Using Defeasible Argumentation in Progression and Regression Planning

Some Preliminary Explorations

Guillemo R. Simari Alejandro J. García

grs@cs.uns.edu.ar agarcia@cs.uns.edu.ar
Laboratorio de Investigación y Desarrollo en Inteligencia Artificial
Departamento de Ciencias e Ingeniería de la Computación
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 is more involved than expected when combined with a defeasible argumentation formalism.

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.

Keywords: Artificial Intelligence, Planning, Defeasible Argumentation.

1 Introduction

The aim of this work is to study an argumentation-based formalism that an agent could use for constructing plans. In [9] we have introduced a formalism where agents have certain knowledge about their environment represented in Defeasible Logic Programming (DeLP) [4]. They also have a set of actions they are capable of executing in order to change the environment where they are performing their tasks. We have also shown [9] that action selection is more involved than expected when combined with an argumentation formalism.

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 here will be to show meaningful examples where those issues are exposed 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 manners of constructing plans introduce interesting details we will be considering progression and regression planning.

2 Knowledge Representation and Actions

The agent's knowledge will be represented by a knowledge base $\mathcal{K} = (\Phi, \Delta)$, where Φ will be a consistent set of facts, and Δ a set of defeasible rules. For example, below we have an agent's knowledge base. This agent, named *tom*, has a match, he is in an explosive place, and the knowledge base contains a pair of defeasible rules representing that *"if the agent wants to produce fire and it has a match, then there is a good reason for striking the match"* and *"being in an explosive place is a good reason for not striking the match"*.

$$\Phi = \{has(tom, match), at(tom, explosive\_place)\}$$
$$\Delta = \left\{ \begin{array}{l} strike(X, match) -\!\prec has(X, match), wants(X, fire) \\ \sim strike(X, match) -\!\prec has(X, match), at(X, explosive\_place) \end{array} \right\}$$

Notice that defeasible rules are the key element for introducing *defeasibility* [6] and they will be 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 provide reasons to believe in the consequent Head*" [10]. *Strong negation* is allowed in the head of defeasible rules, and hence may be used to represent contradictory knowledge.

The agent's knowledge base will be a restricted *Defeasible Logic Program* (DeLP). The results already obtained for such argumentation-based extension of logic programming will be used freely here. A brief description of DeLP follows. The interested reader is referred to [4, 2, 3] for details about DeLP.

In DeLP, a literal h is *warranted* if there exists a non-defeated *argument* $\mathcal{A}$ supporting h. An argument structure $\mathcal{A}$ for a literal h (denoted $\langle \mathcal{A}, h \rangle$) is a minimal and consistent set of defeasible rules that allows to infer h. In order to establish whether $\langle \mathcal{A}, h \rangle$ is a non-defeated argument, *argument rebuttals* or *counter-arguments* that could be *defeaters* for $\langle \mathcal{A}, h \rangle$ are considered, *i.e.*, counter-arguments that by some criterion, are preferred to $\langle \mathcal{A}, h \rangle$. 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* may appear, where each argument defeats its predecessor in the line (see the following example). Usually, each argument has more than one defeater and more than one argumentation line exists. Therefore, a tree of arguments called *dialectical tree* is constructed, where the root is $\langle \mathcal{A}, h \rangle$ and each path from the root to a leaf is an argumentation line. A *dialectical analysis* of this tree is used for deciding whether h is warranted.

Example 2.1 Consider the following knowledge base $\mathcal{K}= (\Phi, \Delta)$

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

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

Here, the literal p has the argument $\mathcal{A}$=$\{p -\!\prec b\}$ supporting it, $\mathcal{A}$ 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 it 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 Φ is trivially warranted, because no counter-argument can defeat a fact. □

Besides its knowledge base $\mathcal{K}$, an agent will have a set of actions Γ 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 [9] are recalled below.

Definition 2.1 [Action] An action A is an ordered triple $\langle \mathsf{P}, \mathsf{X}, \mathsf{C} \rangle$, where P is a set of literals representing preconditions for A, X is a consistent set of literals representing consequences of executing A, and 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 $\{not\ C_1, \ldots, not\ C_k\}$. □

Definition 2.2 [Applicable Action] Let $\mathcal{K} = (\Phi, \Delta)$ be an agent's knowledge base. Let Γ be the set of actions available to that agent. An action A in Γ, defined as before, is applicable if every precondition P_i in P has a warrant built from (Φ, Δ) and every constraint C_i in C fails to be warranted. □

Definition 2.3 [Action Effect] Let $\mathcal{K} = (\Phi, \Delta)$ be an agent's knowledge base. Let Γ be the set of actions available to that agent. Let A be an applicable action in Γ 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 Φ by X, i.e. $\Phi^{*\mathsf{X}} = \Phi^{*\{X_1,\ldots,X_n\}}$. Revision will consist of removing any literal in Φ that is complementary of any literal in X and then adding X to the resulting set. Formally:

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

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

Example 2.2 Let $\mathcal{K}$= (Φ, Δ) be an agent's knowledge base as defined in Example 2.1

$\Phi = \{a, b, c, \sim d\}$

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

And Γ the set of available actions containing only:

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

That action is applicable because every literal in the precondition set has a warrant, and no constraints in {t,∼t, w} are warranted (see Example 2.1). If the action is executed the set of facts becomes:

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

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

In [9] we have shown that the interaction between actions and the defeasible argumentation formalism is twofold. On one hand, as stated by Definition 2.2, 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 Φ) and then have a warrant for a literal h that has no warrant from the current knowledge base (Φ, Δ).

As we have shown in [9] this interaction produces a more powerful formalism. However, some new elements that are not present in traditional planning systems prompt for a deeper analysis.

3 Planning with argumentation

A simple formulation of a planning problem defines three inputs [13]: 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 current agent's representation of the world, and in our case it will be the set Φ. 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 Φ and it will execute a sequence of actions transforming Φ in Φ'. The agent's goal will be a set G of literals that the agent wants to be true in his world. The agent will satisfy its goals when through a sequence of actions it reaches some state Φ' where each literal of G is warranted. The planner will be in charge of obtaining the proper sequence of actions in advance.

3.1 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 C with I
REPEAT
- select an action which its preconditions hold in C
- simulate the action execution modifying C with the action effects
UNTIL $G \subseteq C$

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 C. If there are 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 Φ. In each step the planner will select an applicable action in the current state Φ, that is, an action $a=\langle \mathsf{P}, \mathsf{X}, \mathsf{C} \rangle$, where every precondition P_i in P has a warrant built from (Φ, Δ), and every constraint C_i in C fails to be warranted from (Φ, Δ). The action effect will be calculated with $\Phi^{*\mathsf{X}}$. A progression planner algorithm that uses defeasible argumentation follows:

Let Φ be the initial state and G the agent's goal.
REPEAT
- select an *applicable action* $a=\langle \mathsf{P}, \mathsf{X}, \mathsf{C}\rangle$
- simulate the *action effect* with $\Phi := \Phi^{*\mathsf{X}}$
UNTIL all literals in G are warranted from (Φ, Δ).

The main advantage of combining a progression planner with our approach is that no further considerations are necessary. However, 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 avoid this search problem. In the next section we will analyze the combination of regression planning with our approach.

3.2 Regression Planning

A regression planner searches backward from the goal state G to the initial state I. The outline of a typical regression planner that searches through the space of possible states is given below:

REPEAT
- select an action $a=\langle \mathsf{P}, \mathsf{X}, \mathsf{C}\rangle$, such that one consequent is in G. Formally: $\mathsf{X} \cap (G - I) \neq \emptyset$
- G is recomputed replacing X by P, that is, $G := G - \mathsf{X} \cup \mathsf{P}$
UNTIL $G \subseteq I$

Unfortunately, it is known that $\mathsf{X} \cap (G - I) \neq \emptyset$ 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 of the search, a selected action a could have some consequents that in the order of the plan execution could interact with an action already selected but to be executed after a in the plan. Consider the following example: the initial state is $I = \{c, e, f\}$, the goal state $G = \{a\}$, 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	X	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 an inverse order should be 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, and the action A_1 needs c as a precondition. Observe that c was "assumed" to be present along the search. The problem can be solved is the literal c is "protected". For example, one possibility is to add to the selection mechanism the condition $\overline{\mathsf{X}} \cap G = \emptyset$. That is,

REPEAT
- select an action a=$\langle \mathsf{P}, \mathsf{X}, \mathsf{C} \rangle$, such that $\mathsf{X} \cap (G - I) \neq \emptyset$ and $\overline{\mathsf{X}} \cap G = \emptyset$
- G is recomputed as $G := G - \mathsf{X} \cup \mathsf{P}$

UNTIL $G \subseteq I$

Although this is a well known problem, we recall it here because in our approach the proposed solution is not enough and some other consideration have to be done. We will consider first actions without constraints.

Observe that our approach use a deductive knowledge base (Φ, Δ), so a goal is achieve not just because is a member of a set. A literal g of the goal G is achieved if g is warranted from the current agent's knowledge base (Φ, Δ). In the rest of the paper we will use $w(G)$ to represent the subset of warranted literals of G, formally, $w(G) = \{l | l \in G$ and l is warranted $\}$. Thus, a planning problem will be solved when all the literals of G were warranted, that is, $G = w(G)$. The modified outline follows:

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

UNTIL $G = w(G)$

We will analyze the behavior of this last outline through meaningful examples. In all of the 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 3.1 [Argument Clipping]
Suppose that the agent has the following knowledge base: Φ= { e,x } and Δ= { $c \prec x$}. 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=$\langle \mathsf{P}, \mathsf{X}, \mathsf{C} \rangle$ is selected. The effect of A_2 is X= {b, $\sim$x}, so $\overline{\mathsf{X}} \cap G = \emptyset$ holds, and G becomes { e, c }. Since both literals are warranted, therefore a plan $[A_2, A_1]$ seems to be found.

However, if action A_2 is executed in the initial state, then literal x is removed from Φ. Therefore, no argument for c can be built and the action A_1 can not be executed. □

Example 3.1 shows that the literals to be protected are not only the ones in G. 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, that a new argument exists. This new argument becomes a defeater of an argument that was assumed undefeated.

Example 3.2 [Enabling a Defeater]
Suppose that the agent has the following knowledge base: Φ= { e,x,d } and Δ= { $(c -\!\prec d\})$, $(\sim c -\!\prec \sim x)$ }. In order to achieve the goal "a", action A_1 is selected first, and G becomes { b, c}. The literal c is warrant, 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=$\langle$P, X, C$\rangle$ is selected. The effect of A_2 is X= {b, $\sim$x}, so $\overline{\mathsf{X}} \cap G = \emptyset$ holds, and G becomes { e, c }. Since both literals are warranted, therefore a plan $[A_2, A_1]$ seems to be found.

However, if action A_2 is executed, literal $\sim x$ is added to Φ, 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 the action A_1 can not be executed. □

The Example 3.2 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 warrant of literals need to be protected. The following example shows a different situation where a warrant disappears because a supporting argument defeating a defeater disappears.

Example 3.3 [Disabling a Defeater]
Suppose that the agent has the following knowledge base: Φ= { e,x,g } and Δ= { $(c -\!\prec d\})$, $(d -\!\prec e\})$, $(\sim d -\!\prec e, f)$, $(f -\!\prec g)$, $(\sim f -\!\prec x)$}. In order to achieve the goal "a", action A_1 is selected first, and G becomes { b, c}. The literal c is warrant, 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=$\langle$P, X, C$\rangle$ is selected next, G becomes { e, c }, and a plan $[A_2, A_1]$ seems to be found. However, if action A_2 is executed, literal x is removed from Φ, and then the argument $\mathcal{D}$= { $\sim f -\!\prec x$} can not be obtained. The argument $\mathcal{C}$ is now undefeated and since $\mathcal{C}$ defeats $\mathcal{B}$, there is no warrant for c and the action A_1 can not be executed because c is needed as a precondition. □

As the reader has noticed, when an action is executed new literals can be added or deleted from Φ, and therefore, new defeaters could appear or disappear interfering somehow with the assumed warrants. As it was shown in the previous examples, this could cause that the planner selects an improper sequence of actions that can not 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 is proposed in the following section.

4 Protecting Warrants in Regression Planning

In DeLP an *argumentation line* starting in $\langle \mathcal{A}_0, h_0 \rangle$ is a sequence of arguments

$$[\langle \mathcal{A}_0, h_0 \rangle, \langle \mathcal{A}_1, h_1 \rangle, \langle \mathcal{A}_2, h_2 \rangle \langle \mathcal{A}_3, h_3 \rangle, \ldots]$$

where each element of the sequence $\langle \mathcal{A}_i, h_i \rangle$, $i > 0$, is a defeater of its predecessor $\langle \mathcal{A}_{i-1}, h_{i-1} \rangle$. Then, $\langle \mathcal{A}_0, h_0 \rangle$ becomes a *supporting* argument for h_0, $\langle \mathcal{A}_1, h_1 \rangle$ an *interfering* argument, $\langle \mathcal{A}_2, h_2 \rangle$ a supporting argument, $\langle \mathcal{A}_3, h_3 \rangle$ an interfering one, and so on.

Thus, an argumentation line $\Lambda = [\langle \mathcal{A}_0, h_0\rangle, \langle \mathcal{A}_1, h_1\rangle, \langle \mathcal{A}_2, h_2\rangle \langle \mathcal{A}_3, h_3\rangle, \ldots]$ can be split in two disjoint sets: The set Λ_S= { $\langle \mathcal{A}_0, h_0\rangle$, $\langle \mathcal{A}_2, h_2\rangle$, $\langle \mathcal{A}_4, h_4\rangle$, ... } of supporting arguments, and the set Λ_I= {$\langle \mathcal{A}_1, h_1\rangle$, $\langle \mathcal{A}_3, h_3\rangle$, ...} of interfering arguments. The warrant of a literal h_0 is obtained exploring all possible argumentation lines that starts with $\langle \mathcal{A}_0, h_0\rangle$, what in DeLP terminology is called a dialectical tree (see [4] for 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 [$\langle \mathcal{A}_0, h_0\rangle$,$\langle \mathcal{A}_1, h_1\rangle$,$\langle \mathcal{A}_2, h_2\rangle$,$\langle \mathcal{A}_3, h_3\rangle$,$\langle \mathcal{A}_4, h_4\rangle$]. On the one hand, if an action b selected after (but to be executed before) a delete one of the literals used in the supporting arguments { $\langle \mathcal{A}_0, h_0\rangle$, $\langle \mathcal{A}_2, h_2\rangle$, $\langle \mathcal{A}_4, h_4\rangle$}, then the warrant for p could disappear. On the other hand, is the selected action b adds a fact to the knowledge base, such that a new undefeated argument $\langle \mathcal{A}_i, h_i\rangle$ can be build and $\langle \mathcal{A}_i, h_i\rangle$ defeats any of the supporting arguments { $\langle \mathcal{A}_0, h_0\rangle$, $\langle \mathcal{A}_2, h_2\rangle$, $\langle \mathcal{A}_4, h_4\rangle$}, then the warrant for p could also disappear.

The first problem could be avoided collecting all the facts used in { $\langle \mathcal{A}_0, h_0\rangle$, $\langle \mathcal{A}_2, h_2\rangle$, $\langle \mathcal{A}_4, h_4\rangle$} and protecting them requiring that no action could delete those facts. The second one, ensuring that no new defeaters for the supporting arguments { $\langle \mathcal{A}_0, h_0\rangle$, $\langle \mathcal{A}_2, h_2\rangle$, $\langle \mathcal{A}_4, h_4\rangle$} can be obtained.

Finally observe that if a literal used in an interfering argument for p {$\langle \mathcal{A}_1, h_1\rangle$, $\langle \mathcal{A}_3, h_3\rangle$} 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 will be collected. This information will be used for avoiding in advance the selection of an improper action. Consider the following definitions:

Definition 4.1 Let h_0 be a warranted literal.

- $Protect(h_0)$ will be the set of all literals from Φ used in supporting arguments involved in the warrant of h_0.
- $PossAttack(h_0)$ will be the set of all points of attack of supporting arguments involved in the warrant of h_0. A literal l is a point of attack if $l \notin \Phi$

□

Definition 4.2 Let $\mathcal{K} = (\Phi, \Delta)$ be the agent's knowledge base, G the agent's goal, and $[A_1, A_2, \ldots, A_n]$ the selected actions by the regression planner. Let $\{h_1, h_2, \ldots, h_k\}$ be the set of literals that were assumed to be warranted for the selection of the actions $[A_1, A_2, \ldots, A_n]$. Then, we will define $Protect = \bigcup_{i=1..k} Protect(h_i)$ and $PossAttack = \bigcup_{i=1..k} PossAttack(h_i)$ □

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

REPEAT
- select an action a=⟨P, X, C⟩, such that

1) $\mathsf{X} \cap (G - w(G)) \neq \emptyset$
2) $\overline{\mathsf{X}} \cap \mathit{Protect} = \emptyset$
3) there is no new undefeated argument for a member of $\mathit{PossAttack}$ from $\Phi \cup \mathsf{X}$
- G is recomputed as $G = G - \mathsf{X} \cup \mathsf{P}$
- $\mathit{Protect}$ and $\mathit{PossAttack}$ are updated accordingly. UNTIL $G = w(G)$

Although this last solution averts the mentioned problems, it is not complete. Consider the example below that is a variation of Example 3.1

Example 4.1 [Action Selection]
Consider an agent with the goal $G = \{a\}$, and the actions:

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

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

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

Suppose that the agent has the following knowledge base: Φ= { e,x } and Δ= { $(c -\!\prec x)$ }. 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 $\mathit{Protect} = \{x\}$. Since literal b is not warranted, the planning process continues and action A_2=$\langle \mathsf{P}, \mathsf{X}, \mathsf{C} \rangle$ is considered. The action can not be selected because $\overline{\mathsf{X}} \cap \mathit{Protect} = \{x\}$. Therefore, no plan is found.

As the reader may notice 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 4.1 we propose the following solution. When an applicable action a can not be selected because one of its effects delete a protected literal that is necessary for the warrant of a literal c, then instead of simply discard the action, the planner will search for an other way of obtaining c and insert this subsidiary plan into the main plan.

5 Conclusions and Future Work

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

As showed above, the use of defeasible argumentation in progression planning is straightforward. However, regression planning deserves more attention. The combination of searching backwards with the generation of warrants for literals could produce unexpected results. Several examples to illustrate those problems were introduced, and solutions were proposed.

Future work include the analysis of other planning systems as Partial Order Planning and GraphPlan.

References

[1] 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.

[2] Alejandro J. García. *Defeasible Logic Programming: Definition, Operational Semantics and Parallelism*. PhD thesis, Computer Science Department, Universidad Nacional del Sur, Bahía Blanca, Argentina, December 2000.

[3] Alejandro J. García and Guillermo R. Simari. Parallel defeasible argumentation. *Journal of Computer Science and Technology Special Issue: Artificial Intelligence and Evolutive Computation. http://journal.info.unlp.edu.ar/*, 1(2):45–57, 1999.

[4] Alejandro J. García and Guillermo R. Simari. Defeasible logic programming: An argumentative approach. *Theory and Practice of Logic Programming*, 2002. accepted for publication.

[5] 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.

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

[7] John Pollock. Implementing defeasible reasoing. *workshop on Computation Dialectics*, 1996.

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

[9] 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. ISBN 987-96-288-6-1.

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

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

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

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