# Actions, Planning and Defeasible Reasoning *

**Guillermo R. Simari and Alejandro J. García and Marcela Capobianco**

grs@cs.uns.edu.ar agarcia@cs.uns.edu.ar mc@cs.uns.edu.ar
Artificial Intelligence Research and Development Laboratory
Department of Computer Science and Engineering
Universidad Nacional del Sur
Av. Alem 1253, (8000) Bahía Blanca, Argentina

**Abstract**

The aim of this work is to study an argumentation-based formalism that an agent could use for constructing plans. Elsewhere, we have introduced a formalism for agents to represent knowledge about their environment in Defeasible Logic Programming, and a set of actions that they are capable of executing in order to change the environment where they are performing their tasks. We have also shown that action selection, when combined with a defeasible argumentation formalism, is more involved than expected.

In this paper we will develop a novel way of using argumentation in the definition of actions and combining those actions to form plans. Since our interest here lies in exploring the important issues that need to be addressed, the main contribution will be to show meaningful examples where those issues are exhibited and not in improving current planning implementations. Therefore, we will use simple planning algorithms in an effort to reduce the complexity of the examples. Nevertheless, as the different ways of constructing plans introduce interesting details, we will be considering progression and regression planning.

## General Considerations

The aim of this work is to define actions in a way that would permit the use of an argumentation-based formalism to select among them and to construct plans. Several formalisms have been introduced relating some form of argumentation formalism to reason about action and change (KMT99; KMT00; Bre01). Argumentation will be the device used to introduce defeasibility in the construction of plans. In previous work, we have introduced a formalism where agents represent their knowledge about the environment using the language of Defeasible Logic Programming (DeLP) (SG01; SG02; GS04). They also should have a set of actions that they can execute in order to change the environment where they are performing their tasks. Here, we will explore the consequences of using this kind of formalism in the selection of actions.

*Partially supported by Secretaría General de Ciencia y Tecnología de la Universidad Nacional del Sur and the Agencia Nacional de Promoción Científica y Tecnológica (PICT 2002 Nro 13096)

We will begin by developing a novel way of using argumentation in the definition of actions and combining those actions to form plans. Since our current interest lies in exploring the basic, foundational issues that need to be addressed, the main contribution in this work will be to show meaningful examples where those issues are exposed and not in improving current planning implementations. Therefore, we will consider simple planning algorithms in an effort to reduce the complexity of the examples. Nevertheless, as the different methods of constructing plans introduce interesting details, we will be considering progression and regression planning.

We have shown (SG01) that action selection, when combined with an argumentation formalism, is more involved than expected specially in regression planning. Because of these complexities, any naïve algorithm would be computationally inadequate. After completing the necessary definitions, we will address the implementation concerns succinctly.

## First Step: Actions

We will begin with an example. Suppose an agent named *Yosemite Sam*, or *Sam* for short, is in the dark. He wants to *light* up the place and he has a *match*. *Sam* is lost, but he knows that the room he is in is a *gunpowder shed*. He also has some defeasible rules to guide his behavior: *"if someone is in the dark then he would like to produce light"*, *"if someone wants to produce light and he has a match, then he has a good reason for striking the match"* and *"being in a gunpowder shed is a good reason for not striking a match"*.

Our formalism follows the logic programming paradigm for knowledge representation (GS04). In that setup, an agent's knowledge will be represented by a knowledge base $\mathcal{K} = (\Psi, \Delta)$, where $\Psi$ should be a consistent set of *facts*, and $\Delta$ a set of *defeasible rules*. Below, we have agent *Sam*'s knowledge base as described before.

$$\Psi = \{\ in\_the\_dark(s), lost(s), \\ has(s, match), at(s, gps)\ \}$$

$$\Delta = \{\ wants(s, light) \prec lost(s), in\_the\_dark(s), \\ strike(s, match) \prec has(s, match), wants(s, light), \\ \sim strike(s, match) \prec has(s, match), at(s, gps)\ \}$$

From the knowledge base described, *Sam* could not decide what to do because he has reasons both in favor and against striking the match. Notice that a defeasible rule is the key element for introducing *defeasibility* (Pol95) and it is used to represent a relation between pieces of knowledge that could be *defeated*, after all things are considered. A defeasible rule "$Head \prec Body$" is understood as expressing that *"reasons to believe in the antecedent Body of a rule provide reasons to believe in its consequent, Head*" (SL92). *Strong negation* "$\sim$" is allowed in the head of defeasible rules, and and it can therefore be used to represent potentially contradictory knowledge.

Thus, the agent's knowledge base will be represented using a restricted *defeasible logic program* (*delp*). The results already obtained for such argumentation-based form of logic programming will be used here freely, but a brief description of DeLP will be introduced below. The interested reader is referred to (GS04) for details about DeLP's general framework and to (CCS01; Cap03) for the restricted form of the language used here.

In DeLP, a literal $L$ is *warranted* from knowledge base $\mathcal{K} = (\Psi, \Delta)$ if there exists a non-defeated *argument* $\mathcal{A}$ supporting $L$. An argument structure $\mathcal{A}$ for a literal $L$, denoted $\langle \mathcal{A}, L \rangle$, is a minimal and consistent set of defeasible rules that allows to infer $L$. In order to establish whether $\langle \mathcal{A}, L \rangle$ is a non-defeated argument, *argument rebuttals* or *counter-arguments* that could be *defeaters* for $\langle \mathcal{A}, L \rangle$ are considered, *i.e.*, counter-arguments that by some criterion are preferred to $\langle \mathcal{A}, L \rangle$. Since counter-arguments are arguments, there may exist defeaters for them, and defeaters for these defeaters, and so on. Thus, a sequence of arguments called *argumentation line* appears, where each argument defeats its predecessor in the line (see the example below). Usually, for a given argument more than one defeater exists and in that case more than one argumentation line could appear. Thus, all argumentation lines could be consolidated in a tree of arguments called *dialectical tree*, where the root is $\langle \mathcal{A}, L \rangle$ and each path from the root to a leaf is an argumentation line. Finally, a *dialectical analysis* of this tree is used for deciding whether $L$ is warranted. Next, all of the concepts introduced before are exemplified. We will use a propositional language in order to simplify the example.

**Example 1** Consider the following knowledge base $\mathcal{K} = (\Psi, \Delta)$ with the set of facts

$$\Psi = \{a, b, c, d\}$$

and the set of defeasible rules,

$$\Delta = \{(p \prec b),\ (q \prec r),\ (r \prec d),\ (\sim r \prec s), (s \prec b), \\ (\sim s \prec a, b),\ (w \prec b), (\sim w \prec b, c)\}$$

Here, the literal $p$ has the argument $\mathcal{A}=\{p \prec b\}$ supporting it, and $\mathcal{A}$ is undefeated because there is no counter-argument for it. Hence, $p$ is warranted. The literal $q$ has the argument $\mathcal{A}_1 = \{(q \prec r),\ (r \prec d)\}$, but $\mathcal{A}_1$ is defeated by $\mathcal{A}_2 = \{(\sim r \prec s),\ (s \prec b)\}$, that attacks $r$, an inner point in $\mathcal{A}_1$. The argument $\mathcal{A}_2$ is in turn defeated by $\mathcal{A}_3 = \{(\sim s \prec a, b)\}$. Thus, the argumentation line $[\mathcal{A}_1,\ \mathcal{A}_2,\ \mathcal{A}_3]$ is obtained. The literal $q$ is warranted because its supporting argument $\mathcal{A}_1$ has only one defeater $\mathcal{A}_2$ that is defeated by $\mathcal{A}_3$, and $\mathcal{A}_3$ has no defeaters.

Observe that there is no warrant for $\sim r$ because $\mathcal{A}_2$ is defeated by $\mathcal{A}_3$. The literals $t$ and $\sim t$ have no argument, so neither of them is warranted. Every fact of $\Psi$ is trivially warranted, because no counter-argument can defeat a fact.

Following Lifschitz (Lif96), DeLP rules, strict and defeasible, could be represented as *schematic rules* making abstraction of the object constants. The resulting DeLP programs are therefore *schematic programs*.

Besides its knowledge base $\mathcal{K}$, an agent will have a set of actions $\Gamma$ that it may use to change its world. Once an action has been applied, the effect of the action will change the set $\mathcal{K}$. The formal definitions that were introduced in (SG01) are recalled below.

**Definition 1 [Action]** An action $A$ is an ordered triple $\langle \mathsf{X}, \mathsf{P}, \mathsf{C} \rangle$, where $\mathsf{X}$ is a consistent set of literals representing consequences of executing $A$, $\mathsf{P}$ is a set of literals representing preconditions for $A$, and $\mathsf{C}$ is a set of constraints of the form *not* $L$, where $L$ is a literal. We will denote actions as follows:

$$\{X_1, \ldots, X_n\} \xleftarrow{A} \{P_1, \ldots, P_m\}, not\ \{C_1, \ldots, C_k\}$$

Notice that the notation *not* $\{C_1, \ldots, C_k\}$ represents $\{not\ C_1, \ldots, not\ C_k\}$.

Note that the dialectical process may have different outcomes. When looking for a warrant for a literal $L$ there might be four different answers: YES, if $L$ is warranted; NO, if $\sim L$ is warranted; UNDECIDED, if there is no warrant for $L$ and no warrant for $\sim L$; and UNKNOWN, if $L$ is a literal that is not possible to consider given the knowledge base $\mathcal{K}$, *i.e.*, $L$ is not part of the language of $\mathcal{K}$.

Accordingly, the condition that must be satisfied before an action $A = \langle \mathsf{X}, \mathsf{P}, \mathsf{C} \rangle$ can be executed contains two parts: $\mathsf{P}$, which mentions the literals that *must* be warranted, and $\mathsf{C}$, which mentions the literals whose negations *must not* be warranted. In this way, the satisfaction of the preconditions could also depend on the fact that some information is unknown (*un-warranted*).

For example, using this form of specification of actions, it is possible to express conditions such as the

following: *"If it did not rain today and it is unknown when it might rain, then water the garden"*, using *wg* for *water the garden*:

$$\{wg(today)\} \xleftarrow{wg} \{\sim rain(today)\}, not\ \{rain(X)\}$$

Formally,

**Definition 2 [Applicable Action]** Let $\mathcal{K} = (\Psi, \Delta)$ be an agent's knowledge base. Let $\Gamma$ be the set of actions available to this agent. An action $A$ in $\Gamma$, defined as before, is applicable if every precondition $P_i$ in $\mathsf{P}$ has a warrant built from $(\Psi, \Delta)$ and every constraint $C_i$ in $\mathsf{C}$ fails to be warranted.

**Definition 3 [Action Effect]** Let $\mathcal{K} = (\Psi, \Delta)$ be an agent's knowledge base. Let $\Gamma$ be the set of actions available to this agent. Let $A$ be an applicable action in $\Gamma$ defined by:

$$\{X_1, \ldots, X_n\} \xleftarrow{A} \{P_1, \ldots, P_m\}, not\ \{C_1, \ldots, C_k\}$$

The effect of executing $A$ is the revision of $\Psi$ by $\mathsf{X}$, i.e. $\Psi^{*\mathsf{X}} = \Psi^{*\{X_1,\ldots,X_n\}}$. Revision will consist of removing any literal in $\Psi$ that is complementary of any literal in $\mathsf{X}$ and then adding $\mathsf{X}$ to the resulting set. Formally:

$$\Psi^{*\mathsf{X}} = \Psi^{*\{X_1,\ldots,X_n\}} = (\Psi - \overline{\mathsf{X}}) \cup \mathsf{X}$$

where $\overline{\mathsf{X}}$ represents the set of complements of members of $\mathsf{X}$.

**Example 2** Let $\mathcal{K}$= $(\Psi, \Delta)$ be an agent's knowledge base as defined in Example 1

$\Psi = \{a, b, c, d\}$
$\Delta = \{(p \prec b),\ (q \prec r),\ (r \prec d),\ (\sim r \prec s), (s \prec b),$
$(\sim s \prec a, b), (w \prec b), (\sim w \prec b, c)\}$

And $\Gamma$ the set of available actions containing only:

$$\{\sim a, d, x\} \xleftarrow{A} \{a, p, q\}, not\ \{t, \sim t, w\}$$

This action is applicable because every literal in the precondition set has a warrant, and no constraints in $\{t, \sim t, w\}$ are warranted (see Example 1). If the action is executed, the set of facts becomes:

$$\Psi' = \{b, c, \sim a, d, x\}$$

Observe that the precondition $a$ was "consumed" by the action.

In (SG01), we have shown that the interaction between actions and the defeasible argumentation formalism is twofold. On one hand, as stated by Definition 2, defeasible argumentation is used for testing preconditions and constraints through the warrant notion. On the other hand, actions may be used by agents in order to change the world (actually the set $\Psi$) and then have a warrant for a literal $L$ that has no warrant from the current knowledge base $(\Psi, \Delta)$.

As we have shown in (SG01), this interaction produces a powerful formalism. However, some new elements that are not present in traditional planning systems prompt for a deeper analysis.

## Defeasible Planning through Argumentation

A simple formulation of a planning problem defines three inputs (Wel99): a description of the *initial state* of the world in some formal language, a description of the agent's *goal*, and a description of the possible *actions* that can be performed. The initial state is the agent's current representation of the world, and in our case it will be the set $\Psi$. As stated above, through the execution of actions the agent may change its world. Therefore, in order to achieve its goals, the agent will start in the initial state $\Psi$ and it will execute a sequence of actions transforming $\Psi$ into $\Psi'$. The agent's goals will be represented as a set $G$ of literals. The agent will satisfy its goals when through a sequence of actions it reaches some state $\Psi'$ where each literal of $G$ is warranted. The planner will be in charge of obtaining the proper sequence of actions in advance.

### Progression Planning

A progression planner searches forward from the initial state $I$ to the goal state $G$. The outline of a progression planner that searches through the space of possible states follows:

INITIALIZE the current state $Q$ with $I$
REPEAT
- SELECT an action such that its preconditions hold in $Q$
- SIMULATE the action execution modifying $Q$ with the action effects

UNTIL $G \subseteq Q$

**Remark 1** We will assume, realistically, that none of the goals contained in the initial goal state $G$ is already holding in the initial state $I$, *i.e.*, $G \cap I = \emptyset$.

That is, the planner starts in the initial state $I$, it selects one applicable action and simulates its execution modifying the state with the action effects. The process continues until all the literals in $G$ hold in the current state $Q$. If there is more than one applicable action to select, then a choice point is generated and backtracking is possible. Thus, the planner will explore a space of states rooted in $I$.

Combining our proposed formalism with a progression planner is straightforward. The initial state is represented with the agent's facts $\Psi$. In each step, the planner will select an applicable action in the current state $\Psi$, that is, an action $A$=$\langle \mathsf{X}, \mathsf{P}, \mathsf{C} \rangle$, where every precondition $P_i$ in $\mathsf{P}$ has a warrant built from $(\Psi, \Delta)$, and every constraint $C_i$ in $\mathsf{C}$ fails to be warranted from $(\Psi, \Delta)$. The action effect will be calculated with $\Psi^{*\mathsf{X}}$. A progression planner algorithm that uses defeasible argumentation follows:

LET $\Psi$ be the initial state and $G$ the agent's goal
REPEAT
- SELECT an *applicable action* $A=\langle \mathsf{X}, \mathsf{P}, \mathsf{C}\rangle$
- SIMULATE the *action effect* with $\Psi := \Psi^{*\mathsf{X}}$

UNTIL all literals in $G$ are warranted from $(\Psi, \Delta)$

In progression planning, there are no further considerations, since it is unnecessary to protect the literals used to construct the warrants. Naturally, if there is a considerable number of actions, then the branching factor could be very large and the search problem intractable. Regression planning tries to improve this situation. In the next section we will analyze the combination of regression planning with our approach and study the interaction between the process of constructing warrants and executing planning.

## Regression Planning

A regression planner searches backward from the goal state $G$ to the initial state $I$. As we have remarked, we assume that $G \cap I = \emptyset$. The outline of a typical regression planner that searches through the space of possible states is given below:

REPEAT
- SELECT an action $A=\langle \mathsf{X}, \mathsf{P}, \mathsf{C}\rangle$, such that $\mathsf{X} \cap G \neq \emptyset$
- RECOMPUTE $G$ eliminating $\mathsf{X}$ and adding $\mathsf{P}$, that is, $G := (G - \mathsf{X}) \cup \mathsf{P}$

UNTIL $G \subseteq I$

Unfortunately, it is known that $\mathsf{X} \cap G \neq \emptyset$ is not enough as a selection condition. In a regression planner, the first action to be selected is the last to be executed. Therefore, at any point in the search, a selected action $A$ could introduce some literal that could interfere with an action already selected but that will be executed after $A$ in the plan. Consider the following example: the initial state is $I = \{c, e, f\}$, the goal state is $G = \{a\}$, and the actions are:

$$\{a\} \xleftarrow{A_1} \{b, c\}, not\ \{\}$$

$$\{\sim d, b\} \xleftarrow{A_2} \{d\}, not\ \{\}$$

$$\{\sim c, d\} \xleftarrow{A_3} \{e\}, not\ \{\}$$

$$\{d\} \xleftarrow{A_4} \{f\}, not\ \{\}$$

$$\{c\} \xleftarrow{A_5} \{f\}, not\ \{\}$$

The following table shows one possible trace of the regression planner outline:

| $G$ | selected action | $\mathsf{X}$ | $\mathsf{P}$ |
|---|---|---|---|
| $\{a\}$ | $A_1$ | $\{a\}$ | $\{b, c\}$ |
| $\{b, c\}$ | $A_2$ | $\{\sim d, b\}$ | $\{d\}$ |
| $\{c, d\}$ | $A_3$ | $\{\sim c, d\}$ | $\{e\}$ |
| $\{c, e\}$ | | | |

Since we are using a regression planner, the selected actions, in inverse order, form a plan. However, if the sequence $[A_3, A_2, A_1]$ is executed in this order, action $A_3$ will delete the literal $c$ from the initial state, but action $A_1$ needs $c$ as a precondition. Observe that $c$ was "assumed" to be present along the search. The problem can be solved if the literal $c$ is "protected". One way of achieving this is to add to the selection mechanism the condition $\overline{\mathsf{X}} \cap G = \emptyset$. That is,

REPEAT
- SELECT an action $A=\langle \mathsf{X}, \mathsf{P}, \mathsf{C}\rangle$, such that $\mathsf{X} \cap G \neq \emptyset$ and $\overline{\mathsf{X}} \cap G = \emptyset$
- RECOMPUTE $G$ as $(G - \mathsf{X}) \cup \mathsf{P}$

UNTIL $G \subseteq I$

Although this is a well known problem, we bring it up here because in our approach the standard solution that consists of protecting the necessary literals is inadequate. Thus, some further considerations are needed. First, we will consider examples of actions without constraints.

Observe that our approach uses a deductive knowledge base $(\Psi, \Delta)$, so a goal is not necessarily achieved when it becomes a member of the set. A goal literal $g \in G$ is achieved if and only if $g$ is warranted from the agent's current knowledge base $(\Psi, \Delta)$. From now on we will use $w(G)$ to represent the subset of warranted literals of $G$, *i.e.,* $w(G) = \{g \mid g \in G$ and $g$ is warranted$\}$. Thus, a planning problem will be solved when all the literals in $G$ become warranted, that is, $G = w(G)$. The modified outline follows:

REPEAT
- SELECT an action $A=\langle \mathsf{X}, \mathsf{P}, \mathsf{C}\rangle$, such that $\mathsf{X} \cap (G - w(G)) \neq \emptyset$ and $\overline{\mathsf{X}} \cap G = \emptyset$
- RECOMPUTE $G$ as $(G - \mathsf{X}) \cup \mathsf{P}$

UNTIL $G = w(G)$

We will analyze the behavior of this last outline through meaningful examples. In all of the following examples the agent has the goal $G = \{a\}$, and the actions of the agent are:

$$\{a\} \xleftarrow{A_1} \{b, c\}, not\ \{\}$$

$$\{\sim x, b\} \xleftarrow{A_2} \{e\}, not\ \{\}$$

**Example 3 [Argument Clipping]**
Suppose that the agent has the following knowledge base: $\Psi = \{e, x\}$ and $\Delta = \{\ c \prec x\}$. In order to achieve the goal "$a$", action $A_1$ is selected first, and $G$ becomes $\{b, c\}$. Observe that $\mathcal{B}=\{c \prec x\}$ is an undefeated argument, so the literal $c$ is warranted. Since literal $b$ is not warranted, the planning process continues and action $A_2=\langle \mathsf{X}_2, \mathsf{P}_2, \mathsf{C}_2\rangle$ is selected. The effect of $A_2$ is $\mathsf{X}_2 = \{b, \sim x\}$, so $\overline{\mathsf{X}}_2 \cap G = \emptyset$ holds, and $G$ becomes $\{e, c\}$. Both literals are warranted, so plan $[A_2, A_1]$ appears to be correct.

However, if action $A_2$ is executed in the initial state, the literal $x$ is removed from $\Psi$. Therefore, no argument for $c$ can be built and action $A_1$ cannot be executed.

Example 3 shows that the literals in $G$ are not only the ones that should be protected. All the facts used for constructing arguments involved in the warrant of a literal in $G$ need to be protected. The following example shows a different situation where an action causes, as a side effect, the existence of a new argument. This new argument becomes a defeater of an argument that was assumed undefeated.

**Example 4 [Enabling a Defeater]**
Suppose that the agent has the following knowledge base: $\Psi = \{e, x, d\}$ and $\Delta = \{(c \prec d), (\sim c \prec \sim x)\}$. In order to achieve the goal "$a$", action $A_1$ is selected first, and $G$ becomes $\{b, c\}$. The literal $c$ is warranted because $\mathcal{B}=\{c \prec d\}$ is an undefeated argument that supports it. Observe that although there is a rule with head $\sim c$, there is no defeater for $\mathcal{B}$ because $\sim x$ is not in the knowledge base. Since literal $b$ is not warranted, the planning process continues and action $A_2=\langle \mathsf{X}_2, \mathsf{P}_2, \mathsf{C}_2\rangle$ is selected. The effect of $A_2$ is $\mathsf{X}_2 = \{b, \sim x\}$, so $\overline{\mathsf{X}}_2 \cap G = \emptyset$ holds, and $G$ becomes $\{e, c\}$. Since both literals are warranted, a plan $[A_2, A_1]$ appears to be found.

However, if action $A_2$ is executed, literal $\sim x$ is added to $\Psi$, and then the argument $\mathcal{C}=\{\sim c \prec \sim x\}$ can be obtained. Since $\mathcal{C}$ defeats $\mathcal{B}$, there is no warrant for $c$ and action $A_1$ cannot be executed.

Example 4 shows a case where, as a side effect of one of the selected actions, a new argument can be built. This new argument interferes with the warrant of a literal that was assumed warranted. Therefore, not only the literals but the existence of warrants for literals needs to be protected. The following example shows a different situation where a warrant disappears because a supporting argument defeating a defeater disappears.

**Example 5 [Disabling a Defeater]**
Suppose that the agent has the following knowledge base: $\Psi = \{e, x, g\}$ and $\Delta = \{(c \prec d), (d \prec e), (\sim d \prec e, f), (f \prec g), (\sim f \prec x)\}$. In order to achieve the goal "$a$", action $A_1$ is selected first, and $G$ becomes $\{b, c\}$. The literal $c$ is warranted, because although $\mathcal{B}=\{(c \prec d), (d \prec e)\}$ is defeated by $\mathcal{C} = \{(\sim d \prec e, f), (f \prec g)\}$, a third argument $\mathcal{D}= \{\sim f \prec x\}$ defeats $\mathcal{C}$ reinstating $\mathcal{B}$.

Again, action $A_2=\langle \mathsf{X}, \mathsf{P}, \mathsf{C}\rangle$ is selected next, $G$ becomes {e, c}, and a plan $[A_2, A_1]$ appears to be found. However, if action $A_2$ is executed, literal $x$ is removed from $\Psi$, and then the argument $\mathcal{D}=\{\sim f \prec x\}$ cannot be obtained. Argument $\mathcal{C}$ is now undefeated and since $\mathcal{C}$ defeats $\mathcal{B}$, there is no warrant for $c$ and action $A_1$ cannot be executed because $c$ is needed as a precondition.

It is clear that, when an action is executed new literals can be added or deleted from $\Psi$. As a consequence, new defeaters could appear or disappear interfering with the existence of assumed warrants. As it was shown in the examples above, this could cause the planner to select an improper sequence of actions that cannot be used as a plan. In traditional planning, the solution is to protect the literals. However, since in this approach we are using a deductive knowledge base, we need to protect the warrant of the literals. A solution to this problem is proposed in the following section.

## Protecting Warrants in Regression Planning

In DeLP, an *argumentation line* (GS04) starting at $\langle \mathcal{A}_0, L_0\rangle$ is a sequence of arguments

$$[\langle \mathcal{A}_0, L_0\rangle, \langle \mathcal{A}_1, L_1\rangle, \langle \mathcal{A}_2, L_2\rangle \langle \mathcal{A}_3, L_3\rangle, \ldots]$$

where each element of the sequence $\langle \mathcal{A}_i, L_i\rangle$, $i > 0$, is a defeater of its predecessor $\langle \mathcal{A}_{i-1}, L_{i-1}\rangle$. Then, $\langle \mathcal{A}_0, L_0\rangle$ becomes a *supporting* argument for $L_0$, $\langle \mathcal{A}_1, L_1\rangle$ an *interfering* argument posed against $\langle \mathcal{A}_0, L_0\rangle$, $\langle \mathcal{A}_2, L_2\rangle$ a supporting argument because it attacks $\langle \mathcal{A}_1, L_1\rangle$, $\langle \mathcal{A}_3, L_3\rangle$ an interfering one as it attacks $\langle \mathcal{A}_2, L_2\rangle$, *etc.* Thus, an argumentation line $\Lambda = [\langle \mathcal{A}_0, L_0\rangle, \langle \mathcal{A}_1, L_1\rangle, \langle \mathcal{A}_2, L_2\rangle \langle \mathcal{A}_3, L_3\rangle, \ldots]$ can be split into two disjoint sets: The set $\Lambda_S=\{\langle \mathcal{A}_0, L_0\rangle, \langle \mathcal{A}_2, L_2\rangle, \langle \mathcal{A}_4, L_4\rangle, \ldots\}$ of supporting arguments, and the set $\Lambda_I=\{\langle \mathcal{A}_1, L_1\rangle, \langle \mathcal{A}_3, L_3\rangle, \ldots\}$ of interfering arguments. The warrant of a literal $L_0$ is obtained exploring all possible argumentation lines that start with $\langle \mathcal{A}_0, L_0\rangle$. These argumentation lines could be seen as paths from the root to a leaf in a tree. This tree is called *dialectical tree* in DeLP (see (GS04) for the complete details).

Suppose now that during the planning process the literal $p$ was assumed to be warranted for selecting an action $A$ and that warrant exists because of the argumentation line $[\langle \mathcal{A}_0, L_0\rangle, \langle \mathcal{A}_1, L_1\rangle, \langle \mathcal{A}_2, L_2\rangle, \langle \mathcal{A}_3, L_3\rangle, \langle \mathcal{A}_4, L_4\rangle]$. On the one hand, if an action $B$ selected after but to be executed before $A$ deletes one of the literals used in the supporting arguments $\{\langle \mathcal{A}_0, L_0\rangle, \langle \mathcal{A}_2, L_2\rangle, \langle \mathcal{A}_4, L_4\rangle\}$, then the warrant for $p$ could disappear. On the other hand, if the selected action $B$ adds a fact to the knowledge base, such that a new undefeated argument $\langle \mathcal{A}_i, L_i\rangle$ can be built and $\langle \mathcal{A}_i, L_i\rangle$ defeats any of the supporting arguments $\{\langle \mathcal{A}_0, L_0\rangle, \langle \mathcal{A}_2, L_2\rangle, \langle \mathcal{A}_4, L_4\rangle\}$, then the warrant for $p$ could also disappear.

The first problem could be avoided collecting all the facts used in $\{\langle \mathcal{A}_0, L_0\rangle, \langle \mathcal{A}_2, L_2\rangle, \langle \mathcal{A}_4, L_4\rangle\}$ and protecting them, requiring that no action can delete these facts. The second problem could be solved by ensuring that no new defeaters for the supporting arguments $\{\langle \mathcal{A}_0, L_0\rangle, \langle \mathcal{A}_2, L_2\rangle, \langle \mathcal{A}_4, L_4\rangle\}$ can be obtained.

Finally, observe that if a literal used in an interfering argument for $p$ $\{\langle \mathcal{A}_1, L_1\rangle, \langle \mathcal{A}_3, L_3\rangle\}$ is erased, then although the dialectical tree changes, the warrant for $p$ remains. However, it is important to note that a literal

could be used both in supporting and interfering arguments, so in such a case it should be protected for the supporting argument.

Therefore, to protect a warrant, all the facts used in supporting arguments and all the potential points of attack should be considered. This information will be used for avoiding, in advance, the selection of an improper action. The following definitions introduce the necessary elements.

**Definition 4** Let $L$ be a literal. We will define *warrants*$(L)$ as the set of warrants for that literal, *i.e.*, the set of all arguments $\mathcal{A}$ that are warrants for $L$. Given a set S of literals, *warrants*(S) represents the set of all warrants for literals in S.

**Definition 5** Let $L$ be a warranted literal with warrant $\mathcal{A}$. We define $SuppArg(\langle \mathcal{A}, L\rangle)$ as the set of all supporting arguments in the dialectical tree that shows that $\mathcal{A}$ is a warrant for $L$.

**Definition 6** Let $L$ be a warranted literal with warrant $\mathcal{A}$, and let $SuppArg(\langle \mathcal{A}, L\rangle) = \{\langle \mathcal{A}_1, L_1\rangle, \langle \mathcal{A}_2, L_2\rangle, \ldots \langle \mathcal{A}_k, L_k\rangle\}$. We define $Facts(SuppArg(\langle \mathcal{A}, L\rangle))$ as the set of all facts in $\Psi$ used in the construction of the arguments in $\{\langle \mathcal{A}_1, L_1\rangle, \langle \mathcal{A}_2, L_2\rangle, \ldots \langle \mathcal{A}_k, L_k\rangle\}$.

The set $Facts(SuppArg(\langle \mathcal{A}, L\rangle))$ represents the set of all known literals that are necessary for constructing the supporting arguments. Failure in maintaining these facts in the epistemic state of the agent would result in the loss of the warrant.

**Definition 7** Let $L$ be a warranted literal with warrant $\mathcal{A}$, and let $SuppArg(\langle \mathcal{A}, L\rangle) = \{\langle \mathcal{A}_1, L_1\rangle, \langle \mathcal{A}_2, L_2\rangle, \ldots \langle \mathcal{A}_k, L_k\rangle\}$. We define $Weak(SuppArg(\langle \mathcal{A}, L\rangle))$ as the set of all literals that are heads in rules contained in the arguments in $\{\langle \mathcal{A}_1, L_1\rangle, \langle \mathcal{A}_2, L_2\rangle, \ldots \langle \mathcal{A}_k, L_k\rangle\}$.

The set $Weak(SuppArg(\langle \mathcal{A}, L\rangle))$ represents the set of points that could be subject to attack in the dialectical process. These points were analyzed during the construction of the dialectical tree. Any existing defeater was defeated. We are taking notice of these points because any new defeater must attack one of them.

If we wish to maintain a warrant as such we should do two things. It is necessary to protect the literals in $Facts(SuppArg(\langle \mathcal{A}, L\rangle))$, and we need to avoid introducing new literals that could allow the construction of new defeaters for $SuppArg(\langle \mathcal{A}, L\rangle)$.

**Definition 8** Let $\mathcal{K} = (\Psi, \Delta)$ be the agent's knowledge base, $G$ the agent's goal, *i.e.*, a set of literals, and $[A_1, A_2, \ldots, A_n]$ the actions selected by the regression planner. Let $\{\langle \mathcal{A}_1, L_1\rangle, \langle \mathcal{A}_2, L_2\rangle, \ldots, \langle \mathcal{A}_k, L_k\rangle\}$ be the set of warrants $\mathcal{A}_i$ for the literals $L_i$ that were assumed to be warranted for the selection of the actions $[A_1, A_2, \ldots, A_n]$. We will define:

$$Protect = \bigcup_{i=1..k} Weak(SuppArg(\langle \mathcal{A}_i, L_i\rangle))$$

and

$$PossAttack = \bigcup_{i=1..k} Facts(SuppArg(\langle \mathcal{A}_i, L_i\rangle))$$

The sets *Protect* and *PossAttack* will be used by the planner during the action selection process. In this manner, the planner will not select an action that during the execution of the resulting plan could cause a protected literal to be erased, or a new defeater for a supporting argument to be constructed. Note that if backtracking occurs, the sets *Protect* and *PossAttack* have to be updated accordingly.

Notice that for each literal in *Protect* and *PossAttack* there could be more than one warrant. This could help in the process of selecting the actions to reach the goals in $G$ since a convenient action could still be selected replacing the current warrant by a different one.

Accordingly, we can modify the outline of the planner as follows:

REPEAT
- SELECT an action $A=\langle \mathsf{X}, \mathsf{P}, \mathsf{C}\rangle$, such that:
  1. $\mathsf{X} \cap (G - w(G)) \neq \emptyset$
  2. $\overline{\mathsf{X}} \cap Protect = \emptyset$
  3. there is no new undefeated defeater for a warrant for a member of *PossAttack* from $\Psi \cup \mathsf{X}$
- RECOMPUTE $G$ as $(G - \mathsf{X}) \cup \mathsf{P}$
- UPDATE *Protect* and *PossAttack* accordingly

UNTIL $G = w(G)$

Although this last solution averts the problems we have mentioned, it is not complete. Consider the example below, which is a variant of Example 3

**Example 6 [Action Selection]**
Consider an agent with the goal $G = \{a\}$, and the actions:

$$\{a\} \xleftarrow{A_1} \{b, c\}, not\ \{\}$$

$$\{\sim x, b\} \xleftarrow{A_2} \{e\}, not\ \{\}$$

$$\{c\} \xleftarrow{A_3} \{e\}, not\ \{\}$$

Suppose that the agent has the following knowledge base: $\Psi = \{e, x\}$ and $\Delta$=$\{(c \prec x)\ \}$. In order to achieve the goal "$a$", action $A_1$ is selected first, and $G$ becomes $\{b, c\}$. Observe that the literal $c$ is warranted because $\mathcal{B}$=$\{c \prec x\}$ is an undefeated argument The planner has to protect the warrant of $c$ and therefore sets *Protect* $= \{x\}$.
Since literal $b$ is not warranted, the planning process continues and action $A_2$=$\langle \mathsf{X}, \mathsf{P}, \mathsf{C}\rangle$ is considered. The action cannot be selected because $\overline{\mathsf{X}} \cap Protect = \{x\}$.

Therefore, no plan is found, but clearly a plan exists: $[A_2, A_3, A_1]$. However, this plan was not considered because the literal $c$ was warranted when the action $A_1$ was selected.

In order to avert the problem introduced in Example 6 we propose the following solution. When an applicable action $A = \langle \mathsf{X}, \mathsf{P}, \mathsf{C} \rangle$ cannot be selected because one of its effects deletes a protected literal that is necessary for the warrant of a literal $c$, instead of simply discard the action, the planner will search for another way of obtaining $c$ and insert this subsidiary plan into the main plan. This search must consider the same restrictions regarding *Protect* and *PossAttack* that the main planner is considering in that step.

## Implementation Issues

In (Cap03; CCS04) a restricted version of DeLP, known as Observation based Defeasible Logic Programming (ODeLP), was developed and studied. ODeLP addresses some of the implementation concerns associated with argumentative formalisms, borrowing concepts from the theory of Truth Maintenance Systems (TMS) presented in (Doy79). Under this view, precompiled knowledge may be used to optimize the inference process of ODeLP in the same way truth maintenance systems improve the performance of problem solvers.

Associated with every ODeLP program $\mathcal{K} = (\Psi, \Delta)$ there is a data structure called *Dialectical Base*. Simply put, the dialectical base of a given program stores all the arguments that could be built from the rules in $\Delta$. These arguments are *compiled* in such a way that it is possible to obtain warrants in a computationally less costly way.

Dialectical bases may be seen as a set of *potential arguments*. Potential arguments are different from the notion used in section , where arguments are sets of ground defeasible rules, instantiated accordingly with the literals in the set $\Psi$. Even though defeasible rules are ground, they could also be expressed as "schematic rules" with variables, as it is done in Logic Programming (see (Lif96)). This form of expressing the knowledge base is more convenient for potential arguments, since they must be independent from the current perceptions represented in the set $\Psi$. When using schematic rules it is possible to obtain "schematic arguments", which stand for a set of different arguments, which are instantiated according to a particular $\Psi$.

From $\mathcal{K} = (\Psi, \Delta)$, which represents the agent's epistemic state, *all* the potential arguments could be obtained. As mentioned above, these structures depend only on the set $\Delta$ of defeasible rules. To finish the construction of the Dialectical Base we also need to record the *defeat relation* among potential arguments. Using this relationship, and the potential arguments, it is possible to carry out the dialectical analysis in a seamless manner.

**Example 7 Dialectical Base.** Consider the knowledge base regarding *Sam* and his situation in the *gunpowder shed* given in section . The dialectical base for this program consists of the following potential arguments:

$$\mathsf{A}_1 = \{strike(X,Y) \prec has(X,Y), wants(X,Z)\}$$
$$\mathsf{A}_2 = \{\sim strike(X,Y) \prec has(X,Y), at(X,Z)\}$$

The change in notation reflects the situation that these are not *arguments* but *potential arguments*. Note that the set consisting of both rules is not a potential argument (since once instantiated it may result in an inconsistent set of rules). The defeat relation establishes that $\mathsf{A}_2$ defeats $\mathsf{A}_1$.

When reasoning with a particular knowledge base $\mathcal{K} = (\Psi, \Delta)$, only the potential arguments that can be instantiated using the individual constants appearing in $\Psi$ are considered in the dialectical analysis. Since the defeat relation is already given, and the heavy computational work is done *off-line*, the cost of the inference process is greatly lowered. Integrating a module of pre-compiled knowledge in our planner can therefore optimize the process of deciding when an action is applicable, a key issue for implementing the system.

## Conclusions and Future Work

We have introduced a way in which argumentation can be used in the definition of actions and the combination of those actions to form plans. Our aim was not centered on improving current planning implementations. We have explored how this new approach can be integrated with a simple planning algorithm.

As we have shown above, the use of defeasible argumentation in progression planning is almost straightforward. However, regression planning becomes rapidly more difficult and deserves more attention. The combination of searching backwards for appropriate actions with the task of keeping warrants for literals could produce unexpected results. Several examples that illustrate these problems were introduced, and solutions were proposed.

Future work includes the analysis of other methods and planning systems, and the implementation of a planner based in the framework described.

## References


G. Brewka. Dynamic argument systems: A formal model of argumentation processes based on situation calculus. *Journal of Logic and Computation*, 11(2):257–282, 2001.

M. Capobianco. *Argumentación rebatible en entornos dinámicos*. PhD thesis, Computer Science and Engineering Department, Universidad Nacional del Sur, Bahía Blanca, Argentina, June 2003.

M. Capobianco, C. I. Chesñevar, and G. R. Simari. An argumentative formalism for implementing rational agents. In *Proceedings del 2do Workshop en Agentes*

*y Sistemas Inteligentes (WASI), 7mo Congreso Argentino de Ciencias de la Computación (CACIC)*, pages 1051–1062, El Calafate, Santa Cruz, October 2001. Universidad Nacional de la Patagonia Austral.

M. Capobianco, C. I. Chesñevar, and G. R. Simari. An argument-based framework to model an agent's beliefs in a dynamic environment. In *First International Workshop on Argumentation in Multi-Agent Systems (ArgMAS 2004), in conjunction with AAMAS 2004 (to appear)*, New York, USA, July 2004.

J. Doyle. A Truth Maintenance System. *Artificial Intelligence*, 12(3):231–272, November 1979.

John Fox and Simon Parsons. On using arguments for reasoning about action and values. In *Proceedings of the AAAI Spring Symposium on Qualitative*. Stanford, 1997.

Alejandro J. García and Guillermo R. Simari. Defeasible logic programming: An argumentative approach. *Theory and Practice of Logic Programming*, 4(1):95–138, 2004.

A. Kakas, R. Miller, and F. Toni. An argumentation framework for reasoning about actions and changes. In Michael Gelfond, Nicola Leone, and Gerald Pfeifer, editors, *Proceedings of the 5th International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR-99)*, volume 1730 of *LNAI*, pages 78–91, Berlin, December 2–4 1999. Springer.

Antonis Kakas, Rob Miller, and Francesca Toni. Planning with incomplete information. In Mirek Truszczynski Chitta Baral, editor, *Proceedings of the 8th International Workshop on Non-Monotonic Reasoning*, Breckenridge, Colorado, April 2000.

Vladimir Lifschitz. Foundations of logic programs. In G. Brewka, editor, *Principles of Knowledge Representation*, pages 69–128. CSLI Publications & FOLLI, 1996.

Pablo Noriega and Carles Sierra. Towards layered dialogical agents. In *Proc. of the ECAI'96 Workshop on Agents, Theories, Architectures and Languages (Budapest)*, pages 69–81, 1996.

John Pollock. *Cognitive Carpentry: A Blueprint for How to Build a Person*. MIT Press, 1995.

John Pollock. Implementing defeasible reasoning. *Workshop on Computational Dialectics*, 1996.

Guillermo R. Simari and Alejandro J. García. Actions and arguments: Preliminaries and examples. In *Proceedings of the VII Congreso Argentino en Ciencias de la Computación*, pages 273–283. Universidad Nacional de la Patagonia San Juan Bosco, El Calafate, Argentina, October 2001.

Guillermo R. Simari and Alejandro J. García. Using defeasible argumentation in progression and regression planning: Some preliminary explorations. In *Proceedings of the VIII Congreso Argentino en Ciencias de la Computación*, pages 898–907. Universidad de Buenos Aires, Buenos Aires, Argentina, October 2002.

Murray Shanahan. An abductive event calculus planner. *Journal of Logic Programming*, 44(1-3):207–240, 2000.

Guillermo R. Simari and Ronald P. Loui. A Mathematical Treatment of Defeasible Reasoning and its Implementation. *Artificial Intelligence*, 53:125–157, 1992.

Jordi Sabater, Carles Sierra, Simon Parsons, and Nick Jennings. Engineering executable agents using multi-context systems. *Journal of Logic and Computation (In-press)*, 2001.

Bart Verheij. *Rules, Reasons, Arguments: formal studies of argumentation and defeat*. PhD thesis, Maastricht University, Holland, December 1996.

Gerard A.W. Vreeswijk. Abstract argumentation systems. *Artificial Intelligence*, 90:225–279, 1997.

Daniel S. Weld. Recent advances in AI planning. *AI Magazine*, 20(2):93–123, 1999.