Computación y Sistemas
ISSN: 1405-5546
computacion-y-sistemas@cic.ipn.mx
Instituto Politécnico Nacional
México

Morales Luna, Guillermo
Reseña de "Foundations of Inductive Logic Programming" de S.H.N Cheng, R. de Wolf
Computación y Sistemas, vol. 2, núm. 1, julio-septiembre, 1998, pp. 52-55
Instituto Politécnico Nacional
Distrito Federal, México

Computación y Sistemas Vol. 2 No. 1 pp. 52-55
© 1998, CIC - IPN. *ISSN 1405-5546* Impreso en México

REVISIÓN DE LIBROS

Foundations of Inductive Logic Programming

S. H. N. Cheng, R. de Wolf
(Erasmus University of Rotterdam)
Lecture Notes in Artificial Intelligence 1228 (Subseries of LNCS)
Springer-Verlag, 1997, ISBN 3-540-62927-0

Reviewed by:
Guillermo Morales-Luna
CINVESTAV-IPN

This is a book with two main items: One is *logic*, i.e. deduction, and the other is *learning*, i.e. induction. Hence, the text is divided in two parts. The first part, consisting of eight chapters, can be considered as a complete textbook in a course of computational logic. The second part consists of eleven chapters and introduces the fundamentals of learning from a logical point of view.

There are classical textbooks ([2],[3],[4],[6]) on these topics. However a modern presentation was lacking. For sure the current textbook fulfills this lack.

The first three chapters in the book introduce the basic notions of propositional and predicate calculi. Both presentations are based on semantics. The syntactical aspects are restricted to the rules of formulae well formation. In spite that formal deduction is not treated, the authors present fundamental results as the Deduction Theorem (for any set of formulae Σ and any two formulae ϕ, ψ: $\Sigma \models (\phi \to \psi)$ iff $\Sigma \cup \{\phi\} \models \psi$) and the Compacteness Theorem (any infinite unsatisfiable set of formulae contains an unsatisfiable finite subset). The algorithmic transformation of formulae into their respective Skolem normal forms is presented in detail. Then Herbrand models are constructed in order to prove the fundamental result: Any theory is consistent iff it possesses a Herbrand model.

Resolution and unification are treated in the fourth chapter. For any set of clauses Σ and any clause C, it is written $\Sigma \vdash_r C$ if C can be (formally) derived from Σ by resolution. Obviously, if Σ is unsatisfiable then $\Sigma \vdash_r \square$, where $\square$ is the empty clause, which gives rise to proofs by refutation. Indeed, in a more general way, a soundness theorem for resolution is proved: $(\Sigma \vdash_r C) \Rightarrow (\Sigma \models C)$.

The fifth chapter deals with subsumption and refutation completeness. A clause D subsumes another clause C if for some substitution θ one has $D\theta \subset C$. The authors write $\Sigma \vdash_d C$ to denote the fact that for some D, $\Sigma \vdash_r D$ and D subsumes C. The subsumption theorem entails soundness and completeness: $(\Sigma \models C) \Leftrightarrow (\Sigma \vdash_d C)$. In the text, two interesting proofs of the "completeness" implication are given: The first is done by cases and the second by refutation. The classical use of refutation is based on the logical equivalence: $(\Sigma \models \phi) \Leftrightarrow (\Sigma \cup \{\neg\phi\} \vdash_r \square)$. Nevertheless any of the components of this equivalence is an undecidable predicate, as is stated by the Schmidt-Schauss theorem.

In my opinion, the first five chapters cover the formal basics of a course on mathematical logic addressed to an audience in an undergraduate programme in computing.

The remaining chapters in Part I are devoted to specific strategies for resolution. It is written $\Sigma \vdash_{lr} C$ if C is derived from Σ by linear resolution and $\Sigma \vdash_{ld} D$ if, for some C which subsummes D, $\Sigma \vdash_{lr} C$. The corresponding completeness theorems are proved: Σ is unsatisfiable iff $\Sigma \vdash_{lr} \square$, and $(\Sigma \models D) \Leftrightarrow \Sigma \vdash_{ld} D$. The input resolution coincides with linear resolution except that, at each step, each resolvent is restricted to be selected in the input set of clauses. This procedure, whose search is quicker than the linear one, is not complete, i.e. it maight fail to find a derivation in spite that there exists one. The linear resolution with a selection function, SL-resolution for short, gives rise to SL-resolution for definite clauses, SLD-resolution for short. This is the basic principle of PROLOG programming: restricted to Horn clauses, the current clause is obtained by resolving a selected atom of the former clause and the head atom of a clause in the input set of clauses (indeed the logic program). Corresponding notations $\Sigma \vdash_{sr} C$ and $\Sigma \vdash_{sd} D$ are introduced. For this type of derivation, restricted to Horn clauses, the corresponding completeness theorems also hold. Then the authors present the formal founda-

tions of PROLOG programming. Let us say that a "definite program" is a set of Horn clauses and that a "definite goal" is a Horn clause with no positive atom. Then it is shown that any definite program has a minimal Herbrand model, indeed, any ground atom A is in the minimal model of the definite program Π iff $\Pi \cup \{\neg A\} \vdash_{sr} \square$. Concerning the query satisfaction made by PROLOG, for a given goal $G = \bigwedge_{i=1}^{k} \neg A_i = (\leftarrow A_1, \ldots, A_k)$ and a program Π, let us say that a substitution θ is a "correct answer" if $\Pi \vdash_{sr} \left[\bigwedge_{i=1}^{k} \neg A_i\right] \theta$, and let us say that a "computed answer" is the substitution obtained as composition of the substitutions performed by the resolution process. A soundness result asserts that any computed answer is correct indeed. A completeness result asserts that any correct answer can be computed, up to a substitution consisting basically of a renaming of the involved variables. On this line of development, roughly speaking, a "computation rule" is a criterion to select atoms in the current clause considering the current state of derivation. The strong version of completeness for SLD-resolution states that any correct answer may be computed, up to a substitution, regardless of the stablished composition rule. The derivation procedure may be formalized as a search process in a tree. A derivation is a path leading to a succesful leaf in the tree. In an exhaustive way, it is possible to recognize whether $\Pi \nvdash_{sd} D$. Bad news, with respect to resolution, is that logical derivability between Horn clauses is undecidable. Thus, in spite that there were derivations, there is no a general procedure to find them. Finally, SLDNF-resolution (SLD-resolution with Negation as Failure) is introduced. Two points of great interest in the text are the discussion on the non-monotonic aspects of the Closed World Assumption (CSW) and the formalization of the CSW. Namely, given a definite program Π, it is firstly extended to a logic with equality and then, for each predicate P, each clause of the form $C = [P(\mathbf{t}) \leftarrow L_1(\mathbf{t},\mathbf{y}), \ldots, L_k(\mathbf{t},\mathbf{y})]$ is transformed into the formula $C' = [P(\mathbf{x}) \leftarrow \exists \mathbf{y} : (\mathbf{x} = \mathbf{t}), L_1(\mathbf{t},\mathbf{y}), \ldots, L_k(\mathbf{t},\mathbf{y})] = [P(\mathbf{x}) \leftarrow E]$. The "completion" $comp(\Pi)$ is the program with equality consisting of the clauses $[P(\mathbf{x}) \leftrightarrow \bigvee_E E]$. Then one has that G can be derived from Π using SLDNF-resolution if and only if $comp(\Pi) \vdash G$.

The contents of the first part is a complete course in computational logic with a modern approach. Thus I strongly recommend the first part of this book as a textbook in introductory courses to Computational Logic, in general, and Logic Programming, in particular.

Second part of the book deals with Inductive Logic Programming (ILP). First of all, the problem is stated. Given two sets E^+ and E^- of clauses, indeed they may consist of just ground atoms, called respectively *positive* and *negative examples*, a theory Σ is said to be *complete* with respect E^+ if $\Sigma \models E^+$. Σ is said to be *consistent* with respect E^- if $\Sigma \cup \{\neg e | e \in E^-\}$ is satisfiable. Σ is said to be *correct* with respect to E^+ and E^- if Σ is complete with respect E^+ and consistent with respect E^-. The main problem of ILP takes as instance a triplet $(\mathcal{B}, E^+, E^-)$ of *background knowledge* and positive and negative examples and looks to produce a theory Σ such that $\Sigma \cup \mathcal{B}$ is correct with respect to E^+ and E^-. Alternatively, the *nonmonotonic setting* of this problem takes as instance a pair $(\mathcal{I}^+, \mathcal{I}^-)$ of sets of Herbrand interpretations (the positive and the negative examples) and looks to produce a theory Σ which is true under each interpretation in $\mathcal{I}^+$ but false under each interpretation in $\mathcal{I}^-$.

A theory Σ is *too strong* with respect to a set E^- if Σ is not consistent with respect E^-. Σ is *too weak* with respect to a set E^+ if Σ is not complete with respect E^+. Σ is *overly general* with respect to E^+ and E^- if it is complete with respect to E^+ but not consistent with respect to E^-. Σ is *overly specific* with respect to E^+ and E^- if it is consistent with respect to E^- but not complete with respect to E^+. The inmediate approach to solve the ILP problem is to proceed in an iterative way: Given that Σ is the current candidate solution and it is not still correct, if $\Sigma \cup \mathcal{B}$ is too strong to E^- then specialize Σ, and if $\Sigma \cup \mathcal{B}$ is too weak to E^+ then generalize Σ. In some cases this search may succed to find a correct theory. However the problem in general may have no solution. The authors show, using a cardinality argument, that there are sets E^+ and E^- consisting of ground terms such that there is no theory correct with respect to E^+ and E^-.

The ILP problem can also be stated as the problem to find an axiomatization of a given theory: Let $\mathcal{C}$ be a clausal logic and let $\mathcal{C}_h \subset \mathcal{C}$ be a language consisting of formulae of a certain type. Let $\mathcal{C}_0 \subset \mathcal{C}_h$ be a language consisting of positive and negative examples, usually, ground terms, and let I be a Herbrand interpretation. An $\alpha \in \mathcal{C}_0$ is a "good" example if α is true under I and is "bad" otherwise. Let $\mathcal{C}_0^I$ be the set of good examples for I. A theory $\Sigma \subset \mathcal{C}_h$ is a *$\mathcal{C}_0$-complete axiomatization of I* if Σ is true under I and $\Sigma \models \mathcal{C}_0^I$. The pair $(\mathcal{C}_0, \mathcal{C}_h)$ is *admissible* if for any I and for any satisfiable $\Sigma \subset \mathcal{C}_h$ one has that for each $\alpha \in \mathcal{C}_0$: $\Sigma \models \alpha \Leftrightarrow I(\alpha) = \text{True}$. The axiomatization problem receives as input an admissible pair $(\mathcal{C}_0, \mathcal{C}_h)$ and a Herbrand interpretation I, and asks to build a $\mathcal{C}_0$-complete axiomatization of I. Again, a generalization-specialization strategy to solve this problem is presented. Of particular interest is the backtracking algorithm to find false clauses, when specialization is attempted, and the refinement operators, when generalization is attempted. Indeed, Shapiro's model inference algorithm takes a top-down approach to solve the axiomatization problem.

Induction can be seen as the reverse operation of deduction, hence *inverse resolution* is a form of induction. The V-operator generalizes a pair of clauses $\{C_1, R\}$ to a pair $\{C_1, C_2\}$, in such a way that R would be an instance of a resolvent of C_1, C_2. The W-operator combines two V-operators: Given a pair of clauses $\{R_1, R_2\}$ it produces a triplet $\{C_1, C_2, C_3\}$, in such a way that R_1 would be an instance of a resolvent of C_1, C_2 and R_2 an instance of a resolvent of C_2, C_3. The authors present the algorithms of Muggleton and Buntime ([5]) for inverse resolution, and emphasize its use as a type of generalization.

Unfolding, which is a kind of dual to inverse resolution, is also treated in the text. Given an overly general theory, then it is necessary to specialize it to a correct one. The specialization problem is stated as follows: Given a definite program Π and two disjoint sets of ground atoms E^+ and E^-, such that Π is overly general with respect to E^+ and E^- then the goal is to find a definite program Π' such that $\Pi \models \Pi'$ and Π' is correct with respect to E^+ and E^-.

The aim of unfolding is the construction of resolvents from parent clauses. If $C = (A \leftarrow B_1, \ldots, B_m)$ is a definite clause in a definite program Π and $\{C_1, \ldots, C_n\}$ is the set of clauses in Π whose head can be unified with some B_i then the unfolding of C upon B_i in Π is $U_{C,i} = \{D_1, \ldots, D_n\}$, where each D_j is the resolvent of C_j and C using B_i and the head of C_j as the literals resolved upon. The *type 1* program obtained by unfolding is $\Pi_{1,C,i} = (\Pi - \{C\}) \cup U_{C,i}$ and the *type 2* program obtained by unfolding is $\Pi_{2,C,i} = \Pi \cup U_{C,i}$. An UDS specialization of Π into Π' is a sequence $\Pi = \Pi_1, \ldots, \Pi_j, \ldots, \Pi_n = \Pi'$, $n \geq 1$, such that for each j, Π_{j+1} is either of the form $(\Pi_j)_{2,C,i}$ (Unfolding), or of the form $\Pi_j - \{C\}$ (clause Deletion), or of the form $\Pi_j \cup \{C\}$, where C is subsumed by a clause in Π_j (Subsumption). Then it can be seen that for any Π and Π', where Π' has no tautologies: $\Pi \models \Pi' \Leftrightarrow \Pi'$ is an UDS specialization of Π. This kind of program transformation is used to solve the specialization problem stated above.

Inmediately thereafter the authors present the quasi-ordered structures of atoms and clauses. A quasi-ordered set is a set provided with a reflexive and transitive relation. A lattice is a quasi-ordered set such that any couple of elements $\{x, y\}$ has a greatest lower bound (glb), $x \sqcap y$, and a lowest upper bound (lub), $x \sqcup y$. In a non-linear order one may define also the notions of maximal lower bound (mlb) and minimal upper bound (mub). If x, y are two elements in a quasi-ordered set, $x > y$, and there is no z with $x > z > y$, then x is an upward cover of y and y is a downward cover of x. Let $\mathcal{A}$ be the set of atoms in a language plus other two elements, $\top$ and $\bot$ called the top and the bottom elements respectively. Let $\succeq$ be the relation such that the top element is an upper bound, the bottom element is a lower bound, and for any two atoms A, B, $A \succeq B$ iff there is a substitution θ such that $A\theta = B$, i.e. A subsumes B. Then $(\mathcal{A}, \succeq)$ is a lattice, in fact the notion of upper bound corresponds to the notion of generalization and the notion of lower bound corresponds to the notion of specialization. The greatest specialization of a finite set of atoms can be obtained by the Unification Algorithm, while its least generalization by the so-called Anti-Unification Algorithm. Also, in this lattice each conventional atom can be connected to $\top$ by a chain of consecutive upward covers and to $\bot$ by a chain of consecutive downward covers. The downward covers of $\top$ are the most general atoms, while the upward covers of $\bot$ are the ground atoms. For any two clauses C, D, let $C \succeq D$ if there is a substitution θ such that $C\theta = D$, and let $C \sim D$ if $C \succeq D$ and $D \succeq C$. A clause C is reduced if for no proper subclause $D \subset C$ we have $C \sim D$. The Plotkin's algorithm serves to reduce a given set of clauses. Inverse reduction takes as input a reduced set of clauses and an integer m and produces an equivalent set of clauses with at most m clauses. For purposes of refinements, inverse reduction is an useful tool. Let $\mathcal{C}$ be the set of clauses. Then $(\mathcal{C}, \succeq)$ is a lattice, and the set of Horn clauses $\mathcal{H}$ with the inherited structure is a lattice as well. It may happen, nevertheless that some clauses do not have covers. An important property is that the subsumption relation $\succeq$ is decidable over $\mathcal{C}$.

As an alternative to the subsumption relation, the logical implication relation is treated. Implication is stronger than subsumption and it is undecidable. A sufficient condition for a finite set of clauses to have a least generalization, with respect to implication (LGI), is that it has at least one function-free non-tautological clause. Greatest specializations for finite clause sets always exist. Thus if $\mathcal{C}$ is function-free, $(\mathcal{C}, \models)$ is a lattice. However, the Horn clauses do not form a lattice, even if there is no function symbols. The decision for the existence of LGI under the existence of function symbols is an open question till now.

In the textbook there is also a chapter devoted to the relativization of both subsumption and implication with respect to background knowledge, which is a must on systems based on knowledge. The approaches of Plotkin and Buntine are exposed.

Finally, the text deals with the treatment of the non-monotonic setting of ILP. The PAC algorithm is presented. This is a probabilistic algorithm that given a concept in an explicit way it produces an implicit concept such that almost surely the probabilistic measure of the symmetric difference between those concepts is small.

I have found the book very illustrative of the notions of Inductive Logic Programming. All methods are explained in detail, hence they provide the reader with a lot of possibilities of experimentation. The second part is at present very specialized, thus I recommend it as a text at the graduate level. It is worth to mention that in spite of its high level of specialization, this is a self contained text quite readable.

References

[1] A. Bundy. *The Computer Modelling of Mathematical Reasoning.* Academic Press, 1983.

[2] C. L. Chang, R. C. T. Lee. *Symbolic logic and mechanical theorem proving.* Academic Press. San Diego, CA, 1973.

[3] D. Duffy. *Principles of Automated Theorem Proving.* John Wiley and Sons, 1991.

[4] R. Kowalski. *Logic for problem solving.* North-Holland. New York, 1979.

[5] S. Muggleton, W. Buntime. Machine invention of first-order predicates by inverting resolution. In J. Laird, Editor, *Proceedings of the 5-th International Conference on Machine Learning (ICML-88)*, pp. 339-352. Morgan Kauffman, 1988.

[6] L. Wos, R. Overbeek, E. Lusk, J. Boyle. *Automated Reasoning. Introduction and Applications.* Second Edition, McGraw-Hill, 1992.

Guillermo Morales-Luna *is professor at the Computer Science Section of the Research and Advanced Studies Center (CINVESTAV-IPN) in Mexico City. In 1984 he got a Ph. D. at the Mathematics Institute of the Polish Academy of Sciences, in Warsaw. His main areas of interest are Mathematical Logic, Theoretical Computer Science, Theory of Complexity and Natural Language Processing.*

Morales-Luna's address is: Computer Science, CINVESTAV-IPN, Av. IPN 2508, 07000 México, D.F., e-mail:gmorales@cs.cinvestav.mx