the Gelfond–Lifschitz transformation. $P^{M/not_{STB}}$ contains only not_{SEM} (in other words, it is free of not_{STB}) and therefore the semantics SEM of P^M can be computed. If M is one of the canonical models of P^M under SEM, we say that M belongs to the semantics of P .

Whether or not there is a constructive procedure to generate this collection of models depends on the particular semantics SEM being used to interpret not_2 . A relevant such semantics is the Well-Founded semantics (WFS). The following subsection describes the results we obtained when the WFS is used to interpret not_2 .

Mixing Well-Founded and Stable negations

The well-founded semantics [50] is a widely used semantics for default negation, as is the stable semantics. We defined the semantics WF3STB for programs in which $not_1 = not_{STB}$ and $not_2 = not_{WFS}$ by using the fixpoint equation:

$$WF3STB(P) = \{M \mid M = WFS(P^{M/not_{STB}})\}$$

We prove, among others, the following properties of the WF3STB semantics.

- (i) The WF3STB interpolates between the WFS and the 3-STB semantics. That is,

$$WFS(P^{not}) \in WF3STB(P) \subseteq 3-STB(P^{not})$$

where P^{not} denotes the \mathcal{LP}_1^\sim program obtained from P by replacing the two default negation operators not_{STB} and not_{WFS} with a unique generic default negation operator not .

- (ii) The existence of WF3STB models for any program is guaranteed.
- (iii) The complexity of skeptical reasoning under the WF3STB semantics for the propositional case is quadratic in the length of the program.
- (iv) Credulous reasoning under the WF3STB semantics for the propositional case is an NP-complete problem.

The definition of the WF3STB semantics given above is not constructive. It provides a test to determine whether or not a given model is a WF3STB

model of a program. We describe an effective procedure to construct the collection of WF3STB models of a program. We have shown that the WF3STB semantics of P can be characterized as follows:

$$WF3STB(P) = \{M \in 3-STB(P^{not}) \mid M = WFS(P^{M/not_{STB}})\}.$$

Given that the WFS can be effectively computed, the existence of a procedure that computes the partial stable semantics implies that of a procedure to compute the WF3STB semantics.

3.3.3 Disjunctive LPs with Multiple Kinds of Negations

For the general framework of LPs with rules of the form (2), we concentrate on the interaction of four theories for negation: the *Generalized Closed World Assumption* (GCWA) [26], the *Weak Generalized Closed World Assumption* (WGCWA) [41], an arbitrary semantics SEM for default negation, and explicit negation. In order to do so, we introduce new default operators for negation in the language to explicitly represent the GCWA and the WGCWA. The first is called not_G and the second one is called not_W . So we consider LPs containing four different operators for negation: not_G, not_W, not_{SEM} and \sim .

A constructive definition of the semantics of these programs is provided in [43] for stratified and quasi-stratified LPs. The notion of stratification in LPs with several types of negation is defined similarly to the case of normal disjunctive logic programs by considering the different operators of default negation as if they were the same. Informally, a LP P is stratified if P^{not} , the LP obtained from P by replacing the three operators of default negation not_G, not_W and not_{SEM} with a unique generic default negation operator not is stratified.

There is a proper superclass of stratified LPs for which the semantics is also well-defined. We call the systems in this superclass quasi-stratified LPs. Intuitively, a LP is quasi-stratified if the default negations not_G and the not_W are stratified. (not_{SEM} may be non-stratified.)

4 Logical Rules and Association Rules

A separate useful exploration of the connections between KDD and DDBs is that of analyzing the relative merits of logical rules with respect to other types of rules, for instance probabilistic rules and association rules.

Association rule mining is a widely used technique for knowledge discovery. Association rules were independently introduced by P. Hajek, et al. [15, 16] and by Agrawal, et al. [1]. While [15] and [16] introduce association rule mining as a machine learning approach to the logic of discovery, [1] concentrates on the mining of associations over commercial transactional data.

We have successfully employed association rule mining in a variety of domains ranging from scientific discovery in biomedicine [47, 35] to recommender systems in electronic commerce [21].

5 Conclusions

We have described several different interactions between knowledge discovery and deductive databases. We elaborated on one of them that involves the discovery of patterns across different deductive databases by means of rule-bases integration.

Another useful direction for exploration is that of considering alternative rule formalisms and compare them with logical rules. Among those, association rules are of particular interest.

Further investigation of the connections between KDD and DDBs can yield beneficial approaches to handling background knowledge during the discovery process as well as exciting applications for DDBs to a variety of new domains.

References

- [1] R. Agrawal, T. Imielinski, and A. Swami. Mining association rules between sets of items in large databases. In *Proc. of the ACM SIGMOD Conference on Management of Data*, pages 207–216, Washington, D.C., May 1993. ACM.
- [2] J. Aronis, C. Ruiz, and B. Buchanan. Knowledge and data integration for discovery. In *Proceedings of the Workshop on Research and Development Opportunities in Federal Information Services*, May 1997. Arlington, VA (<http://www.isi.edu/nsf/>).
- [3] C. Baral, J. Lobo, and J. Minker. Generalized disjunctive well-founded semantics: Declarative semantics. In *Proceedings of the Fifth International Symposium on Methodologies for Intelligent Systems*, pages 465–473, Knoxville TN, USA, 1990.
- [4] R. Brachman, P. Selfridge, L. Terveen, B. Altman, F. Halper, T. Kirk, A. Lazar, D. McGuinness, L. Resnick, and A. Borgida. Integrated support for data archaeology. *International Journal of Intelligent and Cooperative Information Systems*, 2(2):159–185, 1993.
- [5] I. Bratko and S. Muggleton. Applications of inductive logic programming. *Communications of the ACM*, 38(11):65–70, 1995.
- [6] S. Džeroski and N. Lavrač. Inductive learning in deductive databases. *IEEE Transactions on Knowledge and Data Engineering. Special Issue on Learning & Discovery in Knowledge-based Databases*, 1993.
- [7] S. Džeroski. Inductive logic programming and knowledge discovery in databases. In U. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy, editors, *Advances in Knowledge Discovery and Data Mining*, pages 118–152. The MIT Press, 1996.
- [8] U.M. Fayyad, G. Piatetsky-Shapiro, and P. Smyth. From data mining to knowledge discovery: An overview. In U.M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy, editors, *Advances in Knowledge Discovery and Data Mining*, chapter 1, pages 1–34. AAAI Press/The MIT Press, 1996.
- [9] P. Flener and S. Yilmaz. Inductive synthesis of recursive logic programs: achievements and prospects. *Journal of Logic Programming*, 41(2–3):141–195, 1999.
- [10] M. Gelfond and V. Lifschitz. The stable model semantics for logic programming. In R. Kowalski and K. Bowen, editors, *Proceedings of the Fifth International Conference and Symposium on Logic Programming*, pages 1070–1080, Seattle, WA, USA, Aug. 1988. The MIT Press.
- [11] M. Gelfond and V. Lifschitz. Logic programs with classical negation. In D.H.D. Warren and P. Szeredi, editors, *Proceedings of the Seventh International Conference on Logic Programming*, pages 579–597, Jerusalem, Israel, June 1990. The MIT Press.
- [12] M. Gelfond and V. Lifschitz. Classical negation in logic programs and disjunctive databases. *New Generation Computing*, 9:365–385, 1991.
- [13] M.R. Genesereth and N.J. Nilsson. *Logical Foundations of Artificial Intelligence*. Morgan-Kaufmann, 1988.
- [14] C. L. Goh and S. Nishio M. Tsukamoto. Knowledge discovery in deductive databases with large deduction results: The first step. *IEEE Transactions on Knowledge and Data Engineering*, 8(6):952–956, Dec. 1996.
- [15] P. Hajek, I. Havel, and M. Chytil. The guha method of automatic hypotheses determination. *Computing*, 1:293–308, 1966.
- [16] P. Hajek and T. Havranek. On generation of inductive hypotheses. *Int. J. Man-Machine Studies*, 9:415–438, 1977.
- [17] J. Han, Y. Cai, and N. Cercone. Knowledge discovery in databases: An attribute-oriented approach, 1992.

- [18] J. Han, Y. Cai, and N. Cercone. Knowledge discovery in object-oriented and active databases. In K. Fuchi and T. Yokoi, editors, *Knowledge Building and Knowledge Sharing*, pages 221–230. Ohmsha, Ltd./IOS Press, July 1994.
- [19] B. Kero, L. Russell, S. Tsur, and W. Shen. An overview of database mining techniques, 1995.
- [20] N. Lavrac, D. Gamberger, and V. Jovanoski. A study of relevance for learning in deductive databases. *Journal of Logic Programming*, 40(2–3):215–249, Aug. 1999.
- [21] W. Lin, S.A. Alvarez, and C. Ruiz. Collaborative recommendation via adaptive association rule mining. In R. Kohavi, M. Spiliopoulou, and J. Srivastava, editors, *Proc. Intl. Workshop on Web Mining for E-Commerce (WebKDD2000)*. In conjunction with the Sixth Conference on Knowledge Discovery in Databases (*KDD'2000*), pages 35–41, Aug. 2000.
- [22] J.W. Lloyd. *Foundations of Logic Programming*. Springer-Verlag, second extended edition, 1987.
- [23] J. Lobo, J. Minker, and A. Rajasekar. *Foundations of Disjunctive Logic Programming*. The MIT Press, 1992.
- [24] J. McCarthy. Circumscription - a form of non-monotonic reasoning. *Artificial Intelligence Journal*, 13:27–39, 1980.
- [25] McDermott and J. Doyle. Non-monotonic logic I. *Artificial Intelligence Journal*, 13:41–72, 1980.
- [26] J. Minker. On indefinite databases and the closed world assumption. In *Proceedings of the Sixth Conference on Automated Deduction*, pages 292–308, 1982. Also in: *Lecture Notes in Computer Science* 138, pages 292–308. Springer Verlag, 1982.
- [27] J. Minker and C. Ruiz. On extended disjunctive logic programs. In J. Komorowski and Z.W. Raś, editors, *Proceedings of the Seventh International Symposium on Methodologies for Intelligent Systems, ISMIS'93*, pages 1–18. Lecture Notes in AI. Springer-Verlag, June 1993.
- [28] J. Minker and C. Ruiz. Semantics for disjunctive logic programs with explicit and default negation. *Fundamenta Informaticae*, 20(3/4):145–192, 1994. Anniversary Issue edited by H. Rasiowa.
- [29] J. Minker and C. Ruiz. Mixing a default rule with stable negation. In *Proceedings of the Fourth International Symposium on Artificial Intelligence and Mathematics*, pages 122–125, Jan. 1996.
- [30] T. M. Mitchell. *Machine Learning*. McGraw-Hill, 1997.
- [31] R. C. Moore. Semantical considerations on non-monotonic logic. *Artificial Intelligence Journal*, 25:75–94, 1985.
- [32] S. Muggleton. Scientific knowledge discovery using inductive logic programming. *Communications of the ACM*, 42(11):42–46, Nov. 1999.
- [33] S. Nishio, H. Kawano, and J. Han. Knowledge discovery in object-oriented databases: The first step. In *Proc. AAAI'93 Workshop on Knowledge Discovery in Databases*, pages 299–313, July 1993.
- [34] P. Pearce and G. Wagner. Logic programming with strong negation. In P. Schroeder-Heister, editor, *Proceedings of the International Workshop on Extensions of Logic Programming*, pages 311–326, Tübingen, FRG, Dec. 1989. Lecture Notes in Artificial Intelligence, Springer -Verlag.
- [35] M.A. Sao Pedro, C.A. Shoemaker, M. Pungliya, C. Ruiz, S.A. Alvarez, M.O. Ward, M. Stevens, E.F. Ryder, and J. Krushkal. Computational population-based techniques in identifying genetic variants associated with simulated complex disorder in a general population. In *Proc. of the Twelfth Genetic Analysis Workshop (GAW12)*. Southwest Foundation for Biomedical Research, Oct. 2000.
- [36] T. C. Przymusinski. Stable semantics for disjunctive programs. *New Generation Computing*, 9:401–424, 1991.
- [37] T.C. Przymusinski. Stationary semantics for disjunctive logic programs and deductive databases. In S. Debray and M. Hermenegildo, editors, *Proceedings of the North American Conference on Logic Programming*, pages 42–59, Austin, TX, USA, Oct. 1990. The MIT Press.
- [38] J.R. Quinlan. Learning logical definitions from relations. *Machine Learning*, 5:239–266, 1990.
- [39] L. De Raedt and M. Bruynooghe. A theory of clausal discovery. In *Proceedings of the 13th International Joint Conference on Artificial Intelligence*. Morgan Kaufmann, 1993.
- [40] L. De Raedt and N. Lavrač. The many faces of inductive logic programming. In J. Komorowski, editor, *Proceedings of the 7th International Symposium on Methodologies for Intelligent Systems: Lecture Notes in Artificial Intelligence*. Springer-Verlag, 1993.
- [41] A. Rajasekar, J. Lobo, and J. Minker. Weak generalized closed world assumption. *Journal of Automated Reasoning*, 5:293–307, 1989.

- [42] R. Reiter. A logic for default reasoning. *Artificial Intelligence Journal*, 13:81–132, 1980.
- [43] C. Ruiz and J. Minker. Combining closed world assumptions with stable negation. *Fundamenta Informaticae*, 32(2):163–181, Nov. 1997.
- [44] C. Ruiz and J. Minker. Logic knowledge bases with two default rules. *Annals of Mathematics and Artificial Intelligence*, 22(3,4):333–361, 1998.
- [45] S. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall, 1995.
- [46] W. Shen, B. Mitbender, K. Ong, and C. Zaniolo. Using metaqueries to integrate inductive learning and deductive database technology. In *Knowledge Discovery in Databases: Papers from the 1994 AAAI Workshop*, 1994. Available as AAAI Tech. Rep. WS-94-03, Menlo Park, CA.
- [47] C.A. Shoemaker, M.A. Sao Pedro, S.A. Alvarez, and C. Ruiz. Prediction vs. description: Two data mining approaches to the analysis of genetic data. In *Proc. of the Twelfth Genetic Analysis Workshop (GAW12)*. Southwest Foundation for Biomedical Research, Oct. 2000.
- [48] E. Simoudis and R. Kerber B. Livezey. Integrating inductive and deductive reasoning for data mining. In U. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy, editors, *Advances in Knowledge Discovery and Data Mining*, pages 353–373. The MIT Press, 1996.
- [49] S.L. Tanimoto. *The Elements of Artificial Intelligence*. Computer Science Press, 1987.
- [50] A. Van Gelder, K.A. Ross, and J.S. Schlipf. Unfounded sets and well-founded semantics for general logic programs. In *Proceedings of the Seventh ACM Symposium on Principles of Database Systems.*, pages 221–230, 1988.
- [51] G. Wagner. A database needs two kinds of negation. In B. Thalheim, J. Demetrovics, and H.D. Gerhardt, editors, *Proceedings of the Third Symposium on Mathematical Fundamentals of Database and Knowledge Base Systems*, pages 1–15, Rostock, Germany, May 1991. Lecture Notes in Computer Science, Springer -Verlag.