5.3 On Complexity of DeLP through Game Semantics

On the Complexity of DeLP through Game Semantics*

Laura A. Cecchi
Depto. Ciencias de la Computación - Fa.E.A.
Universidad Nacional del Comahue
Buenos Aires 1400
(8300) Neuquén - ARGENTINA
lcecchi@uncoma.edu.ar

Pablo R. Fillottrani and **Guillermo R. Simari**
Depto. de Ciencias e Ingeniería de la Computación
Universidad Nacional del Sur
Av. Alem 1253
(8000) Bahía Blanca - ARGENTINA
{prf,grs}@cs.uns.edu.ar

Abstract

Defeasible Logic Programming (DeLP) is a general argumentation based system for knowledge representation and reasoning. Its proof theory is based on a dialectical analysis where arguments for and against a literal interact in order to determine whether this literal is believed by a reasoning agent. The semantics $\mathcal{GS}$ is a declarative trivalued game-based semantics for DeLP that is sound and complete for DeLP proof theory.

Complexity theory is an important tool for comparing different formalism and for helping to improve implementations whenever it is possible. In this work we address the problem of studying the complexity of some important decision problems in DeLP. Thus, we characterize the relevant decision problems in the context of DeLP and $\mathcal{GS}$, and we define data and combined complexity for DeLP. Since DeLP computes every argument from a set of defeasible rules, it is of central importance to analyze the complexity of two decision problems. The first one can be defined as "Is a set of defeasible rules an argument for a literal under a defeasible logic program?". We prove that this problem is **P**-complete. The second decision problem is "Does there exist an argument for a literal under a defeasible logic program?". We prove that this problem is in **NP**. Furthermore, we study data complexity of query answering in the context of DeLP. As far as we know, data complexity has not been introduced in the context of argumentation systems.

KEYWORDS: Argumentation Systems, Defeasible Reasoning, Logic Programming, Game-based Semantics, Complexity

Introduction

Defeasible Logic Programming (DeLP) is a general argumentation based tool for knowledge representation and reasoning (García & Simari 2004)[1]. Its proof theory is based on a dialectical analysis where arguments for and against a literal interact in order to determine whether this literal is believed by a reasoning agent. The semantics $\mathcal{GS}$ is a declarative trivalued game-based semantics for DeLP that links game-semantics (Abramsky & McCusker 1997) and model-theory. Soundness and completeness of $\mathcal{GS}$ with respect to DeLP proof theory have been proved (Cecchi & Simari 2004).

Complexity theory is an important tool for comparing different formalism, and for helping to improve implementations whenever it is possible. For this reason, it is important to analyze the computational complexity and the expressive power of DeLP. The former tells us how difficult it is to answer a query, while the latter gives a precise characterization of the concepts that are definable as queries.

Even thought complexity for nonmonotonic reasoning systems has been studied in depth for several formalisms such us default logic, autoepistemic logic, circumscription, abduction and logic programming (Cadoli & Schaerf 1993; Dantsin *et al.* 2001) until recently not many complexity results for argumentation systems have been reported.

This situation can be explained in part by the fact that, historically, implementations of argumentation systems have been limited to areas with no real time response restriction (see (Verheij 1998; Gordon & Karacapilidis 1997)). Recently, however, several applications have been developed, and implemented using argumentation systems related, for instance, with multiagent systems and web search (Atkinson, Bench-Capon, & Mc Burney 2004; Chesñevar & Maguitman 2004a; 2004b; Bassiliades, Antoniou, & Vlahavas 2004). Scalability and robustness of such approaches heavily depend on the computational properties of the underlying algorithms. It is hence crucial to study these properties in order to expand the application fields of argumentation systems.

Different computational complexity results (Dimopoulos, Nebel, & Toni 2002; Bench-Capon 2003; Amgoud & Cayrol 2002; Dunne & Bench-Capon 2002) have been presented on argumentation abstract framework (Bondarenko *et al.* 1997; Dung 1995), based on admissibility and preferability semantics. However, those results do not apply directly to DeLP, because its semantics are quite different. Another notable study of the computational complexity of defeasible systems

*This research was partially supported by the Secretaría General de Ciencia y Tecnología of the Universidad Nacional del Sur, by the Universidad Nacional del Comahue (Proyecto de Investigación 04/E062), by the Agencia Nacional de Promoción Científica y Tecnológica (PICT 2002 No. 13096, PICT 2003 15043, PAV 076) and by the National Research Council (CONICET), ARGENTINA.

[1]The interested reader can find an on-line interpreter for DeLP in `http://lidia.cs.uns.edu.ar/DeLP`

has been done in (Maher 2001). But, defeasible theory analyzed in this work greatly differs from DeLP in several points, such as knowledge representation (facts and strict rule, defeasible and defeaters rules) and their proof theories.

When measuring the complexity of evaluating queries in a specific language, we distinguish between several kinds of complexity according to (Vardi 1982; Papadimitriou & Yannakakis 1997; Dantsin *et al.* 2001). *Data complexity* is the complexity of evaluating a specific query in the language, when the query is fixed, and we study the complexity of applying this query to arbitrary databases; the complexity is thus given as a function of the size of the database. *Program or Expression complexity* appears when a specific database is fixed, and we study the complexity of applying queries represented by arbitrary expressions in the language; the complexity is given as a function of the length of the expression. *Combined complexity* considers both query and database instance as input variables.

In this work we are concerned with the study of complexity of some important decision problems of DeLP. The system and its asociated game semantics $\mathcal{GS}$ are analyzed introducing relevant decision problems in relation to the possible query answers.

Since DeLP builds the arguments from a defeasible logic program results of central importance to consider and evaluate two questions: "is a set of defeasible rules an argument for a literal under a defeasible logic program?" which has been proved to be **P**-complete, and does there exit an argument for a literal under a defeasible logic program? which has been proved to be in **NP**.

We define data, expression and combined complexity in the context of DeLP, in order to evaluate the efficiency of DeLP implementations. In particular, we study data complexity of query answering to assess DeLP applications over database technologies. As far as we know data complexity has not been introduced in the context of argumentation systems.

The paper is structured as follows. In the following section we briefly outline the fundamentals of DeLP, and describe the declarative game-based semantics $\mathcal{GS}$. Then, we discuss DeLP through $\mathcal{GS}$ semantics pointing out the decision problems that are of central importance, and we define data, expression and combined complexity in the context of DeLP. Afterwards, we give complexity results on the existence of an argument for a literal L under a defeasible logic program $\mathcal{P}$, and on the decision problem of whether a subset of defeasible rules is an argument for a literal L under $\mathcal{P}$. Next, we analyze data complexity for DeLP, and we present complexity results for two decision problems on entailment. In the last section, we summarize the main contributions of this work, and we present our conclusions and future research lines.

DeLP and Game Semantics $\mathcal{GS}$

We will start by introducing some of the basic concepts in DeLP (see (García & Simari 2004)). In the language of DeLP a literal L is a atom A or a negated atom $\sim A$, where $\sim$ represents the strong negation in the logic programming sense. The complement of a literal L, denoted as $\overline{L}$, is defined as follows: $\overline{L} = \sim A$, if L is an atom, otherwise if L is a negated atom, $\overline{L} = A$. Let X be a set of literals, $\overline{X}$ is the set of the complement of every member in X.

Definition 1 A *strict rule* is an ordered pair, denoted "$Head \leftarrow Body$", where "$Head$" is a ground literal, and "$Body$" is a finite set of ground literals. A strict rule with head L_0 and body $\{L_1, \ldots L_n, n > 0\}$ is written as $L_0 \leftarrow L_1, \ldots L_n$. If body is the empty set, then we write $L_0.$, and the rule is called a *Fact*. A *defeasible rule* is an ordered pair, denoted "$Head \prec Body$", where "$Head$" is a ground literal, and "$Body$" is a finite, non-empty set of ground literals. A defeasible rule with head L_0 and body $\{L_1, \ldots L_n, n > 0\}$ is written as $L_0 \prec L_1, \ldots L_n$.
A *defeasible logic program* $\mathcal{P}$, abbreviated *de.l.p.*, is a set of strict rules and defeasible rules. We will distinguish the subsets Π_F of facts, Π_R of strict rules, $\Pi = \Pi_F \cup \Pi_R$ and the subset Δ of defeasible rules.

We denote by Lit the set of all the ground literals that can be generated considering the underlying signature of a *de.l.p.* an we denote by Lit^+ the set of all the atoms in Lit.

Intuitively, whereas Π is a set of certain and exception-free knowledge, Δ is a set of defeasible knowledge, i.e., tentative information that could be used, whenever nothing is posed against it.

By definition a *de.l.p.* may be an infinite set of strict and defeasible rules but for complexity analysis we restrict ourselves to finite defeasible logic programs.

DeLP proof theory is based on developments in non monotonic argumentation systems (Pollock 1987; Simari & Loui 1992). An *argument for a literal* L is a minimal subset of Δ that together with Π consistently entails L. The notion of entailment corresponds to the usual SLD derivation used in logic programming, performed by backward chaining on both strict and defeasible rules, where negated atoms are treated as a new atom in the underlying signature. Thus, an agent can explain a literal L, throughout this argument.

In order to determine whether a literal L is supported from a *de.l.p.* a dialectical tree for L is built. An argument for L represents the root of the dialectical tree, and every other node in the tree is a defeater argument against its parent. At each level, for a given a node we must consider all the arguments against that node. Thus every node has a descendant for every defeater. A comparison criteria is needed for determining whether an argument defeats another. Even though there exist several preference relations considered in the literature, in this first approach we will abstract away from that issue.

We will say that a literal L is warranted if there is an argument for L, and in the dialectical tree each defeater of the root is itself defeated. Recursively, this leads to a marking procedure of the tree that begins by considering the fact that leaves of the dialectical tree are undefeated arguments as a consequence of having no defeaters. Finally, an agent will believe in a literal L, if L is a warranted literal.

There exist four possible answers for a query L: YES if L is warranted, NO if $\overline{L}$ is warranted (i.e., the complement of L is warranted), UNDECIDED if neither L nor $\overline{L}$ are warranted,

and UNKNOWN if L is not in the underlying signature of the program.

We have briefly given an intuitive introduction to the DeLP language and the dialectical procedure for obtaining a warranted conclusion. For complete details on DeLP see (García & Simari 2004).

Games have an analogy with a dispute and, therefore, that analogy extends to argument-based reasoning. A dispute can be seen as a game where in an alternating manner, the player P, the proponent, starts with an argument for a literal. The player O, the opponent, attacks the previous argument with a counterargument strong enough to defeat it. The dispute could continue with a counterargument of the proponent, and so on. When a player runs out of moves, i.e., that player can not find a counterargument for any of his adversary's arguments, the game is over. If the proponent's argument has not been defeated then she has won the game.

The semantics $\mathcal{GS}$ is a declarative trivalued game-based semantics for DeLP that links game-semantics (Abramsky & McCusker 1997) and model theory. Soundness and completeness of $\mathcal{GS}$ with respect to DeLP proof theory have been proved (Cecchi & Simari 2004). In the following we present some notions of $\mathcal{GS}$, for more details see (Cecchi & Simari 2000; 2004).

Let X be a set and $\{x_1, \ldots, x_n\} \subseteq X$, X^* is the set of finite sequences over X and $[x_1 \ldots x_n]$ denotes the sequence of the elements $x_1, \ldots, x_n$. We write $|s|$ for the length of a finite sequence and s_i for the ith element of s, $1 \leq i \leq |s|$. Concatenation of sequences is indicated by juxtaposition. If $t = su$ for some sequences $t,\ s,\ u$, then we say that s is a prefix of t. Let $Pref(S)$ be a set of prefix of S, then S is prefix closed if $S = Pref(S)$.

In order to use a game to capture the dialectical procedure, we need to define in a declarative way the movements of such game: the argument. The followings definitions are based on the notation introduced in (Lifschitz 1996).

Definition 2 Let X be a set of ground literals. The set X is *rigorously closed under a de.l.p.* $\mathcal{P}$, if for every strict rule $Head \leftarrow Body$ of $\mathcal{P}$, $Head \in X$ whenever $Body \subseteq X$, and for every defeasible rule $Head' \prec Body'$ of $\mathcal{P}$, $Head' \in X$ whenever $Body' \subseteq X$.
The set X is *consistent* if there is no literal L such that $\{L, \overline{L}\} \subseteq X$. Otherwise, we will say that X is *inconsistent*.
We say that X is *logically closed* if it is consistent or it is equal to Lit.

Intuitively, if the set of knowledge of an agent is rigorously closed under a *de.l.p.*, the agent will not believe in a literal that she cannot explain.

Definition 3 Let $\mathcal{P}$ be a *de.l.p.*. The *set of rigorous consequences* of $\mathcal{P}$, denoted $Cn_R(\mathcal{P})$, is the least set of literals w.r.t. inclusion, such that it is logically closed and rigorously closed under $\mathcal{P}$.

Even though rigorous consequences do not reflect the underlying ideas of strict and defeasible rules, they are very useful for introducing a declarative definition of argument.

Definition 4 Let $\mathcal{P} = \langle \Pi, \Delta \rangle$ be a *de.l.p.*. We say that $\langle \mathcal{A}, L \rangle$ is an argument structure for a ground literal L, if $\mathcal{A}$ is a set of defeasible rules of Δ, such that:

1. $L \in Cn_R(\Pi \cup \mathcal{A})$
2. $Cn_R(\Pi \cup \mathcal{A}) \neq Lit$
3. $\mathcal{A}$ is minimal w.r.t. inclusion, i.e., there is no $\mathcal{A}' \subseteq \mathcal{A}$ such that satisfies (1) and (2).

For convenience we will simply speak of argument instead of argument structure whenever this does not lead to misunderstandings. Let's introduce game concept and $\mathcal{GS}$ semantics.

Definition 5 Let $\mathcal{P} = (\Pi, \Delta)$ be a *de.l.p.*, L a literal and $\langle \mathcal{A}, L \rangle$ an argument structure for L. A *game* for $\langle \mathcal{A}, L \rangle$ with respect to $\mathcal{P}$, that we denote $G(\langle \mathcal{A}, L \rangle, \mathcal{P})$, is a structure

$$(M_{G(\langle \mathcal{A}, L \rangle, \mathcal{P})}, J_{G(\langle \mathcal{A}, L \rangle, \mathcal{P})}, P_{G(\langle \mathcal{A}, L \rangle, \mathcal{P})})$$

where

- $M_{G(\langle \mathcal{A}, L \rangle, \mathcal{P})}$ is a set of argument structure.
- $J_{G(\langle \mathcal{A}, L \rangle, \mathcal{P})} : M_{G(\langle \mathcal{A}, L \rangle, \mathcal{P})} \times I \rightarrow \{\mathsf{P}, \mathsf{O}\}$ where I is an enumerable index;
- $P_{G(\langle \mathcal{A}, L \rangle, \mathcal{P})} \subseteq M^*_{G(\langle \mathcal{A}, L \rangle, \mathcal{P})}$, where $P_{G(\langle \mathcal{A}, L \rangle, \mathcal{P})}$ is a non-empty, prefix-closed set.
  Each sequences s of $P_{G(\langle \mathcal{A}, L \rangle, \mathcal{P})}$ satisfy:
  1. $s = [\langle \mathcal{A}, L \rangle] s'$, s' possibly empty.
  2. For all $i,\ 1 < i \leq |s|$

$$\begin{array}{rcl} J_{G(\langle \mathcal{A}, L \rangle, \mathcal{P})}(s_1, 1) & = & \mathsf{P} \\ J_{G(\langle \mathcal{A}, L \rangle, \mathcal{P})}(s_i, i) & = & \overline{J_{G(\langle \mathcal{A}, L \rangle, \mathcal{P})}(s_{i-1}, i-1)} \end{array}$$

  $\overline{\mathsf{P}} = \mathsf{O}$ and $\overline{\mathsf{O}} = \mathsf{P}$.
  3. If $s \in P_{G(\langle \mathcal{A}, L \rangle, \mathcal{P})}$, then for each argument structure $\langle \mathcal{A}_2, L_2 \rangle$ that is a legal move for $s_{|s|}$, there exists a sequence $t \in P_{G(\langle \mathcal{A}, L \rangle, \mathcal{P})}$, such that $t = s[\langle \mathcal{A}_2, L_2 \rangle]$.
  4. No other sequence belongs to $P_{G(\langle \mathcal{A}, L \rangle, \mathcal{P})}$.

Movements in a game are the introduction of arguments. A legal move in the game over a sequence s is an argument $\mathcal{A}$ such that strictly defeats $s_{|s|}$ or defeats non strictly $s_{|s|}$ and $s_{|s|}$ strictly defeats $s_{|s|-1}$. Furthermore, such legal move $\mathcal{A}$ cannot be part of another argument in s, ie we cannot introduce more than once an argument neither for nor against the first move. Finally, this move must be consistent with every move made by the same player in the sequence s.

For every argument $\mathcal{A}$ for a literal L we can built a game whose first move is $\langle \mathcal{A}, L \rangle$. Thus, a family of games will be obtained considering all the arguments for L.

Definition 6 Let $\mathcal{P}$ be a *de.l.p.*, L a literal under the signature of $\mathcal{P}$, $\langle \mathcal{A}_1, L \rangle, \ldots, \langle \mathcal{A}_n, L \rangle$ all the argument structures of L under $\mathcal{P}$ and $G(\langle \mathcal{A}_1, L \rangle, \mathcal{P}), G(\langle \mathcal{A}_2, L \rangle, \mathcal{P}), \ldots, G(\langle \mathcal{A}_n, L \rangle, \mathcal{P})$ the corresponding games for the arguments of L.

$$\{G(\langle \mathcal{A}_1, L \rangle, \mathcal{P}), G(\langle \mathcal{A}_2, L \rangle, \mathcal{P}), \ldots, G(\langle \mathcal{A}_n, L \rangle, \mathcal{P})\}$$

is the *game family of* L and we denote it as $\mathcal{F}(L, \mathcal{P})$.

Definition 7 Let a be the first proponent movement in the game. A sequence s is complete if $s = [a]s_1$, with s_1 potentially empty, then there is no movement $b \in M_{G(\langle \mathcal{A}, L\rangle, \mathcal{P})}$ such that $[a]s_1[b] \in P_{G(\langle \mathcal{A}, L\rangle, \mathcal{P})}$. A sequence s is preferred if each opponent movement has a proponent answer. In other words, a sequence s is preferred if $|s|$ is odd.

Definition 8 A strategy over a game G is a set of sequences S, such that for all sequence $s \in S$, either:

- s is preferred; or
- there exists other sequence $s' \in S$, such that s' is preferred and s and s' has a prefix t, $|t| = n$, n is even and $s_{n+1} \neq s'_{n+1}$.

Definition 9 Let $\mathcal{P}$ be a *de.l.p.*, $L \in Lit$ and $G(\langle \mathcal{A}, L\rangle, \mathcal{P}) \in \mathcal{F}(L, \mathcal{P})$. We say that *P wins the game* $G(\langle \mathcal{A}, L\rangle, \mathcal{P})$ or that $G(\langle \mathcal{A}, L\rangle, \mathcal{P})$ *is won by P*, if the set of complete sequences of $P_{G(\langle \mathcal{A}, L\rangle, \mathcal{P})}$ is an strategy. Otherwise, we say that *O wins the game* or that $G(\langle \mathcal{A}, L\rangle, \mathcal{P})$ *is won by O*.

A player can win a game even though he does not win every complete sequence in such game. In (Prakken & Sartor 1997) the authors have developed an argument-based extended logic programming system which differs from DeLP in its winning rule: a player wins a dialogue tree if and only if he wins all the branches of the tree.

Definition 10 Let $\mathcal{P}$ be a *de.l.p.*. A *game-based interpretation for* $\mathcal{P}$, or G-Interpretation for $\mathcal{P}$ for short, is a tuple $\langle T, F\rangle$, such that T and F are subsets of atoms of the underlying signature of $\mathcal{P}$ and $T \cap F = \varnothing$.

In the previous definition T stands for true while F stands for false. The set of atoms UNDECIDED is defined as the set $U = Lit^+ - \{T \cup F\}$.

Each game can finish in two possible ways: won by the proponent P or won by the opponent O. There is no possibility for a draw. As the first move is made by the P, we are interested in those games won by this player.

Definition 11 Let $\mathcal{P}$ be a *de.l.p.*, h an atom of the underlying signature of $\mathcal{P}$, $\mathcal{F}(h, \mathcal{P})$ the game family for h and $\mathcal{F}(\overline{h}, \mathcal{P})$ the game family for $\overline{h}$ under a *de.l.p.* $\mathcal{P}$. A *game-based model for* $\mathcal{P}$, that we name G-Model of $\mathcal{P}$, is a G-interpretation $\langle T, F\rangle$ such that:

- If there exists a game $G(\langle \mathcal{A}, h\rangle$ in the family $\mathcal{F}(h, \mathcal{P})$ won by P, then h belongs to T.
- If there exists a game $G(\langle \mathcal{A}, \overline{h}\rangle, \mathcal{P})$ in the family $\mathcal{F}(\overline{h}, \mathcal{P})$ won by P, then h belongs to F.

Since we only consider literals under the signature of *de.l.p.*, the G-model definition does not contemplate the answer UNKNOWN. The minimal G-model defines a sound and complete semantics $\mathcal{GS}$ for DeLP (Cecchi & Simari 2004). We will say that $\mathcal{GS}$ entails a literal L from a *de.l.p.* $\mathcal{P}$, denoted by $\mathcal{P} \models_{\mathcal{GS}} L$, whenever $L \in T$ or $\overline{L} \in F$, being $\langle T, F\rangle$ the minimal G-model of $\mathcal{P}$.

The following theorem relates proof theory and game-based semantics, showing soundness and completeness.

Theorem 1 Let $\mathcal{P}$ be a *de.l.p.* and L a literal. L is warranted under $\mathcal{P}$ if and only if L belongs to the set T or $\overline{L}$ belongs to the set F of the minimal G-models $\langle T, F\rangle$ of $\mathcal{P}$ under $\mathcal{GS}$ semantics.

We have briefly presented the DeLP language, its proof theory and its declarative game-based semantics $\mathcal{GS}$. Now, we will be able to analyze the system and study some complexity properties.

Discussion on $\mathcal{GS}$ Complexity

DeLP is a defeasible reasoning system where every consequence of a *de.l.p.* is analyzed considering all the arguments for and against it. The trivalued game semantics $\mathcal{GS}$ characterizes such reasoning by two sets T and F, since $T \cup \overline{F}$ is the set of all warranted literals. Undecided literals are the remaining literals L for which there is no warrant for it nor for its complement. When considering DeLP in relation to game semantics, there are two relevant computational decision problems to analyze in the context of a *de.l.p.* $\mathcal{P}$:

- GAMESAT: Deciding whether there is a game for a literal α won by the proponent P in the context of a *de.l.p.* $\mathcal{P}$.
- NOWINGAME: Deciding whether there is no game for a literal α neither for the complement of α won by the proponent P in the context of a *de.l.p.* $\mathcal{P}$.

The former problem involves just finding a game that is won by the proponent. In order to capture the latter, it is necessary to find all the games for the literal and for its complement, and to establish that none of them is won by the proponent.

A positive GAMESAT answer for a given *de.l.p.* $\mathcal{P}$ and a literal L implies that $L \in T$, being $\langle T, F\rangle$ the minimal G-model, i.e., $\mathcal{P} \models_{\mathcal{GS}} L$. A positive GAMESAT answer for $\overline{L}$ means that $L \in F$ in the minimal G-model, i.e., $\mathcal{P} \models_{\mathcal{GS}} \overline{L}$.

The NOWINGAME decision problem for a *de.l.p.* $\mathcal{P}$ and a literal L is equivalent to determining if given the minimal G-model $\langle T, F\rangle$ of $\mathcal{P}$, $L \in Lit^+ - \{T \cup F\}$, i.e., $\mathcal{P} \not\models_{\mathcal{GS}} L$ and $\mathcal{P} \not\models_{\mathcal{GS}} \overline{L}$.

In this case, three interesting situation can be contemplated, and establish the followings decision problems:

- *Whether there is no game for a literal L, neither for its complement $\overline{L}$.* The game families for a literal L and for its complement $\overline{L}$, $\mathcal{F}(L, \mathcal{P})$ and $\mathcal{F}(\overline{L}, \mathcal{P})$ respectively, are empty. L has no argument neither for nor against it. Therefore, the agent has no information about such query.
- *Whether there is no game for a literal L, and the non empty set of all games in the family of its complement $\overline{L}$ are won by the opponent.* The game family for a literal L, $\mathcal{F}(L, \mathcal{P})$, is empty and only games won by the opponent are in the non empty family $\mathcal{F}(\overline{L}, \mathcal{P})$. L has no argument for and all the arguments for its complement are defeated. Therefore, the agent has no information for L, and he cannot defend its complement. In a similarly way, we can define the case where the agent cannot defend a literal L, and has no information about its complement.

- *Whether all games in the non empty families for L and for its complement $\overline{L}$ are won by the opponent.* $\mathcal{F}(L, \mathcal{P})$ and $\mathcal{F}(\overline{L}, \mathcal{P})$ are non empty set and all the argument are defeated. The agent cannot defend any argument neither for nor against the literal L.

In order to determine the computational complexity of the decision problems introduced above, we will study DeLP from two approaches: combined and data complexity. Combined complexity of a fragment of logic programming has been defined and used in (Dantsin *et al.* 2001):

Complexity of (some fragment of) logic programming: is the complexity of checking if for variable programs $\mathcal{P}$ and variable ground atoms A, $\mathcal{P} \models A$.

On the other hand, the notion of data complexity is borrowed from relational database theory (Vardi 1982). Databases are nowadays the main tool for storing and retrieving very large sets of data. Data complexity allows us to study DeLP as a query language measuring its complexity focus on the size of the databases, and using defeasible and strict rules for inference purpose. Data complexity is a key measure to determine the efficiency of argumentation system implementations based on database technologies.

For methodological and complexity issues, it is important to distinguish in a *de.l.p.* the input data from the inference rules. Thus, hereafter, we will denote $\mathcal{P} = \langle \Pi_F, \Pi_R \cup \Delta \rangle$, where Π_F is a finite set of ground facts, and $\Pi_R \cup \Delta$ is a finite set of ground strict and defeasible rules. Making an analogy with database concepts, Π_F represents the input databases, also called the *extensional part*, and $\Pi_R \cup \Delta$ are the inference rules, called the *intensional part* of the database. We define a Boolean query as a finite set of strict and defeasible rules together with a ground literal L. The intended intuitive meaning of defining such query is the following: we want to know whether a literal L is entailed by $\mathcal{GS}$ from $\Pi_R \cup \Delta$ together with the database Π_F.

Following the principle and notions above, in the context of DeLP we will define data, program and combined complexity as follows.

Definition 12 Let Ω be any of the decision problems introduced above, $\mathcal{P} = \langle \Pi_F, \Pi_R \cup \Delta \rangle$ and $(\Pi_R \cup \Delta, L)$ a query:

- The *data complexity* of Ω is the complexity of Ω when the query is fixed, and the database varies, i.e., parameters $\Pi_R \cup \Delta$ and L are fixed.
- The *program or expression* complexity of Ω is the complexity of Ω when the database instance is fixed, and the query varies, i.e., the parameter Π_F is fixed.
- The *combined complexity* of Ω is the complexity where every parameter Π_F, $\Pi_R \cup \Delta$ and L vary.

Expression and combined complexity are quite close and they are rarely differentiated. For this reason we will only discuss data and combined complexity.

In order to carry out this complexity analysis we will first focus on the complexity of determining whether there is an argument $\mathcal{A}$ for a literal L. Then we will study if the game played with initial move $\mathcal{A}$ is won by the proponent.

The complexity of computing arguments

Arguments and counterarguments are the movements in a game, and hence the core of DeLP. Dung's formalism (Dung 1995) and some extensions that have been developed (Bench-Capon 2002; 2003; Amgoud & Cayrol 2002), offer a powerful tool for the abstract analysis of defeasible reasoning. However, these approaches operate with arguments and their attack and defeat relation at an abstract level, avoiding to deal with the underlying logical language used to structure the arguments. On the other hand DeLP does construct the arguments and analyzes the defeater relationship. Thus, studying the decision problem: "is a given subset of defeasible rules an argument for a literal under a *de.l.p.*?" is of central importance.

Following the definition of argument this problem has three parts: is L a consequence of $\Pi \cup \mathcal{A}$?, is $\Pi \cup \mathcal{A}$ consistent?, and is there a subset $\mathcal{A}'$ of $\mathcal{A}$ such that it is consistent with Π and that together with Π derives L?

Let $\mathcal{P} = \langle \Pi_F, \Pi_R \cup \Delta \rangle$ be a *de.l.p.*, L be a literal and $\mathcal{A} \subseteq \Delta$. The first condition of definition 4, that involves rigorous consequences concept is $L \in Cn_R(\Pi \cup \mathcal{A})$. In (Cecchi & Simari 2000), we have defined the following transformation Φ from a *de.l.p.* into a propositional definite logic program, i.e., a propositional logic program with just Horn clauses. Let A be an atom. $\Phi(A) = A$, $\Phi(\sim A) = A'$ where A' is a new atom not in the signature of the *de.l.p.* and the transformation of a conjunction is $\Phi(A, B) = \Phi(A), \Phi(B)$. $\Phi(H \prec B) = \Phi(H) \leftarrow \Phi(B)$ and all other rules remain the same $\Phi(H \leftarrow B) = \Phi(H) \leftarrow \Phi(B)$. We will use this transformation, and the following lemma in order to reduce the rigorous consequences of a *de.l.p.* into consequences of propositional Horn clauses.

Lemma 1 Let DP be a definite logic program, and $\mathcal{M}$ be the minimal model of DP, then $\mathcal{M} = Cn_R(DP)$.

We are interested in computing the time complexity of verifying whether $L \in Cn_R(\Pi \cup \mathcal{A})$. We shall construct a logic program with just Horn clauses, denoted $\mathcal{HP}(\Pi, \mathcal{A}, L)$ such that $L \in Cn_R(\Pi \cup \mathcal{A})$ if and only if $\mathcal{HP}(\Pi, \mathcal{A}, L) \models yes$.

Suppose that $A_1, \ldots, A_n$ are all the atoms in $\Pi \cup \mathcal{A}$. We define $\mathcal{HP}(\Pi, \mathcal{A}, L)$ as follows:

$$\begin{aligned} \mathcal{HP}(\Pi, \mathcal{A}, L) \quad = \quad & \Phi(\Pi) \cup \Phi(\mathcal{A}) \cup \{yes \leftarrow \Phi(L)\} \cup \\ & \{yes \leftarrow \Phi(A_i), \Phi(\overline{A_i}) : 1 \leq i \leq n\} \end{aligned}$$

Even though the SAT decision problem is **NP**-complete, both checking whether a definite propositional logic program $\mathcal{DP}$ satisfies a ground atom A, i.e., $\mathcal{DP} \models A$, and HORNSAT, i.e., the decision problem whether there is a truth assignment that satisfies a collection of Horn clauses, are **P**-complete (Dantsin *et al.* 2001; Papadimitriou & Yannakakis 1997).

Lemma 2 $\mathcal{HP}(\Pi, \mathcal{A}, L)$ is a transformation from a *de.l.p.* P into propositional Horn clauses such that verifying whether a literal L belongs to $Cn_R(\mathcal{P})$ is equivalent to verifying

Algorithm: Minimal
Input: $\mathcal{A}$ an argument for a literal L, and Π a set of strict rules.
Output: true if A is a minimal argument for L, false otherwise

minimal=true
Aux= $\mathcal{A}$
While minimal and not $Aux = \varnothing$ do
select $H \prec B \in$ Aux
$A' = \mathcal{A} - \{H \prec B\}$
if $L \in Cn_R(\Pi \cup A')$
then minimal=false
else Aux= Aux - $\{H \prec B\}$

Figure 1: Algorithm for verifying if a set of defeasible rules is minimal with respect to set inclusion for deriving a literal L.

whether yes is entailed from the transformed propositional Horn program. Thus, $L \in Cn_R(\mathcal{P})$ reduces to $\mathcal{DP} \models yes$, being $\mathcal{DP}$ a propositional Horn program.

Proof: In order to prove our claim, we have to establish that:

1. $L \in Cn_R(\Pi \cup \mathcal{A})$ if and only if $\mathcal{HP}(\Pi, \mathcal{A}, L) \models yes$.

   We will consider two cases:
   - $\Pi \cup \mathcal{A}$ is consistent.
     $L \in Cn_R(\Pi \cup \mathcal{A})$ if and only if $\Phi(L) \in \Phi(Cn_R(\Pi \cup \mathcal{A}))$ if and only if $\Phi(L) \in Cn_R(\Phi(\Pi \cup \mathcal{A}))$(see (Cecchi & Simari 2000)) if and only if, by lemma 1, $\Phi(L)$ is in the minimal model of $\Phi(\Pi \cup \mathcal{A})$ if and only if $\mathcal{HP}(\Pi, \mathcal{A}, L) \models yes$ by the definition of minimal model, the monotonicity property and the use of the rule $yes \leftarrow \Phi(L)$.
   - $\Pi \cup \mathcal{A}$ is inconsistent.
     $L \in Cn_R(\Pi \cup \mathcal{A}) = Lit$ if and only if there exists $i, 1 \leq i \leq n$, such that $\Phi(L_i)$ and $\Phi(\overline{L_i})$ are in $\Phi(Cn_R(\Pi \cup \mathcal{A}))$ if and only if $\Phi(L_i)$ and $\Phi(\overline{L_i})$ are in the minimal model of $\mathcal{HP}(\Pi, \mathcal{A}, L)$ if and only if $\mathcal{HP}(\Pi, \mathcal{A}, L) \models yes$ by definition of minimal models, monotonicity property and the use of the rule $yes \leftarrow \Phi(L_i), \Phi(\overline{L_i})$.
2. $\mathcal{HP}$ is computed in logarithmic space: the transformation is quite simple, and is feasible in logarithmic space, since rules can be generated independently of each other except those of the form $yes \leftarrow \Phi(L_i), \Phi(\overline{L_i})$ which depends on the literal in the input.

Therefore $\mathcal{HP}(\Pi, \mathcal{A}, L)$ is a reduction from $L \in Cn_R(\Pi \cup \mathcal{A})$ into propositional Horn clauses. ■

Theorem 2 Let $\mathcal{P} = (\Pi_F, \Pi_R \cup \Delta)$ a *de.l.p.*, $\mathcal{A} \subseteq \Delta$, and L a literal. Determining whether $L \in Cn_R(\Pi \cup A)$ is **P**-complete.

Proof:
- *Membership:* Given a definite logic program $\mathcal{P}$ the least fixpoint T_P^∞ of the operator T_P can be computed in polynomial time (Papadimitriou 1994; Dantsin *et al.* 2001) : the number of iterations is bounded by the number of rules plus one. Each iteration step is feasible in polynomial time. Thus finding the minimal model of a logic program with just Horn clauses is in **P** (Dantsin *et al.* 2001).
  By lemma 2, $L \in Cn_R(\Pi \cup A)$ has been reduced to propositional logic programming. Therefore, $L \in Cn_R(\Pi \cup A)$ is in **P**.
- *Hardness:* Horn rules are strict rules in a *de.l.p.*, and the minimal model of a definite logic program $\mathcal{DP}$ is equal to $Cn_R(\mathcal{DP})$. Therefore, by applying reduction by generalization, we have that $\mathcal{DP} \models L$ reduce to $L \in Cn_R(\mathcal{DP})$. Propositional logic programming is **P**-complete (Dantsin *et al.* 2001). This suffices to complete the proof. ■

Until now we have proved that the first condition of argumentation definition is **P**-complete. Now we will analyze the rest of the issues we need for computing an argument. We will denote the cardinality of the language by $|Lit|$ and defeasible rules cardinality by $|\Delta|$.

In Figure 1, we present an algorithm for verifying whether a set of defeasible rules is minimal with respect to set inclusion for entail a literal L. Worst case of the minimality condition is considered assuming that the argument has at most $|\Delta|$ defeasible rules, i.e., Δ is an argument for some literal. Computing the minimality condition involves $|\Delta|$ loops verifying that $L \in Cn_R(\Pi \cup A')$, which is in **P**. Thus, this problem is solvable in polynomial time, and, therefore, it is in **P**.

Finally, to check whether the set of defeasible rules is consistent under a *de.l.p.*, we verify that there is no atom such that the atom and its complement are members of $Cn_R(\Pi \cup A)$. In the worst case, when $Cn_R(\Pi \cup A)$ is consistent, this algorithm must control every atom in the signature of the *de.l.p.*. Thus, to check if it is consistent is proportional to the number of atoms $|Lit|/2$ and therefore it is in **P**.

Theorem 3 The decision problem "is a given subset of defeasible rules an argument for a literal under a *de.l.p.*?" is **P**-complete.

Proof:
Membership: (sketch) From the above development it follows membership to **P**.
Hardness: We employ a reduction from $\mathcal{DP} \models L$, being $\mathcal{DP}$ a propositional Horn program. Consider the following transformation $r(\mathcal{DP}) = \mathcal{DP}' = \langle \Pi, \Delta \rangle$, where $\Pi = \mathcal{DP}$ and Δ is empty. r is a transformation computed in logarithmic space such that whenever a literal L is entailed by a propositional Horn program $\mathcal{DP}$, the decision problem "is a given subset of $\Delta = \varnothing$ an argument for L under $\mathcal{DP}'$" finish in an accepting state.

L is in the minimal model of a propositional Horn program if and only if $L \in Cn_R(\mathcal{DP})$ if and only if $\varnothing$ is an argument for L, since is minimal and consistent with Π, if and only if "is a given subset of $\Delta = \varnothing$ an argument for

L under $\mathcal{DP}'$" finish in an accepting state. Thereby establishing that the decision problem "is a given subset of defeasible rules an argument for a literal under a *de.l.p.*?" is **P**-complete. ■

Our final aim is to determine the complexity of computing the set of all the arguments under a *de.l.p.*. This is motivated in that GAMESAT and NOWINGAME require for playing a game to compute every argument that defeats each argument introduced in a previous move. A subset $A \subseteq \Delta$ may be a potential argument of different literals in the language. Thus, the maximum number of checks for potential arguments that depends on the size of the set of defeasible rules and on the size of Lit, is $|Lit| * 2^{|\Delta|}$.

Lemma 3 Let AP be the polynomial time needed for the decision problem "is a given subset of defeasible rules an argument for a literal l under a *de.l.p.*?". Then, the upper bound time for computing all the arguments is $|Lit| * 2^{|\Delta|} * AP$.

The result above states an exponential upper bound for computing $\mathcal{X}$, the set of all the arguments in Dung's formalism (Dung 1995).

Even though we must verify whether every subset of Δ is an argument for every literal in the language of the *de.l.p.*, because the consistency condition in the definition of argument, $A \subseteq \Delta$ cannot be an argument for a literal and for its complement, so we will consider only $\frac{|Lit|}{2} * 2^{|\Delta|} = |Lit| * 2^{|\Delta|-1}$ potential arguments in order to play a game or equivalently to build the dialectical tree. This upper bound could be improved by considering minimality over the arguments, i.e., no $A_1 \subseteq \Delta$ would be an argument for a literal L if A_2 is an argument of L and $A_2 \subseteq A_1$.

Finally, we consider the argument existence decision problem.

Corollary 1 *(Argument Existence)* The decision problem "whether there is an argument for a literal L under a *de.l.p.*" is **NP**.

Proof: We can guess any subset of Δ, and verify whether this subset is an argument for a literal L under *de.l.p.* in polynomial time. This proves membership in **NP**. ■

These results contrast with those of (Parsons, Wooldridge, & Amgoud 2003), where determining whether there is an argument for a formula h is Σ_2^P-complete. Even thought there are some similarities between argument definitions, they differ in the underlying logic. While in DeLP approach an argument is a subset of defeasible rules, and the inference mechanism to obtain it is logic programming based, an argument in the formalism described in (Parsons, Wooldridge, & Amgoud 2003) is a subset of formulas of a propositional language, and $\vdash$ stands for classical inference.

Data Complexity for DeLP

In order to determine the upper bound for the data complexity of the decision problems GAMESAT and NOWINGAME, we will first analyze the dialectical tree structure over the size of the facts and the strict and defeasible rules.

The dialectical tree is explored in a complete depth first way, as minimax does. If the maximum depth of the tree is m, and there are b legal movements at each point, then the time complexity will be $O(b^m)$(Russell & Norvig 2003). If we implement the technique alpha-beta pruning, and we consider that successors are examined in random order, then the time complexity will be roughly $O(b^{\frac{3m}{4}})$(Russell & Norvig 2003). The maximum depth of a dialectical tree for an argument under a *de.l.p.* with $|\Delta|$ defeasible rules is $2^{|\Delta|}$, i.e., we can consider every potential argument in one branch of the tree. Any argument can appear more than once in the tree but at most once in every branch, because of the acceptable argumentation line definition. What about branch factor: there exists $|Lit|/2$ literals that can be in conflict with the last argument. These literals may have at most $2^{|\Delta|}$ potential arguments. So our branching factor is in the worst case $|Lit|/2 * 2^{|\Delta|}$. Thus, exploring the dialectical tree as minimax does has an upper bound of $O((|Lit|*2^{|\Delta|-1})^{2^{|\Delta|}})$.

Every time we must insert a neighbour node B of a node A in the tree structure or equivalently, when a player makes a move, we must check if it is a legal move in the game, i.e., if B attacks and defeats A, and if B does not introduce inconsistency. In order to determine whether B is a defeater of A, we must take into account the preference criterion between arguments. Any preference criterion defined among arguments could be used in DeLP. For this reason, the complexity class of the following decision problem "whether an argument can be considered in the tree structure of a game" will be left parameterized in the class C.

Theorem 4 Let C be the complexity class for the decision problem:"whether an argument can be considered in the tree structure of a game". The upper bound for data complexity of GAMESAT is $\mathbf{NP}^C$.

Proof: For fixed $\Pi_R \cup \Delta$, the size of the dialectical tree for an argument $\langle \mathcal{A}, L \rangle$ is polynomial in the size of the literals in Π_F. Furthermore, computing each argument is in **P**, and considering each argument in the tree structure is in C. In order to decide whether a literal L belongs to the set T of the minimal G-model, we guess for an argument of L such that the game played from this argument is won by the Proponent. The number of arguments is polynomial when $\Pi_R \cup \Delta$ is fixed, and determining whether the game is won by the Proponent can be done with a C oracle. This proves membership in $\mathbf{NP}^C$. ■

Since NOWINGAME is a conjunction of GAMESAT complements an immediate corollary to the result above follows naturally.

Corollary 2 Let C be the complexity class for the decision problem:"whether an argument can be considered in the tree structure of a game". The upper bound for data complexity of NOWINGAME is co-$\mathbf{NP}^C$.

Decision Problem	Complexity
Is $L \in Cn_R(Rules)$?	**P**-complete
Is $\langle \mathcal{A}, L \rangle$ an argument?	**P**-complete
Argument Existence	**NP**
GAMESAT	Data Complexity $\mathbf{NP}^C$
NOWINGAME	Data Complexity $co - \mathbf{NP}^C$

Table 1: Problems studied, and the main complexity results obtained.

Even though we have not analyzed in depth the complexity for computing the preference criterion, we illustrate this concept with two different cases.

In (Chesñevar & Maguitman 2004a; 2004b), the authors use specificity (Simari & Loui 1992) as a syntax-based criterion among conflicting arguments, preferring those arguments which are more informed or more direct, in order to assess natural language usage based on the web corpus and to evaluate and rank search results, respectively. Computing specificity depends strongly on the set $2^{|Lit|}$.

Other DeLP implementations use a static preference relation (Chesñevar *et al.* 2004). In this case, the preference criterion is computed by comparing arguments values. Such values are obtained through different mathematical formulas applied to the certainty of a formula in the language. Computing such preference criterion involves just a comparison between two certainty values. However, an extra cost is considered in the argument construction procedure, since the certainty value is computed keeping a trace of all uncertain information used to derive a goal.

Conclusion and Future Work

We have analyzed complexity of DeLP through the $\mathcal{GS}$ semantics, pointing out some relevant decision problems. In particular, we have analyzed in depth GAMESAT and NOWINGAME. In order to achieve our aim, we have distinguished database and a query from a *de.l.p.*, and we have defined data, expression and combined complexity in the context of DeLP. As far as we know, argumentation systems have not been studied yet as a query language, and, therefore, there is no previous data complexity analysis for defeasible reasoning. Table 1 summarizes the problems studied and the main complexity results obtained.

As DeLP do not assume as input the argument set, the first results that has been established where related to arguments, the movements of a game. We have focused on the existence of an argument in order to play a game, and on verifying whether a set is an argument. We state an exponential upper bound for the set of all the arguments. Because of the underpinning logic of DeLP our complexity results are a bit better than those based on classical logic.

Data complexity results on GAMESAT and NOWINGAME give a guideline for determining expressive power for DeLP. Since our results are parameterized, we can state a lower bound on **NP**, otherwise known as Σ_1^1, which coincides with the class of properties of finite structures expressible in existential second-order logic (Fagin 1974).

When analyzing Data complexity we have fixed the query and we have parametrized the preference criteria. Thus an interesting topic for future research is to study to what extent this results can be applied to others rule-based argumentation systems whose theory proof is rather similar.

As future work we will analyze combined complexity of the decision problems introduced. We are studying the expressive power of DeLP in order to compare this system with other non monotonic formalisms.

References

Abramsky, S., and McCusker, G. 1997. Game Semantics. In Schwichtenberg, H., and Berger, U., eds., *Logic and Computation: Proceedings of the 1997 Marktoberdorf Summer School*. Springer-Verlag.

Amgoud, L., and Cayrol, C. 2002. A reasoning model based on the production of acceptable arguments. *Annals of Math and Artificial Intelligence* 34:197–215.

Atkinson, K.; Bench-Capon, T.; and Mc Burney, P. 2004. A dialogue game protocol for multi-agent argument over proposals for action. Technical Report ULCS-04-007, Department of Computer Science, University of Liverpool, Liverpool, U.K.

Bassiliades, N.; Antoniou, G.; and Vlahavas, I. 2004. A defeasible logic reasoner for the semantic web. In *Proc. of the Workshop on Rules and Rule Markup Languages for the Semantic Web*, 49–64.

Bench-Capon, T. J. M. 2002. Value-based argumentation frameworks. In *NMR 2002*, 443–454.

Bench-Capon, T. J. M. 2003. Persuasion in Practical Argument Using Value Based Argumentation Frameworks. *Journal of Logic and Computation* 13(3):429–448.

Bondarenko, A.; Dung, P.; Kowalski, R.; and Toni, F. 1997. An Abstract, Argumentation-Theoretic Approach to Default Reasoning. *Artificial Intelligence* 93(1-2):63–101.

Cadoli, M., and Schaerf, M. 1993. A survey of complexity results for nonmonotonic logics. *Journal of Logic Programming* 17:127–160.

Cecchi, L. A., and Simari, G. R. 2000. Sobre la Relación entre la Definición Declarativa y Procedural de Argumento. In *VI CACiC*, 465–476.

Cecchi, L. A., and Simari, G. R. 2004. Sobre la relación entre la Semántica GS y el Razonamiento Rebatible. In *X CACiC - Universidad Nacional de La Matanza*, 1883–1894.

Chesñevar, C., and Maguitman, A. 2004a. An Argumentative Approach to Assessing Natural Language Usage based on the Web Corpus. In *Proc. of the European Conference on Artificial Intelligence (ECAI) 2004*, 581–585.

Chesñevar, C., and Maguitman, A. 2004b. ARGUENET: An Argument-Based Recommender System for Solving Web Search Queries. In *Proc. of the 2nd IEEE Intl. IS-2004 Conference*, 282–287.

Chesñevar, C.; Simari, G.; Alsinet, T.; and Godo, L. 2004. A Logic Programming Framework for Possibilistic Argumentation with Vague Knowledge. In *Proc. of the UAI-2004*, 76–84.

Dantsin, E.; Eiter, T.; Gottlob, G.; and Voronkov, A. 2001. Complexity and expressive power of logic programming. *ACM Computing Surveys (CSUR)* 33(3):374 – 425.

Dimopoulos, Y.; Nebel, B.; and Toni, F. 2002. On the Computational Complexity of Assumption-based Argumentation for Default Reasoning. *Artificial Intelligence* 141(1):57–78.

Dung, P. M. 1995. On the acceptability of arguments and its fundamental role in nonmonotonic reasoning and logic programming and n-person games. *Artificial Intelligence* 77:321–357.

Dunne, P. E., and Bench-Capon, T. 2002. Coherence in finite argument systems. *Artificial Intelligence* 141:187–203.

Fagin, R. 1974. Generalized first-order spectra and polynomial-time recognizable sets. In Karp, R., ed., *Complexity of Computation. SIAM-AMS Proceedings*, volume 7, 43–73.

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

Gordon, T., and Karacapilidis, N. 1997. The Zeno Argumentation Framework. In ACM., ed., *The Sixth International Conference on Artificial Intelligence and Law*, 10–18.

Lifschitz, V. 1996. Foundations of logic programming. In Brewka, G., ed., *Principles of Knowledge Representation*. CSLI Publications. 1–57.

Maher, M. J. 2001. Propositional defeasible logic has linear complexity. *Theory and Practice of Logic Programming* 1(6):691–711.

Papadimitriou, C. H., and Yannakakis, M. 1997. On the complexity of database queries (extended abstract). In *PODS '97: Proceedings of the sixteenth ACM SIGACT-SIGMOD-SIGART symposium on Principles of database systems*, 12–19. New York, NY, USA: ACM Press.

Papadimitriou, C. 1994. *Computational Complexity*. Addison-Wesley Publishing Company.

Parsons, S.; Wooldridge, M.; and Amgoud, L. 2003. Properties and complexity of some formal inter-agent dialogue. *Journal of Logic and Computation* 13(3):347–376.

Pollock, J. 1987. Defeasible Reasoning. *Cognitive Science* 11:481–518.

Prakken, H., and Sartor, G. 1997. Argument-based extended logic programming with defeasible priorties. *Journal of Applied Non-Classical Logics* 7:25–75.

Russell, S., and Norvig, P. 2003. *Artificial Intelligence: A modern approach*. New Jersey: Prentice Hall, second edition.

Simari, G. R., and Loui, R. P. 1992. A mathematical treatment of defeasible reasoning and its implementation. *Artificial Intelligence* 53:125–157.

Vardi, M. Y. 1982. The complexity of relational query languages. In *Proceedings of the Fourteenth Annual ACM Symposium on Theory of Computing, STOC82*, 137–146. New York, NY, USA: ACM Press.

Verheij, B. 1998. Argumed - a template-based argument mediation system for laweyers. In Hage, J.; Bench-Capon, T. J.; Koers, A.; de Vey Mestdagh, C.; and Grütters, C., eds., *Legal Knowledge Based Systems. JURIX: The Eleventh Conference*, 113–130.