A model to Coordinate UAVs in urban environments using defeasible logic

Ho-Pun Lam[1,2], Subhasis Thakur[1,3], Guido Governatori[1] and Abdul Sattar[1,3]

[1] NICTA, Australia
[2] ITEE, The University of Queensland, Brisbane, Australia
[3] IIIS, Griffith University, Brisbane, Australia

Abstract. In this paper we show how a non-monotonic rule based system (defeasible logic) can be integrated with numerical computation engines. To this end we simulate a physical system from which we obtain numerical information. The physical system perceives information from its environment and it sends some predicates which are used by the defeasible logic reasoning engine to make decisions and then these decisions are realized by the physical system as it takes action based on the decision made by the reasoning engine. We consider a scenario where UAVs have to navigate through an urban environment. The UAVs are autonomous and there is no centralized control. The goal of the UAVs is to navigate without any collisions with each other or with any building. In case of a possible collision, the concerned UAVs communicate with each other and use background knowledge or some travel guidelines to resolve the conflicts.

1 Introduction

Typically complex systems have to manipulate and to react to different types of data (e.g., numerical and Boolean), and in many scenarios we have to integrate different types of reasoning processes. For example in a UAV (Unmanned Autonomous Vehicle) scenario, a UAV has to use some sensing devices to determine its position and the position of obstacles and the other information about other UAVs (maybe represented as a vector indicating the speed, direction of a UAV). 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 determined by the two vectors, and determine whether the two UAV will reach the intersection point at the same time (or within a proximity threshold). These two operations typically require some numerical computation. In case the previous computation returns that a collision is possible the UAV has to decide whether to change direction and how to change direction. While it is possible to use non logical methods for these two tasks, experience tell us that these are better handle by rule based methods[4].

[4] If you are not convinced about this, please, spend few seconds thinking about it the next time that (1) drive your car home after work, (2) take some form of public transportation, (3) simply cross the road. For a concrete case consider rule 162 of the Australian Civil Aviation Regulations 1988: “When two aircraft are on converging headings at approximately the same height, the aircraft that has the other on its right shall give way, except that (a) power-driven heavier-than-air aircraft shall give way to airships, gliders and balloons; . . . ”

UAV coordination is a large research topic and a huge literature is dedicated to this. 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 for different scenarios as new constraints are introduced about how the UAVs should behave (2) the decision making system has linear computational complexity, it can even be modelled as hardware.

The physical system of a UAV gathers data such as its current position, and the position, velocity and travel direction of UAVs nearby and the alternative path(s) available as contextual information, and will be used to determine whether a possible collision will occur. If it is the case then some of these information will be "transformed" into a set of rules and then merged to some pre-defined guidelines (which appears as a set of rules in a logic and is available to all UAVs) so that the reasoning engine can reason to determine how to evade the possible collision and to arrive the desired destination safely.

The paper is organized as follows: In Section 2 we briefly introduce the scenario that we deal with; then in Section 3 we motivate the reasons why we use defeasible logic and an informal introduction about the subject will be provided. In Section 4 we illustrate the various reasoning techniques used to model the UAV navigation by means of the system we have developed.

2 UAV coordination problem

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

> Given a map of city and specific targets, a group of UAVs have to navigate through the city from a start location to a desired destination without colliding with other UAVs.

In the light of this we have implemented a UAVs navigation system to simulate this situation. Figure 2 captures the essence of components of our UAV navigation system. Each UAV is equipped with a GPS application engine and a reasoner. The application engine is used to request and receive information from the GPS monitor. In each reasoning cycle (see below for details), the application engine will issue request to the GPS monitor on the current traffic situation and information of the UAVs nearby; and if, on the other hand, an accident has occured at a particular location, the GPS monitor will also issue acknowledgement to the UAVs (which are close to the accident location) about the accident.

The reasoning engine is used to control the behavior of a UAV. It share certain aspects of consciousnes and the doctorine behind how the UAVs render their decision under different context. Simply speaking, the UAV's traffic is regulated by a set of 'road rules' (the knowledge base) determining which UAVs have the right of way in case they are travelling to the same location and has a high probability to collide.

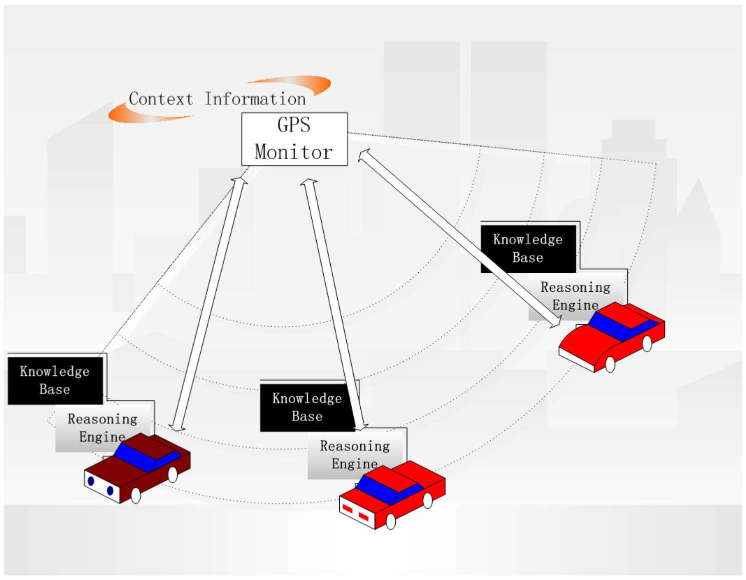

Fig. 1. UAV system overview

The UAVs navigation system uses the following reasoning cycle:

1. A UAV gathers data about its current location, travel direction and inforamtion of UAVs nearby from the GPS monitor.
2. The UAV detects if there is any possible collision (Section 4.2).
3. In case of possible collision, the UAV utilize the ‘right-of-way’ rules to reason the next direction of travel, or to stop its motion for a while. The ‘right-of-way’ rules are modelled in Defeasible Logic (See Section 3 for a short introduction to the logic).
4. If the resulted travel direction D is changed, except the case of stop moving, the UAV should then select a new path (using shortest path length, least number of turns, etc) along the set of paths in the new direction and continuous its travel
5. Go back to step 1 until the UAV reaches its desired destination.

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 formalised 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. 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 reasons with norms [2–4]. 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.

A Defeasible theory [1] 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 $\sim l$.

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

Provability is based on the concept of a derivation (or proof) in $D = (F,R,>)$. 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 $P = (P(1),\dots,P(n))$ of tagged literals satisfying proof conditions (which correspond to inference rules for each of the four kinds of conclusions). $P(1..i)$ denotes the initial part of the sequence P of length i. For a full presentation and proof conditions of DL refer to [1].

The set of conclusions of a defeasible theory is finite[5], and it can be computed in linear time [6]. The reasoning engine can be also implemented as a chip [7]. For the application at hand we have the SPINdle Java implementation of defeasible logic [5]. SPINdle is able to handle defeasible theories with over 1,000,000 rules [5].

4 UAV Navigation Problem

In this section we present a novel version of UAV navigation problem to illustrate the framework that we have developed in the pape. We will also describes some of the rules that appeared in our knowledge base, and will show to the reader under what circumstance new rule(s) will be added to the knowledge base to rebute the norms.

[5] It is the Herbrand base that can be built from the literals occurring in the rules and the facts of the theory.

The objective of our framework is to have all UAVs traveled to the destination (in a city environment) without colliding with each others. To simplify our framework, we considered only four values of travel directions, namely: NORTH, EAST, WEST and SOUTH, as shown in Figure 4 below.

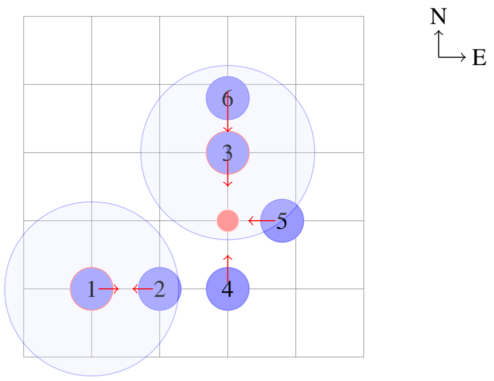

Fig. 2. City with UAVs

4.1 The Knowledge Base

As described in Section 2 a UAV will gather different types of data from the GPS monitor within a proximate range and detects if a possible collision may occur. In case of possible collisionthe UAV will then utilize the information in its Knowledge Base (KB) andd to reason on the next travel direction, or to stop its motion. The KB contains the information about a UAV and is a well-documentated (doctrine) limited set of behavioral interactions that describes the behavior of the UAV under diffferent situation. To be able to travel from one location, and to aviod collision with other UAVs, a UAV should incorporate into the KB the set of context-related information (such as traffic situation, information of other UAVs) and derive a safe direction of travel when a possible collision occur. (Consider the scenario as shown in Figure 4 that V_1 and V_2 are moving towards to the same location and collisions may occur if none of the UAVs alter their travel directions. (the same also applies to (the same also applies to V_3, V_4 and V_5))

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), L, V and θ are the location, velocity and travel direction of U at time T respectively. These information, together with other context-related information will be added to the knowledge base using defeasible rules, as shown below:

$$\begin{array}{ll}
\text{EM01}: & \Rightarrow \neg \mathit{isEmergencyVehicle} \\
\text{DIR01}: & \Rightarrow \mathit{currentDirectionTravelSafe} \\
\text{STT01}: & \Rightarrow \mathit{safeToTravel}(X) \\
\text{TJ01}\ \ : & \Rightarrow \neg \mathit{trafficJam}(X) \\
\text{CM01}: & \rightarrow \mathit{currentMove}(EAST)
\end{array}$$

The above rules[6] describe 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 belived that there exist paths at different directions which is safe to travel and with no traffic jam, and is currently travelling to the EAST.

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

So, under normal situation, if no traffic jam occurs at the current travel direction and the current travel direction is safe to travel, the UAV should then continues its travel without any change of its direction (rules 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 will occurs, i.e., the UAV will collide with another UAV if both continues with their current travel directions, see Section 4.2 below for the details), then the UAV should consider to alter its travel direction (rules VC01 to CD21).

4.2 Collision detection

Now consider two UAVs (U_1 and U_2) traveling through the city to different destinations. Using their current location and travel direction, and with the help of some simple geometry methods, an interception location L_{int} between the two UAVs can be calculated.

Let t_{int} be the time required for a UAV to travel from the UAV's 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_{inf} is more-or-less the same. That is, in our case, if $|t_{int_1} - t_{int_2}| < t_{limit}$, where t_{limit} is the time boundary, then a possible collision may occur between the two UAVs and a new rule indicating this situation will be added to the KB, to indicate the current situation. For example, if there exist a possible collision at direction EAST, the following rule will be added to the KB:

$$STT11 : \Rightarrow \neg \mathit{safeToTravel}(East)$$
$$STT11 > STT01$$

The above rule tells the UAV that it is not safe to travel to EAST and thus should forbidding it from traveling to that direction. Moreover, if the UAV is traveling towards the EAST, it should also consider alter its travel direction, or to stop for a while, as well.

[6] The value of X in DIR01 and STT01 should be interpreted as the four directions of travel that we are considering, i.e. EAST, SOUTH, WEST and NORTH.
Please also note that all rules above (except CM01) are defeasible since they are the common norms that stored in every UAVs. Information about a particular UAV should be added to the KB to rebute the norms as necessary.

4.3 Right of way

However, things in reality are much more complex. We may have some emergency vehicles traveling through the city that cannot afford to pay the price of changing their path. In this situation, vehicles which are not emergency should give their way to the emergency vehicles, which can be represented in defeasible logic as follows:

EM02: $\mathit{emergencyVehicleComing}, \mathit{isEmergencyVehicle} \Rightarrow \mathit{negotiatePathWIthEmergencyVehicle}$
EM03: $\mathit{emergencyVehicleComing}, \neg\mathit{isEmergencyVehicle} \Rightarrow \mathit{changeDirection}$
EM04: $\neg\mathit{emergencyVehicleComing}, \neg\mathit{isEmergencyVehicle} \Rightarrow \mathit{negotiatePathWithVehicle}$
EM06: $\neg\mathit{emergencyVehicleComing}, \mathit{isEmergencyVehicle} \Rightarrow \mathit{requestVehicleToChangeDirection}$

The above rules stated that if the vehicle is not an emergency vehicle, than it should give the right of travel to an emergency vehicle. However, if both vehicle are (not) emergence vehicles, them they should negotiate on who should change their travel direction.

After gathering all the contextual information, as mentioned before, new rules will be added to the KB. The UAV will thus start to to reason out the action on how to evade the possible collision. In most cases, as we are expected, the UAV will alter its travel direction, to avoid possible collision and gives the 'rights-of-way' to emergency vehicles. However, there are in some situations when the UAV(s) has no direction safe to travel, it will stop for a while and let other UAVs to continuous their journery first.

5 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 such as UAV navigation. Future work includes the study of UAV negotiation, the use of Temporal Defeasible Logics, and integrating the rule-based system with reaction-based mechanism.

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. Antoniou, Grigoris and Billington, David and Governatori, Guido and Maher, Michael J. Representation Results for Defeasible Logic. ACM Transactions on Computational Logic,2, 255-287, 2001.
2. Antoniou, Grigoris and Billington, David and Governatori, Guido and Maher, Michael J. On the Modeling and Analysis of Regulations. Proceedings of the Australian Conference Information Systems, 1999.

3. Governatori, Guido. Representing Business Contracts in RuleML. International Journal of Cooperative Information Systems, 14, 181-216, 2005.
4. Governatori, Guido Antonino Rotolo and Giovanni Sartor. Temporalised Normative Positions in Defeasible Logic. 10th International Conference on Artificial Intelligence and Law, 2005.
5. Lam, Ho-Pun and Governatori, Guido. The Making of SPINdle. 3rd International RuleML Symposium, 2009.
6. Maher, Michael J.. Propositional Defeasible Logic has Linear Complexity. Theory and Practice of Logic Programming, 1, 691-711, 2001.
7. Song, Insu and Governatori, Guido Hardware Implementation of Temporal Nonmonotonic Logics. 19th Australian Joint Conference on Artificial Intelligence, 2006.