Game-based approach for modeling dialectical analysis

Preliminary Report

Laura A. Cecchi*

Departamento de Informática y Estadística
Facultad de Economía y Administración
Universidad Nacional del Comahue
e-mail:lcecchi@uncoma.edu.ar

Guillermo R. Simari

Grupo de Investigación en Inteligencia Artificial
Departamento de Ciencias de la Computación
Universidad Nacional del Sur
e-mail: grs@criba.edu.ar

Keywords: Defeasible Logic Programming - Game Semantics - Logic Programming

Abstract

The operational semantics of Defeasible Logic Programming (justification process) is based on a dialectical analysis of arguments and counterarguments.

In [Abramsky, 1997] and [Abramsky(b), 1997], a game semantics is introduced in order to model a computation as a game between two players: the System and the Environment. The main idea is to use a game to model the *interaction* between the participants.

The justification process can be seen as a game where a player proposes an argument for a goal q and tries to defend it while the other player tries to find counterarguments that defeat it. Therefore we can model a dialectical system through the interaction between two players: proponent and opponent.

The purpose of this paper is to introduce a model based on a game structure for the operational semantics of Defeasible Logic Programming. The declarative semantics introduced, models the justification process when there exists a preference relation between contradictory arguments such that given two contradictory arguments it always determines which one is better.

*Supported by a fellowship of the Universidad Nacional del Comahue.

1 Introduction

Logic programs are nowadays widely recognized as a valuable tool for knowledge representation and commonsense reasoning. In order to increase the applicability of logic programming in many fields, several extensions of the class of definite programs have been proposed. Normal programs [Lloyd, 1987] add negation as failure in program clause bodies, basic programs [Gelfond, 1990] allow strong negation not only in the clause bodies but in the clause heads as well and disjunctive programs [Minker, 1982] allow disjunction as heads of program clauses.

Defeasible Logic Programming[García, 1998][García, 1999][Dung, 1995], that allows both notions of negation: default negation and strong negation, adds to basic normal programs a new set of rules, the *defeasible rules*. Defeasible rules are added in order to represent tentative information. Even though the tentative information represented may be contradictory, Defeasible Logic Programming (henceforth DLP) provides a criterion for deciding between contradictory goals.

DLP operational semantics is based on a dialectical analysis of arguments and counterarguments. Thus, a query q will be successful if there exists an argument $\mathcal{A}$ of q that is a justification,i.e., there is no counterargument that defeats $\mathcal{A}$. Since defeaters are arguments there may exit defeaters for the defeaters and so on.

The purpose of this paper is to introduce a model based on a game structure for the operational semantics of a restricted DLP. In [Abramsky, 1997] and [Abramsky(b), 1997], a game semantics is introduced in order to model a computation as a game between two players: the System and the Environment. The main idea is to use a game to model the *interaction* between the participants.

The justification process can be seen as a game where a player proposes an argument for a goal q and tries to defend it while the other player tries to find counterarguments that defeat it. Therefore we can model a dialectical system through the interaction between two players: proponent and opponent.

The outline of this paper is as follows. In section 2 we briefly review some background about DLP. Section 3 describes and illustrates game semantics showing how the interaction between the system and the environment is defined. Section 4 introduces a formal definition of a game-based semantics for the operational semantics of DLP. Finally in section 5, we give our conclusions and mention some possible directions for future research.

2 Defeasible Logic Programming

DLP allows to represent not only certain knowledge but tentative knowledge as well. In order to distinguish both kinds of knowledge, the logic programming language has been extended with a set of defeasible rules. A defeasible rule $Head \prec\!\!- Body$ is understood as expressing *reasons to believe in the antecedent Body provide reasons to believe in the consequence Head.*

Although both notions of negation, default and strong negation, can be represented in DLP, this work will be circumscribed to a language without default negation. Let's define the language.

Definition 2.1. Strong and Defeasible Rules[García, 1999]
A *Strong Rule* is an ordered pair $Head \leftarrow Body$, whose first member $Head$ is a literal[1], and whose second member $Body$ is a finite set of literals. A strong rule with the head L_0 and body $\{L_1, \ldots, L_n\}$ can also be written as: $L_0 \leftarrow L_1, \ldots, L_n$. As usual, if the body is empty, then a strong rule becomes $L \leftarrow true$ (or simply L) and it is called a *fact.*
A *Defeasible Rule* is an ordered pair $Head \prec\!\!- Body$, whose first member $Head$ is a literal,

[1] A literal L is an atom A or a negated atom $\sim A$.

and whose second member *Body* is a finite set of literals. A defeasible rule with head L_0 and body $\{L_1, \ldots, L_n\}$ can also be written as: $L_0 \prec\!\!- L_1, \ldots, L_n$. If the body is empty, we write $L_0 \prec\!\!- true$ and we call it a *presumption*. ■

A *defeasible logic program* [García, 1999] is a finite set of strong and defeasible rules. If $\mathcal{P}$ is a defeasible logic program, we will distinguish the subset $\mathcal{K}$ of strong rules in $\mathcal{P}$, and the subset Δ of defeasible rules in $\mathcal{P}$. When required we will denote $\mathcal{P}$ as $(\mathcal{K}, \Delta)$. A *defeasible query* is a defeasible rule with empty consequent denoted $\prec\!\!- Q_1, \ldots, Q_n$, where each Q_i $(1 \leq i \leq n)$ is a literal.

Example 2.1. We would like to represent the following sentences. Generally an elephant is not a gray elephant. Real elephants generally are gray but real elephants whose mother is gray, usually tend not to be gray . Every real elephant is an elephant. Clyde is an elephant and Trici is its mother who is gray. The following program encodes the above intended meaning.

$$\mathcal{K} = \{e(X) \leftarrow r(X);\ \ r(clyde);\ \ m(trici, clyde);\ \ gray(trici)\}$$

$$\Delta = \{\sim g(X) \prec\!\!- e(X);\ \ g(X) \prec\!\!- r(X);\ \ \sim g(X) \prec\!\!- r(X), m(Y, X), gray(Y)\}$$

When considering the query $\prec\!\!- g(clyde)$, we find a conflict between the rules of Δ. ■

Briefly[2],the operational semantics can be described as follows: when answering a query q, we must build an *argument* that supports the query. An argument $\mathcal{A}$ for q [García, 1999], that we will denote $\langle \mathcal{A}, q \rangle$, is a subset of ground instances of defeasible rules of $\mathcal{P}$, such that: (1) There exists a defeasible derivation for q from $\mathcal{K} \cup \mathcal{A}$, (2) $\mathcal{K} \cup \mathcal{A}$ is non-contradictory, and (3) $\mathcal{A}$ is minimal with respect to set inclusion. Once an argument for q has been built, we must find every argument $\langle \mathcal{A}', h' \rangle$ such that there exists a subargument[3] $\langle \mathcal{B}, h \rangle$ of $\langle \mathcal{A}, q \rangle$ and $\mathcal{K} \cup \{h', h\}$ is contradictory. Those arguments are called rebuttals or counterarguments. Counterarguments for the rebuttals found are considered and so on. The complete dialectical analysis can be described through a dialectical tree. Informally, the dialectical tree's root is an argument for the query and all the counterarguments of the parent node are considered in every level of the tree. A path from the root to a leaf in a dialectical tree is called "an argumentation line".

A query q will succeed if the supported argument for it is not defeated. Such an argument will be called a justification for q.

Since contradictory information can be represented in DLP, a way to decide between arguments should be found. Let $\langle A_1, q_1 \rangle$ and $\langle A_2, q_2 \rangle$ be two arguments. Several ways exists in order to determine which argument is better. For instance, specification [Simari, 1992], and fixed priorities [Prakken, 1996]. Some approaches allow incomparable elements, so they consider two cases:

- $\langle A_1, q_1 \rangle$ is better than $\langle A_2, q_2 \rangle$, then $\langle A_1, q_1 \rangle$ is a *proper defeater* of $\langle A_2, q_2 \rangle$.
- $\langle A_1, q_1 \rangle$ and $\langle A_2, q_2 \rangle$ are unrelated with respect to a preference order, then $\langle A_1, q_1 \rangle$ is a *blocking defeater* of $\langle A_2, q_2 \rangle$.

Example 2.2. Let's consider again example 2.1. Applying the preference approach based on specification [García, 1998], we can build an argument for the query $\prec\!\!- g(clyde)$:

$$\mathcal{A}_1 = \langle \{g(clyde) \prec\!\!- e(clyde)\}, g(clyde) \rangle.$$

[2]The reader interested in a more detailed discuss is advised to read [García, 1999].

[3]An argument $\langle \mathcal{B}, h \rangle$ is a subargument of $\langle \mathcal{A}, q \rangle$ if $\mathcal{B} \subseteq \mathcal{A}$.

Nevertheless, there exists a counterargument

$$\mathcal{A}_2 = \langle \{\sim g(clyde) \ \multimap \ r(clyde)\}, \sim g(clyde) \rangle$$

that is a proper defeater. But we can find a defeater to the above rebuttal:

$$\mathcal{A}_3 = \langle \{g(clyde) \multimap r(clyde), m(clyde, trici), gray(trici)\}, g(clyde) \rangle$$

Since this argument has no counterargument then $\mathcal{A}_1$ is a justification. ■

Our work deals with certain kind of preference relations over the set of argument. Henceforth, the way we decide between an argument and one of its counterarguments is defined through a preference relation $\mathcal{R}$, such that:

- For every pair of contradictory arguments $\mathcal{A}, \mathcal{B}$:

  $$\mathcal{R}(\mathcal{A}, \mathcal{B}) \ \oplus^4 \mathcal{R}(\mathcal{B}, \mathcal{A})$$

  Thus our discuss has been restricted to proper defeaters.

- If $\mathcal{B}$ is a subargument of $\mathcal{A}$ and there exists an argument $\mathcal{C}$ such that $\mathcal{R}(\mathcal{C}, \mathcal{B})$ then it will be the case that $\mathcal{R}(\mathcal{C}, \mathcal{A})$. This condition avoids reciprocal defeaters and therefore cycles.

Intuitively, given an argument $\mathcal{A}$ and one of its counterarguments $\mathcal{B}$, $\mathcal{R}(\mathcal{A}, \mathcal{B})$ means that A is better than B and therefore $\mathcal{A}$ defeats $\mathcal{B}$. In section 4 we formalize the operational semantics defined through the dialectical analysis described above in a game structure.

3 Game semantics

In [Abramsky, 1997] and [Abramsky(b), 1997], a game semantics is introduced in order to model computation as a game between two participants. One of the players in the game represents the *System* and is referred to as *Proponent* (P); the other represents the *Environment* and is referred to as *Opponent* (O). Thus, a single "computation" or "run" involving interaction between Proponent and Opponent is represented by a sequence of *moves*, made alternately by P and O.

Before turning to a formal definition of game semantics, we should fix the notation we will use. Let s and t be sequences, a an element and X a set, then

- st denotes the concatenation of sequences s and t.
- sa denotes the sequence obtained by adding the element a to the sequence s in the last position.
- we write $|s|$ for the length of a finite sequence s, and s_i for the ith element of s, $1 \leq i \leq |s|$.
- X^* is the set of finite sequences over X.
- ϵ denotes the empty sequence.

Definition 3.1. [Abramsky, 1997] A *game* G is a structure (M_G, λ_G, P_G), where

- M_G is the set of *moves* of the game;

[4] The $\oplus$ operator stands for "exclusive disjunction".

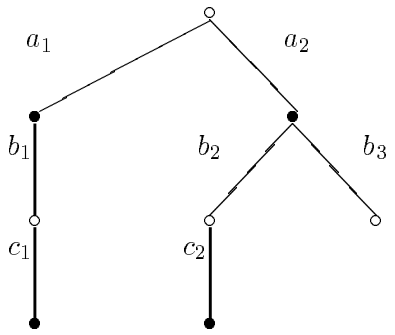

Figure 1: Game tree of the example 3.1

- $\lambda_G : M_G \rightarrow \{P, O\}$ is a labeling function designating each move as by *Proponent* or *Opponent*;
- $P_G \subseteq M_G^{alt}$, P_G is a non-empty, prefix-closed subset of M_G^{alt}, the set of alternating sequences of moves in M_G. More formally, M_G^{alt} is the set of all $s \in M_G^*$ such that

$$\begin{aligned} \forall i : 1 \leq i \leq |s|, \quad & even(i) \text{ then } \lambda_G(s_i) = P \\ \wedge \quad & odd(i) \text{ then } \lambda_G(s_i) = O \end{aligned}$$

  P_G represents the game tree.

■

Example 3.1. Let's consider the following game:

$$\begin{aligned} G = \Big(& \{a_1, a_2, b_1, b_2, b_3, c_1, c_2\}, \\ & \{\lambda(a_1) = P, \lambda(a_2) = P, \lambda(b_1) = O, \lambda(b_2) = O, \lambda(b_3) = O, \lambda(c_1) = P, \lambda(c_2) = P\}, \\ & \{\epsilon, a_1, a_2, a_1b_1, a_2b_2, a_2b_3, a_1b_1c_1, a_2b_2c_2\} \Big) \end{aligned}$$

G represents the tree in figure 1. ■

Games classify *behaviors*, thus *Programs* will be modeled by strategies,i.e., rules specifying how the System should actually play. Formally, [Abramsky, 1997] we define a strategy σ on a game G to be non-empty prefix-closed subset, $\sigma \subseteq P_G^{even}$ [5], satisfying:

$$sab,\ sac \in \sigma \ \Rightarrow \ b = c$$

We can regard a sequence $sab \in \sigma$ as saying: "when given the stimulus a in the context s, respond with b". Note that for every stimulus a strategy defines a unique response, therefore a strategy is deterministic. Abramsky classifies strategies as winning if every possible move of the opponent has some response. Thus the system must always be prepared to respond to any stimulus from the environment.

The next section defines the formalization of game semantics to capture a declarative semantics for a restricted set of defeasible logic programs.

4 DLP game semantics

Games have an analogy with argument-based reasoning. A dialectical tree can be seen as a game where in an alternating way, the player P proposes an argument for a literal and the player O tries to defeat it. A game for a literal h will be denoted G_h.

[5] $\sigma \subseteq P_G^{even}$ means: $\epsilon \in \sigma$ and if $sab \in \sigma$ then $s \in \sigma$.

Thus, informally, if P wins a game G_h whose first move is an argument for a literal h in a defeasible program DP, then h will belong to DP's semantics. In order to capture this game-based semantics, we define the set of moves M_G as a set of arguments. In particular, M_{G_h} will be a subset of the set $Arg_{\langle \mathcal{A},h\rangle}$, where $\langle \mathcal{A}, h\rangle$ is an argument of h:

$$\begin{array}{lcl} Arg^0_{\langle \mathcal{A},h\rangle} & = & \{\langle A, h\rangle\} \ \cup \ \{\langle A, \sim h\rangle\} \\ Arg^i_{\langle \mathcal{A},h\rangle} & = & \{\langle A, l\rangle | \text{ there exists } l \prec l_1, ...l_n \in A' \text{ where } \langle A', h'\rangle \ \in \ Arg^{i-1}_{\langle \mathcal{A},h\rangle}\} \cup \\ & & \{\langle A, \sim l\rangle | \text{ there exists } l \prec l_1, ...l_n \in A' \text{ where } \langle A', h'\rangle \ \in \ Arg^{i-1}_{\langle \mathcal{A},h\rangle}\} \cup \end{array}$$

$$Arg_{\langle \mathcal{A},h\rangle} \ = \ \bigcup_i Arg^i_{\langle \mathcal{A},h\rangle}$$

$Arg_{\langle \mathcal{A},h\rangle}$ contains all possible attack points (argument and counterargument) for the argument $\langle \mathcal{A}, h\rangle$ of h.

Definition 4.1. Let $DP = (\mathcal{K}, \Delta)$ be a defeasible logic program, h a literal and $\langle \mathcal{A}, h\rangle$ an argument for h. A *game* for h with respect to DP, that we denote G_h is a structure $(M_{G_h}, \lambda_{G_h}, P_{G_h})$, where

- $M_{G_h} \subseteq Arg_{\langle \mathcal{A},h\rangle}$;
- $\lambda_{G_h} : \ M_{G_h} \rightarrow \{P, O\}$;
- $P_{G_h} \subseteq M^{alt}_{G_h}$, P_{G_h} is a non-empty, prefix-closed subset of $M^{alt}_{G_h}$, the set of alternating sequences of moves in M_{G_h}. Formally, $M^{alt}_{G_h}$ is the set of all $s \in M^*_{G_h}$ such that

  $$\begin{array}{lll} \forall i : \ 1 \leq i \leq |s|, & & even(i) \ \text{ then } \lambda_G(s_i) = O \\ & \wedge & odd(i) \ \text{ then } \lambda_G(s_i) = P \end{array}$$

  Every sequence s of P_{G_h} satisfies:

  $$(\exists s' \ \ s = \langle \mathcal{A}, h\rangle s') \vee (s = \epsilon)$$

  where $a = \langle \mathcal{A}, h\rangle$.

■

Note that unlike the original game definition, the first move in a game G_h is always done by the proponent. Furthermore, we require that every game begins with an argument for the literal h.

Every move is required to capture supporting or interfering arguments of the initial argument, depending on the player. For this reason we are only interested on move sequences of certain kind: those which capture a dialectical tree. Before defining a *legal sequence* in a game let's introduce some notation. We denote the projection of the first element of s_i as $s^{\mathcal{A}}_i$ and we will denote the projection of the second element of s_i as s^h_i.

Definition 4.2. Let $\langle K, \Delta\rangle$ be a defeasible program where a preference relation holding the conditions discussed in section 2 is defined. A sequence s is *legal* in a game if it satisfies the following conditions:

- Players alternate: i.e., if $s = s_1 mns_2$ then $\lambda_G(m) \neq \lambda_G(n)$
- Avoiding inconsistency between moves of the same player: $1 \leq i \leq |s|$

$$\mathcal{K} \cup \bigcup_{even(i)} s^{\mathcal{A}}_i \not\models \bot \ \ \wedge \ \mathcal{K} \cup \bigcup_{odd(i)} s^{\mathcal{A}}_i \not\models \bot$$

- Every move is a counterargument of the precedent move: $\forall i\ 1 \leq i \leq (|s| - 1)$

$$\mathcal{K} \cup\ s_i^{\mathcal{A}} \cup s_{i+1}^{\mathcal{A}} \models \bot$$

■

By the way we have restricted the preference relation that should be used in order to decide between contradictory information, it is not possible to find reciprocal defeaters and therefore cycles[6] in sequences are avoid.

As we would like to use game semantics to model DLP without default negation, we should find a condition to determine whether a game models a justification or not.

Definition 4.3. A game G_h is legal if and only if:

- *every sequence in* P_{G_h} *is legal* and
- *all counterarguments are played*: if $s'\langle \mathcal{A}_1, h_1\rangle \in P_{G_h}$, s' possibly empty, then there exists a sequence $s \in P_{G_h}$ such that $s = s'\langle \mathcal{A}_1, h_1\rangle\langle \mathcal{A}_2, h_2\rangle$ for every counterargument $\langle \mathcal{A}_2, h_2\rangle$ of $\langle \mathcal{A}_1, h_1\rangle$.

■

Definition 4.4. Let a be the first move of the proponent in the game. A sequence s is complete if either $s = a$ or if $s = as_1b$ then there is no move $c \in M_G$ such that $as_1bc \in P_G$. A sequence s' is total if every move of the opponent has a response from the proponent. Briefly, a sequence s' is total if $|s'|$ is odd. ■

A complete sequence means an argumentation line, i.e., a path from the first move to a move that allows us to reach a leaf. A total sequence s' captures the fact that s' ends with a proponent move.

Definition 4.5. We define a strategy σ on a game G to be a non-empty prefix-closed set of total sequences of P_G,i.e., $s \in \sigma$ if s is odd-length. ■

Definition 4.6. Let G_h be a legal game for a literal h in a defeasible logic program DP where a preference relation holding the conditions discussed in section 2 is defined. If the set of complete legal sequences in P_{G_h} is a strategy then h belongs to the game semantics. ■

The above definition introduces a declarative way of analyzing a dialectical tree. By requiring that every argumentation line ends with a proponent move, we are able to capture a line won by the proponent. Let's illustrate these definitions.

Example 4.1. Let $(\mathcal{K}, \Delta)$ be the defeasible program introduced in 2.1. We define the game $G_g = (M_{G_g}, \lambda_{G_g}, P_{G_g})$ for $g(clyde)$, where

- $M_{G_g} = \{\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3\}$;
- $\lambda_{G_g}(\mathcal{A}_1) = P,\ \lambda_{G_g}(\mathcal{A}_2) = O,\ \lambda_{G_g}(\mathcal{A}_3) = P$;
- $P_{G_g} = \{\epsilon,\ \mathcal{A}_1,\ \mathcal{A}_1\mathcal{A}_2,\ \mathcal{A}_1\mathcal{A}_2\mathcal{A}_3\}$

The unique complete sequence is $\mathcal{A}_1\mathcal{A}_2\mathcal{A}_3$ and is a strategy, therefore $g(clyde)$ belongs to the game-based semantics. ■

[6] A sequence s contains a cycle if $s = s's''s'''s''s'^v$.

5 Conclusion and Future Work

A game-based semantics for modeling a dialectical analysis has been introduced and formalized. Games have an analogy with argument-based reasoning. A dialectical tree can be seen as a game where in an alternating way, the player P proposes an argument for a literal and the player O tries to defeat it.

Thus if P wins a game G_h whose first move is an argument for a literal h in a defeasible program DP, then h belongs to DP's semantics. Although the scope of this formalization is circumscribed to preference relations where only proper defeaters are considered, we belive that this approach is a fitting one to capture the operational semantics, even when considering incomparable contradictory arguments.

DLP has been extended with default negation. A possible future work would be to decide whether it is possible to capture the operational semantics for the "not" operator.

References

[Abramsky, 1997] Abramsky, S. (1997). Semantics of Interaction. In A.Pitts and Dibyer, P., editors, *Semantics and Logic Computation*. Cambridge.

[Abramsky(b), 1997] Abramsky, S. and McCusker, G. (1997). Game Semantics. In *Proceedings of Marktoberdorf'97 - Summer School*.

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

[García, 1999] García, A. and Simari, G. R. (1999). Strong and Default Negation in Defeasible Logic Programming. In *4th Dutch/German Workshop on Nonmonotonic Reasoning Techniques and their applications*, Amsterdam.

[García, 1998] García, A., Simari, G. R., and Chesñevar, C. (1998). An Argumentative Framework for Reasoning with Inconsistent and Incomplete Information. In *ECAI 98, Proceedings de Workshop on Practical Reasoning and Rationality, 13th. European Conference on Artificial Intelligence*, Brighton, England.

[Gelfond, 1990] Gelfond, M. and Lifschitz, V. (1990). Logic programs with classical negation. In Warren, D. and Szeredi, P., editors, *Logic Programming: Proceedings of the Seventh International Conference*, pages 579–597.

[Lloyd, 1987] Lloyd, J. W. (1987). *Foundations of Logic Programming*. Springer-Verlag, New York, second edition.

[Minker, 1982] Minker, J. (1982). On Indefinite Data Base and the Closed World Assumption. In Loveland, D., editor, *Proc. 6th Conf. on Automated Deduction (CADE'82)*, pages 292–308, LNCS 138, New York. Springer.

[Prakken, 1996] Prakken, H. and Sartor, G. (1996). A dialectical model of assessing conflicting arguments in legal reasoning. *Artificial Intelligence and Law*, 4:331–368.

[Simari, 1992] Simari, G. R. and Loui, R. (1992). A mathematical treatment of defeasible reasoning and its implementation. *Artificial Intelligence*, pages 125–157.