Designing Intelligent Database Systems through Argumentation

Marcela Capobianco Carlos I. Chesñevar Alejandro J. García
Guillermo R. Simari

Dpto. de Cs. e Ing. de la Computación – Universidad Nacional del Sur
Tel.: 0291-4595135 – Av. Alem 1253, Bahía Blanca
{mc, cic, ajg, grs}@cs.uns.edu.ar

1 Introduction

Information systems play an ever-increasing key role in our society. In particular, data intensive applications are in constant demand and there is need of computing environments with much more intelligent capabilities than those present in today's Data-base Management Systems (DBMS). Nowadays intelligent applications require better reasoning capabilities than those present in older data processing systems[7].

Argumentation frameworks appear to be an excellent starting-point for building such systems. Research in argumentation has provided important results while striving to obtain tools for common sense reasoning. As a result, argumentation systems have substantially evolved in the past few years, and this resulted in a new set of argument-based applications in diverse areas where knowledge representation issues play a major role. Clustering algorithms [6], intelligent web search [3], recommender systems [4, 3], and natural language assessment [2] are the outcome of this evolution.

We claim that massive data processing systems can be combined with argumentation to obtain systems that administer and reason with large databases. These would result in a system that can extract and process information from massive databases and exhibit intelligent behavior and common sense reasoning as a by-product of the argumentation system used for the reasoning process.

In this work, we present a specialization of the DeLP [5] system, called Database Defeasible Logic Programming (DB_DeLP). This framework could be easily integrated with a relational database component to achieve a system capable of massive data processing and intelligent behavior. Next, we formally define this system.

2 DBDeLP: Database Defeasible Logic Programing

In this section we present an specialization of the DeLP language, called *Database Defeasible Logic Programing* (DB_DeLP), that can be used as a reasoning module in a Database integrated system. DB_DeLP has common elements with the ODeLP language [1], another specialization of DeLP useful for dynamic environments that provides perception mechanisms to detect the changes in the world and integrate them into the knowledge base.

The language of DB_DeLP is based on the language of logic programming. Standard logic programming concepts (such as signature, variables, functions, etc) are defined in the usual way. Literals are atoms that may be preceded by the symbol "$\sim$" denoting *strict* negation, as in extended logic programming.

DB_DeLP programs are formed by a set of *defeasible rules*. These rules provide a way of performing tentative reasoning as in other argumentation formalisms. A *defeasible rule* has the form $L_0 \prec L_1, L_2, \ldots, L_k$, where L_0 is a literal and $L_1, L_2, \ldots, L_k$ is a non-empty finite set of literals. An *DB_DeLP program*, noted as Δ, is a finite set of defeasible rules.

Example 2.1 Lets consider the following program: $\Delta = \{\mathsf{feline(X)} \prec \mathsf{lion(X).}, \mathsf{climbs(X)} \prec \mathsf{feline(X).}, \sim\mathsf{climbs(X)} \prec \mathsf{lion(X).}, \sim\mathsf{climbs(X)} \prec \mathsf{sick(X).}, \mathsf{climbs(X)} \prec \mathsf{lion(X),\ cub(X).}\}$ which shows general information about lions' behavior. The rules establish that felines usually climb trees, while lions usually do not. Exceptionally, lion cubs could climb trees and usually sick animals cannot climb trees.

Defeasible rules are combined to build *potential arguments*. Since these rules are not instantiated, potential argument may become arguments if the necessary evidence is available, they can be roughly described as schematic arguments that may be instantiated to become concrete arguments. Therefore every potential argument represents a set of instantiated arguments that are obtained using *different* instances of the *same* defeasible rules. These instances are summarized in the concept of argument (instantiated), that is later used to define potential arguments:

Definition 2.1 Let Δ be a set of defeasible rules and Ξ a set of ground literals. An *argument* $\mathscr{A}$ for a ground literal $\overline{Q}$, also denoted $\langle \mathscr{A}, \overline{Q} \rangle$, is a subset of ground instances of defeasible rules such that: (1) there exists a defeasible derivation [5] for $\overline{Q}$ from $\Xi \cup \mathscr{A}$ (denoted as $\Xi \cup \mathscr{A} \mid\!\sim \overline{Q}$), (2) $\Xi \cup \mathscr{A}$ is non-contradictory, and (3) $\mathscr{A}$ is minimal with respect to set inclusion in satisfying (1) and (2). Given two arguments $\langle \mathscr{A}_1, Q_1 \rangle$ and $\langle \mathscr{A}_2, Q_2 \rangle$, we will say that $\langle \mathscr{A}_1, Q_1 \rangle$ is a *sub-argument* of $\langle \mathscr{A}_2, Q_2 \rangle$ iff $\mathscr{A}_1 \subseteq \mathscr{A}_2$.

Definition 2.2 Let Δ be a set of defeasible rules. A subset A of Δ is a *potential argument* for a literal Q, noted as $\langle\langle \mathsf{A}, \mathsf{Q} \rangle\rangle$, if there exists a non-contradictory set of ground literals Ξ, an instance $\mathscr{A}$ of the rules in A, and a grounded instance R of Q such that $\langle \mathscr{A}, R \rangle$ is an argument wrt Δ and Ξ.

Given a particular set of defeasible rules Δ we can obtain a finite set of potential arguments based on it. This set will be recorded to speed-up inference, but with a particular structure. In the inference process arguments sustain theirs conclusions trough a dialectical process. To do this, arguments in favor and against a certain conclusion are contested. These conflicts are formally defined in the notion of attack or counter-argument between potential arguments. A potential argument $\langle\langle \mathsf{A}_1, \mathsf{Q}_1 \rangle\rangle$ *counter-argues* an potential argument $\langle\langle \mathsf{A}_2, \mathsf{Q}_2 \rangle\rangle$ at a literal Q if and only if there is a sub-argument $\langle\langle \mathsf{A}, \mathsf{Q} \rangle\rangle$ of $\langle\langle \mathsf{A}_2, \mathsf{Q}_2 \rangle\rangle$ such that Q_1 and Q are complementary with respect to strong negation, denoted as $\sim$.

Defeat among potential arguments is defined combining the counter-argument relation and a preference criterion "$\preceq$". A potential argument $\langle\langle \mathsf{A}_1, \mathsf{Q}_1 \rangle\rangle$ *defeats* $\langle\langle \mathsf{A}_2, \mathsf{Q}_2 \rangle\rangle$ if $\langle\langle \mathsf{A}_1, \mathsf{Q}_1 \rangle\rangle$ is a counterargument of $\langle\langle \mathsf{A}_2, \mathsf{Q}_2 \rangle\rangle$ at a literal Q and $\langle\langle \mathsf{A}, \mathsf{Q} \rangle\rangle \preceq \langle\langle \mathsf{A}_1, \mathsf{Q}_1 \rangle\rangle$ (proper defeater) or $\langle\langle \mathsf{A}_1, \mathsf{Q}_1 \rangle\rangle$ is unrelated to $\langle\langle \mathsf{A}, \mathsf{Q} \rangle\rangle$ (*blocking defeater*).

Defeaters may in turn be defeated by others potential arguments. Thus, a complete dialectical analysis is required to determine which arguments are ultimately accepted. But before turning to the particular details of the inference process, we must define the supporting structure that will be used in it. The dialectical process will be guided by the dialectical graph associated to the set of rules Δ.

Definition 2.3 Let Δ be a set of defeasible rules. The *dialectical graph* of Δ, denoted as $\mathscr{DG}_\Delta$, is a 3-tuple $(PotArgs(\Delta), D_p, D_b)$ such that: (1) $PotArgs(\Delta)$ is the set $\{\langle\langle \mathsf{A}_1, \mathsf{Q}_1 \rangle\rangle, \ldots, \langle\langle \mathsf{A}_k, \mathsf{Q}_k \rangle\rangle\}$ of every potential argument that can be built from Δ. (2) D_p and D_b are relations over the elements of $PotArgs(\Delta)$ such that for every pair $(\langle\langle \mathsf{A}_1, \mathsf{Q}_1 \rangle\rangle, \langle\langle \mathsf{A}_2, \mathsf{Q}_2 \rangle\rangle)$ in D_p (respectively D_b) it holds that $\langle\langle \mathsf{A}_2, \mathsf{Q}_2 \rangle\rangle$ is a proper (respectively blocking) defeater of $\langle\langle \mathsf{A}_1, \mathsf{Q}_1 \rangle\rangle$.

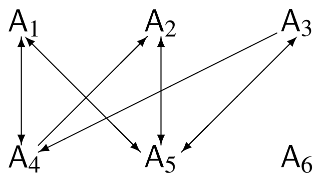

Figure 1: Dialectical graph corresponding to Example 2.2.

Example 2.2 Consider the rules in Ex. 2.1, its dialectical graph is composed by:
$\langle\langle A_1, \mathsf{climbs(X)}\rangle\rangle$, where $A_1 = \{\mathsf{climbs(X)} \prec \mathsf{feline(X)}\}$;
$\langle\langle A_2, \mathsf{climbs(X)}\rangle\rangle$, where $A_2 = \{\mathsf{climbs(X)} \prec \mathsf{feline(X)}, \mathsf{feline(X)} \prec \mathsf{lion(X)}\}$;
$\langle\langle A_3, \mathsf{climbs(X)}\rangle\rangle$, where $A_3 = \{\mathsf{climbs(X)} \prec \mathsf{lion(X),\ cub(X)}\}$;
$\langle\langle A_4, \sim\mathsf{climbs(X)}\rangle\rangle$, where $A_4 = \{\sim\mathsf{climbs(X)} \prec \mathsf{lion(X)}\}$;
$\langle\langle A_5, \sim\mathsf{climbs(X)}\rangle\rangle$, where $A_5 = \{\sim\mathsf{climbs(X)} \prec \mathsf{sick(X)}\}$;
$\langle\langle A_6, \mathsf{feline(X)}\rangle\rangle$, where $A_6 = \{\mathit{feline(X)} \prec \mathit{lion(X)}\}$.
The relations $D_b = \{(A_1,A_4),(A_4,A_1),(A_1,A_5),(A_5,A_1),(A_2,A_5),(A_5,A_2),(A_3,A_5),(A_5,A_3)\}$ and $D_p = \{(A_2,A_4),(A_4,A_3)\}$ are shown in Figure 2, where blocking defeat is indicated with a double headed arrow. In this case the specificity criterion is used as the preference criterion for defeat.

A dialectical graph for a given set of rules collects a set of potential arguments and the defeat relation among them. Bear in mind that every potential argument functions like an schema that represents a set of arguments that can be obtained using *different* instances of the *same* defeasible rules. This speeds-up the inference process functioning like a precompiled knowledge component [1], without need of generating and storing many arguments which are structurally identical, only differing in the constant names being used to build the corresponding derivations.

Now we can finally define our argumentation framework:

Definition 2.4 Let Δ be a set of defeasible rules. The database argumentation framework of Δ is a pair $AF = (\mathscr{DB}_\Delta, \Xi)$, where $\mathscr{DB}_\Delta$ is the dialectical graph obtained from Δ and Ξ, called the evidential base, is a relational database.

The facts derived from Ξ will be used as evidence to instantiate the potential arguments in the dialectical graph. This gives raise to the inference process of the system.

3 Inference process

The inference process starts when a new query is posed to the system, regarding the derivation of a given conclusion. Consider the dialectical graph in example 2.2 and suppose the following set of facts can be derived from the evidential base: $\{$*lion(simba).*, *lion(mufasa).*, *cub(simba).*$\}$ Lets see how the system works when faced with the query *climbs(simba)*.

First the set of potential arguments in the dialectical graph is searched to see if there exists an element whose conclusion can be instantiated to match the query. It finds $\langle\langle A_2, \mathsf{climbs(X)}\rangle\rangle$, $A_2 = \{\mathsf{climbs(X)} \prec \mathsf{feline(X)}, \mathsf{feline(X)} \prec \mathsf{lion(X)}\}$. A_2 can be instantiated to $\mathscr{A}_2 = \{\mathit{climbs(simba)} \prec \mathit{feline(simba)}, \mathit{feline(simba)} \prec \mathit{lion(simba)}\}$.

Now, to see if *climbs(simba)* is inferred by the system from Δ and the evidential base, we must check whether $\mathscr{A}_2$ can sustain its conclusion when confronted with its counterarguments. Using the links in the dialectical graph we find one defeater for $\mathscr{A}_2$, instantiating $A_4 = \{\sim\mathsf{climbs(X)} \prec \mathsf{lion(X)}\}$ to $\mathscr{A}_4 = \{\sim\mathit{climbs(simba)} \prec \mathit{lion(simba)}\}$. But defeaters are arguments and so may in turn be defeated as well. This gives raises to a dialectical process [5] in which we recursively find defeaters for the defeaters and so on. Such analysis results in a tree structure called *dialectical tree*, in which arguments are nodes labeled as undefeated (*U-nodes*) or defeated (*D-nodes*) according to a marking procedure.

Definition 3.1 The *dialectical tree* for an argument $\langle \mathscr{A}, \overline{Q} \rangle$, denoted $\mathscr{T}_{\langle \mathscr{A}, Q \rangle}$, is recursively defined as follows: (1) A single node labeled with an argument $\langle \mathscr{A}, \overline{Q} \rangle$ with no defeaters (proper or blocking) is by itself the dialectical tree for $\langle \mathscr{A}, \overline{Q} \rangle$. (2) Let $\langle \mathscr{A}_1, Q_1 \rangle, \langle \mathscr{A}_2, Q_2 \rangle, \ldots, \langle \mathscr{A}_n, Q_n \rangle$ be all the defeaters (proper or blocking) for $\langle \mathscr{A}, \overline{Q} \rangle$. The dialectical tree for $\langle \mathscr{A}, \overline{Q} \rangle$, $\mathscr{T}_{\langle \mathscr{A}, Q \rangle}$, is obtained by labeling the root node with $\langle \mathscr{A}, \overline{Q} \rangle$, and making this node the parent of the root nodes for the dialectical trees of $\langle \mathscr{A}_1, Q_1 \rangle, \langle \mathscr{A}_2, Q_2 \rangle, \ldots, \langle \mathscr{A}_n, Q_n \rangle$.

For the marking procedure we start labeling the leaves as *U-nodes*. Then, for any inner node $\langle \mathscr{A}_2, Q_2 \rangle$, it will be marked as *U-node* iff every child of $\langle \mathscr{A}_2, Q_2 \rangle$ is marked as a *D-node*. If $\langle \mathscr{A}_2, Q_2 \rangle$ has at least one child marked as *U-node* then it is marked as a *D-node*.

Given a query Q for a given framework $AF = (\mathscr{DB}_\Delta, \Xi)$, we will say that Q is *warranted* wrt $\mathscr{P}$ iff there exists an argument $\langle \mathscr{A}, \overline{Q} \rangle$ such that the root of its associated dialectical tree $\mathscr{T}_{\langle \mathscr{A}, Q \rangle}$ is marked as a U-node. Solving a query Q in DB_DeLP accounts for trying to find a warrant for Q, that is, an argument for Q that is warranted wrt to the DB_DeLP under consideration.

Example 3.1 Continuing with example 2.1, let *climbs(simba)* be a query to that framework. In this case argument $\langle \mathscr{A}_2$, *climbs(simba)*$\rangle$ (an instantiation of $\langle\langle \mathsf{A}_2, \mathsf{climbs(X)} \rangle\rangle$) is attacked by argument $\langle \mathscr{A}_4$, ~*climbs(simba)*$\rangle$ (an instantiation of $\langle\langle \mathsf{A}_4, \sim\mathsf{climbs(X)} \rangle\rangle$), which is attacked by $\langle \mathscr{A}_3$, *climbs(simba)*$\rangle$ (an instantiation of $\langle\langle \mathsf{A}_3, \mathsf{climbs(X)} \rangle\rangle$).
Using specificity as the preference criterion, $\langle \mathscr{A}_4$, ~*climbs(simba)*$\rangle$ is a proper defeater for argument $\langle \mathscr{A}_2$, *climbs(simba)*$\rangle$, but $\mathscr{A}_4$ is properly defeated by $\langle \mathscr{A}_3$, *climbs(simba)*$\rangle$. Thus, *climbs(simba)* is warranted. The structure of these arguments and the dialectical tree are shown in Fig. 2 (left), where defeated nodes are marked with a D and undefeated nodes with an U.

The evidential base is subject to constant changes as every practical database does. The only restriction is that it must not be changed while a query is being solved. Note that the dialectical graph is not affected by changes in the database, as the following example shows:

Example 3.2 Suppose now that a new fact is added to the evidential base stating that simba is sick. Even though the dialectical graph remains unchanged, now the potential argument $\langle\langle \mathsf{A}_5, \sim\mathsf{climbs(X)} \rangle\rangle$ can be instantiated to argument $\langle \mathscr{A}_5$, ~*climbs(simba)*$\rangle$ (see Fig. 2 (right)).
Using specificity as the preference criterion, $\langle \mathscr{A}_5$, ~*climbs(simba)*$\rangle$ is a blocking defeater for both $\mathscr{A}_2$ and $\mathscr{A}_3$. The resulting dialectical tree is shown Fig. 2 (right). Now, the marking procedure classifies the root as a *D-node* and thus *climbs(simba)* is no longer warranted.

4 Conclusions and future work

We have defined the DB_DeLP system as the first step to achieve an intelligent database system capable of massive database processing. The system shows several advantages. First, it allows a seamless connection with massive databases. Precompiled knowledge is integrated in the framework since its conception, and this speeds-up the inference process. This is a important step for solving the efficiency problem in argumentation systems. Besides, changes in the database can be easily accommodate since they don't change the dialectical graph. Therefore no up-keeping process in necessary for the dialectical graph, given that defeasible rules remain unchanged.

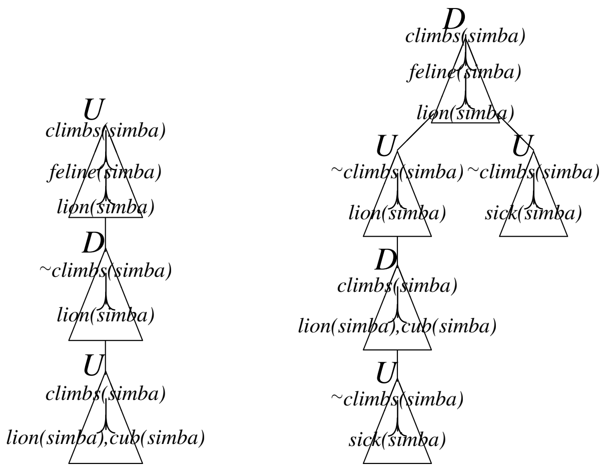

Figure 2: First and second dialectical trees from Example 3.1

Considering these facts we claim that DB_DeLP is an optimal argumentation system to be build an intelligent system architecture capable of reasoning with and manage massive data. To test this hypothesis empirically, we plan to implement the DB_DeLP and build a prototype that interact with a MySQL database.

Intelligent information systems are today's most critical tools for survival in the business environment [7]. Having an implementation of the DB_DeLP system can help us achieve a new set of practical application based on argumentation-driven information systems.

References

[1] Capobianco, M., Chesñevar, C., and Simari, G. Argumentation and the dynamics of warranted beliefs in changing environments. *Journal of Autonomous Agents and Multiagent Systems 11* (2005), 127–151.

[2] Chesñevar, C., and Maguitman, A. An argumentative approach to assesing natural language usage based on the web corpus. In *Proc. of European Conference on Artificial Intelligence (ECAI 2004). Valencia, Spain* (Aug. 2004), ECCAI.

[3] Chesñevar, C., and Maguitman, A. ArgueNet: An Argument-Based Recommender System for Solving Web Search Queries. In *Proc. of Intl. IEEE Conference on Intelligent Systems IS-2004. Varna, Bulgaria* (June 2004).

[4] Chesñevar, C. I., Maguitman, A. G., and Simari, G. R. Argument-based critics and recommenders: A qualitative perspective on user support systems. *Data & Knowledge Engineering 59*, 2 (2006), 293–319.

[5] García, A., and Simari, G. Defeasible Logic Programming: An Argumentative Approach. *Theory and Practice of Logic Programming 4*, 1 (2004), 95–138.

[6] Gomez, S., and Chesñevar, C. A Hybrid Approach to Pattern Classification Using Neural Networks and Defeasible Argumentation. In *Proc. of Intl. 17th FLAIRS Conference. Palm Beach, FL, USA* (May 2004), AAAI, pp. 393–398.

[7] Zaniolo, C. Intelligent Databases: Old Challenges and New Opportunities. *Journal of Intelligent Information Systems 1* (1992), 271–292.