

# Towards a model of UAVs Navigation in urban canyon through Defeasible Logic\*

Ho-Pun Lam and Guido Governatori

February 21, 2011

## Abstract

This paper shows how a non-monotonic rule based system (defeasible logic) can be integrated with numerical computation engines, and how this can be applied to solve the Vehicle Routing Problem (VRP). To this end, we have simulated a physical system from which we can obtain numerical information. The physical system perceives information from its environment and generates predicates that can be reasoned by a defeasible logic engine. The conclusions/decisions derived will then realized by the physical system as it takes actions based on the conclusion derived. Here we consider a scenario where a “flock” of UAVs have to navigate within an urban canyon environment. The UAVs are self-autonomous without centralized control. The goal of the UAVs is to navigate to their desired destinations without colliding with each other. In case of possible collision, the UAVs concerned will communicate with each other and use their background knowledge or travel guidelines to resolve the conflicts.

## 1 Introduction

Typical complex systems have to manipulate and react to different types of data (e.g., numerical and boolean), and in many occasions we have to integrate different types of reasoning processes. For example in a UAV (Unmanned Autonomous Vehicle) scenario, a UAV has to use sensing devices to determine its position, the positions of obstacles and other information such as the current traffic situation and other UAVs’ information (that maybe represented as a vector indicating the speed, travel direction, etc.). Based on this data a UAV has to decide whether to change its course of direction or to continue with its trajectory. Accordingly, given two vectors, the UAV has to compute the intersection of the lines and has to determine whether the two UAVs will reach the intersection point at the same time (or within a proximity threshold). These two operations typically required some numerical computations. In case the

---

\*An earlier version of this paper has been presented at the 3<sup>rd</sup> International RuleML-2009 Challenge, Las Vegas, Nevada, USA, 5-7 Nov 2009

previous computation returns that a collision is possible the UAV has to decide whether to alter its travel direction and how to change it. Even though it is possible to use non-logical methods for these two tasks, experience tell us that these can be handled better with rule based methods.

Vehicle Routing Problem (VRP) is a large research topic and a huge literature is dedicated to this. According to [20], the “classical VRP” is to determine a set of least cost vehicle routes starting and ending at the depot, such that each customer is visited exactly once, and satisfying a number of side constraints (such as vehicle capacity, relative time constraints, and servicing time windows). It has played a central role in distribution management.

However, the focus of this paper is not on modelling UAV. Indeed, the novelty of this paper is, how a (non-monotonic) rule-based system can be used for coordination, which constantly seeks information from some numeric manipulation units which can be hardware such as radar, different engine monitors, GPS, etc. The advantages of using a rule based system are (1) it can be modified by an external user according to different context as new constraints about how the UAVs should behave can be introduced, (2) the decision making system has linear computational complexity with respect to the number of rules and can even be modelled as hardware.

The physical system of a UAV gathers data such as its current location, velocity, acceleration and travel directions of UAVs nearby and the alternative routes available as the contextual information, and will be used to determinate whether a collision may occur. If it is the case then some of these information will be “transformed” into a set of rules. These rules, together with some pre-defined guidelines which is available to all UAVs as a set of rules in a logic, will then be used by the reasoning engine to decides whether to continue traveling with its route, change to a new routes, or to stop its motion for a while.

The paper is organized as follows. Section 2 describes the problem scenario that we deal with; Section 3 describes the logic on which our system based and the motivation behind. An informal introduction about defeasible logics, its extension with modal operator and rules with multiple conclusions will presented. We discuss properties of the logic and we present the algorithm on which the implementation of the reasoning mechanism of the UAV is based on. Then in section 4 we will illustrate the various reasoning techniques used to model UAV navigation by means of the system we have developed. Followed by the simulation results and conclusion.

## 2 UAV Routing Problem

Typical UAVs operate at high altitudes where the space is of obstacle-free. However, when navigating within an urban environment, a UAV would have to deal with canyons that have 90-degrees bends, T-junctions and dead-ends [18], which impose another level of difficulties to the autonomous control of UAVs. In the light of this, to be able to travel from one location to another, a UAV should be able to mimic the limited and usually repetitive conditioned responses, and



Figure 1: City Map model

be smart enough to make proper decisions when a possible collision might occur.

To illustrate the combination of techniques to model the VRP we have the following problem scenario:

Given a city map with specific targets and obstacles (Figure 1), a group of UAVs has to navigate through the city from a starting location to a destination without colliding with each other. There is a GPS enabled application that informs the UAVs about the current traffic situations and the locations of other UAVs. To navigate successfully, the UAVs have to follow some guidelines about how and when they should alter their route.

The above scenario also revealed how a UAV should interact with its environment. It presumes an execution cycle consisting of a phase where the UAV collects information through its sensors, decides an action and then applies this action [23].

To this end, we have implemented a UAV navigation system to simulate the VRP mentioned above. In our simulator, each UAV is equipped with a GPS-based application that determines the current location of the UAV, direction of travel and information about other UAVs within a proximity range. The UAV traffic is thus regulated by a set of ‘road rules’ determining which vehicles have ‘right of way’ in case they are travelling to the same location within a certain time frame. In addition to this, each UAV has a route optimizer to optimize the route when navigating through the city to the desired destination.

The UAVs navigation system uses the following execution cycle:

1. A UAV gathers data about the location, travel direction and velocity of other UAVs within some proximity range from the GPS monitor.
2. The UAV detects if a collision may occur.

3. In case of a possible collision, the UAV will utilize the ‘right of way’ rules to reason the next direction of travel, or to stop its motion for a while (The mechanism to change direction is described in section 4.4). The ‘right of way’ rules are modelled in a defeasible logic theory (see Section 3).
4. If the travel direction  $D$  has to be changed the UAV will then select an optimal route (using shortest path, least numbers of turns, etc) along the set of routes in the new direction. It is also possible that the UAV remains temporarily stationary.

### 3 Defeasible Logic

The coordination of the UAV is achieved by a set of norms creating the convention UAVs have to follow to prevent collision. Given that the behaviour of the UAVs is governed by a set of norms, we believe that the best choice to formalise the rules encoding the norms is with a logic that has proved able to handle norms. Thus we have decided to encode such rules in Defeasible logic (DL). Defeasible logic is a simple and efficient skeptical rule-based non-monotonic formalism, and it has been argued that it is suitable to represent and reason with norms [1, 9, 17]. Two important features of Defeasible logic are its ability to represent exceptions (and typically normative systems leave room for exceptions) and it is possible to draw (tentative) conclusions with partial information.

#### 3.1 Basics of Defeasible Logic

A Defeasible theory<sup>1</sup> [3]  $D$  is a triple  $(F, R, >)$  where  $F$  is a finite set of facts,  $R$  is a finite set of rules, and  $>$  is a superiority relation on  $R$ . The language of defeasible logic consists of a finite set of literals,  $l$ , and their complement  $\neg l$ .

A rule  $r$  in  $R$  is composed of an antecedent (body)  $A(r)$  and a consequent (head)  $C(r)$ , where  $A(r)$  consists of a finite set of literals and  $C(r)$  contains a single literal.  $A(r)$  can be omitted from the rule if it is empty. There are three types of rules in  $R$ , namely  $\rightarrow$  (strict rules),  $\Rightarrow$  (defeasible rules), and  $\rightsquigarrow$  (defeaters). Furthermore, Defeasible logic is equipped with a binary relation between the set of rules, called superiority relation ( $>$ ), to be used to determine the relative strength of two rules. The superiority relation is used when we have two conflicting rules that fire simultaneously, and it tells us that one rule prevails over the other, thus its conclusion has to be derived.

A conclusion derived from the theory  $D$  is a tagged literal and is categorized according to how the conclusion can be proved:

- $+\Delta q$ :  $q$  is definitely provable in  $D$ .
- $-\Delta q$ :  $q$  is definitely unprovable in  $D$ .
- $+\partial q$ :  $q$  is defeasibly provable in  $D$ .

---

<sup>1</sup>Please note that the rules, predicates and propositions described in this section are for illustration only. The vocabulary and rules used in our system are those defined in Section 4.

- $-\partial q$ :  $q$  is defeasibly unprovable in  $D$ .

Provability is based on the concept of a derivation (or proof) in  $D$ . Informally, definite conclusions can be derived from strict rules by forward chaining, while defeasible conclusions can be obtained from defeasible rules iff all possible “attacks” are rebutted due to the superiority relation or defeater rules. A derivation is a finite sequence of tagged literals satisfying proof conditions (which correspond to inference rules for each of the four kinds of conclusions). For a full presentation and proof conditions of DL refer to [3].

The set of conclusions of a defeasible theory is finite<sup>2</sup>, and it can be computed in linear time [21]. The reasoning engine can be also implemented as a chip [22]. For the application at hand we have the SPINdle Java implementation of defeasible logic [19]. SPINdle is able to handle defeasible theories with over 1,000,000 rules.

To illustrate the inferential mechanism of DL, let us assume we have a theory containing the following rules:

$$\begin{aligned} r_1: & \quad \textit{SpecialOrder}(X) \Rightarrow \neg \textit{Discount}(X) \\ r_2: & \quad \textit{PremiumCustomer}(X) \Rightarrow \textit{Discount}(X) \\ r_3: & \quad \textit{Promotion}(X) \Rightarrow \neg \textit{Discount}(X) \end{aligned}$$

where the superiority relation is thus defined as:  $r_3 > r_2$  and  $r_2 > r_1$ . The theory states that services in promotion are not discounted, and so are special orders except when the order is placed by a premium customers, who are normally entitled to a discount.

In a scenario where all we have is that we received a special order, then we can conclude that the price has to be calculated without a discount since rule  $r_2$  is not applicable (we do not know whether the customer is a premium customer or not). In case the special order is received from a premium customer for a service that is not in promotion, we can derive that the customer is entitled to a discount. Indeed rule  $r_2$  is now applicable and is stronger than rule  $r_1$ , and  $r_3$ , which is stronger than  $r_2$  is not applicable (i.e., the service is not in promotion).

### 3.2 Extending Defeasible Logic with Modal Operators

Having the basics of DL is not sufficient enough. The most expensive part of developing a UAV navigation system is on creating the framework, norms, etc. for representing the behaviors of the UAVs. So the next step is to integrate modal logic in DL.

Modal logics have been put forward to capture many different notions somehow related to the intensional nature of agency as well as many other notions. Usually modal logics are extensions of classical propositional logic with some intensional operators. Thus any modal logic should account for two components: (1) the underlying logical structure of the propositional base; and (2) the logical behavior of the modal operators. Alas, as is well-known, classical propositional

<sup>2</sup>It is the Herbrand base that can be built from the literals occurring in the rules and the facts of the theory.

logic is not well suited to deal with real life scenarios. The main reason is that the descriptions of real-life cases are, very often, partial and somewhat unreliable. In such circumstances, classical propositional logic is doomed to suffer from the same problems.

On the other hand, the logic should specify how modalities can be introduced and manipulated. Some common rules for modalities are, e.g., **Necessitation** (from  $\vdash \phi$  infer  $\vdash \Box\phi$ ) and **RM** (from  $\vdash \phi \rightarrow \psi$  infer  $\vdash \Box\phi \rightarrow \Box\psi$ ). Both dictate conditions to introduce modalities purely based on the derivability and structure of the antecedent. These rules are related to well-known problem of omniscience and put unrealistic assumptions on the capability of an agent. However, if we take a constructive interpretation, we have that if an agent can build a derivation of  $\psi$  then she can build a derivation of  $\Box\psi$ .

To this end, we follow the idea presented by [15] and introduced three modal operators: BEL (Believe), INT (Intentions) and OBL (Obligation), to the theory when formalizing the behaviors of UAVs. Based on the above idea we also introduced multiple (defeasible) consequence relations, each corresponding to one of the above modal operators.

Alternatively we could say that we have a subset of defeasible rules that produce conclusions labelled with BEL, a subset of rules for conclusions labelled with INT and a subset of rules for conclusions labelled with OBL. Thus, for example, the rule

$$police(X) \Rightarrow_{BEL} emergencyVehicle(X)$$

means that if  $X$  is recognized as a police vehicle, then the agent equipped with the rule believes that the vehicle is an emergency vehicle. Similarly the reading of

$$leftIndicatorOn(X) \Rightarrow_{INT} turnLeft(X)$$

is that if a vehicle has its left turn indicator on, the vehicle intends to turn left. Finally, the rule

$$emergencyVehicle(X) \Rightarrow_{OBL} giveWayTo(X)$$

states that it is obligatory to give way to emergency vehicles.

Now we have to identify the properties of the single modal operators and their mutual relationships. The BEL operator is used to encode the environment where an agent is embedded in and to reason about it. Furthermore we assume that our agents (vehicle) have completely reliable sensors, thus we assume that they have truthful beliefs. This means that BEL is a reflexive operator (i.e., we can derive  $\phi$  from  $BEL\phi$ ). For intentions and obligations (INT and OBL) we only ask that they are internally consistent. That is, it is not possible to have both  $\Box\phi$  and  $\Box\neg\phi$ . Finally, we work with realistic social agents. A realistic agent is an agent that in case of a possible conflict between a rule for BEL and a rule for a different cognitive attitude, prefer the rule for BEL (see [13, 15, 7] for detailed discussion of agent types in defeasible logic and issues related to

them and [6] for how to use agent types to solve moral dilemmas). For example, given the rules

$$highSpeed(X) \Rightarrow_{BEL} \neg turnLeft(X)$$

and

$$leftIndicatorOn(X) \Rightarrow_{INT} turnLeft(X)$$

a realistic agent aware that there is an approaching vehicle travelling at a very high speed (and it would not be possible to have a safe turn at that speed) and with indicator on prefers the first rule to the second, and thus the agent derives  $BEL \neg turnLeft$ .

A social agent is an agent that, in case of a conflict between an obligation and an intention, gives up the intention. Consider the scenario where a vehicle is on a long street and at the end there is a T-intersection, and the vehicle has planned to turn left. However, approaching the intersection the vehicle discovers that the crossing road is one way, and it is thus forbidden turning left. Given that the agent is social the agent has to give up its intention.

$$oneWay \Rightarrow_{OBL} \neg turnLeft \qquad \Rightarrow_{INT} turnLeft$$

The above two rules encode the scenario just described. The ‘sociality’ of the agent means that the agent prefers the obligation rule to the intention rule.

The last aspect we want to have for our vehicles/agents is that they subscribe to the notion of intentionality proposed by [10].<sup>3</sup> The intuition is that if an agent intends something, let us say  $\phi$ , and the agent is aware that something else (let us say  $\psi$ ) follows from its intention  $\phi$ , and the agent does not try to avoid/minimize/prevent  $\psi$ , then the agent intends  $\psi$ . Consider the following scenario. There is a dangerous bend left, and normally it is possible to turn left without crashing at low speed, and our agent is aware of this. However, our agent intends to travel and to turn at high speed. This situation can be described by the rule:

$$highSpeed, turnLeft \Rightarrow_{BEL} crash$$

plus the intentions  $INT highSpeed$  and  $INT turnLeft$ . The intuition advanced by [15] is that the agent crashes intentionally. What we have is that we can use a belief rule to derive an intention when all the premises of the belief rule are intended. Now suppose that the agent is a skilled driver, and the agent knows how to do an hand-brake turn, and the hand-brake turn, while risky, when executed by a skilled driver might not result in a crash. This piece of information can be encoded as

$$handBrake \rightsquigarrow_{BEL} \neg crash$$

In this situation, if the agent intended to do a hand-brake turn (i.e., the agent has  $INT handBrake$ ), then we have two conflicting rules and we can no longer conclude that the crash was intentional.

<sup>3</sup>This phenomenon is also known as the expected side effects problem.

We have introduced obligations, and obligations can be violated, thus we have to be able to reason with violations. [12, 14] introduced a sub-structural operator  $\otimes$  to capture an obligation and the obligations (or another normative position) arising in response to the violation of the obligation. Thus given an expression like  $\phi \otimes \psi$ , the intuitive reading is that  $\phi$  is obligatory (i.e.,  $\text{OBL}\phi$ ), but if the obligation of  $\phi$  has been violated, then  $\psi$  is obligatory ( $\text{OBL}\psi$ ). [7, 8] proposes to use the same operator not only for deontic operators but also for other cognitive attitudes (intentions, beliefs, desires, ...), to model graded preferences<sup>4</sup>. However, the  $\otimes$  operator cannot be used for rules for reflexive operators.

To illustrate the use to  $\otimes$  for obligation rules, suppose, again the scenario of a T-intersection, where the bifurcation on the right is a lane reserved to emergency vehicles. This can be encoded by the rule

$$\text{emergencyLaneRight}, \neg\text{emergencyVehicle} \Rightarrow_{\text{OBL}} \text{turnLeft} \otimes \text{payFine}$$

The meaning is that a non emergency vehicle has the obligation to turn left, but if it does not, i.e., it turns right, then the agent has to pay a fine.

The use of  $\otimes$  to model preferences over intention is exemplified in the scenario depicted in Figure 2.

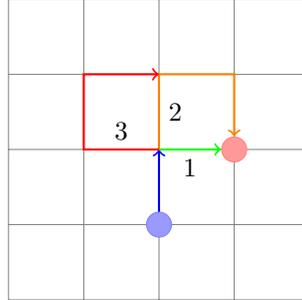


Figure 2: Preferences over Intentions

A vehicle is currently travelling North, and it is approaching an intersection, and its intended destination is straight east to the intersection. Accordingly, the preferences of the vehicle can be order as follows: (1) turn East (*moveE*), (2) continue to travel North and then change direction (*moveN*) and finally (3) take a longer detour, moving first in the opposite direction (*moveW*), then moving North, and then turning East before going South.

$$\text{currentN}, \text{INTdestinationE} \Rightarrow_{\text{INT}} \text{moveE} \otimes \text{moveN} \otimes \text{moveW}$$

The idea behind the rule is that given that the current direction of travel North, and that the intended destination is East, then the primary intention of the

<sup>4</sup>A similar approach, but with a different motivation has been proposed in the context of logic programming by Brewka and co-worker in their logic of ordered disjunction [5].

vehicle is to move East. However, if for some reason, this is not possible, i.e., we have  $\neg moveE$ , then the second best plan is to continue with the current direction, thus the vehicle should form the intention to move North. If even this is not possible, then the last resort is to move West.

### 3.3 Modal Defeasible Logic

In the previous section we paved the way to Modal Defeasible Logic, and we discussed the various properties the modal operators have to satisfy to model the application at hand. In this section we lay the formal foundations of Modal Defeasible Logic and we show how to instantiate the various options to model UAVs.

As we mention in the previous section the intuitive reading of an expression like  $a \otimes b \otimes c$  is that  $a$  is preferred, but if  $\neg a$  is the case, then  $b$  is preferred; if  $\neg b$  is the case, given  $\neg a$ , then the third choice is  $c$ . Accordingly, we will refer to  $\otimes$  also as preference operator, whose properties are given in Definition 1 below.

**Definition 1.** *A preference operator  $\otimes$  is a binary operator satisfying the following properties:*

1.  $a \otimes (b \otimes c) = (a \otimes b) \otimes c$  (associativity);
2.  $\bigotimes_{i=1}^n a_i = (\bigotimes_{i=1}^{k-1} a_i) \otimes (\bigotimes_{i=k+1}^n a_i)$  where exists  $j$  such that  $a_j = a_k$  and  $j < k$  (duplication and contraction).

**Definition 2.** *Let  $MOD = \{BEL, OBL, INT\}$  be a set of modal operators and  $P$  a set of propositional atoms.*

- If  $p \in P$ , then  $p$  and  $\neg p$  are literals.
- If  $l$  is a literal, and  $\square$  is a modal operator, then  $\square l$  and  $\neg \square l$  are modal literals.
- If  $l_1, \dots, l_n$  are literals, then  $l_1 \otimes \dots \otimes l_n$  is an  $\otimes$ -expression.

Notice that we do not allow nesting of modal operators. This is a simplification aimed at keeping the system manageable, but it does not pose severe limits for our purposes (see [16] and [15] for discussions about this limitation, and how to extend the language to overcome this limitation).

**Definition 3.** *A rule is an expression*

$$r : \phi_1, \dots, \phi_n \leftrightarrow_{\square} \psi$$

where

- $r$  is a unique label, identifying the rule;
- $\phi_i$ ,  $1 \leq i \leq n$ , is either a literal or a modal literal;
- $\leftrightarrow \in \{\rightarrow, \Rightarrow, \rightsquigarrow\}$ , and  $\square$  is a modal operator;

- $\psi$  is either a literal or, if  $\leftrightarrow \Rightarrow$ , an  $\otimes$ -expression.

According to the above definition it is not possible to have modal literals in the conclusion of a rule (so modal operators cannot occur in  $\otimes$ -expressions). This is in agreement with the idea that rules are used to introduce modal operators with their conclusions.

Given a set  $R$  of rules, we denote the set of all strict rules in  $R$  by  $R_s$ , the set of strict and defeasible rules in  $R$  by  $R_{sd}$ , the set of defeasible rules in  $R$  by  $R_d$ , and the set of defeaters in  $R$  by  $R_{df}$ .  $R[q]$  denotes the set of rules in  $R$  with consequent  $q$ . For some  $i$ ,  $1 \leq i \leq n$ , such that  $c_i = q$ ,  $R[c_i = q]$  denotes the set of rules with the head  $\otimes_{j=1}^n c_j$ . Finally,  $R^\square$  denotes the subset of  $R$  where the arrow of the rule is labelled with the modal operator  $\square$ .

**Definition 4.** The conversion relation *Convert* is defined as follows:

$$\text{Convert} \subseteq \text{MOD} \times \text{MOD}$$

The conflict relation *Conflict*  $\subseteq \text{MOD} \times \text{MOD}$  is such that

$$\forall X, Y \in \text{MOD}, \text{Conflict}(X, Y) \Rightarrow \neg(\text{Conflict}(Y, X)) \text{ (asymmetry)}$$

Specifically, for realistic social agents, we are going to set the following conflicts and conversions:

$$\mathcal{C} = \{\text{Convert}(\text{BEL}, \text{INT}), \text{Conflict}(\text{BEL}, \text{OBL}), \\ \text{Conflict}(\text{BEL}, \text{INT}), \text{Conflict}(\text{OBL}, \text{INT})\}$$

**Definition 5.** A defeasible modal theory is a structure

$$D = (F, R^{\text{BEL}}, R^{\text{INT}}, R^{\text{OBL}}, >, \mathcal{C})$$

where

- $F$  is a finite set of facts, i.e., a set of literals and modal literals;
- $R^{\text{BEL}}, R^{\text{INT}}, R^{\text{OBL}}$  are three finite sets of rules such that each rule has a unique label;
- The superiority relation  $>$  is such that  $> = >^{sm} \cup >^{\text{Conflict}}$ , where  $>^{sm} \subseteq R^X \times R^X$  such that if  $r > s$ , then if  $r \in R^X[p]$  then  $s \in R^X[\sim p]$  and  $>$  is acyclic; and  $>^{\text{Conflict}}$  is such that

$$\forall r \in R^X[p], \forall s \in R^Y[\sim p], \text{if } \text{Conflict}(X, Y), \text{ then } r >^{\text{Conflict}} s$$

- $\mathcal{C}$  is the set of conflict and conversion relations given above for realistic social agents.

The construction of the superiority relation combines two components: the first  $>^{sm}$  considers pairs of rules of the same mode. This component is usually given by the designer of the theory and captures the meaning of the single rules, and thus encodes the domain knowledge of the designer of the theory. The second component,  $>^{\text{Conflict}}$  is obtained from the rules in a theory and depends on the meaning of the modalities.

**Definition 6.** Given a defeasible modal theory  $D$ , a proof in  $D$  is a linear derivation, i.e., a sequence of labelled formulas of the type  $+\Delta_{\square}q$ ,  $-\Delta_{\square}q$ ,  $+\partial_{\square}q$  and  $-\partial_{\square}q$ , where the proof conditions defined in the rest of this section hold.

The interpretation of the proof tags for modal defeasible logic is the same as that of standard defeasible logic, the only modification is that the conclusions has been obtain, at least in the last step, using rules of mode  $\square$ .

We start with some terminology. As was explained, the following definition states the special status of belief rules, and that the introduction of a modal operator corresponds to being able to derive the associated literal using the rules for the modal operator.

**Definition 7.** Let  $\# \in \{\Delta, \partial\}$ ,  $\square \in \text{MOD}$  and  $P = (P(1), \dots, P(n))$  be a proof in  $D$ . A (modal) literal  $q$  is  $\#$ -provable in  $P$  if there is a line  $P(m)$  of  $P$  such that either

1.  $q$  is a literal and  $P(m) = +\#\text{BEL}q$  or
2.  $q$  is a modal literal  $\square p$  and  $P(m) = +\#\square p$  or
3.  $q$  is a modal literal  $\neg\square p$  and  $P(m) = -\#\square p$ .

A literal  $q$  is  $\square$ -rejected in  $P$  if there is a line  $P(m)$  of  $P$  such that either

1.  $q$  is a literal and  $P(m) = -\#\text{BEL}q$  or
2.  $q$  is a modal literal  $\square p$  and  $P(m) = -\#\square p$  or
3.  $q$  is a modal literal  $\neg\square p$  and  $P(m) = +\#\square p$ .

The definition of  $\Delta_{\square}$  describes just forward chaining of strict rules:

- $+\Delta_{\square}$ : If  $P(n+1) = +\Delta_{\square}q$  then
- (1)  $q \in F$  if  $X = \text{BEL}$  or  $\square q \in F$  or
  - (2)  $\exists r \in R_s^{\square}[q] : \forall a \in A(r) a$  is  $\Delta$ -provable or
  - (3)  $\exists r \in R_s^{\square_i}[q] : \text{Convert}(\square_i, \square) \in \mathcal{C}$ ,  $A(r) \neq \emptyset$ ,  $\forall a \in A(r) \square a$  is  $\Delta$ -provable.
- $-\Delta_{\square}$ : If  $P(n+1) = -\Delta_{\square}q$  then
- (1)  $q \notin F$  if  $X = \text{BEL}$  and  $\square q \notin F$  and
  - (2)  $\forall r \in R_s^{\square}[q] \exists a \in A(r) : a$  is  $\Delta$ -rejected and
  - (3)  $\forall r \in R_s^{\square_i}[q] : \text{if } \text{Convert}(\square_i, \square) \in \mathcal{C} \text{ then } \exists a \in A(r) \square a$  is  $\Delta$ -rejected.

For a literal  $q$  to be definitely provable we need to find a strict rule with head  $q$ , whose antecedents have all been definitely proved previously. And to establish that  $q$  cannot be proven definitely we must establish that for every strict rule with head  $q$  there is at least one of antecedent which has been shown to be non-provable. Condition (3) says that a belief rule can be used as a rule for a different modal operator in case all literals in the body of the rules are modalised with the modal operator we want to prove. Thus, for example, given the rule  $p, q \rightarrow_{\text{BEL}} s$ , we can derive  $+\Delta_{\text{INT}}s$  if we have  $+\Delta_{\text{INT}}p$  and  $+\Delta_{\text{INT}}q$ .

Conditions for  $\partial_{\square}$  are more complicated. First we have to account for  $\otimes$ -expressions. Then we have to define when a rule is applicable or discarded. A rule for a belief is applicable if all the literals in the antecedent of the rule are provable with the appropriate modalities, while the rule is discarded if at least one of the literals in the antecedent is not provable. As before, for the other types of rules we have to take conversions into account. We have thus to determine conditions under which a rule for  $Y$  can be used to directly derive a literal  $q$  modalised by  $\square$ . Roughly, the condition is that all the antecedents  $a$  of the rule are such that  $+\partial_{\square}a$ .

**Definition 8.** *Given a derivation  $P$ ,  $P(1..n)$  denotes the initial part of the derivation of length  $n$ . Let  $X, Y, Z \in \text{MOD}$ .*

- A rule  $r \in R_{sd}[c_i = q]$  is applicable in the proof condition for  $\pm\partial_X$  iff
  1.  $r \in R^X$  and  $\forall a \in A(r)$ ,  $+\partial_{\text{BEL}}a \in P(1..n)$  and  $\forall Za \in A(r)$ ,  $+\partial_Za \in P(1..n)$ , or
  2.  $r \in R^Y$ ,  $\text{Convert}(Y, X) \in \mathcal{C}$ ,  $A(r) \neq \emptyset$  and  $\forall a \in A(r)$ ,  $+\partial_Xa \in P(1..n)$ .
- A rule  $r$  is discarded in the condition for  $\pm\partial_X$  iff
  1.  $r \in R^X$  and  $\exists a \in A(r)$  such that  $-\partial_{\text{BEL}}a \in P(1..n)$  or  $\exists Za \in A(r)$  such that  $-\partial_Za \in P(1..n)$ ; or
  2.  $r \in R^Y$  and, if  $\text{Convert}(Y, X)$ , then  $\exists a \in A(r)$  such that  $-\partial_Xa \in P(1..n)$ , or
  3.  $r \in R^Z$  and either  $\neg\text{Convert}(Z, X)$  or  $\neg\text{Conflict}(Z, X)$ .

We are now ready to provide proof conditions for  $\partial_X$ :

- $+\partial_X$ : If  $P(n+1) = +\partial_Xq$  then
- (1)  $+\Delta_Xq \in P(1..n)$  or
    - (2.1)  $-\Delta_X\sim q \in P(1..n)$  and
    - (2.2)  $\exists r \in R_{sd}[c_i = q]$  such that  $r$  is applicable, and  $\forall i' < i$ ,  $+\partial_{\text{BEL}}\sim c_{i'} \in P(1..n)$ ; and
    - (2.3)  $\forall s \in R[c_j = \sim q]$ , either  $s$  is discarded, or  $\exists j' < j$  such that  $-\partial_{\text{BEL}}\sim c_{j'} \in P(1..n)$ , or
    - (2.3.1)  $\exists t \in R[c_k = q]$  s.t.  $r$  is applicable,  $t > s$   $\forall k' < k$ ,  $-\partial_{\text{BEL}}c_{k'} \in P(1..n)$  and either  $t, s \in R^Z$  or  $\text{Convert}(Y, X)$  and  $t \in R^Y$ .

- $-\partial_X$ : If  $P(n+1) = -\partial_X q$  then
- (1)  $-\Delta_X q \in P(1..n)$  and either
    - (2.1)  $+\Delta_X \sim q \in P(1..n)$  or
    - (2.2)  $\forall r \in R_{sd}[c_i = q]$ , either  $r$  is discarded or
      - $\exists i' < i$  such that  $-\partial_{\text{BEL}} \sim c_{i'} \in P(1..n)$ , or
    - (2.3)  $\exists s \in R[c_j = \sim q]$ , such that  $s$  is applicable and
      - $\forall j' < j$ ,  $+\partial_{\text{BEL}} \sim c_{j'} \in P(1..n)$  and
      - (2.3.1)  $\forall t \in R[c_k = q]$  either  $t$  is discarded, or  $t \not\prec s$ , or
        - $\exists k' < k$  such that  $+\partial_{\text{BEL}} c_{k'} \in P(1..n)$  or
        - $t \in R^Z$ ,  $s \in R^{Z'}$ ,  $Z \neq Z'$  and
        - if  $t \in R^Y$ , then  $\neg\text{Convert}(Y, X)$ .

For defeasible rules we deal with  $\otimes$  formulas. To show that  $q$  is provable defeasibly we have two choices: (1) We show that  $q$  is already definitely provable; or (2) we need to argue using the defeasible part of a theory  $D$ . For this second case, three (sub)conditions must be satisfied. First, we require that there must be a strict or defeasible rule for  $q$  which can be applied (2.1); in case the conclusion of the rule is an  $\otimes$ -expression, and the conclusion we want to prove is not the first element, we have to ensure that for every element of the  $\otimes$ -expression that precedes  $q$  the complement if provable with belief mode. Second, we need to consider possible reasoning chains in support of  $\sim q$ , and show that  $\sim q$  is not definitely provable (2.2). Third, we must consider the set of all rules which are not known to be inapplicable and which permit to get  $\sim q$  (2.3); the second part of this condition checks that we can use the  $\sim q$  if this is not the first element of the  $\otimes$ -expression corresponding to the head of the rule. The idea is that an obligation has not been violated, similarly for intentions. Essentially, each such a rule  $s$  attacks the conclusion  $q$ . For  $q$  to be provable,  $s$  must be counterattacked by a rule  $t$  for  $q$  with the following properties: (i)  $t$  must be applicable, and (ii)  $t$  must be stronger than  $s$ . Thus each attack on the conclusion  $q$  must be counterattacked by a stronger rule. In other words,  $r$  and the rules  $t$  form a team (for  $q$ ) that defeats the rules  $s$ . However, since we can have rules for different modes, we have to ensure we have the appropriate relationships among the rules. Thus clause (2.3.1) prescribes that either the rule that attacks the conclusion we want to prove ( $s$ ) and the rule used to counterattack it (i.e.,  $t$ ) have the same mode (i.e.,  $s, t \in R^Z$ ), or that  $t$  can be used to produce a conclusion of the mode we want to prove (i.e.,  $t \in R^Y$  and  $\text{Convert}(Y, X)$ ).  $-\partial_X q$  is defined in an analogous manner.

### 3.4 Properties of Modal Defeasible Logic

In this section we are going to explore some properties of the logic that make the logic amenable for applications.

The first property is that Modal Defeasible Logic with the preference operator  $\otimes$  is consistent and coherent. Consistency means that for any literal and proof tag it is not possible to prove and reject the literal (with one and the same proof tag). Coherence means that the inferential mechanism of defeasible logic

does not allow us to defeasible derive both  $p$  and  $\neg p$  with the same mode, unless the monotonic part of a theory (i.e., facts and strict rules) is inconsistent; and even in that cases the inconsistency does not propagate to the entire theory. In this respect defeasible logic is paraconsistent.

**Proposition 1.** *Let  $D$  be modal defeasible theory,  $l$  be a literal,  $\# \in \{\Delta, \partial\}$ , and  $\square \in \text{MOD}$ . It is not the case that  $D \vdash +\#_{\square}l$  and  $D \vdash -\#_{\square}l$ .*

*Proof.* The proof is an immediate consequence of the framework used to define the proof tags [10]. The proof tags for the negative proof tags, i.e.,  $-\Delta$  and  $-\partial$ , are the ‘strong’ negation of that for the positive proof tags [2]. The strong negation of a formula is closely related to the function that simplifies a formula by moving all negations to an innermost position in the resulting formula and replaces the positive tags with the respective negative tags and vice-versa.  $\square$

**Proposition 2.** *Let  $D$  be modal defeasible theory,  $l$  be a literal,  $\square \in \text{MOD}$ .  $D \vdash +\partial_{\square}l$  and  $D \vdash +\partial_{\square}\neg l$  iff  $D \vdash +\Delta l$  and  $D \vdash +\Delta\neg l$ .*

*Proof.* The right to left direction is immediate given clause (1) of the definition of the proof conditions for  $+\partial$ .

For the left to right direction, for BEL, since we cannot have  $\otimes$  is the conclusion of rules, see Proposition 5.5 [3]. For the other cases let us reason as follows: Let us suppose that we have left hand side of the iff but not the right hand one. This means that either we have (i) only one of  $D \vdash +\Delta l$  and  $D \vdash +\Delta\neg l$  or (ii) none of them.

For (i), let us assume that we have  $D \vdash +\Delta l$  (the case where we have  $D \vdash +\Delta\neg l$  is symmetrical). Since we do not have  $D \vdash +\Delta\neg l$ , then condition (2.1) of  $+\partial$  holds, that is,  $-\Delta\neg l$ , thus condition (1) of  $-\partial$  holds, and we know  $+\Delta l$ , thus it holds that  $D \vdash -\partial\neg l$ , and thus from Proposition 1 we get a contradiction.

For (ii): First of all, it is easy to verify that no rule can be at the same time applicable and discarded for the derivation of  $\pm\partial_{\square}l/\neg l$ . Then, since both  $+\partial_{\square}l$  and  $+\partial_{\square}\neg l$  holds, then we have that there are applicable rules for both  $l$  and  $\neg l$ . This means that clause (2.3.1) holds for both  $+\partial_{\square}l$  and  $+\partial_{\square}\neg l$ . Therefore, for every applicable rule for  $l$  there is an applicable rule for  $\neg l$  stronger than the rule for  $l$ , and symmetrically, for every applicable rule for  $\neg l$  there is an applicable rule for  $l$  stronger than the rule for  $\neg l$ . The set of rules in a theory is finite, thus, this means that the situation we have just described is only possible if there is a cycle in the transitive closure of the superiority relation. Accordingly, we have a contradiction, because the superiority relation of defeasible theory is acyclic (the transitive closure of the relation does not contains cycles.).  $\square$

In [15], it was proved that the modal defeasible logic obtained by extending defeasible logic with BIO modal operators (BEL, INT, and OBL) and the mechanism of conversion and conflict still has liner complexity (in the size of a theory, where the size of the theory is given by the number of rules and number of distinct literals). In [11] the complexity result was extended to deontic defeasible logic with the  $\otimes$ -operator. The techniques used to prove the two

complexity results for modal operators and  $\otimes$  can be combined. In the rest of this section we propose algorithms for computing the extension of a modal defeasible theory based on the result just discussed. The algorithms proposed hereto implement both modal operators and  $\otimes$ -expressions extend the algorithm proposed by Maher [21] and they have been implemented in SPINdle [19]

Before presenting the algorithms we introduce some notation used in it. Given a defeasible modal theory  $D$ , a modal operator  $\square \in \text{MOD}$ , and a set of rules  $R$  (such that the rules appears in the theory); for  $r \in R$ ,  $r_{sup} = \{s : (s, r) \in >\}$  and  $r_{inf} = \{s : (r, s) \in >\}$ . The Herbrand Base of  $D$ ,  $HB_D$  is the set of literals such that the literal or its complement appears in  $D$ , where ‘appears’ means that it is could be a subformula of a modal literal occurring in the theory. The Modal Herbrand Base of  $D$ ,  $HB^\square = \{\square_i l : \square_i \in \text{MOD}, l \in HB_D\}$ . Given an  $\otimes$ -expression  $c = a_1 \otimes \dots \otimes a_{i-1} \otimes a_i \otimes a_{i+1} \otimes \dots \otimes a_n$ , the operations  $c!a_i$  (truncation at  $a_i$ ) and  $c \ominus a_i$  (removal of  $a_i$ ) are defined as follows:

$$\begin{aligned} c!a_i &= a_1 \otimes \dots \otimes a_{i-1} \\ c \ominus a_i &= a_1 \otimes \dots \otimes a_{i-1} \otimes a_{i+1} \otimes \dots \otimes a_n \end{aligned}$$

The idea of the algorithms is to perform a series of transformations that reduce a defeasible modal theory into a simpler equivalent theory.

To simplify the presentation of the algorithm to compute the definite extension, i.e., the set of conclusions that can be proved with  $\pm\Delta$ , we assume that the set of facts is empty. This can be easily achieved by replacing every literal  $l$  in  $F$  with the rule  $\rightarrow_{\text{BEL}} f$ , and every modal literal  $\square l$  with the rule  $\rightarrow_{\square} l$ .

Algorithm 1 starts by initialising the global sets of definite conclusions to the empty sets (lines 1 and 2). In addition for any conversion relationship it identifies the set of rules that can be used for a conversion (line 3). Given a conversion  $\text{Convert}(\square_i, \square_j)$  a rule for  $\square_i$  can be used to derive a conclusion of type  $\square_j$ , if the body of the rule is not empty and all the elements of the body are literals (not modal literals) and they are derived with mode  $\square_j$  (see clause 3 of the conditions for  $+\Delta_{\square}$ ).

After the initialisation, the algorithm begins its main loop. At each cycle of the loop we reset the set of definite conclusion that are derived during a cycle of the loop (lines 5–6). Then, we scan the modal literals in the Modal Herbrand Base. For each of such modal literals, we check whether there are strict rules that can produce it. This means that for a modal literal  $\square l$  we have to check the rule of mode  $\square$  as well as the rule for all modes involved in a conversion  $\text{Convert}(\square_i, \square)$ . If there are no such rules (line ), then clauses (2) and (3) of the conditions for  $-\Delta_{\square}$  are vacuously satisfied, thus we know that we can prove  $-\Delta_{\square} l$ , thus we add it to the set of local negative conclusions (line 9). Lines 10–12 take care of the effects of the conclusion  $-\Delta_{\square} l$ . If a rule has  $\square l$  in the body (or  $l$  if  $\square = \text{BEL}$ , given the reflexivity of the BEL operator), then the rule is discarded (it cannot longer be used to prove a positive conclusion), and thus we can remove such rules without affecting the conclusions that can be derived from the theory. Similarly if  $l$  appears in the body of a conversion rule (line 12). In addition  $-\Delta_{\square} l$  makes  $\neg \square l$   $\Delta$ -provable, thus we can safely remove

---

**Algorithm 1** ComputeDefinite

---

```
1:  $+\Delta_{\square} \leftarrow \emptyset$ 
2:  $-\Delta_{\square} \leftarrow \emptyset$ 
3:  $R_s^{\square_i, \square_j} \leftarrow \{r \in R_s^{\square_i} : \text{Convert}(\square_i, \square_j), A(r) \neq \emptyset, A(r) \text{ is a set of literals}\}$ 
4: repeat
5:    $\Delta_{\square}^{\dagger} \leftarrow \emptyset$ 
6:    $\Delta_{\square}^{-} \leftarrow \emptyset$ 
7:   for  $\square l \in HB^{\square}$  do
8:     if  $R_s^{\square}[l] \cup R_s^{\square_i, \square}[l] = \emptyset$  then
9:        $\Delta_{\square}^{-} \leftarrow \Delta_{\square}^{-} \cup \{l\}$ 
10:       $R_s \leftarrow R_s - (\{r \in R_s : \square l \in A(r)\} \cup \{r \in R_s : l \in A(r), \square = \text{BEL}\})$ 
11:       $R_s \leftarrow \{A(r) - \{\neg \square l\} \rightarrow_X C(r) : r \in R_s\}$ 
12:       $R_s^{\square_i, \square} \leftarrow R_s^{\square_i, \square} - \{r \in R_s^{\square_i, \square} : l \in A(r)\}$ 
13:       $HB^{\square} \leftarrow HB^{\square} - \{\square l\}$ 
14:     end if
15:     if  $\exists r \in R_s^{\square}[l] \cup R_s^{\square_i, \square}[l] : A(r) = \emptyset$  then
16:        $\Delta_{\square}^{\dagger} \leftarrow \Delta_{\square}^{\dagger} \cup \{l\}$ 
17:        $R_s \leftarrow \{A(s) - \{\square l\} \rightarrow_X C(s) : s \in R_s\}$ 
18:        $R_s \leftarrow \{A(s) - \{l\} \rightarrow_X C(s) : s \in R_s, \square = \text{BEL}\}$ 
19:        $R_s^{\square_i, \square} \leftarrow \{A(r) - \{l\} \rightarrow_{\square_i} C(r) : r \in R_s^{\square_i, \square}\}$ 
20:        $HB^{\square} \leftarrow HB^{\square} - \{\square l\}$ 
21:     end if
22:   end for
23:    $+\Delta_{\square} \leftarrow +\Delta_{\square} \cup \Delta_{\square}^{\dagger}$ 
24:    $-\Delta_{\square} \leftarrow -\Delta_{\square} \cup \Delta_{\square}^{-}$ 
25: until  $\Delta_{\square}^{\dagger} = \emptyset$  and  $\Delta_{\square}^{-} = \emptyset$ 
```

---

the occurrences of such modal literal from the body of the rules. The modal literal no longer contributes for the rule being applicable or rejected, these two properties now depends on the other (modal) literals in the body of the rules. After this step we remove  $\Box l$  from the modal Herbrand Base (line 13). We have already assessed the status of it and there is no need to further iteration on it.

The next step is to figure out whether the literal is provable positively. The strategy we use is to see whether there is a rule with empty body among the rules that can lead the modal literal, clauses (2) and (3) of  $+\Delta_{\Box}$ . Notice that for conversion, we first create a set of rules that can be used for conversion  $(R_i^{\Box}, B_j)$ , and we do not look for rules that satisfy the condition among all rules for a mode involved in a conversion. In lines 17–19, we remove (modal) literal that do not further influence the outcome of the computation. At the end of this phase we remove the modal literal for the Modal Herbrand Base, thus we do not have to consider it again in the loop.

In the main loop (lines 7–22) we have two blocks: lines 7–14 and lines 15–21. The applicability conditions for those two blocks are mutually exclusive, and when of the two succeeds, the output of the cycle is not empty, and at the same time we reduce the complexity of the theory, either by removing rules or by removing literals from the body of rules. In case that both blocks are unsuccessful, then there are no modifications in the theory, thus any other repetition of the loop would not produce any further conclusion, thus the exit condition in line 25. All conditions to be verified and the operation to be performed in the algorithm requires either constant time or in the worst case linear time. Thus the complexity of the algorithm is dominated by the number of times we have to repeat the main loop; but this is bounded by the number of rules. To have a successful cycle we either had to remove a rule (making the set of rule for a literal  $l$  empty) or reduced a rule to a rule with empty body (and after that we could remove the rule<sup>5</sup>). In the worst case we remove a single rule at every cycle and at every cycle we have to scan Modal Herbrand Base. In addition the set of rules and the Modal Herbrand Base are finite, thus the algorithm terminates and the complexity of the main loop is  $O(|HB^D| * |R_s|)$ .

The discussion of Algorithm 1 above shows that

**Proposition 3.** *The definite extension of a Defeasible Modal Theory  $D$  can be computed in time linear to the size of the theory, where the size of the theory is determined by the number of literals and rules in time.*

Before moving to the algorithm to compute the defeasible extension of a Defeasible Modal Theory we introduce an auxiliary procedure that is used several time in the computation of the defeasible extension. The procedure in Algorithm 2 do the ‘housekeeping’ operation related to when we prove  $-\partial_{\Box} l$ . In nature it is very similar to the operations in lines 9–12 of Algorithm 1, the difference is that it operates on all types of rules and not only on strict rules. In addition when it removes a rule it removes the rule from the superiority relation.

---

<sup>5</sup>In the algorithm this effect is achieved by remove the literal for the Modal Herbrand Base, thus while the rule is not remove, effectively is removed, since we do not scan the literal

Furthermore, when  $\square = \text{BEL}$  it operates on  $\otimes$ -expression in the head of rules. For these expressions it truncates the  $\otimes$ -expression keeping only the part of the expression preceding the first occurrence of the complement of then literal the procedure takes as parameter (line 9). This operation implement clause (2.3) of  $+\partial_{\square}$  and the or part of clauses (2.2) and (2.3.1) of  $-\partial_{\square}$ .

---

**Algorithm 2** Discard

---

```

1: procedure DISCARD( $l, \square$ )
2:    $\partial_{\square} \leftarrow \partial_{\square} \cup \{l\}$ 
3:    $R \leftarrow R - \{s \in R : \square l \in A(r)\}$ 
4:    $R \leftarrow \{A(r) - \{\neg \square l\} \leftrightarrow_X C(r) : r \in R\}$ 
5:    $R^{\square, \square_i} \leftarrow R^{\square, \square_i} - \{s \in R^{\square, \square_i} : l \in A(r)\}$ 
6:    $>\leftarrow -\{(r, s), (s, r) \in > : \square l \in A(r)\}$ 
7:   if  $\square = \text{BEL}$  then
8:      $R \leftarrow R - \{s : l \in A(s)\}$ 
9:      $R \leftarrow \{A(r) \leftrightarrow_X C(r)! \sim l : r \in R\}$ 
10:     $>\leftarrow \{(r, s), (s, r) \in > : l \in A(r)\}$ 
11:  end if
12: end procedure

```

---

We are now ready to present the Algorithm (Algorithm 3) to compute the defeasible extension of a Defeasible Modal Theory, i.e., the set of conclusions that can be proved with  $\partial_{\square}$ .

Most of the aspects of Algorithm 3 are essentially identical to the corresponding operation and condition in Algorithm 1. Thus the focus here is just on the aspect specific to the task at hand.

For the initialisation we take the output of ComputeDefinite Algorithm and we populate the global set of positive and negative defeasible conclusion according to clause (1) of  $+\partial + \square$  and clause (2.1) of  $-\partial_{\square}$ . We then remove all literal included in the two set just initialised from the Modal Herbrand Base. We also generate clones of the rules for the set of rules that can be applied in a conversion. The last initialisation step is to create a set where to store the rules that are weaker than applicable rules (line 5).

As in ComputeDefinite we have a main loop, where, for each cycle we initialise the set of conclusions that can compute at that iteration of the main loop. Again the main loop has two mutually exclusive blocks. The first block (lines 10–13) corresponds to the similar block in ComputeDefinite. The difference here is that we operate on all types of rules and not only on strict rules, but the operations performed are the same.

The real differences are in the block at lines 15–34. The starting point is to figure out if there are applicable rules (of an appropriate mode). The rules we have to consider are the rule for a modal operator plus all rules that can be used in a conversion to that modal operator. In this case, since defeasible rules can have  $\otimes$ -expressions as they heads, the literal we are interested to is the first element of the  $\otimes$ -expression. For all applicable rules we collect the rules weaker

---

**Algorithm 3** ComputeDefeasible
 

---

```

1:  $+\partial_{\square} \leftarrow +\Delta_{\square}$ 
2:  $-\partial_{\square} \leftarrow \{l : \sim l \in -\Delta_{\square}\}$ 
3:  $R^{\square_i, \square_j} \leftarrow \{r \in R^{\square_i} : \text{Convert}(\square_i, \square_j), A(r) \neq \emptyset, A(r) \text{ is a set of literals}\}$ 
4:  $HB^{\square} \leftarrow HB^{\square} - \{\square l : l \in +\partial_{\square} \cup -\partial_{\square}\}$ 
5:  $R_{infd} \leftarrow \emptyset$ 
6: repeat
7:    $\partial_{\square}^+ \leftarrow \emptyset$ 
8:    $\partial_{\square}^- \leftarrow \emptyset$ 
9:   for  $\square l \in HB^{\square}$  do
10:    if  $R_{sd}^{\square}[l] \cup R_{sd}^{\square_i, \square}[l] = \emptyset$  then
11:      DISCARD( $l, \square$ )
12:       $HB^{\square} \leftarrow HB^{\square} - \{\square_i l\}$ 
13:    end if
14:  end for
15:  for  $r \in R^{\square} \cup R^{\square_i, \square}$  do
16:    if  $A(r) = 0$  then
17:       $R_{infd} \leftarrow R_{infd} \cup r_{inf}$ 
18:      Let  $l$  be the first literal of  $C(r)$ 
19:      if  $R^{\square}[\sim l] \cup R^{\square_i}[\sim l] - R_{infd} \subseteq r_{inf}$ , for  $\square_i$  s.t.  $\text{Conflict}(\square_i, \square)$ 
20:    then
21:      DISCARD( $\sim l, \square_i$ ) for  $\square_i$  s.t.  $\text{Conflict}(\square, \square_i)$ 
22:       $HB^{\square} \leftarrow HB^{\square} - \{\square_i \sim l : \text{Conflict}(\square, \square_i)\}$ 
23:      if  $r \in R_{sd}$  then
24:         $\partial_{\square}^+ \leftarrow \partial_{\square}^+ \cup \{l\}$ 
25:         $R \leftarrow \{A(s) - \{\square l\} \leftrightarrow_X C(s) : s \in R\}$ 
26:         $R^{\square, \square_i} \leftarrow \{A(s) - \{l\} \leftrightarrow_X C(s) : s \in R^{\square, \square_i}\}$ 
27:        if  $\square = \text{BEL}$  then
28:           $R \leftarrow \{A(s) - \{l\} \leftrightarrow_X C(r) \ominus l : s \in R\}$ 
29:           $R^{\square_i, \square_j} \leftarrow \{A(s) \leftrightarrow_X C(r) \ominus l : s \in R^{\square_i, \square_j}\}$ 
30:        end if
31:         $HB^{\square} \leftarrow HB^{\square} - \{\square l\}$ 
32:      end if
33:    end if
34:  end for
35:   $+\partial_{\square} \leftarrow +\partial_{\square} \cup \partial_{\square}^+$ 
36:   $-\partial_{\square} \leftarrow -\partial_{\square} \cup \partial_{\square}^-$ 
37: until  $\partial_{\square}^+ = \emptyset$  and  $\partial_{\square}^- = \emptyset$ 

```

---

than them and we group them in the set  $R_{inf\d}$ . This set contains all rules for which clause (3.2.1) of  $+\partial_{\square}$  holds.

The next step, line 19, is to check whether there are rules for the complement of the literal of the same mode, or of a mode conflicting with the mode we want to prove the literal with. The rules for the complement should not be defeated by an applicable rule, i.e., they should not be in  $R_{inf\d}$ . When the condition in line 19 is satisfied, then clauses (2.2) and (2.3) of  $-\partial_{\square}$  are satisfied, and thus can assert that the complement of the literal is rejected with mode  $\square$ , thus we have  $-\partial_{\square}\sim l$ , and we call the Discard procedure to do the required housekeeping.

After that, what we have to do is to check if the rule is a rule that can support a positive conclusion, i.e., not a defeater. This is the final step to verify that the condition for deriving  $+\partial_{\square}$  is satisfied. The last thing to do is to do the housekeeping to reduce the complexity of the theory. Thus we remove all instances of  $\square l$  from the body of rules,  $l$  form the bodies of the appropriate rules for conversion as we did for the case of ComputeDefinite. However, we have to do an additional operation. In case  $\square = \text{BEL}$  we have to remove the literal from  $\otimes$ -expressions occurring in the head of rules, to ensure that we properly capture the second conjunct of clause (2.2) of  $+\partial_{\square}$ , the third of clause (2.3.1) of the same condition, and the second conjunct of clause (2.3) of  $-\partial_{\square}$ .

The termination analysis is the same as that of ComputeDefinite. For the complexity, in this case the complexity of main loop is dominated by the for loop in lines 15–34. For this we have to repeat the loop one time for each literal in  $HB^{\square}$ , hence the complexity of this loop is again  $O(|HB^{\square}| * |R|)$ .

From the discussion about Algorithm 3 above we can conclude the following proposition.

**Proposition 4.** *The defeasible extension of a Defeasible Modal Theory  $D$  can be computed in time linear to the size of the theory, where the size of the theory is determined by the number of literals and rules in time.*

## 4 The UAV Navigation System

In this section we present a novel version of UAV routing problem to illustrate the framework that we have developed in the paper. The objective of our framework is to have that all UAVs travel to the destination (in an urban canyon environment) without colliding with each other. To simplify our framework, we considered only four values of travel directions<sup>6</sup>, namely: NORTH, EAST, WEST and SOUTH, as shown in Figure 3.

### 4.1 Perception-Action Cycle

As discussed in Section 2 a UAV will gather different types of data from the GPS monitor within a proximity range and detects if a possible collision occur. In

<sup>6</sup>The other two directions, UP and DOWN, can easily be extended in our system simply by adding their associated set of rules in the knowledge base.

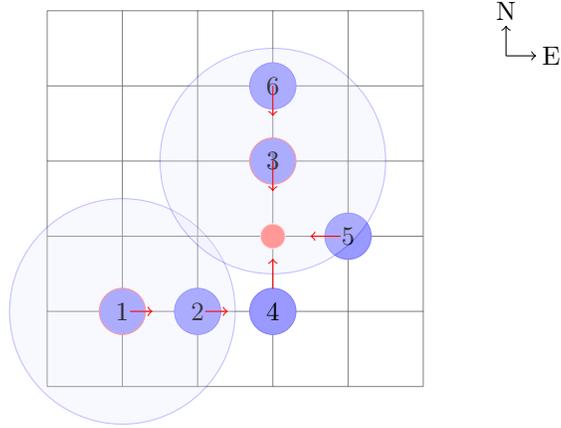


Figure 3: City with UAVs

case of a possible collision, a UAV will utilize the information in its Knowledge Base to reason on the next travel direction or eventually to temporarily stop its motion.

The Knowledge Base (KB) contains information about a UAV and is a well-documented limited set of behavioral interactions that describes the behavior of a UAV under different situations, in particular it contains (prescriptive) rules for determining who has right of way over who. It is complemented with formal logics (and in particular DL) to represent significant issues regarding our domain of operations.

To be able to travel from one location to another, and to avoid collisions with other UAVs, a UAV should incorporate into the KB the set of context-related information (such as traffic situation, information about the vehicles nearby, etc) and derive a safe direction of travel when a possible collision occurs. (Consider the scenario as shown in Figure 3 where vehicles  $V_3$ ,  $V_4$  and  $V_5$  are moving towards the same location (the red circle) and collisions may occur if none of the vehicles alter their route.) This perception-action cycle (Figure 4) can be conceived not only as an issue of control, but also lays out the interface on how the UAVs should interact with the environment [4]. Here, the *sensors* (in our case the GPS monitor) collect information about the environment (as described above) and which is then combined with the knowledge base for common-sense decision making. The *behavior controller* then reasons on these information and triggers the associated *actuator(s)* (whether to change its current travel direction, speed or even to stop its current motion) based on the conclusions derived.

Besides, the perception-action cycle also allows us to incorporate sensor information on the same aspect of the environment from multiple sensors using DL with appropriate superiority relations, which differentiate our approach from others, for example from *reactive system* where reactive agents typically do not

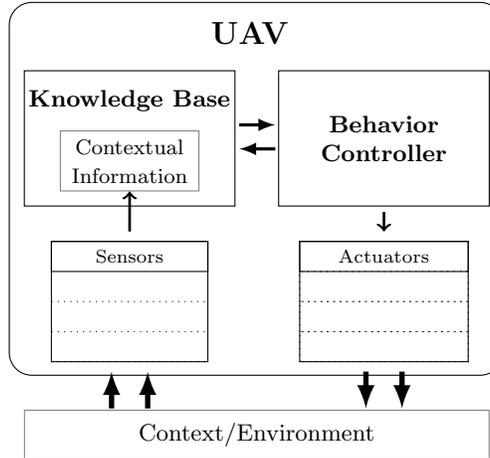


Figure 4: Behavior control under an perception-action cycle

build plans, carry out no reasoning and rarely represent knowledge in any formal logics [4]. In addition, similar to subsumption using priority discrimination, our system organize behaviors in a hierarchical way using the types of behaviors or actions suitable for a certain setting.

## 4.2 The Knowledge Base

A UAV instance at a particular time is a tuple  $U = (t, T, L, V, \theta)$  where  $t$  is the time,  $T$  is the vehicle type (emergency or non-emergency in our case),  $L$ ,  $V$  and  $\theta$  are the location, velocity and travel direction of  $U$  at time  $T$  respectively. In addition, it is initially believed that all directions are safe to travel and no traffic jam occurs for the direction. These information, together with other context-related information can be represented in the KB using defeasible rules, as shown below:

$$\begin{aligned}
 EM &: \Rightarrow_{\text{BEL}} \neg \text{isEmergencyVehicle} \\
 DIR &: \Rightarrow_{\text{BEL}} \text{currentDirectionTravelSafe} \\
 PT &: \Rightarrow_{\text{BEL}} \text{pathTo}(X) \\
 STT &: \Rightarrow_{\text{BEL}} \text{safeToTravel}(X) \\
 TJ &: \Rightarrow_{\text{BEL}} \neg \text{trafficJam}(X) \\
 CM &: \Rightarrow_{\text{BEL}} \text{currentMove}(EAST)
 \end{aligned}$$

The above rules<sup>7</sup> describes the propositional concerns of a particular UAV at a particular context of travel. It is assumed that the UAV is not an emergency vehicle and believed that there exist paths at different directions which are safe to travel and without traffic jams; the last rule states that the UAV is current moving towards EAST.

<sup>7</sup>The value of  $X$  in STT and TJ should be interpreted as the four directions of travel that we are considering, i.e., EAST, SOUTH, WEST and NORTH

Please note that all rules above are defeasible since they are the common norms stored in every UAVs. Information about a particular vehicle at particular time should be changed accordingly. New rules should be added (to be discussed in the following section) to the KB to rebute the norms as necessary.

```

DIR04:                                currentDirectionTravelSafe  $\Rightarrow_{\text{INT}}$  continuousTravel
DIR05:                                INT continuousTravel  $\Rightarrow_{\text{INT}}$   $\neg$ changeDirection
CD01 :    INT continuousTravel, safeToTravel(X), currentMove(X)  $\Rightarrow_{\text{INT}}$  Move(X)
VC01 :                                vehicleCollisionAt(X)  $\Rightarrow_{\text{BEL}}$   $\neg$ safeToTravel(X)
DIR11:                                 $\neg$ safeToTravel(X), currentMove(X)  $\Rightarrow_{\text{BEL}}$   $\neg$ currentDirectionTravelSafe
DIR03:                                 $\neg$ currentDirectionTravelSafe  $\Rightarrow_{\text{INT}}$  changeDirection
CD11 : INT changeDirection, safeToTravel(X), pathTo(X),  $\neg$ trafficJam(X)  $\Rightarrow_{\text{INT}}$  Move(X)
CD21 :                                INT changeDirection, currentMove(X)  $\Rightarrow_{\text{INT}}$   $\neg$ Move(X)
DIR03 > DIR05
CD21 > CD11

```

Under normal situation, if no traffic jam occurs and the current travel direction is safe to travel, the UAV should continues to travel without changing its travel direction (DIR04, DIR05 and CD01). However, if a traffic jam appears in the current travel direction or if the current travel direction is not safe to travel (for example, a collision may occur, i.e., the UAV may collide with another UAV if both continues with their current travel directions, see Section 4.3 below for details), then the UAV should consider to change its travel direction (VC01 to CD21).

### 4.3 Collision detection

Now consider two UAVs ( $U_1$  and  $U_2$ ) traveling through the city to different destinations. Using simple geometry methods, an interception location  $L_{int}$  of the two UAVs can be calculated based on their current locations and directions of travel. Let  $t_{int}$  be the time required for a UAV to travel from its current location to  $L_{int}$ .

So two UAVs are considered to be possibly collided if their time required to travel from their respective locations to  $L_{int}$  is within a proximate time interval. That is, in our system, if  $|t_{int_1} - t_{int_2}| < t_{limit}$ , where  $t_{limit}$  is the time boundary, then a collision between the two UAVs will occur. To avoid the collision from occurs a new rule indicating this situation should be added to the UAV's KB. For example, after some numerical calculation, if there exist a possible collision at direction EAST, the following rule should be added to the KB:

```

STT11:  $\Rightarrow_{\text{BEL}}$   $\neg$ safeToTravel(EAST)
STT11 > STT

```

The above rule tells the UAV that it is not safe to travel to the EAST and thus forbidding it from traveling to that direction. Moreover, if the UAV is current traveling towards the EAST, it should also also its travel direction (also the route) as well.

## 4.4 Right of way

But things are not that simple. In reality, we may have some vehicles that cannot afford to pay the price of changing their route. Consider the scenario where an accident occur, an ambulance is required to arrive at the location of incident within certain time limit and start the rescue operations. In this situation, vehicle(s) which are not emergency should give their way to the emergency vehicle(s), and can be represented using defeasible rules as follows:

EM01:  $BEL_{isEmergencyVehicle} \Rightarrow_{OBL} \neg changeDirection \otimes negotiatePathWithVehicle$   
EM02:  $BEL_{emergencyVehicleComing} \Rightarrow_{OBL} changeDirection$   
EM02 > EM01

The above rules state that, under normal situations, there is an obligation that a vehicle should give way to an emergency vehicle and an emergency vehicle should not alter its route. However, when two emergency vehicles travelling in different directions meet, they should then start negotiating with each other on who should change their travel directions.

## 5 Simulation results

The simulator we have developed can be configured with an arbitrary number of UAVs in the environment and the environment can also be configured according to the characteristics and behaviors that we want to evaluate. In this section, we present several simulation results based on the discussions above.

Each UAV is configured to update the current traffic condition (from the GPS monitor) every 200ms and will update the rules for the contextual information (in the knowledge base) accordingly. As described before, based on these information, the perception-action cycle will trigger the behavior controller to decide the new traveling direction in case some unexpected circumstances (such as a traffic jam) may occurred. Besides, in order to make sure that the UAV is traveling at its optimum path, the UAV is also configured to update its travel direction every 4 seconds, based on the current traffic conditions.

Figure 5a shows a typical scenario of the response time required for a UAV to update its current travel status. On average a UAV took about 280ms to update its current travel status; however, these value may increase as the number of vehicles in the simulator increase (Figure 5b). The high value at the beginning are due the initialization of the behavior controller (i.e., the rule engine) while the high value at time=6 are due to the fact that some unexpected circumstances occurred and travel status update is required.

### 5.1 Escape from Traffic Jam

A traffic jam occurs when the number of vehicles in a road exceeds its capacity within a certain time frame. As shown in 6a, if there is no traffic jam, the

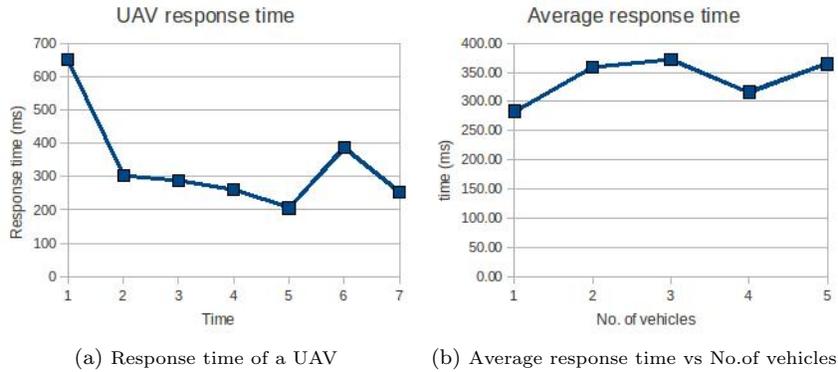


Figure 5: UAV response time

red car should continuous its current travel direction and moving downward. However, if a car exists and blocks its traffic, the UAV then update its travel status and continuous its journey by traveling to the left.

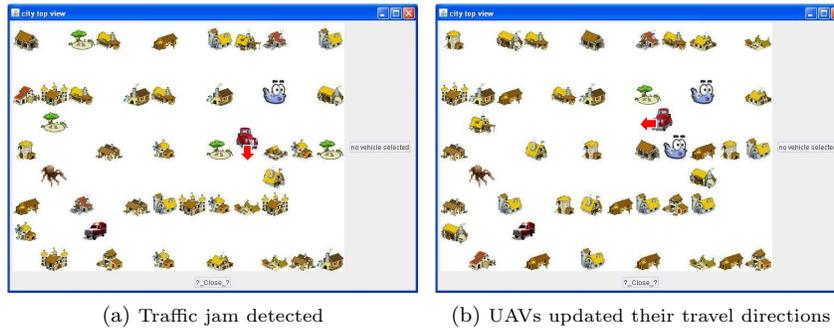


Figure 6: Traffic Jam

## 5.2 Collision detection

As discussed before, it is not uncommon for two UAVs traveling through the city at different directions to be on a collision path. A UAV should be able to identify whether its current travel direction is safe, and should be able to determine whether a possible collision with other UAV may occur and update its travel status accordingly. To address this issue we have conducted the scenario inclusive of two different UAVs. As shown in Figure 7 the two UAVs move away from each other when possible collision are detected.

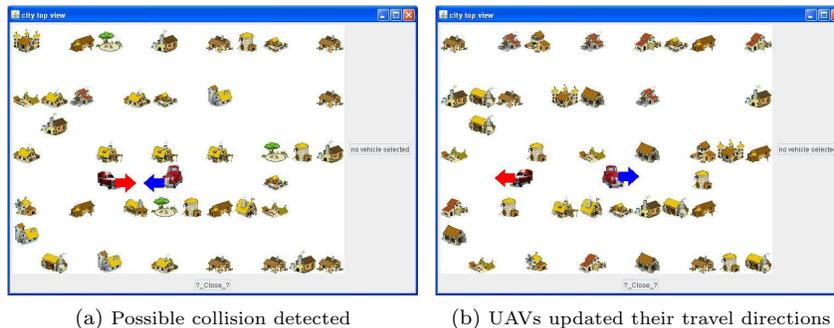


Figure 7: Collision detection

## 6 Conclusion

In this paper we have demonstrated how to use a combination of numerical computation methods and rule based logic computation to simulate a complex environment to solve the Vehicle Routing Problem in an urban canyon environment. Future work includes the study of UAV negotiation, the use of Temporal Defeasible Logic and integrating the rule-base system with reaction-based mechanism.

Besides the combination of numerical and logical techniques, in particular an extension of the BIO approach to the development of agents in defeasible logic. The extension combines the ability of handle violations and degree of preferences. Furthermore, we presented a novel algorithm for the computation of the extension of a defeasible modal theory. The novelty of the algorithm is that it does not require to transform a theory to remove defeaters and superiority relation as need in previous instalments of the algorithm [15], and the algorithm maintains a linear time computation in the size of the theory. The avoidance of the transformations just mentioned allows us to reduce the size of the theory by at least a factor 12, with the consequent speed up of the execution of the computation (this was also confirmed by experimental results were speed us of a few order of magnitude were possible on theory with particular shapes.)

## Acknowledgements

NICTA is funded by the Australian Government as represented by the Department of Broadband, Communications and the Digital Economy and the Australian Research Council through the ICT Centre of Excellence program.

## References

- [1] Grigoris Antoniou, David Billington, Guido Governatori, and Michael J. Maher. On the modeling and analysis of regulations. In *Proceedings of the*

*Australian Conference Information Systems*, pages 20–29, 1999.

- [2] Grigoris Antoniou, David Billington, Guido Governatori, and Michael J. Maher. A flexible framework for defeasible logics. In *Proc. American National Conference on Artificial Intelligence (AAAI-2000)*, pages 401–405, Menlo Park, CA, 2000. AAAI/MIT Press.
- [3] Grigoris Antoniou, David Billington, Guido Governatori, and Michael J. Maher. Representation results for defeasible logic. *ACM Transactions on Computational Logic*, 2(2):255–286, 2001.
- [4] David Billington, Vladimir Estivill-Castro, René Hexel, and Andrew Rock. Architecture for hybrid robotic behavior. In Emilio Corchado, Xindong Wu, Erkki Oja, Álvaro Herrero, and Bruno Baruque, editors, *HAIS*, volume 5572 of *Lecture Notes in Computer Science*, pages 145–156. Springer, 2009.
- [5] G. Brewka, S. Benferhat, and D. Le Berre. Qualitative choice logic. *Artificial Intelligence*, 157:203–237, 2004.
- [6] Matteo Cristani and Elisa Burato. A complete classification of ethical attitudes in multiple agent systems. In *8th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2009)*, pages 1217–1218, 2009.
- [7] Mehdi Dastani, Guido Governatori, Antonino Rotolo, and Leendert van der Torre. Preferences of agents in defeasible logic. In *AI 2005: Advances in Artificial Intelligence, 18th Australian Joint Conference on Artificial Intelligence*, pages 695–704. Springer, 2005.
- [8] Mehdi Dastani, Guido Governatori, Antonino Rotolo, and Leendert van der Torre. Programming cognitive agents in defeasible logic. In *Logic for Programming, Artificial Intelligence, and Reasoning, 12th International Conference*, pages 621–636. Springer, 2005.
- [9] Guido Governatori. Representing business contracts in RuleML. *International Journal of Cooperative Information Systems*, 14(2–3):181–216, 2005.
- [10] Guido Governatori, Vineet Padmanabhan, Antonino Rotolo, and Abdul Sattar. A defeasible logic for modelling policy-based intentions and motivational attitudes. *Logic Journal of the IGPL*, 17(3):227–265, 2009.
- [11] Guido Governatori and Duy Hoang Pham. A defeasible logic for modelling policy-based intentions and motivational attitudes. *International Journal of Business Process Integration and Management*, 5, 2009.
- [12] Guido Governatori and Antonino Rotolo. A Gentzen system for reasoning with contrary-to-duty obligations. a preliminary study. In Andrew J.I. Jones and John Horty, editors, *Δeon’02*, pages 97–116, London, May 2002. Imperial College.

- [13] Guido Governatori and Antonino Rotolo. Defeasible logic: Agency, intention and obligation. In Alessio Lomuscio and Donald Nute, editors, *Deontic Logic in Computer Science*, number 3065 in LNAI, pages 114–128, Berlin, 2004. Springer.
- [14] Guido Governatori and Antonino Rotolo. Logic of violations: A Gentzen system for reasoning with contrary-to-duty obligations. *Australasian Journal of Logic*, 4:193–215, 2006.
- [15] Guido Governatori and Antonino Rotolo. BIO logical agents: Norms, beliefs, intentions in defeasible logic. *Journal of Autonomous Agents and Multi Agent Systems*, 17:36–69, 2008.
- [16] Guido Governatori and Antonino Rotolo. A computational framework for institutional agency. *Artificial Intelligence and Law*, 16(1):25–52, 2008.
- [17] Guido Governatori, Antonino Rotolo, and Giovanni Sartor. Temporalised normative positions in defeasible logic. In *ICAAIL '05: Proceedings of the 10th international conference on Artificial intelligence and law*, pages 25–34, New York, NY, USA, 2005. ACM.
- [18] Stefan Hrabar, Gaurav S. Sukhatme, Peter Corke, Kane Usher, and Jonathan Roberts. Combined optic-flow and stereo-based navigation of urban canyons for a UAV. In *Proceedings of the IEEE International Conference on Intelligent Robots and Systems, 2005 (IROS 2005)*, Aug 2005.
- [19] Ho-Pun Lam and Guido Governatori. The making of SPINdle. In Adrian Paschke, Guido Governatori, and John Hall, editors, *Proceedings of The International RuleML Symposium on Rule Interchange and Applications (RuleML 2009)*, pages 315–322, Las Vegas, Nevada, USA, 5-7 Nov. 2009. RuleML, Springer.
- [20] Gilbert Laporte and Ibrahim H. Osman. Routing problems: A bibliography. *Annals of Operations Research*, 61(1):227–262, 1 December 1995.
- [21] Michael J. Maher. Propositional defeasible logic has linear complexity. *Theory and Practice of Logic Programming*, 1(6):691–711, 2001.
- [22] Insu Song and Guido Governatori. Hardware implementation of temporal nonmonotonic logics. In *19th Australian Joint Conference on Artificial Intelligence*, pages 808–817, 2006.
- [23] Michael Wooldridge. *An introduction to Multi-Agent Systems*. John Wiley & Sons, 2009.