An application of Defeasible Logic Programming to Decision Making in a Robotic Environment

Edgardo Ferretti[1], Marcelo Errecalde[1],
Alejandro J. García[2,3], and Guillermo R. Simari[3]

[1] Laboratorio de Investigación y Desarrollo en Inteligencia Computacional
Universidad Nacional de San Luis, Argentina.
{ferretti, merreca}@unsl.edu.ar
[2] Consejo Nacional de Investigaciones Científicas y Técnicas (CONICET)
[3] Department of Computer Science and Engineering
Universidad Nacional del Sur, Bahía Blanca, Argentina.
{ajg, grs}@cs.uns.edu.ar

Abstract. Decision making models for autonomous agents have received increased attention, particularly in the field of intelligent robots. In this paper we will show how a Defeasible Logic Programming approach with an underlying argumentation based semantics, could be applied in a robotic domain for knowledge representation and reasoning about which task to perform next. At this end, we have selected a simple application domain, consisting of a micro-world environment using real and simulated robots for cleaning tasks.

1 Introduction

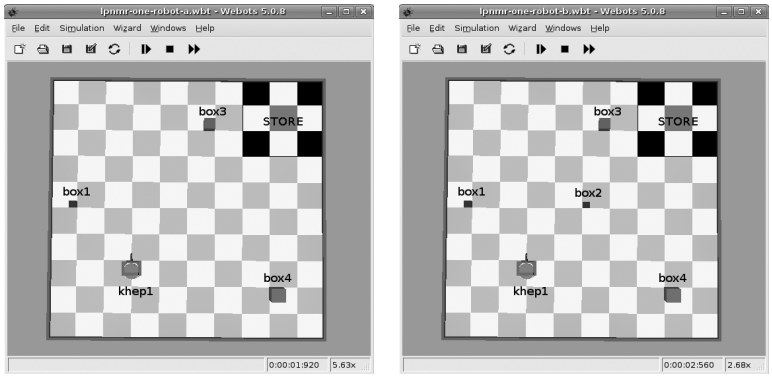
In this paper we will show how a Defeasible Logic Programming approach could be applied in a robotic domain for knowledge representation and reasoning about which task to perform next. At this end, we have selected a simple application domain, consisting of a micro-world environment using real and simulated robots for cleaning tasks. We use the *Khepera* 2 robot [1], a miniature mobile robot ideal for this kind of experimentation. We also use a professional simulator (see Fig. 1) called *Webots* [2], which allows behavior simulation prior to physical experimentation with the robot.

The experimental environment (see Fig. 1(a)) is a square arena of 100 units per side which is conceptually divided into square cells of 10 units per side each. There is a global camera which provides the necessary information to perform their activities. The *store* is a 30×30 units square on the top-right corner and represents the target area where boxes should be transported. There are boxes of three different sizes (*small*, *medium* and *big*) spread over the environment.

As the robot is not able of measuring the state of its battery, it cannot perform a globally optimized task. In this way, the robot will reason about which box is more convenient to select next trying to minimize the time spent in moving boxes. To reason, the robot will use perceptual information about the boxes and its preferences (represented with defeasible rules). For example, the robot could

prefer the smallest box, or the nearest one, or the box that it is nearest to the store. As we will show below, arguments for and against selecting a box will be considered to select the more appropriate one.

A robot capable of solving this kind of problems must at least address the following issues: to perceive the surrounding world, to decide which goal to reach and to have the capabilities for reaching this goal. Several architectures providing the agents with these skills have been proposed [3–5]. In this work, we only consider the necessary reasoning processes to make decisions about which is the most suitable box to be transported by the robots. We will not address the low-level aspects related to sensorial perception and the implementation of low-level actions for the Khepera robots, because they have been presented elsewhere [6].



(a) (b)

Fig. 1. Two possible different environments

2 Knowledge Representation and Defeasible Reasoning

Figure 1(a) shows an example with one robot ($khep1$) and three boxes to be carried: a small one ($box1$) near to the robot, $box3$ that is medium size and it is near to the store, and $box4$ that is big and it is far from both, robot and store. Taking into account its preferences the robot will consider reasons for and against selecting each box. We will refer to these reasons as *arguments*.

For example, there is an argument *for* selecting $box3$ because *"it is near to the store"* but there is an argument *against* selecting $box3$ because $box1$ *"is near to the robot and it is smaller than* $box3$*."* As it will be shown below a dialectical analysis involving arguments and counter-arguments will be performed to decide which argument prevails. In this case $box1$ will be chosen, because it is the smallest box near to the robot. Since the environment is dynamic, when it changes new arguments could be generated and others could be invalidated.

Thus, the robot might select different boxes in different circumstances. For instance, let us consider Fig. 1(b), that differs from Fig. 1(a) in that there is one more small box ($box2$) in the environment. Here, in the new situation, the robot $khep1$ will choose $box2$ because it has a new argument against selecting $box1$: *"there is another small box ($box2$) that it is nearer to the store than $box1$."*

The robot's knowledge about the environment and its preferences for selecting a box will be represented using Defeasible Logic Programming (DeLP) a formalism that combines logic programming and defeasible argumentation (for a detailed presentation see [7]). In DeLP, knowledge is represented using facts, strict rules or defeasible rules. *Facts* are ground literals representing atomic information or the negation of atomic information using the strong negation "$\sim$". *Strict Rules*, are denoted $L_0 \leftarrow L_1, \ldots, L_n$, where the *head* L_0 is a ground literal and the *body* $\{L_i\}_{i>0}$ is a set of ground literals. In the same way, *Defeasible Rules*, are denoted $L_0 \prec L_1, \ldots, L_n$, where the *head* L_0 is a ground literal and the *body* $\{L_i\}_{i>0}$ is a set of ground literals. In this work, facts will be used for representing perceptual information about the environment, (*e.g.*, $box(box2)$ or $near(box2, store)$), strict rules will be used for representing non-defeasible information (*e.g.*, $\sim far(box1, khep1) \leftarrow near(box1, khep1)$), and defeasible rules will be used for representing tentative reasons for (or against) selecting a box (*e.g.*, $choose(X) \prec small(X)$). The symbol "$\prec$" distinguishes a defeasible rule from a strict one with the pragmatic purpose of using a defeasible rule to represent defeasible knowledge, *i.e.*, tentative information.

A Defeasible Logic Program $\mathcal{P}$ is a set of facts, strict rules and defeasible rules. When required, $\mathcal{P}$ is denoted (Π, Δ) where $\Pi = \Pi_f \cup \Pi_r$, distinguishing the subset Π_f of facts, strict rules Π_r and the subset Δ of defeasible rules. Observe that strict and defeasible rules are ground following the common convention [9]. Some examples will use "schematic rules" with variables. As usual in Logic Programming, variables are denoted with an initial uppercase letter.

Strong negation is allowed in the head of program rules, and hence may be used to represent contradictory knowledge. From a program (Π, Δ) contradictory literals could be derived, however, the set Π (which is used to represent non-defeasible information) must possess certain internal coherence. Therefore, Π has to be non-contradictory, *i.e.*, no pair of contradictory literals can be derived from Π. Given a literal L the complement with respect to strong negation will be denoted $\overline{L}$ (*i.e.*, $\overline{a} = \sim a$ and $\overline{\sim a} = a$). To deal with contradictory and dynamic information, in DeLP, *arguments* for conflicting pieces of information are built and then compared to decide which one prevails. The prevailing argument provides a *warrant* for the information that it supports (A DeLP interpreter satisfying the semantics of [7] is accessible online at http://lidia.cs.uns.edu.ar/DeLP).

In DeLP a literal L is *warranted* from (Π, Δ) if a non-defeated argument $\mathcal{A}$ supporting L exists. An *argument* for a literal L, denoted $\langle \mathcal{A}, L \rangle$, is a minimal set of defeasible rules $\mathcal{A} \subseteq \Delta$, such that $\mathcal{A} \cup \Pi$ is non-contradictory and there is a derivation for L from $\mathcal{A} \cup \Pi$. To establish if $\langle \mathcal{A}, L \rangle$ is non-defeated, *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, defeaters for them may exist, and defeaters for these defeaters, and so on. Thus, a sequence of arguments called *argumentation line* is constructed, where each argument defeats its predecessor in the line. Given a query Q there are four possible answers: YES, if Q is warranted; NO, if the complement of Q is warranted; UNDECIDED, if neither Q nor its complement are warranted; and UNKNOWN, if Q is not in the language of the program.

3 Robot Decision Making: A Simple Example

In this section we describe the components used by the robot to decide which box to transport next. Consider the situation depicted in Fig. 1(b). The knowledge of the robot, referring to this particular scenario, will be represented with the defeasible logic program $\mathcal{P} = (\Pi, \Delta)$ shown in Fig. 2.

$robot(khep1)$
$self(khep1)$
$box(box1)$
$box(box2)$
$box(box3)$
$box(box4)$
$small(box1)$
$small(box2)$
$medium(box3)$
$big(box4)$ (1)
$near(box1, khep1)$ (2)
$near(box2, khep1)$ (3)
$near(box2, store)$ (4)
$near(box3, store)$ (5)
$far(box1, store)$ (6)
$far(box3, khep1)$ (7)
$far(box4, store)$ (8)
$far(box4, khep1)$ (9)

(a) Π_f

$\sim near(X, O) \leftarrow far(X, O)$ (10)
$smaller(X, Y) \leftarrow small(X), medium(Y)$ (11)
$smaller(X, Y) \leftarrow small(X), big(Y)$ (12)
$smaller(X, Y) \leftarrow medium(X), big(Y)$ (13)
$\sim smaller(X, Y) \leftarrow same\_size(X, Y)$ (14)
$same\_size(X, Y) \leftarrow small(X), small(Y)$ (15)
$same\_size(X, Y) \leftarrow medium(X), medium(Y)$ (16)
$same\_size(X, Y) \leftarrow big(X), big(Y)$ (17)
(18)

(b) Π_r

$choose(X) \prec near(X, store)$ (19)
$choose(X) \prec self(Z), near(X, Z)$ (20)
$\sim choose(X) \prec self(Z), near(Y, Z), near(X, store), diff(X, Y)$ (21)
$\sim choose(X) \prec big(X)$ (22)
$choose(X) \prec big(X), self(Z), near(X, Z)$ (23)
$choose(X) \prec big(X), near(X, store)$ (24)
$choose(X) \prec small(X)$ (25)
$\sim choose(X) \prec small(X), far(X, store), self(Z), far(X, Z)$ (26)
$\sim choose(X) \prec self(Z), near(X, Z), near(Y, Z), near(Y, store),$
$diff(X, Y), same\_size(X, Y)$ (27)
$\sim choose(X) \prec choose(Y), smaller(Y, X)$ (28)

(c) Δ

Fig. 2. Defeasible Logic program $\mathcal{P} = (\Pi, \Delta)$

The defeasible rules of Δ describe the robot's preferences about which box to choose in different situations. In this case, the defeasible rules model preference criteria with respect to the size and location of the boxes. For instance, rules (27) and (28) represent the robot's preferences on those boxes near to the store or near to itself. Moreover, rule (29) states that boxes near to the robot are more desirable than those near to the store. Furthermore, rules (30)-(34) represent the preferences of the robot with respect to the boxes' size as well as in which situations, boxes of a determined size are eligible. In addition, rule (35) states that when two boxes of the same size, X and Y, are near to the robot, but Y is also near to the store, then Y is preferred over X. Finally, rule (36) provides defeasible reasons not to choose X if a smaller box Y was already chosen.

In the scenario depicted in Fig. 1(b), the robot should choose $box2$ because it is near to itself and it is also near to the store. Observe that although from $\mathcal{P}$, there are two arguments ($\mathcal{A}_1$ and $\mathcal{A}_2$) supporting $choose(box1)$, these arguments are defeated by $\mathcal{A}_3$. Thus, the answer for $choose(box1)$ is NO.

$$\mathcal{A}_1 = \{\ choose(box1) \prec small(box1)\ \} \quad \mathcal{A}_2 = \{\ choose(box1) \prec self(khep1), near(box1, khep1)\ \}$$

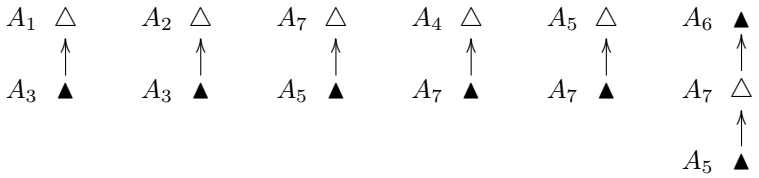
$$\mathcal{A}_3 = \left\{ \begin{array}{c} \sim choose(box1) \prec self(khep1), near(box1, khep1), near(box2, khep1), near(box2, store), \\ diff(box1, box2), same\_size(box1, box2) \end{array} \right\}$$

The interaction among these arguments is shown below (where white triangles represent defeated arguments, black triangles non-defeated ones, and arrows the defeat relation). From $\mathcal{P}$ three arguments *for* $choose(box2)$ can be obtained ($\mathcal{A}_4$, $\mathcal{A}_5$ and $\mathcal{A}_6$) and one argument *against* it ($\mathcal{A}_7$). Here, $\mathcal{A}_7$ is a blocking defeater of $\mathcal{A}_4$ and $\mathcal{A}_5$ and it is a proper defeater of $\mathcal{A}_6$, but $\mathcal{A}_5$ is also a blocking defeater of $\mathcal{A}_7$, therefore, the answer to $choose(box2)$ is YES because $\mathcal{A}_6$ is defeated by $\mathcal{A}_7$ which is in turn defeated by $\mathcal{A}_5$, reinstating $\mathcal{A}_6$. Finally, the answers for $choose(box3)$ and $choose(box4)$ are NO because arguments that use rule (36) are built to support $\sim choose(box3)$ and $\sim choose(box4)$.

$$\mathcal{A}_4 = \{\ choose(box2) \prec small(box2)\ \} \quad \mathcal{A}_5 = \{\ choose(box2) \prec self(khep1), near(box2, khep1)\ \}$$

$$\mathcal{A}_6 = \{\ choose(box2) \prec near(box2, store)\ \}$$

$$\mathcal{A}_7 = \{\ \sim choose(box2) \prec self(khep1), near(box1, khep1), near(box2, store), diff(box2, box1)\ \}$$



4 Related work

Our proposal is closely related to the approach adopted by Parsons *et al.* [10]. In particular, in our work we follow some ideas exposed in [10] about the integration of high-level reasoning facilities with low-level robust robot control. We share the approach of seeing the low-level module as a black box which receives goals to be achieved from the high-level component, and plans to reach goals are internally generated. However, our work differs from the proposal of [10] in that we do not use a BDI deliberator as high-level reasoning layer, instead we use a non-monotonic reasoning module based on a defeasible argumentation system.

With respect to this last issue, our approach to decision making is related to other works which use argumentative processes as a fundamental component in the decision making of an agent [11–13]. It is important to note that these argumentation systems have been usually integrated in *software* agents. On the other hand, in our approach, defeasible argumentation is applied in a robotic domain where the uncertainty generated by noisy sensors and effectors, changes in the physical environment and incomplete information about it, make this kind of problems a more challenging test-bed for the decision processes of an agent.

5 Conclusions and Future Work

In this paper we have shown how a Logic Programming approach could be applied in a robotic domain for knowledge representation and reasoning about which task to perform next. Our approach considers the ability of Defeasible Logic Programming to reason with incomplete and potentially inconsistent information. The simple application domain described consists of a micro-world environment using real and simulated robots for cleaning tasks. We have presented a problem and its solution when there is only one robot in the environment. Future work includes considering more complex environments, such as more than one robot with different abilities working in the same environment with the inclusion of obstacles.

Acknowledgment

We thank the Universidad Nacional de San Luis and the Universidad Nacional del Sur for their unstinting support. This work is partially supported by, CONICET (PIP 5050), and ANPCyT (PICT 2002, Nro.13096 and Nro.12600).

References

1. K-Team: Khepera 2. http://www.k-team.com (2002)
2. Michel, O.: Webots: Professional mobile robot simulation. Journal of Advanced Robotics Systems **1**(1) (2004) 39–42
3. Gat, E.: On three-layer architectures. In: Artificial Intelligence and Mobile Robots. (1998)
4. Estlin, T., Volpe, R., Nesnas, I., Muts, D., Fisher, F., Engelhardt, B., Chien, S.: Decision-making in a robotic architecture for autonomy. In: International Symposium, on AI, Robotics and Automation for Space. (2001)
5. Rotstein, N.D., García, A.J.: Defeasible reasoning about beliefs and desires. In: Proc. of the 11th Int. Workshop on Non-Monotonic Reasoning. (2006) 429–436
6. Ferretti, E., Errecalde, M., García, A., Simari, G.: Khedelp: A framework to support defeasible logic programming for the khepera robots. In: ISRA06. (2006)
7. García, A.J., Simari, G.R.: Defeasible logic programming: an argumentative approach. Theory and Practice of Logic Programming (2004)
8. Simari, G.R., Loui, R.P.: A mathematical treatment of defeasible reasoning and its implementation. Artificial Intelligence (1992)
9. Lifschitz, V.: Foundations of logic programming. In: Principles of Knowledge Representation. CSLI (1996)
10. Parsons, S., Pettersson, O., Saffiotti, A., Wooldridge, M.: Robots with the Best of Intentions. In: Artificial Intelligence Today: Recent Trends and Developments. Springer (1999)
11. Atkinson, K., Bench-Capon, T.J.M., Modgil, S.: Argumentation for decision support. In: DEXA. (2006) 822–831
12. Kakas, A., Moraitis, P.: Argumentation based decision making for autonomous agents. In: AAMAS. (2003)
13. Parsons, S., Fox, J.: Argumentation and decision making: A position paper. In: FAPR. (1996)