Defeasible Reasoning in Dynamic Domains

M. Capobianco G. R. Simari

Grupo de Investigación en Inteligencia Artificial (GIIA)
Departamento de Ciencias de la Computación
Universidad Nacional del Sur
e-mail: {mc,grs}@cs.uns.edu.ar

Abstract

The design of *intelligent agents* is a key issue for many applications. Since there is no universally accepted definition of intelligence, the notion of *rational agency* was proposed by Russell as an alternative for the characterization of *intelligent agency.*

A rational agent must have models of itself and its surroundings to use them in its reasoning. To this end, this paper develops a formalism appropriate to represent the knowledge of an agent. Moreover, if dynamic environments are considered, the agent should be able to observe the changes in the world, and integrate them into its existing beliefs. This motivates the incorporation of perception capabilities into our framework.

The abilities to perceive and act, essential activities in a practical agent, demand a timely interaction with the environment. Given that the cognitive process of a rational agent is complex and computationally expensive, this interaction may not be easy to achieve. To solve this problem, we propose inference mechanisms that rely on the use precompiled knowledge to optimize the reasoning process.

Keywords: knowledge representation, defeasible reasoning, rational agents.

1 Introduction

The design of *intelligent agents* is a key issue for many applications. Since there is no universally accepted definition of intelligence, the notion of *rational agency* was proposed by Russell [11] as an alternative for the characterization of *intelligent agency.* In short, an agent is said to be rational if it performs the *right* actions according to the information it possesses and the goals it wants to achieve.

A rational agent must have models of itself and its surroundings to use them in its reasoning [7]. To this end, this paper develops a formalism appropriate to represent the knowledge of our agent. When a dynamic environment is considered, the agent should be able to perceive the changes in the world and integrate them into its existing beliefs [7]. This involves to provide the agent with the capability of sensing its surroundings, and to define a way of incorporating the new observations into its knowledge. The specification of the former depends on the application domain, but it should be broad enough to comprise the input by a human operator. To accomplish the latter, we have adapted the knowledge representation system to handle perceptions properly.

The abilities to perceive and act, essential activities in practical agents, demand a timely interaction with the environment. Given that the cognitive process of rational agents is complex and computationally expensive, this interaction may not be easy to achieve. To solve this problem, we have defined inference mechanisms that rely on the use of precompiled knowledge to optimize the reasoning process.

The problem of optimizing a Knowledge Representation and Reasoning framework was analyzed by the authors in a prior work [3]. In that article, the basic idea was to maintain a repository of the conclusions previously computed to avoid repeating the reasoning process on the same input. The disadvantages of that formulation were twofold. Firstly, its applicability was restricted given that it only helped in the solution of recurrent queries. Secondly, the recorded information had to be updated each time the agent acquired a new observation; and the restoring operation was a demanding task. In this work, we have developed a new proposal that addresses those shortcomings.

The remainder of this paper is organized as follows. Section 2 redefines the system of *defeasible logic programming*, tailoring it for modeling the epistemic state of an agent in a dynamic domain. Next, section 3 focuses on the incorporation of perception mechanisms and section 4 describes how to use precompiled knowledge to speed-up the deduction process. Finally, section 5 states the conclusions.

2 Observation based DeLP

The formalism of *Defeasible Logic Programming* [5] (DeLP) combines the advantages of logic programming and *defeasible argumentation* [1]. In this framework, an *argumentation system* is used to decide between contradictory goals through a *dialectical analysis*.

Codifying the knowledge base of the agent by means of a DeLP program is a promising alternative, since it provides a good trade-off between expressivity and implementability. For this reason, we have chosen to use a modification of DeLP that incorporates perception abilities. This modification is called *Observation based* DeLP (ODeLP). Next, we define the ODeLP formalism, detailing the differences with respect to the original system.

The language of ODeLP is composed by a set of *observations* encoding the knowledge the agent has about the world, and a set of *defeasible rules* representing tentative information, *i.e.*, information that can be used if nothing is posed against it. In the followings definitions, we consider literals as atoms that may be preceded by the symbol "$\sim$" denoting classical negation.

Definition 2.1. (*Observation*) *An* observation *is a ground literal representing some fact about the world that the agent believes to be correct.* ■

Definition 2.2. (*Defeasible rule*) *A* defeasible rule *is an ordered pair, conveniently denoted as* `Head` < `Body`*, where* `Head` *is a ground literal and* `Body` *is a non-empty finite set of ground literals.* ■

[1]For a more detailed analysis of this concept, see [13, 8, 10, 2, 4].

Definition 2.3. (*Defeasible Logic Program*) *A* defeasible logic program (DLP) *is a finite set of observations and defeasible rules. In a* DLP $\mathcal{P}$, *we distinguish the set* Ψ *of observations and the set* Δ *of defeasible rules. The set* Ψ *must be non-contradictory, i.e., it cannot contain a literal and its complement. When required, we denote* $\mathcal{P}$ *as* $\langle\Psi, \Delta\rangle$. ■

Note that we only focus on propositional programs. As stated in [6], rules with variables are viewed as "schemata" that represent a set of ground instances.

Notice the difference between ODeLP and the DeLP language. A DeLP program consists of a set of *defeasible rules* and a set of *strict rules*. A strict rule is defined as an ordered pair, denoted as `Head` $\leftarrow$ `Body`. Facts are represented by strict rules with an empty body. Syntactically, the symbol "$\leftarrow$" is all that distinguishes strict from defeasible rules. However, they are semantically different: strict rules stablish a strong connection between its antecedent and its consequent. In ODeLP these rules are reduced to a set of observations. This simplifying assumption will help in understanding the problems related with a dynamic environment. The final goal of this research is to use full DeLP.

Having defined the concept of defeasible programs, we focus on the consequence operator for this programs. The inferences that can be obtained from a ODeLP program are ground literals. Accordingly, a *defeasible query* (or simply a query) is a defeasible rule "$\prec l$" with empty head and a ground literal l in its body.

Definition 2.4. (*Defeasible derivation*) *Let* $\mathcal{P}$ *be a* ODeLP *program and let* q *be a ground literal. A finite sequence of ground literals,* $s = q_1, q_2, \ldots, q_{n-1}, q$, *is said to be a* defeasible derivation *for* q *from* $\mathcal{P}$ *(abbreviated* $\mathcal{P} \mid\!\sim q$*) if for every* q_i, $1 \leq i \leq n$, *it holds that* $q_i \in \Psi$, *or* q_i *is a consequent of a defeasible rule* $r \in \Delta$, $r = q_i \prec l_1, \ldots, l_m$, *where* $l_1, \ldots, l_m$ *are ground literals previously occurring in* s. ■

The defeasible rules belonging to Δ play the role of inference rules in the derivation. Although the set Ψ must be non-contradictory, $\mathcal{P}$ may allow the defeasible derivation of complementary literals. Thus, the system should have a mechanism for deciding between them. In what follows, we characterize this mechanism.

In ODeLP, the existence of a defeasible derivation for a literal q is not enough to accept it. Answers to queries must be supported by arguments:

Definition 2.5. (*Argument Sub-argument*) *Given a* ODeLP *program* $\mathcal{P}$, *an* argument $\mathcal{A}$ *for a ground literal* q, *also denoted* $\langle\mathcal{A}, q\rangle$, *is a subset of the defeasible rules in* $\mathcal{P}$ *such that:*

1. *there exists a defeasible derivation for* q *from* $\Psi \cup \mathcal{A}$,
2. $\Psi \cup \mathcal{A}$ *is non-contradictory,*
3. $\mathcal{A}$ *is the minimal set with respect to set inclusion that satisfies the previous conditions.*

An argument $\langle\mathcal{A}_1, q_1\rangle$ *is a* sub-argument *of* $\langle\mathcal{A}_2, q_2\rangle$, *if* $\mathcal{A}_1 \subseteq \mathcal{A}_2$. ■

Example 2.1. In the following DLP, observations and defeasible rules are separated for the sake of clarity.

Ψ	Δ
`cat(tom).`	`has-tail(X)`$\prec$ `cat(X).`
`cat(sylvester).`	`~has-tail(X)`$\prec$ `cat(X), manx-cat(X).`
`cat(grace).`	`~social(X)`$\prec$ `aloof(X).`
`manx-cat(grace).`	`social(X)`$\prec$ `cat(X), young(X).`
`young(tom).`	`aloof(X)`$\prec$ `cat(X).`

From this program, the following arguments can be built:

- $\langle \mathcal{A}_1, \texttt{has-tail(grace)} \rangle$, where $\mathcal{A}_1 = \{\text{has-tail(grace)} \prec \text{cat(grace)}\}$.
- $\langle \mathcal{A}_2, \sim\texttt{has-tail(grace)} \rangle$, where $\mathcal{A}_2 = \{\sim\text{has-tail(grace)} \prec \text{cat(grace)}$, `manx-cat(grace)`$\}$
- $\langle \mathcal{A}_3, \sim \texttt{social(tom)} \rangle$, where $\mathcal{A}_3 = \{\sim\text{social(tom)} \prec \text{aloof(tom)}$, $\text{aloof(tom)} \prec \text{cat(tom)}\}$
- $\langle \mathcal{A}_4, \texttt{social(tom)} \rangle$, where $\mathcal{A}_4 = \{\text{social(tom)} \prec \text{cat(tom)}, \text{young(tom)}\}$

The first argument supports the conclusion that Grace has a tail, because she is a cat and normally cats have tails. The second one claims that Grace is tailless given that she belongs to a special breed of cats whose members do not have a tail. $\mathcal{A}_3$ asserts that Tom is not a social creature since Tom is a cat, and cats are aloof. On the other hand, $\mathcal{A}_4$ supports that Tom is social, considering that young cats are friendly animals.

To give an answer to a query q, the system tries to build an argument $\mathcal{A}$ for q, but arguments that contradict or attack $\mathcal{A}$ could also exist (as shown in the example above).

Definition 2.6. (*Counter-argument*) *An argument* $\langle \mathcal{A}_1, q_1 \rangle$ counter-argues *an argument* $\langle \mathcal{A}_2, q_2 \rangle$ *at a literal* q *if and only if there is a sub-argument* $\langle \mathcal{A}, q \rangle$ *of* $\langle \mathcal{A}_2, q_2 \rangle$ *such that the set* $\{q_1, q\}$ *is contradictory*[2]. ■

Informally, a query for a literal q succeeds if there exists an undefeated argument supporting q; that argument becomes a *justification*. To establish whether $\mathcal{A}$ is an undefeated argument we must analyze the *defeaters* of $\mathcal{A}$, *i.e.*, the counter-arguments prefered to $\mathcal{A}$ under a given criterion. Any partial order on the set of possible arguments can be used to formulate this criterion. In particular, we define the following one based on specificity [9, 5].

Definition 2.7. (ODeLP *Specificity*) *Let* $\mathcal{P}$ *be a* ODeLP *program and let Lit be the set of ground literals belonging to the signature of* $\mathcal{P}$. *An argument* $\langle \mathcal{A}_1, h_1 \rangle$ *is more specific than an argument* $\langle \mathcal{A}_2, h_2 \rangle$ *(denoted* $\langle \mathcal{A}_1, h_1 \rangle \succeq \langle \mathcal{A}_2, h_2 \rangle$*) if and only if for every* $H \subseteq Lit$ *it holds that* $H \cup \mathcal{A}_1 \mid\!\sim h_1$ *and* $H \not\mid\!\sim h_1$ *implies that* $H \cup \mathcal{A}_2 \mid\!\sim h_2$. $\langle \mathcal{A}_1, h_1 \rangle$ *is said to be strictly more specific than* $\langle \mathcal{A}_2, h_2 \rangle$ *(denoted* $\langle \mathcal{A}_1, h_1 \rangle \succ \langle \mathcal{A}_2, h_2 \rangle$*) if and only if* $\langle \mathcal{A}_1, h_1 \rangle \succeq \langle \mathcal{A}_2, h_2 \rangle$ *and* $\langle \mathcal{A}_2, h_2 \rangle \not\succeq \langle \mathcal{A}_1, h_1 \rangle$ ■

[2]We say that two literals are contradictory if they are complementary with respect to classical negation.

This syntactic comparison allows us to discriminate two conflicting arguments, prefering the argument with greater information content or less use of defeasible rules. Lets analyze definition 2.7. The condition $H \cup \mathcal{A}_1 \mathrel{|\!\sim} h_1$ normally holds with a nonempty set H, because arguments do not contain facts. In this case, the set H is said to *activate* $\mathcal{A}_1$. The expression $H \mathrel{|\!\not\sim} h_1$ is called the *non-triviality* requisite, because it forces the effective use of the set H for deriving h_1. Hence, the previous definition may be read as $\langle \mathcal{A}_1, h_1 \rangle$ is more specific than $\langle \mathcal{A}_2, h_2 \rangle$ if and only if for every set H which non-trivially activates $\langle \mathcal{A}_1, h_1 \rangle$ it holds that H non-trivially activates $\langle \mathcal{A}_2, h_2 \rangle$.

Example 2.2. Consider the arguments in example 2.1. $\langle \mathcal{A}_2$, `~has-tail(tom)`$\rangle$ is strictly more specific than $\langle \mathcal{A}_1$, `has-tail(tom)`$\rangle$. Applying definition 2.7, it holds that every subset of *Lit* that activates $\mathcal{A}_1$ also activates $\mathcal{A}_2$, but there are subsets (*e.g.*, {`cat(tom)`}) that activate $\mathcal{A}_2$ and do not activate $\mathcal{A}_1$.

Although a criterion is required, the notion of *defeat* can be independently formulated.

Definition 2.8. (*Defeater*) *An argument* $\langle \mathcal{A}_1, q_1 \rangle$ *defeats* $\langle \mathcal{A}_2, q_2 \rangle$ *at a literal* q*, if and only if there exists a sub-argument* $\langle \mathcal{A}, q \rangle$ *of* $\langle \mathcal{A}_2, q_2 \rangle$ *such that* $\langle \mathcal{A}_1, q_1 \rangle$ *counter-argues* $\langle \mathcal{A}_2, q_2 \rangle$ *at* q*, and either:*

1. $\langle \mathcal{A}_1, q_1 \rangle$ *is prefered to* $\langle \mathcal{A}, q \rangle$ *by the preference criterion (then* $\langle \mathcal{A}_1, q_1 \rangle$ *is a* proper defeater *of* $\langle \mathcal{A}_2, q_2 \rangle$*), or*

2. $\langle \mathcal{A}_1, q_1 \rangle$ *is unrelated to* $\langle \mathcal{A}, q \rangle$ *by the preference criterion (then* $\langle \mathcal{A}_1, q_1 \rangle$ *is a* blocking defeater *of* $\langle \mathcal{A}_2, q_2 \rangle$*).* ■

Since defeaters are arguments, there may be defeaters for the defeaters and so on. For this reason, a complete dialectical analysis is required to determine which arguments are ultimately accepted. This analysis consists of the construction and marking of the *dialectical tree*. As an output of the marking process, the undefeated arguments are labeled as `U-nodes`, and the defeated ones as `D-nodes`.

Definition 2.9. (*Dialectical Tree*) *Let* $\mathcal{A}$ *be an argument for* q*. A*dialectical tree *for* $\langle \mathcal{A}, q \rangle$*, denoted* $\mathcal{T}_{\langle \mathcal{A}, q \rangle}$*, is recursively defined as follows:*

1. *A single node labeled with an argument* $\langle \mathcal{A}, q \rangle$ *with no defeaters (proper or blocking) is by itself the dialectical tree for* $\langle \mathcal{A}, q \rangle$*.*

2. *Let* $\langle \mathcal{A}_1, q_1 \rangle, \langle \mathcal{A}_2, q_2 \rangle, \ldots, \langle \mathcal{A}_n, q_n \rangle$ *be all the defeaters (proper or blocking) for* $\langle \mathcal{A}, q \rangle$*. The dialectical tree for* $\langle \mathcal{A}, q \rangle$*,* $\mathcal{T}_{\langle \mathcal{A}, q \rangle}$*, is obtained by labeling the root node with* $\langle \mathcal{A}, q \rangle$*, and making this node the parent of the roots nodes for the dialectical trees corresponding to* $\langle \mathcal{A}_1, q_1 \rangle, \langle \mathcal{A}_2, q_2 \rangle, \ldots, \langle \mathcal{A}_n, q_n \rangle$*.* ■

Definition 2.10. (*Marking of the dialectical tree*) *Let* $\langle \mathcal{A}_1, q_1 \rangle$ *be an argument and* $\mathcal{T}_{\langle \mathcal{A}_1, q_1 \rangle}$ *its dialectical tree, then:*

1. *All the leaves in* $\mathcal{T}_{\langle \mathcal{A}_1, q_1 \rangle}$ *are marked as a* ***`U-node`****.*

2. *Let* $\langle \mathcal{A}_2, q_2 \rangle$ *be an inner node of* $\mathcal{T}_{\langle \mathcal{A}_1, q_1 \rangle}$*. Then* $\langle \mathcal{A}_2, q_2 \rangle$ *is marked as* ***`U-node`*** *if and only if every child of* $\langle \mathcal{A}_2, q_2 \rangle$ *is marked as a* ***`D-node`****. The node* $\langle \mathcal{A}_2, q_2 \rangle$ *is marked as a* ***`D-node`*** *if and only if it has at least a child marked as* ***`U-node`****.* ■

Example 2.3. The dialectical tree for $\langle \mathcal{A}_1, \texttt{has-tail(grace)} \rangle$ is composed by the argument $\langle \mathcal{A}_1, \texttt{has-tail(grace)} \rangle$ and its sole defeater $\langle \mathcal{A}_2, \sim\texttt{has-tail(grace)} \rangle$. Therefore, $\langle \mathcal{A}_2, \sim\texttt{has-tail(grace)} \rangle$ is marked as a `U-node` and $\langle \mathcal{A}_1, \texttt{has-tail(grace)} \rangle$ is marked as a `D-node`.

The notion of acceptable argumentation lines, required to avoid the so-called *fallacious argumentation* [12], is defined below. Let $\langle \mathcal{A}_0, q_0 \rangle$ be an argument, and let $\mathcal{T}_{\langle \mathcal{A}_0, q_0 \rangle}$ be its associated dialectical tree. Every path from the root $\langle \mathcal{A}_0, q_0 \rangle$ to a leaf $\langle \mathcal{A}_n, q_n \rangle$ in $\mathcal{T}_{\langle \mathcal{A}_0, q_0 \rangle}$, denoted $\lambda = [\langle \mathcal{A}_0, q_0 \rangle, \langle \mathcal{A}_1, q_1 \rangle, \ldots, \langle \mathcal{A}_n, q_n \rangle]$, is an argumentation line of $\mathcal{T}_{\langle \mathcal{A}_0, q_0 \rangle}$.

In each argumentation line $\lambda = [\langle \mathcal{A}_0, q_0 \rangle, \langle \mathcal{A}_1, q_1 \rangle, \ldots, \langle \mathcal{A}_n, q_n \rangle]$ the argument $\langle \mathcal{A}_0, q_0 \rangle$ is supporting the main query q_0, and every argument $\langle \mathcal{A}_i, q_i \rangle$ defeats its predecessor $\langle \mathcal{A}_{i-1}, q_{i-1} \rangle$. Thus, for $k \geq 0$, $\langle \mathcal{A}_{2k}, q_{2k} \rangle$ is a supporting argument for q_0 and $\langle \mathcal{A}_{2k+1}, q_{2k+1} \rangle$ is an interfering argument for q_0. In other words, every argument in the line supports q_0 or interferes with it. As a result, an argumentation line can be split in two disjoint sets: λ_S of supporting arguments, and λ_I of interfering arguments. Following, we define which argumentation lines are valid:

Definition 2.11. (*Acceptable Argumentation Line*) *Let* $\lambda = [\langle \mathcal{A}_0, q_0 \rangle, \langle \mathcal{A}_1, q_1 \rangle, \ldots, \langle \mathcal{A}_n, q_n \rangle]$ *be an argumentation line in* $\mathcal{T}_{\langle \mathcal{A}_0, q_0 \rangle}$. *We say that* λ *is an* acceptable argumentation line *if and only if:*

1. *The sets* λ_S *and* λ_I *are both non-contradictory sets of arguments,*
2. *No argument* $\langle \mathcal{A}_j, q_j \rangle$ *in* λ *is a sub-argument of an earlier argument* $\langle \mathcal{A}_i, q_i \rangle$ *in* λ *(*$i < j$*).* ■

An *acceptable dialectical tree* is defined in turn as a tree where every argumentation line is *acceptable*. Finally, the notion of justification follows:

Definition 2.12. (*Justification*) *Let* $\mathcal{A}$ *be an argument for a literal* q*, and let* $\mathcal{T}_{\langle \mathcal{A}, q \rangle}$ *be its associated acceptable dialectical tree.* $\mathcal{A}$ *is a* justification *for* q *if and only if the root of* $\mathcal{T}_{\langle \mathcal{A}, q \rangle}$ *is marked as a* `U-node`. ■

3 Modeling an agent's epistemic state

This section establishes the foundations for the use of the ODeLP system in defining the set of the agent's beliefs. Next, it details the perception capabilities added in the formalism and deals with its associated problems.

3.1 Using the ODeLP system

The ODeLP system exhibits interesting properties when used as a Knowledge Representation and Reasoning system. In contrast with other models, no mechanism to enforce consistency is required. The ODeLP program $\mathcal{P}$ representing the knowledge base of the agent is able to express the following doxastic attitudes with respect to a ground literal q in the signature of $\mathcal{P}$.

- Believe that q is true.
- Believe that q is false, *i.e.*, believe in $\overline{q}$, where $\overline{q}$ means the complement of q with respect to classical negation.
- Neither believe that q is true nor that it is false.

To formalize these states, we adopt a *metalanguage* based on a modal operator $\mathcal{B}$ standing for "the agent believes". This modal language consists of expressions of the form $\mathcal{B}q$, where q is a ground literal belonging to the signature of the program. In the semantics of this language, $\mathcal{B}q$ is true if and only if there is a justification for q from the ODeLP program encoding the knowledge of the agent, *i.e.*, if there exists an argument $\mathcal{A}$ such that $\mathcal{A}$ is a justification for q.

Based on this, we say that the agent believes in q when $\mathcal{B}q$ is true, does not believe in q when $\mathcal{B}\,\overline{q}$ is true, and is undecided on q when neither $\mathcal{B}q$ nor $\mathcal{B}\,\overline{q}$ are true. Note that for any ODeLP program $\mathcal{P}$, $\mathcal{B}q$ being true implies the falsity of $\mathcal{B}\,\overline{q}$: an agent cannot believe a literal and its complement. Another interesting property involves the certainty of the literals present in the observation set Ψ. It can be shown that for every literal $q \in \Psi$, it holds $\mathcal{B}q$. This is consistent with definition 2.1, in which observations are facts the agent believes.

3.2 Handling perceptions

To act in a dynamic world, our agent must be able to constantly update its beliefs: acquire new ones and revise or give up old ones. To do this, the agent must be able to perceive. However, the acquisition of non-certain beliefs is a problem associated with perception. To deal with it, we assume the perception mechanism to be flawless. This prevents situations where false inferences are obtained. Albeit we realize that this assumption needs not to be veridical, there are many interesting domains where this is a reasonable precondition

The task of perceiving can be carried out by a mechanism that detects the changes in the world and reports the literals representing those changes. This mechanism depends on the particular application domain, and its definition is not addressed in this paper. The perceived literals are added to the knowledge of the agent, into the set of observations Ψ. Notice that if new facts are blindly added to Ψ, it may become inconsistent.

Example 3.1. Suppose that ~`young(tom)` is to be added to the ODeLP program in example 2.1. (Tom may have become a grown-up cat). This contradicts the existing observation that tom is young.

We avoid this situation using a revision function that controls the updating of Ψ when new information is added. Prior to the addition of a new literal α to Ψ, this function removes the element of Ψ that contradicts α, if one exists.

In this revision, we apply an implicit criterion that favors new perceptions. This stems from the fact that given our initial assumption, both of the observations in disagreement were correct at the time of their incorporation. Accordingly, the only reason for the conflict to arise is a change in the state of world, and the new fact should be preferred over the old one.

Definition 3.1. (*Revision*) *Let $\mathcal{P} = \langle \Psi, \Delta \rangle$ be a* ODeLP *program and α an observation. The* revision *of Ψ by α, denoted as $\Psi{*}\alpha$, is defined as follows: $\Psi{*}\alpha = (\Psi - \{\overline{\alpha}\}) \cup \{\alpha\}$* ■

Continuing with the previous example, we can see how a revision performed over the set of observations modifies the set of beliefs. In a sense, the agent can change its previous picture of the world when faced with new information.

Example 3.2. Before the revision of Ψ with respect to ~`young(tom)`, the conclusion `social(tom)` was inferred by the system. This conclusion is withdrawn with the elimination of `young(tom)`, and the argument $\langle \mathcal{A}_3, \sim social(tom) \rangle$ (see example 2.1) becomes a justification for ~`social(tom)`.

4 Introducing Dialectical Bases

As mentioned in the introduction, integrating precompiled knowledge may help to optimize the inference process. In this section, we address how to build this component and how to use it to speed-up reasoning in ODeLP. We maintain a repository containing every possible argument that could be built from a given set Δ of defeasible rules. This repository should also keep the defeat relation between these arguments. When the reasoning process starts over a certain ODeLP program $\mathcal{P} = \langle \Psi, \Delta \rangle$, the system uses the precompiled arguments that are valid in this situation, *i.e.*, those which can be constructed from the set Ψ. This prevents the construction of the arguments and the search for their defeaters that takes place when a query is being solved.

It may be argued that keeping track of every potential argument for a given program is costly. However, this task is performed only once since this structure is independent from the current set of perceptions, and it does not have to be rebuilt or modified every time Ψ changes. In what follows, we formalize this idea.

Definition 4.1. (*Hypothetical Argument*) *Let Δ be a set of defeasible rules. A subset $\mathcal{A}$ of Δ is said to be* hypothetical argument *for a literal q, also denoted $\langle \mathcal{A}, q \rangle_h$, if there exists a consistent subset Φ of the literals in Δ, such that $\langle \mathcal{A}, q \rangle$ is an argument with respect to $\mathcal{P} = \langle \Phi, \Delta \rangle$.* ■

In the definition above, the set Φ represents a possible state of the world in which the hypothetical argument can be constructed. It is clear that hypothetical arguments depend only on the set of defeasible rules of the program. Nevertheless, not every subset of the defeasible rules is a hypothetical argument: some restrictions must be satisfied. The next proposition helps to find the hypothetical arguments in a certain ODeLP program.

Definition 4.2. (*Base*) *Let $\mathcal{A}$ be a set of defeasible rules, and let heads($\mathcal{A}$) (respectively bodies($\mathcal{A}$)) denote the literals occurring in the head (respectively body) of a defeasible rule in $\mathcal{A}$. A set $\mathcal{G}(\mathcal{A})$ of ground literals is said to be* the base of $\mathcal{A}$ *if and only if $\mathcal{G}(\mathcal{A}) = bodies(\mathcal{A}) - heads(\mathcal{A})$.* ■

Proposition 1. Let $\mathcal{P} = \langle \Psi, \Delta \rangle$ be an ODeLP program, $\mathcal{A}$ be a subset of Δ and q a literal such that:

1. $\mathcal{G}(\mathcal{A}) \cup \mathcal{A} \mid\!\sim q$,

2. the set of literals occurring in the rules in $\mathcal{A}$ is consistent, and

3. $\mathcal{A}$ is the minimal set with respect to set inclusion that fulfills the previous conditions.

Then $\mathcal{A}$ is a hypothetical argument for q based on Δ. ■

To understand the proposition above, consider a subset $\mathcal{A}$ of Δ that meets the enumerated conditions. If we take $\Phi = \mathcal{G}(\mathcal{A})$, it can be shown that $\mathcal{A}$ is an argument with respect to $\langle \Phi, \Delta \rangle$ (This satisfy the existence condition in definition 4.1.). Consequently, the computation of hypothetical arguments can be reduced to finding subsets of Δ based on this proposition.

We need to define how the defeat relation among these elements is stored. To this purpose, we overload the term *counter-argues*, using it for hypothetical arguments too. A hypothetical argument $\langle \mathcal{A}_1, q_1 \rangle_h$ *counter-argues* another hypothetical argument $\langle \mathcal{A}_2, q_2 \rangle_h$ at a literal q if and only if there is a sub-argument $\langle \mathcal{A}, q \rangle_h$ of $\langle \mathcal{A}_2, q_2 \rangle_h$ such that $\{q_1, q\}$ is contradictory. This kind of counter-arguments represents only a potential attack between the arguments in contest. It might be the case that these arguments cannot co-exists in any scenario.

To check whether a hypothetical argument defeats one of its counter-arguments, the criterion used to compare pairs of arguments must be adapted to pairs of hypothetical arguments. In particular, we have redefined specificity (definition 2.7) to compare arguments independently from the set of observations. Note that the meaningful activation sets of an argument $\mathcal{A}$ must be subsets of the literals present in the rules of $\mathcal{A}$.

Definition 4.3. (*Precompiled Specificity*) *A hypothetical argument $\langle \mathcal{A}_1, h_1 \rangle_h$ is more specific than a hypothetical argument $\langle \mathcal{A}_2, h_2 \rangle_h$ (denoted $\langle \mathcal{A}_1, h_1 \rangle \succeq \langle \mathcal{A}_2, h_2 \rangle$) if and only if for every $H \subseteq$ literals($\mathcal{A}_1$), it holds that $H \cup \mathcal{A}_1 \mid\!\sim h_1$, and $H \not\mid\!\sim h_1$ implies $H \cup \mathcal{A}_2 \mid\!\sim h_2$. $\langle \mathcal{A}_1, h_1 \rangle_h$ is said to be strictly more specific than $\langle \mathcal{A}_2, h_2 \rangle_h$ (denoted $\langle \mathcal{A}_1, h_1 \rangle_h \succ \langle \mathcal{A}_2, h_2 \rangle_h$) if and only if $\langle \mathcal{A}_1, h_1 \rangle_h \succeq \langle \mathcal{A}_2, h_2 \rangle_h$ and $\langle \mathcal{A}_2, h_2 \rangle_h \not\succeq \langle \mathcal{A}_1, h_1 \rangle_h$* ■

The notion of defeat between hypothetical arguments can be defined in an analogous way to definition 2.8. The definition below describes the notion of dialectical bases. This component subsumes the precompiled knowledge of the agent, according to the previous discussion.

Definition 4.4. (*Dialectical Base*) *Let $\mathcal{P} = \langle \Psi, \Delta \rangle$ be a* ODeLP *program. The tuple $(\mathbb{A}, D_p, D_b)$ is said to be the* dialectical base *of $\mathcal{P}$ with respect to Δ, denoted as $\mathbb{B}_\Delta$, if and only if:*

1. *$\mathbb{A}$ is a set of hypothetical arguments, such that $\mathcal{A} \in \mathbb{A}$ if and only if $\mathcal{A}$ is based on Δ.*

2. *D_p and D_b are relations over the elements of $\mathbb{A}$, such that for every pair $(\mathcal{A}_1, \mathcal{A}_2)$, it holds that $\mathcal{A}_2$ is a proper (resp. blocking) defeater of $\mathcal{A}_1$, if and only if $(\mathcal{A}_1, \mathcal{A}_2)$ belongs to D_p (resp. D_b).* ■

The dialectical base of a ODeLP program can be constructed after the knowledge of the agent is encoded in a ODeLP program $\mathcal{P}$. First we generate the set $\mathbb{A}$, including in it

Algorithm 4.1. Inference process

input: $\mathcal{P} = \langle \Psi, \Delta \rangle$, q
output: $\langle \mathcal{A}_1, q \rangle_h$ (a justification for q, if any)

```
For every hypothetical argument $\langle \mathcal{A}_1, q \rangle_h$ in $\mathbb{A}$ such that
acceptable($\mathcal{A}_1$,$\mathcal{P}$)
    state := undefeated
    For every $\mathcal{A}_2$ in $\mathbb{A}$ such that $(\mathcal{A}_1, \mathcal{A}_2) \in D_p$ or $(\mathcal{A}_1, \mathcal{A}_2) \in D_p$
    and acceptable($\mathcal{A}_2$,$\mathcal{P}$)
        if state($\mathcal{A}_2$, $\mathcal{P}$, $\emptyset$, $\{\mathcal{A}_1\}$) = undefeated
            then state := defeated
    if state = undefeated
        then return($\langle \mathcal{A}_1, q \rangle_h$)
```

every subset of the defeasible rules in $\mathcal{P}$ satisfying the properties stated in proposition 1. Next, we compute the defeat relation among the elements of $\mathbb{A}$ using the precompiled comparison criterion. For every pair $(\mathcal{A}_1, \mathcal{A}_2)$, such that $\mathcal{A}_1, \mathcal{A}_2 \in \mathbb{A}$ and $\mathcal{A}_2$ is a proper (resp. blocking) defeater for $\mathcal{A}_1$, we add $(\mathcal{A}_1, \mathcal{A}_2)$ to D_p (resp. D_b). When implementing this framework, appropriate data structures must be chosen to optimize the creation and use of the dialectical base.

Suppose the system is faced with a query q with respect to a ODeLP program $\mathcal{P} = \langle \Psi, \Delta \rangle$. The traditional procedure starts by building an argument $\mathcal{A}_1$ for q from the rules in $\mathcal{P}$. Then it looks for the defeaters of $\mathcal{A}_1$ that may prevent $\mathcal{A}_1$ from becoming a justification for q. Following, the state of $\mathcal{A}_1$ is decided based on the condition of its defeaters (see definition 2.10).

If $\mathbb{B}_\Delta$ is used, the inference process may be carried out as stated in algorithm 4.1. Since every feasible argument is already recorded in $\mathbb{B}_\Delta$, there is no need of constructing the arguments for q nor its defeaters. The system selects the hypothetical arguments for q that constitute arguments based on $\langle \Psi, \Delta \rangle$ (*i.e.*, that are valid for the particular Ψ under consideration). To this purpose, every $\langle \mathcal{A}_1, q \rangle_h \in \mathbb{B}_\Delta$ is checked using the function `acceptable(`$\mathcal{A}_1$`,`$\mathcal{P}$`)` (see algorithm 4.2). Next, each acceptable hypothetical argument $\langle \mathcal{A}_1, q \rangle_h$ is analyzed to see if it is a justification for q. This task is addressed using the relations D_p and D_b to look for the defeaters of $\mathcal{A}_1$. The `state` function depicted in figure 4.3 decides whether they are defeated or not, and this information is used to determine the state of $\mathcal{A}_1$ according to definition 2.10.

The `state` algorithm takes as input an argument $\mathcal{A}_1$, a ODeLP program $\mathcal{P}$ such that $\mathcal{A}_1$ is based on $\mathcal{P}$, and the interference and support argumentative lines up to this point, `IL` and `SL`. It works in a similar manner to procedure 4.1, analyzing the defeaters of $\mathcal{A}_1$ to define its state. However, one more condition must be met: for a defeater to be satisfactory, it must also comply the rules established in definition 2.11, regarding acceptability of the argumentative lines. The function `valid` tests these conditions.

Algorithm 4.2. Acceptable

input: $\mathcal{A}_1$, $\mathcal{P} = \langle\Psi, \Delta\rangle$
output: acceptable

```
acceptable := false
if base($\mathcal{A}_1$) $\subseteq \Psi$ and consistent($\mathcal{A}_1$,$\mathcal{P}$) and minimal($\mathcal{A}_1$,$\mathcal{P}$)
    then acceptable := true
```

Algorithm 4.3. State

input: $\mathcal{A}_1$, $\mathcal{P} = \langle\Psi, \Delta\rangle$, IL, SL
output: state

```
state := undefeated
For every $\mathcal{A}_2$ in $\mathbb{A}$ such that (($\mathcal{A}_1$,$\mathcal{A}_2$) $\in D_p$ or ($\mathcal{A}_1$,$\mathcal{A}_2$) $\in D_p$) and
acceptable($\mathcal{A}_2$,$\mathcal{P}$) and valid($\mathcal{A}_2$,$IL$,$SL$)
    if $\mathcal{A}_1$ is a supporting argument and state($\mathcal{A}_2$,$\mathcal{P}$,$IL$,$SL \cup \{\mathcal{A}_1\}$)
    = undefeated
        then state := defeated
    if $\mathcal{A}_1$ is an interfering argument and state($\mathcal{A}_2$,$\mathcal{P}$,$IL \cup \{\mathcal{A}_1\}$,$SL$)
    = undefeated
        then state := defeated
return(state)
```

5 Conclusions

We have developed a framework for representing the epistemic state of an agent, adapting the DeLP system for this task. The expressiveness of the defined language allows the description of complex domains. This formalism also provide mechanisms to acquire information perceptually, making the agent adaptable to a dynamic world.

The use of precompiled knowledge can improve the performance of argument-based systems. For this reason, we have defined the notion of dialectical bases and discussed the main issues in the integration of this component into ODeLP.

Solid theoretical foundations of agent design should be based on proper formalisms for knowledge representation and reasoning [1]. The incorporation of our framework into an agent architecture results in a model with interesting theoretical and practical features.

References

[1] Baral, C., and Gelfond, M. Reasoning Agents in Dynamic Domains. To be published.

[2] Bondarenko, A. G., Dung, P. M., Kowalski, R. A., and Toni, F. An Abstract, Argumentation-Theoretic Approach to Default Reasoning. *Artificial Intelligence 93*, 1–2 (1995), 63–101.

[3] Capobianco, M., Chesñevar, C. I., and Simari, G. R. A Formalization of Dialectical Bases for Defeasible Logic Programming. In *Proceedings of the VI Internacional Congress of Informatics Engineering* (Apr. 2000).

[4] Dung, P. M. On the Acceptability of Arguments and its Fundamental Role in Nonmonotonic Reasoning and Logic Programming and n-Person Games. *Artificial Intelligence 77*, 2 (1995), 321–357.

[5] García, A. J. La Programación en Lógica Rebatible: su definición teórica y computacional. Master's thesis, Departamento de Ciencias de la Computación, Universidad Nacional del Sur, Bahía Blanca, Argentina, June 1997.

[6] Lifschitz, V. Foundations of Logic Programming. In *Principles of Knowledge Representation*, G. Brewka, Ed. CSLI Publications, 1994, pp. 69–127.

[7] Pollock, J. L. The Phylogeny of Rationality. *Computational Intelligence 17* (1993), 568–588.

[8] Pollock, J. L. *Cognitive Carpentry: a blueprint for how to build a person.* Bradford/MIT Press, 1995.

[9] Poole, D. L. On the Comparison of Theories: Preferring the Most Specific Explanation. In *Proceedings of the Ninth International Joint Conference on Artificial Intelligence* (1985), IJCAI, pp. 144–147.

[10] Prakken, H., and Sartor, G. A System for Defeasible Argumentation with Defeasible Priorities. In *Proceedings of the International Conference on Formal and Applied Practical Reasoning* (1996), Springer Verlag, pp. 510–524.

[11] Russell, S. J. Rationality and Intelligence. *Artificial Intelligence 94*, 1–2 (1997), 57–77.

[12] Simari, G. R., Chesñevar, C. I., and García, A. J. The Role of Dialectics in Defeasible Argumentation. In *Proceedings of the XIV Conferencia Internacional de la Sociedad Chilena para Ciencias de la Computación* (Nov. 1994), pp. 111–121. http://cs.uns.edu.ar/giia.html.

[13] Simari, G. R., and Loui, R. P. A Mathematical Treatment of Defeasible Reasoning and its Implementation. *Artificial Intelligence 53*, 1–2 (1992), 125–157.