PROGRAMMAZIONE LOGICA IN DATALOG: UN LUNGO PERCORSO DALLA TEORIA ALLA PRATICA
LOGIC PROGRAMMING IN DATALOG: A LONG TOUR FROM THEORY TO PRACTICE

Sergio Greco, Luigi Palopoli, Nicola Leone, Pasquale Rullo, Domenico Saccà

SOMMARIO/*ABSTRACT*

In questo articolo si descrivono le linee di ricerca sviluppate a Cosenza nell'ambito della programmazione logica in un arco temporale di oltre 20 anni e che hanno portato a recenti interessanti e promettenti sviluppi industriali. Tali linee di ricerca sono cambiate nel tempo ma hanno mantenuto l'interesse iniziale per accoppiare la programmazione logica con le tecnologie della basi di dati, interesse che si continuamente rinnovato per affrontare nuove sfide nell'uso della teoria per risolvere problemi pratici.

In this paper, we describe the research lines in logic programming, carried out in Cosenza over a period of more than 20 years, which have recently produced promising industrial exploitation follow-ups. The research lines have changed over the time but they have kept the initial interest on combining logic programming with databases techniques, that has been continuously renewed to cope with new challenges, in our attempt to use theory to solve practical problems.

Keywords: Logic programming, DATALOG, bottom-up execution, stable models, disjunctive logic programming, answer set programming, ontology

1 Introduction

The research on logic programming in Cosenza started in the middle eighties at CRAI, an industrial consortium for information technology research and applications at Rende, Italy. With the enthusiasm of young researchers, we decided to enlarge our original competencies in database technology to exploit the great, at that time promising potentialities of logic programming. We then set up a research group on DATALOG (a version of logic programming particularly suited for database applications) and we started to study the problem of the efficient computation of answers to logic queries over relational databases.

Later on, while moving from CRAI to University of Calabria, we also moved to more theoretical issues in DATALOG. In particular, we worked at endowing DATALOG with the capability of handling non monotonic reasoning and defeasible knowledge, and we concentrated on providing a non-classical interpretation for negation and disjunction (with the two perspectives not necessarily disjoint).

When the age of maturity came, our interest for theory was more and more urged to be combined with the necessity of providing evidence of its utility by means of "running" prototypes. So, our group promoted the construction of the DLV system, one of the most efficient implementation of logic programming available to date, which is being used in many applications.

Finally, getting old and desiring to leave a more tangible effect of our research to the economy of our Region, along with other researchers from University of Calabria, we founded a research spin-off, named Exeura, whose mission is to transform research results in the field of Knowledge Management into industrial products.

In this paper we make a quick tour of our research in DATALOG during the last 20 years, starting from the infancy of our work on efficient query compilation, to the youth of the contribution on non monotonic reasoning and the maturity of developing the DLV system, eventually arriving to the old age of exploiting results within an industrial framework. But our story does not end here: we are positive and ready to add another chapter!

2 INFANCY: efficient compilation of DATALOG

DATALOG is essentially logic programming without function symbols using tuples of a relational database as facts: a database $\mathcal{D}$ is seen as a set of facts , whose predicate symbols (*extensional predicates*) coincide with the relation names, and all other predicate symbols (*intensional predicates*), defined by rules, correspond to views of the

database.

We have investigated the efficient computation of answers to logic queries over relational databases since the the middle of eighties of the past century when we were all working for CRAI. The research has concentrated firstly on the definition of algorithms for the efficient computation of the semantics of programs and answers. The problem can be stated as follows: given a query $\mathcal{Q} = \langle q(X), \mathcal{P} \rangle$ and a database $\mathcal{D}$, seen as a set of facts, compute the atoms matching $q(X)$ which are logic consequence of $\mathcal{P} \cup \mathcal{D}$. Two main approaches have been proposed in the literature, known as *top-down* computation (used by Prolog-like system) and *bottom-up* computation (used by deductive database and answer-set systems).

The advantages of top-down systems is that only rules and atoms relevant to the query goal are considered, but there are several problems as termination and duplicated computation. On the other side, the bottom-up strategy always terminates, but it computes atoms which are not relevant for the query goal (i.e. first computes all atoms which are logic consequence of $\mathcal{P} \cup \mathcal{D}$ and, afterwards, selects the atoms matching the goal $q(X)$). Concerning the definition of optimization techniques avoiding duplicated computation, the most important contributions were top-down methods with memoing strategy and the semi-naive (bottom-up) algorithm.

Moreover, several optimization techniques combining top-down and bottom-up strategies were proposed as well. These techniques try to compute only atoms which may be "relevant" for the query goal in a bottom-up strategy. The key idea of all these techniques consists in the rewriting of deductive rules with respect to the query goal so that answering the query without actually computing irrelevant facts. General rewriting techniques (e.g. magic-set and supplementary magic set) can be applied to all queries, but their efficiency is limited. On the other side, there are specialized techniques which are very efficient, but they can be applied to limited classes of queries.

We investigated a particular interesting class of queries, known as *chain queries*, i.e., queries where bindings are propagated from arguments in the head to arguments in the tail of the rules, in a chain-like fashion. For these queries, which are rather frequent in practice (e.g., graph applications), insisting on general optimization methods (e.g., the *magic-set* method) does not allow to take advantage of the chain structure, thus resulting in rather inefficient query executions. Specialized methods for subclasses of chain queries have been proposed in the literature, but, unfortunately, these methods do not fully exploit bindings.

We proposed a *counting* method that is particularly specialized for bound chain queries; however this method, although proposed in the context of general queries [Saccà(13,17,20), Greco(8)], preserves the original simplicity and efficiency only for a subset of chain queries whose recursive rules are linear. We later proposed a new method exploiting the relationship between chain queries, context-free languages and pushdown automata, which permits to rewrite queries into a format that is more suitable for the bottom-up evaluation [Greco(18, 43)]. The so-called *pushdown* method translates a chain query into a factorized left-linear program implementing the pushdown automaton recognizing the language associated with the query. A nice property of this method is that it reduces to the counting method in all cases where the latter method behaves efficiently and introduces a unified framework for the treatment of special cases, such as the factorization of right-, left-, mixed-linear programs, as well as the linearization of non-linear programs.

These techniques defined for standard DATALOG queries can be also applied, or easily extended, to disjunctive logic queries (queries whose associated program is a disjunctive DATALOG program) [Greco(36, 87, 110), Leone(144)].

We also elaborated optimization techniques for queries with aggregates (expressing, for instance, optimization problems) [Greco(15,38,45,60,75)]. These techniques rewrite queries so that the simple modification of the semi-naive algorithm emulates classical optimization strategies such as greedy and dynamic programming.

Further on, we investigated optimizations techniques for queries with complex terms such as sets and, in particular, we analyzed the computation of optimal sets of matchers and unifiers for atoms with "bounded" set terms [Greco(21,24)].

3 YOUTH: non monotonic reasoning in DATALOG

The realization of common-sense reasoning systems has been, since the beginning, one of the natural application realm of logic programming. However, common-sense reasoning requires non-monotonicity, that is, the capability for the reasoning system to cancel or retract previously attained conclusions, in the light of new evidence the system becomes aware of (that is, generally, the knowledge must be defeasible). Unfortunately, plain DATALOG is indeed monotonic and, therefore, unapt to the purpose. Therefore, mechanisms had to be devised in order to endow DATALOG-based languages with the capability of representing and managing non-monotonicity.

Loosely speaking, in order to endow DATALOG with the capability of handling defeasible knowledge, one might resort to non standard semantics for the language as a whole or concentrate on providing a non-classical interpretation for negation or disjunction (with the two perspectives not necessarily disjoint). In the logic programming community the second line of research received a much larger deal of attention than the first one, and we followed this lines in the research developed when all of us moved from CRAI to University of Calabria.

Important results derived by the work developed in our group on ordered logic programs, that are, programs con-

sisting of a poset of modules, each of which is itself a logic program, under the assumption that programs lying higher in the hierarchy are semantically more trustable than lower ones [Leone(6), Rullo(29)] and the related issue of inheritance [Leone(91), Rullo(18)] and, moreover, on the circumscription-based interpretation of negation-free DATALOG [Palopoli(32)]. The semantics of (disjunctive) logic programs with preferences on atoms was later investigated in [Greco(136)].

But we also investigated the semantics of negation in DATALOG-like languages. The simpler form of non-classical interpreting negation in DATALOG is encoded in the notion of *stratified* (aka, *perfect*) models due, among others, to Przyimusinski. This semantics is defined when in a program there is no recursion through negation, that is, the program is *stratified*. In this case, one can divide the program in an ordered list of layers, such that each predicate occurring negated in the rule of any layer does not occur in the head of rules of that and higher layers. Then, the intended model is obtained by evaluating the program layer-wise, beginning with the lowest one. The drawback of stratified programs rests on their limited generality and expressive power. A more general notion is that of *locally stratified* programs, where recursion through negation is allowed as long as it gets resolved at the ground level. Contrary to stratification, however, local stratification is in general undecidable, even if sufficient conditions for it can be given [Palopoli(3)].

Van Gelder and others proposed a solution to the problem of providing a clean semantics to programs with recursive negation, by defining the concept of *well-founded model*. Loosely speaking, a model of a DATALOG program is well-founded is the model does not contain any subset of unjustifiable atoms. Well-founded semantics is in general polynomial-time computable, but its implementation is not at all trivial [Rullo(9)]. Also, the well-founded model of a DATALOG program is unique (but may not exist).

In order to attain a significant boost in expressivity, though, one has to consider a further, simple yet powerful, semantics for logic program with negation, namely, that of *stable models* of Gelfond and Lifschitz. To informally illustrate, a model of a program is stable if the program regenerates it when the knowledge encoded in the model is assumed from granted. In general, a DATALOG program may have none, one or multiple stable models. Therefore, differently from the aforementioned semantics, entailment under stable model semantics can be intuitively defined in two forms, that are, cautious reasoning, which tells an information to be implied by a program if it is indeed implied in all the stable models of the program, and brave reasoning, which makes the information implied by the program if implied by at least one of its stable models. These definitions endow DATALOG programs with a much larger expressiveness (allowing to capture classes like coNP) than that of well-founded models but, at the same time, renders the entailment and related problem intractable [Saccá(49)], thereof including that of computing one single stable model of a DATALOG program. Fortunately, there are indeed cases when the stable model semantics can be computed efficiently [Palopoli(22)].

But besides the cautious and the brave forms of reasoning, the possibility for a program to have multiple stable models can be interpreted in a different and rather appealing manner, that is, that each stable model of the program *non-deterministically* encode one possible status of the world. This view of stable model semantics prompted through years some of us to study the formal properties and the potential application of exploiting non-determinism as encoded in DATALOG programs under the stable model semantics, which resulted in several interesting research papers [Greco(28,29), Greco(44), Saccá(43,48)].

A second depart from more standard forms of semantics is determined by allowing more than two truth values for literals. To illustrate, in two-valued semantics, each literal of a program must be either declared true or false. There are cases and applications, though, where it appears sensible to introduce a third truth value, say "unknown", into play, to be assigned to a literal if neither itself nor its negation is entailed by a set of rules. All that is conducive to the notion of *partial* model, which is precisely one that tells some atoms true, some atoms false and some undefined in the status of the world it encodes for. Also in this new setting it is worth analyzing the formal properties of the resulting formalisms [Saccá(58)]. Moreover, in this setting, stable models semantics also allows to express search and optimization problems [Saccá(54,63,83)].

To suitably deal with negation is not enough for some application though. Theoretically speaking, this happens when one deals with problems which are complete for the second level of the polynomial hierarchy. From the more practical viewpoint, this more simply happens when the application context naturally calls for the exploitation of disjunctive statements, that are, statements which declare the (possibly conditional) truth of at least one of a group of atoms. The resulting language, usually referred to as *Disjunctive DATALOG* allows disjunct to occur in rule heads and (possibly) negation in rules bodies. Several of the issues discussed above for (disjuntion-free) DATALOG carry over to Disjunctive DATALOG, and the development of the associated research lines has witnessed a relevant contribution of our group. To illustrate, the papers [Rullo(22), Greco(98)] include fundamental results about the semantics, complexity and expressive power of Disjunctive DATALOG programs, [Palopoli(27,50)] discuss tractability issues about Disjunctive DATALOG, while [Leone(103,113)] tackles with the issue of selecting some of the models of a Disjunctive program as the preferred ones and, finally, [Rullo(33)] deals with the enhancement of the expressive and representational capabilities of Disjunctive DATALOG using constraints.

4 MATURITY: the DLV system

DLV [Leone(139)] is an advanced system for Knowledge Representation and Reasoning which is based on Disjunctive DATALOG under the stable model semantics (also called Answer Set Programming). Roughly, a Disjunctive DATALOG program is a set of disjunctive rules, i.e., clauses of the form

$$a_1 \mathtt{v} \cdots \mathtt{v} a_n \mathtt{:-} b_1, \cdots, b_k, \mathtt{not}\ b_{k+1}, \cdots, \mathtt{not}\ b_m$$

where atoms $a_1, \ldots, a_n, b_1, \ldots, b_m$ may contain variables. The intuitive reading of such a rule is "If all $b_1, \ldots, b_k$ are true and none of $b_{k+1}, \ldots, b_m$ is true, then at least one atom in $a_1, \ldots, a_n$ must be true." Disjunctive DATALOG has a very high expressive power – it allows to express all problems in the complexity class Σ_2^P (i.e., NP^{NP}). Thus, under usual complexity conjectures, Disjunctive DATALOG is strictly more expressive than both SAT and CSP, the power of which is "limited" to NP, and it can naturally represent a large class of relevant problems ranging from artificial intelligence to advanced database applications.

DLV is generally considered the state-of-the-art implementation of Disjunctive DATALOG. Its efficiency has been confirmed by the results of First Answer Set Programming System Competition (`http://asparagus.cs.uni-potsdam.de/contest/`), where DLV won the "disjunctive" category. Moreover, DLV turned out to be very efficient also on (non-disjunctive) DATALOG programs, as it finished first also in the general category MGS (Modeling, Grounding, Solving – also called *royal* competition, open to all ASP systems).

The implementation of the DLV system is based on very solid theoretical foundations, and exploits major results that have been achieved by the Deductive Databases group of University of Calabria in the last 20 years. The system has been recently engineered for industrial exploitation, and is successfully employed in many challenging real-world applications, for instance in the area of Knowledge Management [Leone(141)], and advanced Information Integration [Leone(127,144)] (see next section).

Among the many features of the system, it is worth remarking the following:

Advanced knowledge modeling capabilities. DLV provides support for declarative problem solving in several respects:

- High expressiveness in a formally precise sense (Σ_2^P), so any such problem can be uniformly solved by a fixed program over varying input.
- Rich language for knowledge modeling, extending Disjunctive DATALOG with weak constraints (for preferences handling) [Leone(68)], powerful aggregate functions [Leone(132,110,124,148)], and other useful KR constructs.
- Full declarativeness: ordering of rules and subgoal is immaterial, the computation is sound and complete, and its termination is always guaranteed.
- Declarative problem solving following a "Guess&Check" paradigm [Leone(139)] where a solution to a problem is guessed by one part of a program and then verified through another part of the program.
- A number of front-ends for dealing with specific AI applications [Leone(57,109,105,125)], information extraction [Leone(141)], Ontology Representation and Reasoning [Leone(146,130)].

Solid Implementation. Much effort has been spent on sophisticated algorithms and techniques for improving the performance, including

- Database optimization techniques: indexing, join ordering methods [Leone(85)], Magic Sets [Leone(144,124)].
- Artificial intelligence computation techniques: heuristics [Leone(87,149,133)], backjumping techniques [Leone(138,119)], pruning operators [Leone(137)].

DLV is able to solve complex problems and efficiently deal also with large input data [Leone(147)].

Database Interfaces. The DLV system provides a general ODBC interface to relational database management systems [Leone(120)].

For up-to-date information on the system and a full manual we refer to `http://www.dlvsystem.com`, where also download binaries of the current release and various examples are available.

5 OLD AGE: industrial applications of DLV

We are finally at the end of the story. In 2002, along with other researchers from University of Calabria, we founded a research spin-off, named Exeura. Since the beginning, the mission of Exeura was to transform into commercial products research results in the field of Knowledge Management (KM). Topics of interest include: (1) Knowledge Representation and Reasoning (e.g., ontologies, automatic reasoning, etc.); (2) Data, Text, and Process Mining (e.g., data discovery in databases, document classification, web mining, workflow mining, etc.); (3) Information Extraction and Wrapping; (4) Heterogeneous Information Sources Integration.

Thanks to a vast scientific and technological know how in KM and, in general, in advanced Information Systems, Exeura has implemented a number of industrial prototypes, currently under productization. Some of those exploit the DLV reasoning capabilities, notably, OntoDLV, Olex and Hylex.

OntoDLV is a system for ontology specification and reasoning [Leone(146)]. Ontologies are abstract models of complex domains that have been recognized to be a fundamental tool for conceptualizing business enterprise information. The World Wide Web Consortium (W3C) has already provided recommendations and standards related to ontologies, like RDF(S) and OWL. In particular, OWL has been conceived for the Semantic Web, with the goal to enrich Web pages with machine-understandable descriptions of the presented contents. OWL is based on expressive Description Logics (DL); distinguishing features of its semantics w.r.t. Logic Programming languages are the adoption of the Open World Assumption (OWA) and the non-uniqueness of names (different names can denote the same individual). However, while the semantic assumptions of OWL make sense for the Web, they are unsuited for enterprise ontologies. Since an enterprise ontology describes the knowledge of specific aspects of the closed world of the enterprise, it turns out that the Closed World Assumption (CWA) is more appropriate than the OWA (appropriate for the Web, which is an open domain). Moreover, the presence of naming conventions, often adopted in enterprises, can guarantee name uniqueness, making also the Unique Name Assumption (UNA) plausible. Importantly, enterprise ontologies often are the evolution of relational databases, where both CWA and UNA are mandatory.
OntoDLV supports a powerful ontology representation language, called OntoDLP, extending (disjunctive) Answer Set Programming (ASP) with all the main ontology features including classes, inheritance, relations and axioms. OntoDLP is strongly typed, and includes also complex type constructors, like lists and sets. The semantic peculiarities of ASP, like the Closed World Assumption (CWA) and the Unique Name Assumption (UNA), allow to overcome both the above mentioned limits of OWL, thus making OntoDLV suitable for enterprise ontology specification. It is worth noticing that OntoDLV supports a powerful interoperability mechanism with OWL, allowing the user to retrieve information from OWL ontologies, and build rule-based reasoning on top of OWL ontologies. The system is already used in a number of real-world applications including agent-based systems, information extraction, and text classification.
Olex is a rule-based text classification system [Rullo(43)]. It supports a hypothesis language of the form

$$c \leftarrow T_1 \in d \vee \cdots \vee T_n \in d \wedge \neg(T_{n+1} \in d \vee \cdots \vee T_{n+m} \in d)$$

where each T_i is a conjunction of terms (n-grams). The meaning of a classifier as above is "classify document d under category c if any of $T_1, \cdots, T_n$ occurs in d and none of $T_{n+1}, \cdots, T_{n+m}$ occurs in d". The execution of a classifier relies on the DLV system.
One important feature of the Olex system is the integration of the manual approach with the automatic rule induction. Thanks to the interpretability of the produced classifiers, the domain expert can participate in the refinement of a classifier, by manually specifying a set of rules to be used in conjunction with those automatically learned. This cooperation, in fact, may be very effective, since both approaches have some limits, that can be overcome if used in synergy. To this end, the expressive power of the DLV language turns out of great advantage, as it allows the (manual) specification of complex classification rules (not restricted to the hypothesis language), e.g., rules with aggregate functions (such as *count*, *sum*, etc.) that are very useful in text categorization.
Olex has been applied to a number of real world applications in various industries including: health-care, tourism, and insurance.
HiLeX supports a semantic-aware approach to information extraction from unstructured data (i.e., documents in several formats, e.g., html, txt, doc, pdf, etc), that is currently used in many applications of Text Analytics. The semantic approach of HiLeX basically relies on [Sacc(115)]:

- the ontology representation formalism OntoDLP used for describing the knowledge domain;
- a logic-based pattern matching language relying on a two-dimensional document representation; in fact, a document is viewed as a Cartesian plane composed by a set of nested rectangular regions called portions. The DLV system is used as the pattern recognition engine.

REFERENCES

[1] Greco: Papers in the DBLP entry of Sergio Greco: 8, 15, 18, 21, 24, 28, 29, 36, 38, 43, 44, 45, 60, 75, 87, 98, 110, 136.

[2] Leone: Papers in the DBLP entry of Nicola Leone: 6, 57, 68, 85, 87, 91, 103, 105, 109, 110, 113, 119, 120, 124, 125, 127, 130, 132, 133, 137, 138, 139, 141, 144, 146, 147, 148, 149.

[3] Palopoli: Papers in the DBLP entry of Luigi Palopoli: 3, 22, 27, 32, 50.

[4] Rullo: Papers in the DBLP entry of Pasquale Rullo: 9, 18, 22, 29, 33, 43.

[5] Sacca: Papers in the DBLP entry of Domenico Saccà: 13, 17, 20, 43, 48, 49, 58, 115.

NOTE: *for the sake of space, citations are given referring to the DBLP bibliography* (http://www.informatik.uni-trier.de/∼ley/db/index.html), *using the format* [author(n)] *to mean the reference n of the DBLP bibliography list of author.*

6 Contacts

Sergio Greco, Luigi Palopoli, Domenico Saccà
DEIS Department, University of Calabria
Via P. Bucci 41/C, 87036 Rende (CS), Italy
Exeura s.r.l., Rende - Italy
{greco,palopoli,sacca}@deis.unical.it

Nicola Leone, Pasquale Rullo
Department of Mathematics, University of Calabria
Via P. Bucci 30/B, 87036 Rende (CS), Italy
Exeura s.r.l., Rende - Italy
{leone,rullo}@mat.unical.it

7 Biography

- Sergio Greco is full professor of Computer Engineering at the DEIS (Electronics, Computer and Systems Sciences) Department of the University of Calabria. His present research interests are: logic programming for knowledge representation and reasoning, imprecise answering systems, data mining, XML.
- Nicola Leone is full professor of Computer Science at the Department of Mathematics of the University of Calabria. His present research interests are: Knowledge Representation and Reasoning, Logic Programming and NonMonotonic Reasoning, Logics in Databases, Computational Complexity in Artificial Intelligence and Databases
- Luigi Palopoli is full professor of Computer Engineering at the DEIS (Electronics, Computer and Systems Sciences) Department of the University of Calabria. His present research interests are: bioinformatics, computational game theory, knowledge representation and data mining.
- Pasquale Rullo is full professor of Computer Science at the Department of Mathematics of the University of Calabria. His present research interests are: Data and Text Mining, Knowledge Representation
- Domenico Saccà is full professor of Computer Engineering at the DEIS (Electronics, Computer and Systems Sciences) Department of the University of Calabria. His present research interests are: scheme integration in data warehousing, compressed representation of datacubes, workflow and process mining, logic-based query languages for data mining and information extraction.