

25 Years of Applications of Logic Programming

A. Dal Palù¹ and P. Torroni²

¹ Dip. di Matematica, Università di Parma, alessandro.dalpalu@unipr.it

² Dip. di Informatica, Elettronica e Sistemistica, Università di Bologna,
paolo.torroni@unibo.it

Abstract. We present a review of practical applications of Logic Programming appeared in Italy since 1985. We classify them according to their area of application and discuss some trends emerged in the latest developments. Notwithstanding this survey is far to be comprehensive, it shows that LP successfully evolved and quickly adapted to new challenges offered by the notable variety of application areas.

1 Introduction

The beginning of LP applications was driven by the investments and interest by private industries, attracted by the novelty and potentialities of this technology. The enthusiasm of those years was great. Around the time GULP was founded in 1987, the main Italian ICT and consumer electronics event, SMAU, was discovering Artificial Intelligence [14]. AI made its debut in the Italian market from a variety of stands. The heterogeneous mix of AI promoters included small enterprises of academic roots, such as Delphi, a University of Pisa's spin-off then based in Viareggio, and big actors such as IBM, and included many more in between. Back then AI mainly meant Expert Systems, and the use of Prolog inference engines and the adoption of declarative technologies in general was considered a very promising approach. Nixdorf Italia, involved in Esprit-2 research projects and in the development of air fleet optimization tools for Alitalia, was using a development environment written in Prolog, called Twice. IBM, Unisys, Pirelli Informatica and Datitalia Processing, among others, were all promoting expert systems for configuration and diagnosis which made use of knowledge bases and declarative rules. IBM was pushing expert systems technologies by announcing a series of AI courses.

The number of applications in the early years reflected the interest of industries. In Figure 2 it can be seen how the number of applications grew constantly till the middle of 90s.

1.1 Late 90s

In the 90s, the initial enthusiasm from the industry decreased. Already at that time (see [102]) there was a clear perception of the limitations of current applications and the potential to exploit. For example,

- the high level modeling and specification allow effective development of industrial and commercial applications;
- integration between top level language (Prolog) and other languages provide powerful tools for real applications (e.g, CLP).
- the applications are stand-alone projects, not integrated with other technologies. It is difficult to find applications which are in the mainstream of the production line or which have to interact with other crucial applications. We think that the latest Prolog releases, offering the interfaces with other languages could partially solve this situation.

It is interesting to note that many applications appeared afterwards were based on the smooth integration of a LP technology core into a larger project.

In this period (see Figure 2), the number of applications slightly decreased and definitely broke the initial trend, partly because of the lack of collaboration with the industrial world.

1.2 Today

In the latest 15 years, the scenario evolved and expanded widely. We are witnessing the birth and rise of new application domains, alongside the evolution of older application domains in which LP had been present since the 80s, addressed with renewed vigor motivated by more modern LP-based solutions. Probably, the technology is now ripe for adoption of LP technologies. We can rely on more efficient algorithms, on the achievements of CLP, on more powerful machines, and on a better understanding of the theory. Another reason why there are so many applications of LP in Italy nowadays, may be that recently funding agencies privilege applied research projects rather than basic research. This happens both at the National level, and especially at the European level, where most of Italian research is seeking funding.

On the downside, comparatively little Prolog is used for applications - despite the availability of many implementations of it. We still note, as 15 years ago, that only some companies use Prolog. Developers and software experts are still not familiar with it.

Logic programming has not yet broken through the wall that divides research from low level programmers. This is related to how computer science is taught in the universities and high schools. The situation has fortunately improved since 1995, even if this did not echoed yet completely outside the academic walls.

In 2007, GULP ran a survey to evaluate the extent of computational logic teaching at Italian universities. It turns out that nowadays declarative programming is being taught in 20 Italian universities at around 50 courses, at various levels in computer science and engineering curricula. Some of these courses have been running for as long as 20 years. In 80% of the cases, the syllabus includes practical lab sessions that teach students how to use SWI Prolog, SIC-Stus, ECLIPSe or other Prolog engines, ASP solvers such as DLV, SAT solvers and model checkers.

Notwithstanding the increased education and diffusion of LP at the students level, there is still a remarkable gap between the growth of academic research and industrial applications (see next section).

In our opinion, this may be due to the difficult international situation of computer and software industries, worsened by a specific Italian weakness in advanced industrial research, the crisis of AI technology and its influence on logic programming.

2 An overview

In this chapter, we give both a thematic overview and a historical perspective of LP applications in Italy. The material is huge, and we had to leave out many very interesting applications. However, we preferred to keep this presentation as broad as possible. Indeed, this is the first work that surveys LP applications in Italy in 25 years. Applications are organized by domains. At the end we draw conclusions and discuss future trends.

In the presentation we (arbitrarily) divided the various applications into 14 categories, which are expanded in corresponding and detailed subsections of this chapter:

1. industrial and commercial applications;
2. knowledge and information extraction, management and integration;
3. time tabling and rostering;
4. robotics;
5. graphics & design;
6. agent systems;
7. education, learning and cultural heritage;
8. software engineering;
9. verification;
10. natural language;
11. health care;
12. reasoning;
13. bioinformatics;
14. decision support, risk analysis and alarms.

It is complex task to create independent partitions, thus some applications may fall into various classes. We tried to cluster them according to their preponderant application area.

In Figure 1 we depict, for each year, the counting of applications partitioned by our categorization. For each project we considered its life-span over the years and classified it according its relevant application area. This plot aims only to track the development and evolution of logic programming applications. We only provide a rough depiction, merely counting instances. We do *not* account for person-months spent on projects nor fundings nor impact.

It is interesting to note that every mentioned application area was developed by at least one project in very recent years. Starting from a situation where,

between 1975 and 1985, the main applications were in the domain of robotics and industrial applications, the years 1985 – 1995 witnessed the stemming of many other applications areas. From 2000 on, it is notable the birth and rise of many applications based on agent systems, as well as bioinformatics and reasoning.

The plot also highlights the industrial applications trend, which notably decreased in proportion to the other areas of application. Witnessed by the examples reported below, pure industrial applications of Logic Programming are rather unfrequent. This indicates that the LP technology alone does not penetrate the market and often reveals its potential only when supported by academic groups that promote it during collaborations with external parties. Considering also the relevant number of students formed with (at least) the basics of LP, it is reasonable to imagine a number of independent solutions that we are not aware of, due to the difficulties in getting a detailed plot of the current situation.

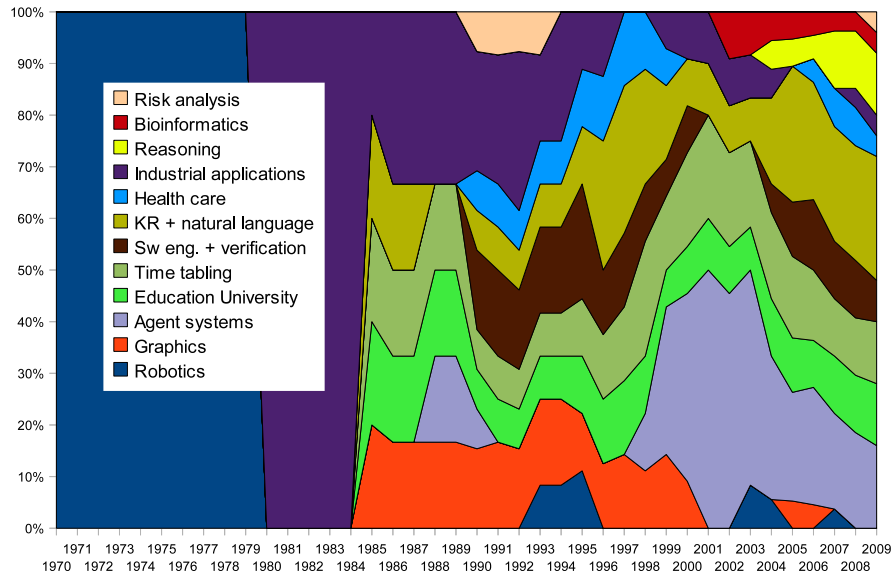


Fig. 1. The evolution of applications from 1970 to 2010

Figure 2 plots the summary of the total counting of applications (in black) described in this section. The temporal distribution gives an idea of the trend among the years. It is important to see that the growth has been constant, apart a small plateau started in mid 90s, that could be correlated to the APPIA-GULP-PRODE era (cfr the Historical Perspective chapter). Moreover, we show the fraction of applications that involve optimization techniques (time tabling,

scheduling, CLP(FD), etc). The peak, reached in the very last years, is supported by 28 different applications and projects, most of them currently alive.

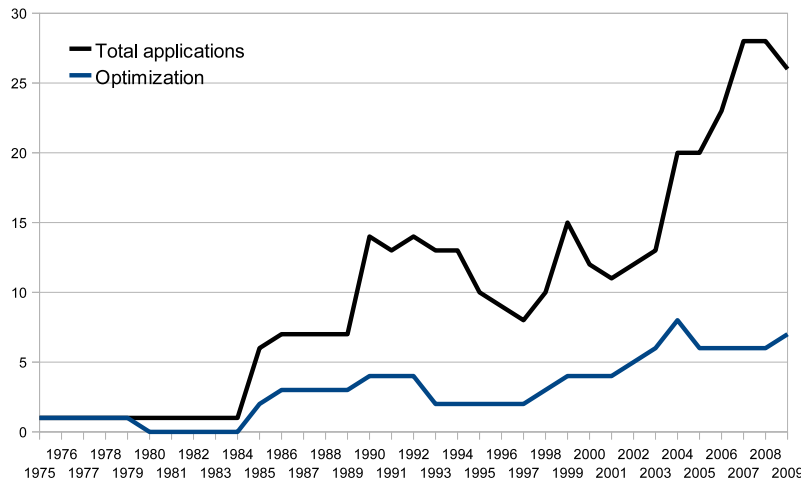


Fig. 2. 35 years of applications

3 Applications

We present here the list of applications, divided according to the main application areas. We would like to clarify the intended meaning for the concept of “application”, since this could be interpreted in different manners. The common semantics associated to the concept of “application” usually refers to some kind of project where Logic Programming takes a central role and the technology is developed outside of an academic context. This kind of applications is classified as *Industrial and commercial applications*. However this category is still poorly populated, if compared to other projects where, for example, LP is used to solve problems arising from other academic disciplines. Therefore, in this manuscript we also consider as application any project that was inspired by any context (including academic and public administration), that involved some LP technology and that developed at least to the stage of a working prototype.

3.1 Industrial and commercial applications

OMAR (late 80s) was an interactive system for predictive and reactive routing of the Alitalia fleet(s). Its kernel was developed entirely in Quintus Prolog and it consisted of 20.000 lines of code. It was the largest system of its kind. OMAR

was not an expert system, but a scheduler in which routing was modeled as a CSP, and constraints were about maintenances, schedule optimization/flexibility, geopolitical issues and so on.

It had a GUI written in C that made use of the Quintus interface, whereas the interface with the Alitalia flight information system was handled separately in SQL.

Since at that time there were no dedicated CLP languages yet, Prolog proved to be an ideal modeling and solving tool. Performance was improved by OMAR relying on approximated CSP algorithms (backward/forward checking, network consistency) rather than by standard backtracking, and by making non-determinism as local as possible. Prolog thus proved to be a powerful prototyping and delivery language.

Scientifically, the project was a success. The authors proved and experimentally confirmed that worst-case complexity was n^2 in the number of aircrafts and tasks. That meant that for example, fleet routing was completed in half a minute for the DC-9 fleet (26 aircrafts, 170 flights plus associated constraints) including the time to compute the dynamic data structures from the Alitalia database. Human experts took half to one hour to complete the task. Quality-wise, OMAR's routing solutions were comparable with those of a (human) senior scheduler.

Unfortunately, in spite of the technical quality and academic recognition of the OMAR project, because of mere data ownership issues, OMAR could never be used in practice.

In the early 90s we find two notable applications which also made it to the market.

IDEA [97] was an intelligent data retrieval system designed by Scancassani and colleagues at DS Logics srl in Bologna for the Epidemiological Observatory of Emilia Romagna. IDEA attempted to formalize the interaction between the data logical view and its real physical organization in databases. SICStus-Objects (a Prolog distribution extended with object-oriented features) was used to achieve an efficient implementation of an intelligent inference agent, whose task was to translate an epidemiologist's request for data into appropriate database queries and vice versa. Logic played a central role. The use of Prolog, with its well-understood semantics, made the semantics of IDEA reasoning rules clear and easy to debug. Prolog was used both to implement logical inference but also other non-logical components.

At the same time as IDEA, the SECR_eTS expert system [26] was being developed, in Prolog, by Chiopris and colleagues at ICON srl in collaboration with an Italian credit institute (BPS). SECR_eTS was sold to and used by several Italian banks to support the analysis of client-specific data. The application was successful in dealing with large amounts of data coming from the Italian Central Bank's Risk Center, whereas other traditional tools used earlier often failed to identify meaningful events and to set a clear boundary between monitoring and diagnosis. Logic programming could address situations that required non-trivial reasoning, which languages such as SQL could not, and it was suitable

for implementing classification and diagnosis procedures that help providing the user with an intuitive general view of the framework, instead of a sequence of raw alarms.

ELAND [20] was an expert LAN designer, that is, a rule-based expert system, written in Prolog, for configuring distributed information systems operating on a LAN. ELAND was developed using a real scenario but did not go beyond the prototyping stage.

In 2008, Mascardi and colleagues at the University of Genova implemented a MAS prototype [68, 17] for Ansaldo STS, that monitors processes running in a railway signalling plant, detects functioning anomalies, and provides support to the early notification of problems to the Command and Control System Assistance. The MAS has been implemented using DCaseLP, a multi-language prototyping environment developed at DISI, that provides libraries for integrating TuProlog agents into Jade (see Chapter by Baldoni et al for more information on these tools). Prolog has proven to be extremely well suited to the implementation of monitoring agents because of their intrinsic rule-based nature. The developed prototype was essential to demonstrate the functioning of the railway signalling system, which is now being developed using procedural languages.

Another optimization problem coming from a famous shoe industry was solved by Meneghetti (Udine) [69], with the application of Constraint Logic Programming. Floor storage system is used in the shoe industry to store fashion products of seasonal collections with low quantity and high variety. Since space is precious and order picking must be sped up, stacking of shoe boxes should be optimized. The problem is modeled by assigning an integer code to each box basing on shoe characteristics (model, type, color, and size) and trying to force similar boxes into near locations to improve workers ability of fast order retrieval. The model is encoded in Constraint Logic Programming and solved comparing different strategies, also using Large Neighborhood Search. Mixing CLP and LNS revealed a powerful methodology for solving allocation problems in floor storage systems for the shoe industry. Furthermore, the declarative nature of CLP allows the programmer to easily describe what properties are required to the desired solution. Requirements can be modified, added or deleted to adhere to a dynamic industrial environment without changing the basic model, but only declaring new constraints, making it adaptable and transferrable to different industrial realities.

3.2 Knowledge and information extraction, management and integration

Logic Programming technologies have been used in the late 90s until 2004 by Milanese and colleagues in the realization of MIRAGGIO: a prototypical design support system oriented to the dynamic generation and modification of design data and process models [32, 33, 73]. MIRAGGIO was implemented in Prolog++. It was based on a mixed object-Oriented and Frame-based approach, with the

aim of defining a flexible environment for design process modeling, for the reuse of previous project results, the dynamic reconfiguration and consistency check of data structures and the reuse previously developed design policies. To this end MIRAGGIO was using a two-level KDB. At the base level lied a knowledge database managing rule-based engineering and design knowledge, whereas at the meta-level lied the design system management rules, aimed to implement autonomous design strategy planning and decision support capabilities based on historical records and to specify its interactive behaviour with the designer. Although the system remained at a prototypical level, its mixture of classes and frames is still used in modern KBSE systems for engineering design support.

More recently information integration has largely been oriented towards ontologies. The used of Logic Programming languages such as Prolog to implement taxonomic reasoning is not new. We mention the Omega system for taxonomies by Attardi and Simi dating back to 1986 [8]. However, it is with the recent integration of ontological reasoners into high-performance logic programming reasoners such as SWI-Prolog and DLV that we start to have the first applications with concrete potential for real-world exploitation. In the Italian landscape, Leone and colleagues (Calabria) have been developing ontology-based knowledge management solutions for several application domains based on the DLV system. We mention some of them, and forward the interest reader to [57].

The industrial exploitation of DLV [62] in the area of Knowledge Management is explored by University of Calabria's spin-off company Exeura s.r.l. It maintains three industrial products: OntoDLV (ontology management) [91, 92], OLEX (document classification) [98, 36], and HiLeX (information extraction) [96, 95], incorporating the DLV system as the computational core. OntoDLV is an ontology management and reasoning system; OLEX is a document classification system; and, HiLeX is an information extraction system.

DLV has also been applied in the context of knowledge management at CERN (the European Laboratory for Particle Physics) for knowledge manipulation on large databases, in the context of the Automatic Itinerary Search, developed for the Government of the Calabria Region, and in the Team Building application, developed for the port authority of the Gioia-Tauro Seaport.

Finally, INFOMIX [61] is an information integration application.

In all cases, according to Leone [60], the key factor for the success of this sort of applications was the expressiveness of the ASP language and the ease of use of the ASP system DLV, rather than its efficiency.

3.3 Time tabling and rostering

Time tabling and rostering are some of the oldest applications of Logic Programming. Since 1985, Monfroglio has been developing and maintaining a high school time-tabling application written in LPA³ Prolog [76]. The application has been successfully experimented and adopted in educational institutions with

³ Logic Programming Associated ltd, London

more than 50 classes. In the 80s the application would take a couple of hours to return a results (much less on today's hardware). The advantages of this application with respect to other commercially available software is that it is flexible, since the user can propose an initial personal timetable, and that its output is of higher quality, since it is possible to achieve a better distribution of the teaching load.

Since 2004, Gavanelli (Ferrara) has been applying CLP for timetabling at Ferrara University's School of Engineering [55]. The application was written in CLP(FD) ECLiPSe. This is because timetabling is a classical combinatorial optimization problem, for which CLP(FD) is particularly suited. The system accommodates non-overlapping constraints about students, teaching staff and venues, and other constraints specifying student and teaching staff requirements, such as those regarding the equipment, capacity of classrooms, distance, etc. Gavanelli's system is currently used. The application is successful. It provides timely results, months before the start of lessons, thus enabling students to plan their curriculum well in advance. Moreover, it was able to eliminate the overlapping between elective and compulsory courses, thus addressing an existing unfortunate situation which was previously unsolved. Finally, CLP made it possible to specify some constraints which were out of bound for hand-written timetables, such as constraints on the timetables of students who switch curriculum from one year to another. The application, which has produced its output as a HTML document, is now interfaced instead with Google calendar: a more flexible format which enables the integration with other applications developed at the University of Ferrara.

Outside of the education environment, among the first timetabling applications we also mention the industrial-level one developed by Momigliano and colleagues within the OMAR project, discussed earlier on.

Similar to the school timetabling problem, the crew rostering problem was also addressed using Logic Programming by Caprara, Focacci, Lamma, Mello, Milano, Toth and Vigo (Bologna) in 1998. To this end, the authors use a mixture of AI and OR techniques, namely OR efficient procedures based on a mathematica approach to the problem, and CLP for knowledge representation, achieving declarativeness, non-determinism and an incremental style of programming. This line of research would later very successfully evolve into the integration of CP, AI, and OR techniques [13].

A recent project by Cipriano, Di Gaspero, and Dovier (Udine) [31], is a rostering system currently in use in the Neurology Dept. of the Udine University Hospital (2006). The project combined the employment of local search to improve a solution found by constraint programming and the exploitation of a constraint model to perform the exploration of the local neighborhood. Constraint programming is used for searching an initial feasible solution and subsequently it is improved by means of local search, using classical algorithms like hill climbing, steepest descent and tabu search. Moreover, a new local search algorithm (called hybrid steepest descent) is designed. It exploits a constraint programming model

for the exploration of neighborhood fragments. The results highlight the benefits of the hybridization approach w.r.t. their component algorithms. Acceptable solutions (with 20 doctors and a temporal horizons of one month) are obtained in a couple of minutes on a (average) PC.

3.4 Robotics

The very birth of Logic Programming in Italy happened in tight connection with research on Robotics. The first Prolog installation dates back in 1974, when Pagello brought from Grenoble a couple of tapes and two boxes full of punchcards with a copy of Alain Colmerauer's Prolog interpreter written in FORTRAN IV. Pagello had the interpreter running on a Univac 1108 machine at Politecnico di Milano. Later in 1975 a Prolog interpreter was also available at the University of Padova. In this way, Pagello and his colleagues could start developing Prolog applications for planning in robotic environments, addressing the Robot planformation problem introduced in Edinburgh by Bobrow, at the time the road of procedural constructs was mostly pursued, by Hewitt, Sussman, McDermott and others in the US. The results of this line of research was tested on a physical robot: a Cartesian manipulator provided by Olivetti. The main purpose was the automatic generation of action sequences for robotic mechanical assembly. The Olivetti robot was programmed in a language developed by Pagello and colleagues at the University of Padova and Politecnico di Milano. Logic Programming would provide the tools needed to design more easily a higher level planning system to automatically generate lower level instruction for the robot. The practical impact of this application was however limited. First order predicate calculus turned out to be suitable only for a not too complex and quite stable working environment with little indeterminism, whereas more complex assembly problems requiring sophisticated sensors, and mobile robotics, would be out of reach. The dichotomy between procedural and declarative programming grew larger while the limitations of the former were emphasized by the increasing complexity of robotic devices.

The interest in robotic application did not die with this first experience. In 1993 Natali, Omicini and Zanichelli (Bologna) [83] propose an architecture for a robot programming environment based on Prolog, extended with control capability toward program structuring and concurrence. The architecture reflect the interest, growing at that time, for the intelligent agent programming paradigms and theories. Thanks to 15 years of advances in Logic Programming, this research had overcome much of the limitations of earlier attempts, however its outcomes were not directly applied to physical robots. Ten years later, Galizia [54] uses disjunctive logic programming, and answer set planning in particular, to improve the Space Shuttle's planning capabilities under system malfunction conditions. Galizia implemented a system in DLV, building on work previously conducted at Texas Tech University's KR Lab [84]. Experimental evaluation was done by comparing the outputs of the newer system with those of the older one, but not on physical robots.

Interestingly, we find that in recent times the application of Logic Programming techniques is attracting the attention of strong Robotics research groups again. In 2007 Scalzo, Nardi and colleagues [19] define a robotic architecture that enables the design of robotic systems that exploit context information to adapt the behaviour of all their subsystems. Context information is modeled explicitly, to allow for automated reasoning to operate and suitably affect the robot's behaviour. Logic programming rules specify the robot subsystems' control. During a control cycle, the system acquires data from the sensors and from the navigation and planning subsystems, extracts symbolic contextual information from the analysis of data, and feeds it into a knowledge base and rule-based reasoning system, which in turn feeds control parameters back to the robot's subsystems. The approach has been implemented and validated on simulated and on physical robots.

3.5 Graphics & Design

In the graphics domain, we find applications of logic programming research mainly on computer-aided design and manufacturing and in 3D recognition.

In the late 80s, Milanese and colleagues (Udine) investigated the integration of the Graphics Kernel System (GKS) standard and Prolog, both for graphical entities modeling and for the implementation of a distributed graphics system [71]. Prolog was considered for its descriptive programming style, enabling modeling of graphical entities via their basic elements and relations. Besides, it would accommodate simple transformation rules to modify or create objects of increasing complexity. In fact, one of the limitations of GKS, which was starting to become a wide-spread standard, was its inability to build complex objects bottom-up starting from simpler ones, to modify parts of graphical objects, and to associate them with information regarding their nature and functionality. Milanese and colleagues extended Prolog with communication and modularization constructs to propose a distributed, multi-processor GKS model, using a two-level architecture. The base level would manage general implementation schemas, while the meta-level would accommodate their instantiation in the overall graphics scheme. Meta-programming would accommodate customization and reconfiguration methodologies in a flexible way.

In the early 90s, Milanese worked with Dulli and Visentin (Padova) on the definition of the KADMOS declarative language and its KAMPE developing environment to create and manipulate hierarchical models of graphical entities for CAD/CAM applications [72, 45, 44, 46]. KADMOS was a mixed logic programming and object-oriented language. Inference was used to create graphical entities that need to be validated with respect to design constraints, whereas object-orientation was needed for inheritance, classification, and modularization. Prolog was chosen as the implementation language for KADMOS.

In 1999, Cucchiara (Modena), Lamma, Mello, Milano (Bologna) and Piccardi (Ferrara) apply the CSP paradigm to 3D object recognition [35]. They propose

an approach for recognizing 3D CAD-made objects in complex range images containing several overlapped and different objects. The reasoning engine is based on Interactive CSP, to guide the acquisition of surfaces on-demand and focus only on significant image parts.

More recently, Farenzena and Fusiello (Verona) and Dovier (Udine) study the use of interval analysis and CLP to obtain an accurate geometric model of a scene that rigorously takes into account the propagation of data errors and roundoff [52].

3.6 Agent systems

Logic programming has given a large contribution to the development of agent and multi-agent systems specifications and verification languages and techniques. Yet, before agents had become so popular, at the end of the 80s Terna (Torino) implemented a Prolog for microeconomic behaviour simulation which we can also include in this section [103]. The program was modeling the Bank of Italy, Industry and Unions, to build an interaction system and implement in this way a sort of economics game.

Towards the end of the 90s, Ciampolini, Mello and Torroni (Bologna) and Lamma (Ferrara) developed an architecture and language for multi-agent hypothetical reasoning based on abductive logic programming (ALP). The architecture, ALIAS [27], was considering agents made of two modules: at the bottom level one for hypothetical reasoning, and at the top level one for communication. The language, LAILA [28], was offering communication primitives to accommodate distributed reasoning in collaboration, to produce globally consistent results, or in competition, to produce locally consistent results. ALIAS has been only applied to toy examples in agent negotiation, recommendation systems and judicial evaluation of criminal evidence [29]. However, the ideas proposed in ALIAS have been further developed in the DARE system developed at Imperial College London [64].

Contemporary to the development of ALIAS, we find work on logic-based agents by Torroni (Bologna) and Toni and Sadri (Imperial College London) [99, 100]. The use of ALP was crucial to the definition of negotiation policies in a declarative way, with an operational execution model underneath. These ideas remained central in the later EU-funded SOCS project, in which 6 universities in Europe, including Ferrara, Bologna and Pisa, collaborated in the definition of a computational logic model for the description, analysis and verification of global and open societies of heterogeneous logic based agents, named computees [104]. The SOCS models of agents and agent interaction were heavily based upon proof procedures for (various extensions of) logic programming. In particular, the operational model for KGP agents [59] relies upon CIFF [65], a proof procedure for abductive logic programming with constraints, and Gorgias, for logic programming with priorities. The operational model for agent societies [1] instead relies upon SCIFF [2], a proof procedure for abductive logic programming with arbitrarily quantified variables, CLP constraints, dynamic event handling and

reasoning with expectations. KGP and SCIFF have been applied to a variety of domains, including normative multi-agent system, recommendation systems, ambient intelligence, business process interaction, medical guidelines, Web service choreographies [24, 25], agent-oriented requirements engineering [18], and argumentation [105].

One of the most recent applications of SCIFF is the modeling and verification of declarative and open interaction models specified in the graphical ConDec language [89]. A mapping has been defined between ConDec and SCIFF [79]. Thanks to such a mapping, and to a number of SCIFF-based tools and extensions, it is possible to monitor and verify at run-time business process executions with respect to the model, and analyze traces after execution (process mining) [21, 22]. To this end, a Pro-M⁴ plug-in has been implemented based on SCIFF. The fully-fledged specification, verification and analysis framework is called CLIMB (Computational Logic for the verification and Modeling of Business processes and choreographies)⁵. CLIMB was used in national projects and on some case studies of chemical and physical analysis of waste water [63] in collaboration with HERA, a private agency, and ENEA, the Italian authority for energy and environment, and also in collaboration with other private manufacturing companies. The application of SCIFF for the static verification of business processes has been evaluated and successfully compared with state-of-the-art model checking techniques [80] to find that it offers greater flexibility and scalability in a number of realistic cases.

Since 1999, Costantini and Tocchio (L'Aquila) have been developing the DALI platform to specify agents and multi-agent systems based on computational logics. DALI is interoperable with other agent platforms such as FIPA. DALI is a general-purpose and agent-oriented logical language implemented in SICStus Prolog. DALI has been used for industrial applications and is patent pending. It is also used to teach AI and agents at L'Aquila.⁶

Since 1998, Omicini, Ricci, Denti, Viroli (Bologna, Cesena campus) Zambonelli (Modena and Reggio-Emilia) and Cremonini (Milano, Crema campus) have been working on the TuCSon service infrastructure⁷ [87, 88] for the coordination and communication among independent and concurrent software components, such as agents. Interaction relies on tuple centers, i.e., programmable tuple spaces characterized by a reactive behaviour that can be programmed at run-time. Tuples are first-order Prolog terms. The behaviour of tuple spaces is expressed via the ReSpecT language [86] TuCSon is written in Java, while ReSpecT relies on the tuProlog library [43, 90], a Prolog engine also written in Java. To date, TuCSon has been used in a number of applications, including the implementation of the Agent Coordination Context (ACC) [93], to manage the interaction space between agent and environment, of the minority game [85], experimentation of new simulation models for systems biology, based on multiagent

⁴ <http://prom.win.tue.nl/tools/prom/>

⁵ <http://lia.deis.unibo.it/research/climb/>

⁶ <http://www.di.univaq.it/stefcost/>

⁷ TuCSon website. <http://alice.unibo.it/xwiki/bin/view/TuCSon/>

systems, especially in relation with the Agents & Artifacts model (A&A) [77], in the implementation of a workflow management system prototype for virtual organizations [94], and for pervasive smart environments.

For more insights about the relations between the declarative and the agents and multi-agent system communities in Italy, we point to chapter by Baldoni and colleagues, in this volume.

3.7 Education, Learning and Cultural Heritage

Since its early days, dating back to 1985, Logic Programming has been used for education of high school teachers and in experimental projects with their students by Casadei (Bologna). Prolog was introduced and used for problem solving in diverse disciplines, to foster discussion and improve general problem solving skills, and to run problem solving competitions. From the previously mentioned enquiry made by GULP in 2007, about the teaching of Logic Programming at Italian Universities, it emerges that Prolog and other LP languages are currently being used in many curricula for teaching subjects such as Ai reasoning, knowledge representation, logics and theorem proving. Among them, Bandini, Mosca and Palmonari (Milano Bicocca), who have also used DLV for education-related initiatives, such as archeological analysis and classification of antique ceramics [82, 67, 81]. This project was conducted in collaboration with Bologna's Archeology department and more recently with the Archeometry research group at the University of Barcelona (EURAB). They have developed a system for the automatic generation of stratigraphic diagrams, and for related abductive (diagnostic) reasoning tasks [66]. DLV lends itself very well to the integration with external computational resources, that is components, written in other procedural languages, for the efficient computation of functions and data structures such as lists and sets. Data integration in DLV was particularly useful. Preliminary results on this application are available at the *Ipotesi di Preistoria* web site⁸

Since 2007, Gennari and colleagues (Bolzano) have been developing an educational Web-based tool oriented to learning impaired children, especially deaf children, called LODE [56, 70, 7]. LODE presents children with tales and exercises aimed to make them reason, globally, on temporal aspects of the narrative. On the server side, LODE uses the ECLIPSe CLP system, to generate exercises and to check their temporal coherence. The project is at an experimental stage.⁹

3.8 Software engineering

The Oikos project [4, 6, 5, 30] was carried on mainly by Montangero (Pisa) and Ciancarini (Bologna) in the first half of the 90s. The goal of the Oikos Project was to describe, in a declarative style, the software development processes and to use Logic Programming to program and execute software processes.

⁸ <http://ipotesidipreistoria.cib.unibo.it/>

⁹ LODE-CARITRO Web site: <http://lode.fbk.eu>

Oikos is a distributed software development environment where the activities' workflow can be modeled. It is specified and implemented using Extended Shared Prolog, a parallel logic language that deals with concurrency and distribution. It provides a blackboard-based communication framework in which experiments about different architectures can be performed and evaluated. The processes modeled by Oikos are the multi-user distributed nature, a long life span, open endedness and executability of models. Oikos predefines a number of services offering basic facilities, like access to data bases, workspaces, user interfaces, etc. Services are customizable, in a declarative way that matches naturally the way ESP defines and controls the software process. ESP allows to define services, to structure them in a dynamic hierarchy, and to coordinate them according to the blackboard paradigm.

The project produced a real case application where it was possible to enact software processes. The example considered a non trivial task of specification of a small language and the implementation of its compiler.

After this project the technology evolved towards the notion of workflow and workflow engine. Notable in this sense is the work done by Greco, Guzzo and Saccà on workflow executions [58] in which a rich graph representation of workflow schemes is combined with simple (i.e., stratified), yet powerful DATALOG rules to express complex properties and constraints on executions. The high expressive power of both the graphical and rule-based formalism provides the designer with powerful mechanisms for reasoning on workflows. Another notable body of work is the CLIMB framework by Montali and colleagues (Bologna), discussed in a previous section, and presented in Montali's PhD thesis [78] which received the GULP 2009 distinguished dissertation award.

3.9 Verification

An application of Logic Programming to software verification has been carried on by Bagnara, Zaffanella (Parma), Hill (Leeds) [11]. The project is still alive and started in the 90s. From 2005 the project switched gear and currently a prototype for industrial application is being developed.

Logic programming is the framework used for definition, analysis and automatic verification of syntactic and semantic properties of imperative languages (e.g., C, C++ and Java). In particular, the specification of concrete semantics is based on structured operational semantics and it can model runtime exceptions and non-structured flow control mechanisms. The specification, being a logic program, is executable and it is thus possible to verify the adherence of the specification against the reference standards.

Moreover, abstract semantics can be applied as well by implementing the techniques from Abstract Interpretation. This creates a general static analyzer that deals with an abstract specification.

Finally, it is possible to define some additional rules that restrict some syntactic and semantic possibilities that are a known possible source of errors, ranging from some unfortunate lexical choices in the language (identifiers containing "l"

and “1”) to some syntactical rules (each *switch* must have a *default* case) and some runtime errors (e.g., deallocation of null pointers). These kind of rules are seen as important contribution to standard compilers in industrial applications.

The project produced some prototypes that are currently being merged to a tool for showing the feasibility in industrial applications. Logic Programming reveals to be successful for these kind of applications also from the performances point of view: a prototype is able to verify the compliance of the Linux kernel to 80 coding rules in a few minutes.

Another important body of work concerns run-time interaction verification in open systems. This relates to the SOCS project mentioned earlier on. The main outcomes of the project are definitions of the agent and society models and the proof-procedures implementing the operational models. Run-time interaction verification accounts to monitoring and checking whether a particular implemented, running agent does indeed operate according to its specification. The SOCS project has produced logic-based tools that reason upon the externally observable behavior of interacting agents and verify whether it complies to predefined norms or protocols. In particular, the SOCS-SI tool [3] uses the abductive logic programming SCIFF proof-procedure [2] to consider messages exchanged by agents plus other events and carries out such a run-time interaction verification task.

3.10 Natural Language Processing

The application area based on natural language processing has flourished in the most recent years. We report here four different applications that are stimulated by the exponential growth and availability of text in natural language from the web and repositories of the last years.

Since 1996, Stefano Ferilli (Bari) coordinated a project for a general-purpose system for automatic learning of Datalog programs, starting from positive and negative concepts that are possibly correlated [49, 48, 47, 50]. It supports multi-strategies (namely induction, abduction and abstraction) and it is inherently incremental. It is completely written in Prolog and it is currently being developed and maintained. The system is integrated in the DOMINUS system (intelligent and automatic handling of electronic documents) and included in phases of analysis and elaboration of documents.

Another project, started in 2004, by Bos (La Sapienza) [37], focuses on natural language processing, in particular on computational semantics, i.e. mapping syntactic structures produced by a parser to first-order languages. The system designed is based on syntactic and semantic formalisms from theoretical linguistics and the implemented prototype is able to analyze the entire Gigaword corpus (1 billion words) in less than 5 days. The system is built around a wide-coverage Combinatory Categorial Grammar (CCG) parser and connected to the Boxer module [16] to produce interpretable structures in the form of Discourse Representation Structures (DRSs).

The resulting open-domain QA system, is well suited to analyzing large amounts of text containing a potential answer, because of its efficiency. The grammar is also well suited to analyzing questions, because of CCGs treatment of long-range dependencies. The system is active and available online and it has been downloaded by 800 people until the time of writing.

Another recent project is a prototype for the semantic search module of the LC3 project (MIUR Public-Private Lab). In particular, the subgoal coordinated by Di Martino (Napoli II) since 2007 is to build a natural language query parser, written in Prolog, that is able to detect conceptual patterns and to build a Query Ontology (<http://lc3.spacespa.it>). The project is still active.

Mnemosine is a new project, started in 2007, by Costantini and Paolucci (L'Aquila) [34]. They designed a semantic search engine capable of accepting natural language queries and of answering as well in natural language. Results are divided in classes of pertinence. The system is based on an iteration of refinement steps, where the user specifies interactively his/her search query. Mnemosine has been implemented into a working prototype and tested on the Italian pages of Wikipedia. The main features related to Logic Programming are: the use of SE-DCG, that are an extension of Definite Clause Grammar (DCG) of Prolog; answers to queries are generated from a knowledge base where a Prolog reasoner computes the results. The prototype is stable and scalable and currently being extended.

3.11 Health care

We have seen the IDE project on health-care support systems in the early 90s. Many years later, we again find LP laying at the core of some health-care applications. We report on three applications that involve tumor prevention, assisted living and classification of clinical diagnoses. They are all quite recent and they show that logic programming can be effectively employed to improve the quality of life.

The first project named SPINNER and PRITT SPRING [23], is a collaboration between academic and private parties. The project was developed between 2004 and 2006 by Mello, Montali, Chesani (Bologna), Storari (Ferrara), in collaboration with Dianoema S.p.A (Bologna).

The goal of the project was to realize a careflow system that implements workflow concepts in the clinical domain in order to administer, support and monitor the execution of health care services performed by different health care professionals and structures. The project concentrated on the monitoring aspects and provides a solution for the conformance verification of careflow process executions.

Given a careflow model, expressed with a simple graphical language for the specification of the careflow (GOSPEL), the system translates it to a formal language based on computational logic and abductive logic programming (SCIFF).

The main advantage of this formalism lies in its operational proof-theoretic counterpart, which is able to verify the conformance of a given careflow process execution (in the form of an event log) w.r.t. the model.

The feasibility of the approach has been tested on a case study related to the careflow process described in the cervical cancer screening protocol, based on the data provided by regional Health District.

The second application, started in 2007, is part of the *Secure and INDependent Living* (SINDI) system. The system offers advanced tools for monitoring dynamical, clinical and physical parameters. SINDI caters to two kinds of people: elder people that are clinically stable and chronic patients that can stay at home. The latter type of people often need to be educated to correct behaviors in order to limit health risks. SINDI offers medical professionals the tools to act before the verification of potential events that may limit the autonomy of the patients. The system is made of three parts: a wireless sensor network, an interface with the patient and Reasoning Component that is in charge of understanding the context by applying inference rules for meaningful data aggregation and interpretation.

The work of Bisiani, Mileo, Merico and Pinaridi (Nomadis Lab, Milano-Bicocca)[15, 74, 75], used non monotonic reasoning as part of SINDI's reasoning component.

SINDI is implemented in two prototypes, the first one focussing on the data retrieval and comparison with the clinical knowledge base. The second one takes the information inferred from the logic reasoner and integrates it to enable a better understanding of the collected data. The latest version of the prototype is currently under experimentation at the Monza Hospital.¹⁰

Finally, another DLV-based application. OLEX was employed for developing a system able to classify automatically case histories and documents containing clinical diagnoses. The system was commissioned, with the goal of conducting epidemiological analyses, by a local health authority in the Veneto region (ULSS of Asolo). The system classifies available case histories, in order to help the analysts while browsing and searching documents regarding specific pathologies, supplied services, or patients living in a given place etc. The application exploits an ontology of clinical case histories based on both the MESH (Medical Subject Headings) ontology and ICD9-CM a system employed by the Italian Ministry of the Health for handling data regarding medical services (e.g. X-Rays analyses, plaster casts, etc.). The analyzed documents are stored in PDF documents and contain medical reports, hospital discharge forms, clinical analysis results etc. Classification rules were manually devised and taken into account, beside the extracted linguistic information, also the metadata contained in the case history forms. The system has been deployed and is currently employed by the personnel of the ULSS of Asolo.

¹⁰ More details are available at: <http://www.nomadis.unimib.it/flex/cm/pages/ServeBLOB.php/L/IT/IDPagina/20>

3.12 Reasoning

Prolog is an programming language that lends itself particularly well to the implementation of other languages for reasoning. We mention some of them.

In the 90s Costantini (L'Aquila), Dell'Acqua (Linköping), Lanzarone (Insubria), Barklund (Uppsala) worked on a Prolog extension, named Reflective Prolog [53, 12]. The system allows to express meta-knowledge and includes an evaluation meta-level that is invoked when needed from the base level. The language supports three different kinds of variables: object variables, predicate meta-variables and function meta-variables. The rules of substitution ensure that these may only be substituted by, respectively, an object term, a representation of a predicate, and a representation of a function. There are syntactic restrictions to keep the meta-levels distinct and prevent self reference within a single atom. A reflective Prolog program distinguishes between the meta-evaluation level and the base level. The former is at the top of the meta-level architecture and the latter, containing an amalgamated theory, comprises the remaining meta-levels below it and can not refer to any predicates in the meta-evaluation level. Procedurally, a definite Reflective Prolog program uses SLD-resolution whenever possible but automatically switches between the levels in certain circumstances. The declarative semantics for such programs, called the Least reflective Herbrand Model, is an adapted form of the well-known least Herbrand model. The prototype, written in Quintus, has been later used to start new projects, e.g. DALI.

Started in 2004, Baldoni, Giacomini and Falda (Padova) realized a Temporal Reasoner capable of handling quantitative and qualitative uncertainty and vagueness [10, 51]. Temporal uncertainty is modeled in terms of possibility distributions and fuzzy relations. A Fuzzy Temporal Constraint Network is used to represent the knowledge about the considered scenario. Temporal reasoning inferences are performed by checking the consistency of the underlying network. The user interface is written in SWI Prolog, with less than 3K lines of code. The constraint solver is written in C++ and connects to the interface with XML files. In particular, the knowledge base manager normalizes the temporal expressions and defines a method for the consistent interpretation of expressions involving uncertainty. User scenarios are described with a simplified language and passed to the solver by XML files. Solver's output is used to generate answers in the same language. The prototype can handle fuzzy constraints (quantitative intervals and points, both precise and/or uncertain) and to generate temporal expression similar to natural language.

Since 2007, Costantini (L'Aquila) and Formisano (Perugia) have been developing the PRASP system (Resourced Answer Set Programming with Preferences): an extension of ASP to manage reasoning with bounded resources. The authors have developed a PRASP inference engine.

3.13 Bioinformatics

Bioinformatics, in broad terms, deals with the use of computational techniques to organize and extract knowledge from biological data; bioinformatics has successfully addressed problems in areas like recognition and analysis of DNA sequences, biological systems simulations, prediction of the spatial conformation of biological polymers, and ontological analysis of biomedical knowledge. An application of logic programming to bioinformatics started in 2003 by Dal Palù (Parma), Dovier (Udine), Pontelli (NMSU) and Fogolari (Udine). They address the problem of tertiary structure prediction using *ab initio* techniques, from the perspective of folding a protein sequence in a discretized representation of the three-dimensional space (viewed as a crystal lattice structure), optimizing an objective function which is related to the potential energy function of the resulting configuration. The problem translates into a CSP, where constraints are derived from physical properties of the molecules, and a set of heuristics that explore the search space effectively. A survey on the project is in [42]. A prototype was developed using Sicstus Prolog, CLP(FD) and parallelism [40, 39]. Another optimized solver was entirely rewritten in C++ [41] and extended traditional FD variables to three dimensional point variables. The work was also presented in Dal Palù's PhD thesis [38] which received the GULP 2006 distinguished dissertation award.

3.14 Decision Support, Risk Analysis and Alarms

A project carried on in the early 90s, by Sardu (System & Management), Serrecchia (La Sapienza), Omodeo (La Sapienza), Li (ECRC), Schuerman (ECRC), Véron (ECRC), was an application for Decision Support System (DSS) for the environmental pollution in the Venice lagoon [101]. The project was about the specification and design of an application based on parallel constraint logic programming. The DSS includes a database describing pollution sources and a lagoon hydrodynamic model, integrated through a knowledge-based core. The prototyping of the knowledge-based core was implemented in ElipSys (developed by ECRC), a parallel constraint logic programming system derived from CHIP.

Another application was developed by Avanzini, Rocchesso, Belussi, Dal Palù and Dovier (Verona) [9] and aimed at creating a new auditory alert system for high tides in Venice designed to replace the existing network of electromechanical sirens. The work was developed in collaboration with the Municipality of Venice (Center for Tide Prediction and Warning) in 2003. The project is composed of different parts including the analysis of the current alert system (sound simulation); the realization of a constraint logic programming tool to determine the optimal placement of loudspeakers in Venice, a complex task with many physical, economic, and social constraints (modeled with FD variables); the creation of alert sounds for the demanding listening environment. The final phase of the project involved iteratively validating and redesigning the alert signals using human testing. After some years, the project was actually installed in Venice, in particular the location of the loudspeakers followed the results of the optimization program.

A very recent collaboration, started in 2009, between Mascardi, Martelli and one of their students, Traverso (University of Genova), and Montolivo (Elsag-Datamat, a FinMeccanica company), focuses on risk analysis of complex infrastructures (harbors, airports, etc). Prolog was used to implement a first prototype for evaluating the feasibility of the approach. The prototype is able to computationally evaluate whether an attacker can violate the security apparatus of a given, simplified, infrastructure. A second prototype, implemented in Java extended with a Prolog-like backtracking mechanism, is much more sophisticated and might develop into a product. The project is protected by a NDA and the patent application has been recently filed.

4 Conclusions

In conclusion, Logic Programming applications are growing more numerous and diverse in several, old and new application domains. However, we note that the source of application domains is often coming from the academic world and that the collaborations with industrial partners are still the minority.

Now time is ripe for pushing the adoption of LP outside of academic entourage. While most of the private investors that were interested in LP 25 years ago have apparently left the stage, other new actors are coming into play. Exeura s.r.l. is a successful example of a company that is actually doing business and providing services with LP technology. There are many collaborations and projects with Public Administrations, such as municipalities and hospitals, that rely on LP and extensions. Indeed, we are now in a very different situation from that of 15 year ago. The main obstacles to LP adoptions, such as lack of education and problems of efficiency and integration, seem to have been overcome in many cases. Fifteen years ago we were wondering why LP is not used and what was missing, while today we can get some insights from many successful LP applications.

It is still true that average programmers and engineers are unable to write (correct and efficient) declarative programs, although we believe that the situation is better than it used to be. Programming methodologies and environments, debugging techniques, friendly interfaces did not evolve significantly compared to other popular imperative languages. However, these issues are confined to the production of LP-based solutions and do not affect the quality of the solutions themselves. LP technologies can now rely on efficient implementations, and offer unique degrees of flexibility. We can observe that the current trend is to develop competitive LP-based solutions for hard problems, which requires a solid background, education and high programming skills. This high quality profile, in the perspective of market globalization and considering the constant increase in the number of new and complex applications, is not necessarily a negative and penalizing aspect. We believe that competencies in declarative programming will become even more valuable in the next years.

Acknowledgements We would like to thank all the colleagues who helped us by providing some comments, material and summary of their activities. A particular thank to (in alphabetical order): Roberto Bagnara, Johan Bos, Giorgio Casadei, Paolo Ciancarini, Marco Colombetti, Stefania Costantini, Beniamino Di Martino, Agostino Dovier, Marco Falda, Stedano Ferilli, Marco Gavanelli, Rosella Gennari, Giuseppina Gini, Maria Gini, Viviana Mascardi, Paola Mello, Vitaliano Milanese, Alessandra Mileo, Alberto Momigliano, Angelo Monfroglio, Marco Montali, Alessandro Mosca, Daniele Nardi, Andrea Omicini, Enrico Pagello, Francesco Ricca, Carlo Matteo Scalzo, Pietro Terna.

References

1. M. Alberti, F. Chesani, M. Gavanelli, E. Lamma, P. Mello, and P. Torroni. The socs computational logic approach to the specification and verification of agent societies. In *Global Computing*, pages 314–339, 2004.
2. M. Alberti, F. Chesani, M. Gavanelli, E. Lamma, P. Mello, and P. Torroni. Verifiable agent interaction in abductive logic programming: The sciff framework. *ACM Trans. Comput. Log.*, 9(4), 2008.
3. M. Alberti, M. Gavanelli, E. Lamma, F. Chesani, P. Mello, and P. Torroni. Compliance verification of agent interaction: a logic-based software tool. *Applied Artificial Intelligence*, 20(2-4):133–157, 2006.
4. V. Ambriola, P. Ciancarini, and C. Montangero. Enacting software processes in Oikos. In *Proc. ACM SIGSOFT Conf. on Software Development Environments*, volume 15:6 of *sigsoft*, pages 12–23, 1990.
5. V. Ambriola, P. Ciancarini, and C. Montangero. Software Processes as a Hierarchy of Services in the Oikos Meta Environment. In *Proc. 6th Int. Software Process Workshop*, pages 57–60, Japan, 1990.
6. V. Ambriola, P. Ciancarini, and C. Montangero. The Logic Language ESP and its Programming Environment. In T. Kusalik and J. Levy, editors, *Proc. Workshop on Logic Programming Environments*, volume Technical Report IR-LP-31-25 of *ECRC (European Computer-Industry Research Centre)*, Eilat, Israel, June 1990.
7. B. Arfé, R. Gennari, and O. Mich. Before, while and after with lode and hearing novice readers. Tech Rep KRDB09-1, University of Bolzano, 2009.
8. G. Attardi and M. Simi. A description-oriented logic for building knowledge bases. *IEEE*, 74(10), 1986.
9. F. Avanzini, D. Rocchesso, A. Belussi, A. Dal Palù, and A. Dovier. Designing an urban-scale auditory alert system. *Computer*, 37(9):55–61, 2004.
10. S. Badaloni, M. Falda, and M. Giacomini. Integrating quantitative and qualitative constraints in fuzzy temporal networks. *AI Communications*, 17(4):183–272, 2004.
11. R. Bagnara, P. M. Hill, A. Pescetti, and E. Zaffanella. On the design of generic static analyzers for imperative languages. Quaderno 485, Dipartimento di Matematica, Università di Parma, Italy, 2008. Available at <http://www.cs.unipr.it/Publications/>.
12. J. Barklund, S. Costantini, P. Dell’Acqua, and G.A. Lanzarone. Reflection principles in computational logic. *Journal of Logic and Computation*, 10:6, December 2000.
13. R. Barták and M. Milano, editors. *Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems, Second International Conference, CPAIOR 2005, Prague, Czech Republic, May 30 - June 1,*

- 2005, *Proceedings*, volume 3524 of *Lecture Notes in Computer Science*. Springer, 2005.
14. L. Bazzocchi. Lo smau scopre l'intelligenza artificiale. *Office Automation*, pages 86–90, Nov 1988.
 15. R. Bisiani, D. Merico, A. Mileo, and S. Pinardi. A logical approach to home healthcare with intelligent sensor-network support. *The computer journal advance access*, 2009.
 16. J. Bos. Towards wide-coverage semantic interpretation. In *Proceedings of IWCS-6*, page 4253, Tilburg, The Netherlands, 2005.
 17. D. Briola, V. Mascardi, M. Martelli, G. Arecco, R. Caccia, and C. Milani. A prolog-based mas for railway signalling monitoring: Implementation and experiments. In M. Baldoni, M. Cossentino, F. De Paoli, and V. Seidita, editors, *Atti del Workshop Dagli Oggetti agli Agenti, WOA'08*. Seneca Edizioni, 2008.
 18. V. Bryl, P. Mello, M. Montali, P. Torroni, and N. Zannone. B-tropos: Agent-oriented requirements engineering meets computational logic for declarative business process modelling and verification. In *Proceedings of the 8th International Workshop on Computational Logic in Multi-Agent Systems (CLIMA-VIII), LNAI 5056*, pages 157–176, 2007.
 19. D. Calisi, L. Iocchi, D. Nardi, C. M. Scalzo, and V. A. Ziparo. Context-based design of robotic systems. *Robotics and Autonomous Systems*, 56(11):992–1003, 2008.
 20. Stefano Ceri and Letizia Tanca. Expert design of local area networks. *IEEE Expert: Intelligent Systems and Their Applications*, 5(5):23–33, 1990.
 21. F. Chesani, E. Lamma, P. Mello, M. Montali, F. Riguzzi, and S. Storari. Exploiting inductive logic programming techniques for declarative process mining. *Transactions on Petri Nets and Other Models of Concurrency II: Special Issue on Concurrency in Process-Aware Information Systems*, pages 278–295, 2009.
 22. F. Chesani, P. Mello, M. Montali, F. Riguzzi, M. Sebastianis, and S. Storari. Checking compliance of execution traces to business rules. In *Business Process Management Workshops*, pages 134–145, 2008.
 23. F. Chesani, P. Mello, M. Montali, and S. Storari. Testing careflow process execution conformance by translating a graphical language to computational logic. In *AIME*, pages 479–488, 2007.
 24. F. Chesani, P. Mello, M. Montali, S. Storari, and P. Torroni. On the integration of declarative choreographies and commitment-based agent societies into the sciff logic programming framework. *Multiagent and Grid Systems*, 2, 2010.
 25. F. Chesani, P. Mello, M. Montali, and P. Torroni. Verification of choreographies during execution using the reactive event calculus. In *WS-FM*, pages 55–72, 2008.
 26. C. Chiopris. The secrets banking expert system from phase 1 to phase 2. In *LPSS '92: Proceedings of the Second International Logic Programming Summer School on Logic Programming in Action*, pages 91–99, London, UK, 1992. Springer-Verlag.
 27. A. Ciampolini, E. Lamma, P. Mello, F. Toni, and P. Torroni. Co-operation and competition in alias: a logic framework for agents that negotiate. *Annals of Mathematics and Artificial Intelligence*, 37(1-2):65–91, January 2003.
 28. A. Ciampolini, E. Lamma, P. Mello, and P. Torroni. Laila: a language for coordinating abductive reasoning among logic agents. *Comput. Lang.*, 27(4):137–161, 2001.
 29. A. Ciampolini and P. Torroni. Using abductive logic agents for modeling the judicial evaluation of criminal evidence. *Applied Artificial Intelligence*, 18(3-4):251–275, 2004.

30. P. Ciancarini. Coordinating rule-based software processes with esp. *ACM Trans on Sw Engineering and Methodolgy*, 2(3):203–227, 1993.
31. R. Cipriano, L. Di Gaspero, and A. Dovier. Hybrid approaches for rostering: a case study in the integration of constraint programming and local search. In Maria J. Blesa Aguilera, Christian Blum, Andrea Roli, and Micheal Sampels, editors, *Hybrid Metaheuristics, Third International Workshop, HM 2006, Gran Canaria, Spain, October 13-15, 2006, Proceedings*, volume 4030 of *Lecture Notes in Computer Science*, pages 110–123. Springer Verlag, 2006.
32. G. Concheri and V. Milanese. Miraggio: a system for the dynamic management of product data and design models. *Advances in Engineering Software*, 32(7):527–543, 2001.
33. G. Concheri and Milanese V. Interaction as an issue in the development of effective tools for the management of the engineering knowledge base. In *XI ADM Conference*, volume B, pages 101–108, 1999.
34. S. Costantini and A. Paolucci. Semantically augmented dcg analysis for next-generation search engine. In *Proc. of CILC, Italian Conference on Computational Logic*, July 2008.
35. R. Cucchiara, E. Lamma, P. Mello, M. Milano, and M. Piccardi. 3d object recognition by vc-graphs and interactive constraint satisfaction. In *ICIAP '99: Proceedings of the 10th International Conference on Image Analysis and Processing*, page 508, Washington, DC, USA, 1999. IEEE Computer Society.
36. C. Cumbo, S. Iiritano, and P. Rullo. Olex - a reasoning-based text classifier. In *Proceedings of JELIA 2004*, pages 722–725, 2004.
37. J. R. Curran, S. Clark, and J. Bos. Linguistically motivated large-scale nlp with c&c and boxer. In *Proceedings of the ACL - Demonstrations Session (ACL-07 demo)*, pages 29–32, 2007.
38. A. Dal Palù. *Constraint Programming approaches to the Protein Structure Prediction Problem*. PhD thesis, University of Udine, 2006.
39. A. Dal Palù, A. Dovier, and F. Fogolari. Constraint logic programming approach to protein structure prediction. *BMC Bioinformatics*, 5(1):186, 2004.
40. A. Dal Palù, A. Dovier, and E. Pontelli. Heuristics, optimizations, and parallelism for protein structure prediction in clp(fd). In *PPDP '05: Proceedings of the 7th ACM SIGPLAN international conference on Principles and practice of declarative programming*, pages 230–241, New York, NY, USA, 2005. ACM.
41. A. Dal Palù, A. Dovier, and E. Pontelli. A constraint solver for discrete lattices, its parallelization, and application to protein structure prediction. *Softw. Pract. Exper.*, 37(13):1405–1449, 2007.
42. A. Dal Palù, A. Dovier, and E. Pontelli. Logic programming techniques in protein structure determination: Methodologies and results. In *LPNMR*, pages 560–566, 2009.
43. Enrico Denti, Andrea Omicini, and Alessandro Ricci. tu prolog: A light-weight prolog for internet applications and infrastructures. In *PADL*, pages 184–198, 2001.
44. S. Dulli, G. Galbiati, and V. Milanese. Hierarchical data structures and geometric modeling: a unified approach. *YUGRAPH'90 - The Fourth International Conference of Computer Graphics Automatika*, 31(1/2):37–42, 1990.
45. S. Dulli and V. Milanese. A graphic programming environment based on kadmos. *Comput. Graph. Forum*, 11(1):3–16, 1992.
46. S. Dulli, V. Milanese, and A. Visentin. A multiple windows user interface. *CAD/Graphics'93 New Advances in Computer Aided Design*, pages 186–188, 1993.

47. F. Esposito, N. Fanizzi, S. Ferilli, T.M.A. Basile, and N. Di Mauro. Incremental multistrategy learning for document processing. *Applied Artificial Intelligence Journal*, 17(8/9):859–883, 2003.
48. F. Esposito, N. Fanizzi, S. Ferilli, T.M.A. Basile, and N. Di Mauro. Incremental learning and concept drift in inthelex. *Intelligent Data Analysis Journal*, 8:3:213–237, 2004.
49. F. Esposito, N. Fanizzi, S. Ferilli, T.M.A. Basile, and N. Di Mauro. Multistrategy operators for relational learning and their cooperation. *Fundamenta Informaticae Journal*, 69:4:389–409, 2006.
50. F. Esposito, N. Fanizzi, S. Ferilli, and N. Di Mauro. Multistrategy theory revision: Induction and abduction in inthelex. *Machine Learning Journal*, 38(1/2):133–156, 2000.
51. M. Falda. Translating fuzzy temporal constraints in more natural expressions. In *ECAI 2008 workshop on Spatial and Temporal Reasoning*, pages 11–15, Patras, Greece, 2008.
52. M. Farenzena and A. Dovier. Reconstruction with interval constraints propagation. In *CVPR '06: Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 1185–1190, Washington, DC, USA, 2006. IEEE Computer Society.
53. D. M. Gabbay, C.J. Hogger, and J.A. Robinson, editors. *Handbook of Logic in Artificial Intelligence and Logic Programming*, volume 5 Logic Programming. Clarendon Press, Oxford, 1998.
54. S. Galizia. Generazione automatica di manovre per lo space shuttle mediante la programmazione logica disgiuntiva. In *APPIA-GULP-PRODE*, pages 97–109, 2003.
55. M. Gavanelli. University timetabling in ECLiPSe. *ALP Newsletter*, 19(3), August 2006.
56. R. Gennari and O. Mich. Lode- a logic-based e-learning tool for deaf children. Tech Rep KRDB07-3, University of Bolzano, 2007.
57. G. Grasso, S. Iiritano, N. Leone, and F. Ricca. Some dlw applications for knowledge management. In *LPNMR '09: Proceedings of the 10th International Conference on Logic Programming and Nonmonotonic Reasoning*, pages 591–597, Berlin, Heidelberg, 2009. Springer-Verlag.
58. G. Greco, A. Guzzo, and D. Sacca. A logic framework for reasoning on workflow executions. In *AGP 2004*, 2004.
59. A. C. Kakas, P. Mancarella, F. Sadri, K. Stathis, and F. Toni. Computational logic foundations of kgp agents. *J. Artif. Intell. Res. (JAIR)*, 33:285–348, 2008.
60. N. Leone. Exploiting asp in real-world applications: Main strengths and challenges. In *LPNMR '09: Proceedings of the 10th International Conference on Logic Programming and Nonmonotonic Reasoning*, pages 628–630, Berlin, Heidelberg, 2009. Springer-Verlag.
61. N. Leone, G. Greco, G. Ianni, V. Lio, G. Terracina, T. Eiter, W. Faber, M. Fink, G. Gottlob, R. Rosati, D. Lembo, M. Lenzerini, M. Ruzzi, E. Kalka, B. Nowicki, and W. Staniszki. The infomix system for advanced integration of incomplete and inconsistent data. In *Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD 2005)*, pages 915–917, 2005.
62. N. Leone, G. Pfeifer, W. Faber, T. Eiter, G. Gottlob, S. Perri, and F. Scarcello. The dlw system for knowledge representation and reasoning. *ACM Trans. Comput. Logic*, 7(3):499–562, 2006.

63. L. Luccarini, G. L. Bragadin, M. Mancini, P. Mello, M. Montali, and D. Sotara. Formal verification of wastewater treatment processes using events detected from continuous signals by means of artificial neural networks. *Environmental Modelling and Software*, in press, 2009.
64. J. Ma, A. Russo, K. Broda, and K. Clark. Dare: a system for distributed abductive reasoning. *Autonomous Agents and Multi-Agent Systems*, 16(3):271–297, 2008.
65. P. Mancarella, G. Terreni, F. Sadri, F. Toni, and U. Endriss. The ciff proof procedure for abductive logic programming with constraints: Theory, implementation and experiments. *CoRR*, abs/0906.1182, 2009.
66. G. Mantegari, A. Mosca, and M. Cattani. Formal knowledge representation and automated reasoning for the study of archaeological stratigraphy. In *12th International Congress Cultural Heritage and New Technologies*, 2007.
67. G. Mantegari, A. Mosca, B. Rondelli, and G. Vizzari. A semantic based approach to gis: the po-basyn project. In *36th Annual Conference on Computer Applications and Quantitative Methods in Archaeology: On the Road to Reconstructing the Past (CAA 2008)*, 2008.
68. V. Mascardi, D. Briola, M. Martelli, R. Caccia, and C. Milani. Monitoring and diagnosing railway signalling with logic-based distributed agents. In R. Zunino E. Corchado, editor, *Proceedings of the International Workshop on Computational Intelligence in Security for Information Systems CISIS08*, volume 53 of *Lecture Notes in Computer Science*, pages 108–115. Springer, Berlin, 2009.
69. A. Meneghetti. Optimizing allocation in floor storage systems for the shoe industry by constraint logic programming. In *Proceedings of ISDA, International Conference on Intelligent Systems Design and Applications*, 2009.
70. O. Mich. Constraint-based temporal reasoning and e-learning tools for deaf users. Tech Rep KRDB08-1, University of Bolzano, 2008.
71. V. Milanese. A prolog environment for gks-based graphics. *Comput. Graph. Forum*, 7(1):9–20, 1988.
72. V. Milanese. Kadmos: A clausal language for cad modeling systems with morphological constraints. *Comput. Graph. Forum*, 9(1):39–51, 1990.
73. V. Milanese. Using semantics in engineering design. In *Proceedings of CIM 2003*, pages 369–378, 2003.
74. A. Mileo, D. Merico, and R. Bisiani. A logic programming approach to home monitoring for risk prevention in assisted living. In *ICLP*, pages 145–159, 2008.
75. A. Mileo, D. Merico, and R. Bisiani. Wireless sensor networks supporting context-aware reasoning in assisted living. In *PETRA*, page 54, 2008.
76. A. Monfroglio. Timetabling through a deductive database: a case study. *Data and Knowledge Engineering*, 3(1):1–27, 1988.
77. S. Montagna, A. Ricci, and A. Omicini. A&a for modelling and engineering simulations in systems biology. *Int. J. Agent-Oriented Softw. Eng.*, 2(2):222–245, 2008.
78. M. Montali. *Specification and Verification of Open Declarative Interaction Models: a Logic-Based Framework*. PhD thesis, University of Bologna, 2009.
79. M. Montali, M. Pesic, W. M. P. van der Aalst, F. Chesani, P. Mello, and S. Storari. Declarative specification and verification of service choreographies. *ACM Transactions on the Web*, 2009.
80. M. Montali, P. Torroni, M. Alberti, F. Chesani, M. Gavanelli, E. Lamma, and P. Mello. Verification from declarative specifications using logic programming. In *ICLP '08: Proceedings of the 24th International Conference on Logic Programming*, pages 440–454, Berlin, Heidelberg, 2008. Springer-Verlag.

81. A. Mosca and D. Bernini. Ontology-driven geographic information system and dlvh reasoning for material culture analysis. In *R.i.C.e.R.c.A. 2008: RCRA Incontri E Confronti, online proceedings of the Italian Workshop RiCeRcA 2008*, 2008.
82. A. Mosca, Rondelli, and G. Mantegari. Integrating a knowledge-based system and a geographical information system for the study of the archaeological material culture. In *Artificial Intelligence in Cultural Heritage Workshop, 10th Symposium on Advances in Artificial Intelligence of the Italian Association for Artificial Intelligence*, pages 84–91, 2008.
83. A. Natali, A. Omicini, and F. Zanichelli. Exploiting logic programming in robot applications. In *GULP*, pages 535–548, 1993.
84. M. Nogueira, M. Balduccini, M. Gelfond, R. Watson, and M. Barry. An a prolog decision support system for the space shuttle. In *Answer Set Programming*, 2001.
85. E. Oliva, M. Viroli, and A. Omicini. Simulation of minority game in tucson. In *WOA*, 2006.
86. A. Omicini and E. Denti. From tuple spaces to tuple centres. *Sci. Comput. Program.*, 41(3):277–294, 2001.
87. A. Omicini and F. Zambonelli. Coordination of mobile information agents in tucson. *Internet Research: Electronic Networking Applications and Policy*, 8(5):400–413, 1998.
88. A. Omicini and F. Zambonelli. Coordination for internet application development. *Autonomous Agents and Multi-Agent Systems*, 2(3):251–269, 1999.
89. M. Pesic and W. M. P. van der Aalst. A declarative approach for flexible business processes management. In *Business Process Management Workshops*, pages 169–180, 2006.
90. G. Piancastelli, A. Benini, A. Omicini, and A. Ricci. The architecture and design of a malleable object-oriented prolog engine. In *SAC*, pages 191–197, 2008.
91. F. Ricca, L. Gallucci, R. Schindlauer, T. Dell’Armi, G. Grasso, and N. Leone. Ontodlv: an asp-based system for enterprise ontologies. *J. Log. and Comput.*, 19(4):643–670, 2009.
92. F. Ricca and N. Leone. Disjunctive logic programming with types and objects: The dlv+ system. *Journal of Applied Logics*, 5(3):545–573, 2007.
93. A. Ricci and A. Omicini. Agent coordination contexts: Experiments in tucson. In *WOA*, pages 14–21, 2002.
94. A. Ricci, A. Omicini, and E. Denti. The tucson coordination infrastructure for virtual enterprises. In *WETICE*, pages 348–353, 2001.
95. M. Ruffolo, N. Leone, M. Manna, D. Saccà, and A. Zavatto. Exploiting asp for semantic information extraction. In *Answer Set Programming*, pages 248–262, 2005.
96. M. Ruffolo and M. Manna. Hilex: A system for semantic information extraction from web documents. *Enterprise Information Systems, LNBIP*, 3(3):194–209, 2008.
97. C. Ruggieri, M. Sancassani, N. Dore, F. Russo, and U. Manfredi. Intelligent data retrieval in prolog: An illuminating idea. *J. Log. Program.*, 26(2):169–198, 1996.
98. P. Rullo, V.L. Policicchio, C. Cumbo, and S. Iiritano. Olex: Effective rule learning for text categorization. *IEEE Transactions on Knowledge and Data Engineering, in press TKDE-2007-07-0386.R3*, 2007.
99. F. Sadri, F. Toni, and P. Torroni. Dialogues for negotiation: Agent varieties and dialogue sequences. In *ATAL*, pages 405–421, 2001.
100. F. Sadri, F. Toni, and P. Torroni. An abductive logic programming architecture for negotiating agents. In *JELIA*, pages 419–431, 2002.

101. G. Sardu, G. Serrecchia, E. Omodeo, L. Li, K. Schuerman, and A. Véron. Safeguarding the venice lagoon: An application of a knowledge-based dss. In *GULP*, pages 519–534, 1993.
102. M.I. Sessa, editor. *1985 – 1995: Ten years of Logic Programming in Italy*. Palladio, 1995.
103. P. Terna. *Rassegna di strumenti informatici*. Giappichelli, Torino, 1988.
104. F. Toni and P. Torroni, editors. *Computational Logic in Multi-Agent Systems, 6th International Workshop, CLIMA VI, London, UK, June 27-29, 2005, Revised Selected and Invited Papers*, volume 3900 of *Lecture Notes in Computer Science*. Springer, 2006.
105. P. Torroni, M. Gavanelli, and F. Chesani. Argumentation in the semantic web. *IEEE Intelligent Systems*, 22(6):66–74, 2007.