Symbolic Reasoning About Dynamic Systems in Conflict Alert Situations

Mei Li

August 5, 2007

Contents

1 Introduction of the Problem **1**

1.1 Goal of Research 1

1.2 The Example Problem 2

1.3 Current Solutions 3

1.3.1 System Components 4

1.3.2 System History 8

1.3.3 TCAS Vertical Resolution Advisory Selection 10

1.3.4 TCAS Horizontal RA 15

1.4 Design Considerations 16

2 Literature Review **19**

2.1 Categorization 19

2.2 Selected Papers and Comments 22

2.3 Summary 27

3 Mathematical Formulation of a Quantitative Method **30**

3.1 Vertical vs. Horizontal Maneuvers 30

3.2 Components of the Goal Function of the Optimization Problem 31

3.3 Optimization Problem Considerations 32
3.3.1 Mathematical Definitions of PROCON, LINCON and MANCON 33
3.3.2 Mathematical Formulation of the Goal Function . . . 35
3.4 Optimization and Possible Maneuvers 36
3.5 Complexity 38

4 Q^2 Approach **42**
4.1 Introduction to General Dynamic Systems and Q^2 Symbolic Reasoning 43
4.1.1 General Dynamic System 46
4.1.2 Q^2 Representation and Consistency of Symbolic Reasoning 48
4.1.3 Qualitative Regions 51
4.2 Application of Q^2 to Conflict Alert Resolution 52
4.2.1 Partitioning of System Spaces in Conflict Alert Situation 53
4.2.2 Partitioning of Output Space and Qualitative Outputs 56
4.2.3 Partitioning of State Space and Qualitative States . . 57
4.2.4 Partitioning of Input Space and Qualitative Inputs . 60
4.2.5 A Moore Machine Representation of the Dynamical System 60
4.2.6 Partitioning of Input Space with MANCON Included 62
4.2.7 Converting the Mealy Machine to a Moore Machine . 64
4.3 Pseudo Code for the Level 1 Partitioning Algorithm 68
4.3.1 Algorithm (P_Ω) for Partitioning the Output Space for the Conflict Resolution Problem 69
4.3.2 Algorithm (P_Θ) for Partitioning the State Space . . . 71

4.3.3 Algorithm (P_{Λ_1}) for Level 1 Partitioning of the Input Space 72
4.4 Pseudo Code for the Level 2 Partitioning Algorithm (MANCON included) 73
4.4.1 Algorithm (P_{Λ^2}) for the Level 2 Partitioning of the Input Space 73
4.5 Consistency of Partitioning by Multiple Criteria 75

5 Symbolic Reasoning for Con

p163rtC36Tc90op4969s3644f-276ninDpA-

6.1.2 ATC Quantitative Method 93
6.1.3 Middle Ground Method 94
6.1.4 Time Distribution of the Methods 96
6.2 Quality of Maneuvers 99

7 Results **101**
7.1 Scenarios 101
7.2 Computed Maneuvers 105
7.3 Computation Time 107
7.4 Evaluation of Maneuver Quality 111

8 Generalization of the Q^2 Approach **113**
8.1 Types of General Dynamic Systems 113
8.1.1 Static and Dynamic Systems 113
8.1.2 Linear and Non-linear Systems 114
8.1.3 Example for Testing 115
8.2 Generalization of Partitioning and Reasoning 115
8.2.1 Pseudo Code for Partitioning the Output Space of Linear Dynamic Systems 117
8.2.2 Pseudo Code for Partitioning the State Space of Linear Dynamic System 118
8.2.3 Pseudo Code for Partitioning the Input Space of Linear Dynamical Systems 120
8.2.4 Partitioning Complicated Cases of Linear Dynamical Systems 121
8.2.5 Generalization of Reasoning 123
8.3 Testing of Generalized Approach 125

8.3.1 The Circuit Problem 126
8.3.2 The ATC problem 129

9 Conclusion **132**

9.1 Contributions 132
9.2 Possible Future Research 133

Abstract

This thesis was motivated by the problem of computational complexity of conflict alert and conflict resolution in air traffic control (ATC). A symbolic reasoning approach is proposed to solve this problem with higher computational efficiency. Then this approach is generalized to solve similar problems for linear dynamic systems. Finally this general method is tested against both the ATC conflict alert problem and another example of the linear dynamic system. Even though the final product of this thesis is a general method of symbolic reasoning, the emphasis is on the ATC example.

Organization of the thesis is as follows. The background information about ATC conflict alert handling is given first. Then some literature review on this topic is presented. Next the mathematical formulation of our example problem is constructed. This forms the base of the research problem. From the complexity analysis, the urgency of a more efficient approach appears. At this point the qualitative approach is introduced. The symbolic reasoning is performed for this ATC example. After this, quantitative and qualitative methods are discussed and compared. The results from the simulations are shown. As the final step, generalization of the symbolic reasoning approach is formed and tested.

Chapter 1

Introduction of the Problem

1.1 Goal of Research

It was shown in [36] that for a given continuous dynamic system, a consistent qualitative representation (an abstraction of the continuous dynamic system) can be constructed such that the results of symbolic logical reasoning within the qualitative representation hold in the underlying quantitative dynamic system. In that formalization, an example of a consistent qualitative/quantitative representation of a simple dynamic system (for which a closed form model was known) and of reasoning using such a representation, was provided. The main idea of the approach presented in [36] was to partition the spaces – input, state and output - of a dynamic system (Cartesian products of variables) into subregions and assign a unique symbol to each of such partitions. An automaton was then constructed to capture the qualitative behavior of a dynamic system. For more complicated processes, complete knowledge, such as known mathematical models, may not always be available, and thus other means are needed to develop qualitative repre-

sentation proposed in [36].

In the research described in this thesis, an example from the air traffic control field [76] is used. In this example, the objective is to generate conflict alerts whenever two aircraft are too close or predicted to become too close, and to generate maneuver advisories such that, if executed, will cause avoidance of a collision under arbitrary behavior of the other aircraft. It is our goal to provide experimental evidence of the appropriateness of the qualitative/quantitative approach and its computational efficiency to solving problems like the conflict alert and avoidance problem. Furthermore, a method for establishing consistent partitions of linear dynamic system(GDS) spaces that define qualitative abstractions through associating the partitions with the inputs, states and outputs of a qualitative dynamic system (QDS) are developed. Finally, an algorithm for finding solutions through qualitative reasoning is presented. The complexity of the qualitative approach is compared with the quantitative approach.

1.2 The Example Problem

In air traffic control, two or more aircraft approaching each other within a close distance will create a conflict alert situation.

Traditionally, alerts are issued using quantitative algorithms based upon tracking and prediction of the aircraft positions. There are a few systems that can generate alerts. One is an automated system used by air traffic controllers in the tower to track aircraft positions measured by radars. This automation system performs conflict alert functions (without generating maneuver instructions) and the human controllers give maneuver instructions. Another system is used by pilots. It is an on-board system known as Traffic-

alert and Collision Avoidance System (called TCAS [1, 2]). The TCAS uses the sensor mounted on the aircraft to monitor positions of intruders. It generates alerts when close encounters occur. When an alert is issued, TCAS gives the pilot maneuver instructions, known as resolution advisories (RA's). Traditional methods have their limitations, which will be discussed in the next sections.

Our intent is to develop a new algorithm to compute resolution advisories assuming positions and velocities of the targets in the encounter are known by means of sensor measurement and tracking. This algorithm can be either in the avionics on board, or it can be in the automation system on the ground. The sensor for such purpose can be either a regular short range or long range radar, or it can be a GPS based on ADS-B broadcasting. This algorithm should overcome some of the limitations of traditional methods.

1.3 Current Solutions

The Traffic Alert and Collision Avoidance System (TCAS) is an airborne system developed by the FAA that operates independently from the ground-based Air Traffic Control (ATC) system. Other systems, such as Automatic Collision Avoidance System (ACAS), operate more or less in a similar way. More recent developments, such as the "Sense and Avoid" concept, also borrow basic concepts from TCAS. In fact, any type of an alerting and resolution system is equipped with some sort of sensor to get the measurements, then it checks the alert logic, then (when necessary) it computes maneuvers to avoid conflicts. Note that the alerting and resolution algorithm itself is not limited to avionics or to aircraft-based systems, such as TCAS. Since TCAS is a typical collision avoidance system, we briefly overview it below.

TCAS was designed to increase cockpit awareness of the proximity of an aircraft to other aircraft and to serve as a "last line of defense" for the prevention of mid-air collisions [30]. There are two levels of TCAS systems: TCAS I and TCAS II (all future developments are at the level of TCASII):

- TCAS I was developed to accommodate the needs of the general aviation (GA) community and of the regional airlines. This system issues Traffic Advisories (TAs) to assist pilots in the visual acquisition of intruder aircraft. TCAS I is mandated on aircraft with 10 to 30 seats, although TCAS II may be installed, instead.

- TCAS II is a more sophisticated system which provides the information of TCAS I, and also analyzes the projected flight path of approaching aircraft and issues Resolution Advisories(RAs) to the pilot to resolve potential mid-air collisions. TCAS II is required internationally in aircraft with more than 30 seats or weighing more than 15,000 kg.

1.3.1 System Components

The system components and various options that are implemented in airline aircraft (cf. [65]) (see Figure 1.1), are described below.

TCAS Computer Unit (CAS Logic): This unit performs airspace surveillance, intruder and own aircraft tracking, threat detection and resolution, advice generation. Pressure and radar altimeter inputs, and other aircraft configuration discrete inputs (not shown in Figure 1.1), are used by the computer to control the collision avoidance logic parameters that determine the protecting volume around the TCAS aircraft. If a tracked aircraft is a collision threat, the computer selects the

best avoidance maneuver, and if the threat aircraft is also equipped with TCAS II, this maneuver is coordinated.

Mode S Transponder: The Mode S Transponder performs the normal ATC functions of existing Mode A&C transponders (explained below). It basically encodes numbers into signals. Because of its selective address capability the mode S transponder is also used to provide air-to-air data exchange between TCAS-equipped aircraft to ensure coordinated, complementary resolution advisories.

Antennas: The Antennas used by TCAS II include a directional antenna which is mounted on top of the aircraft, and an omni-directional antenna. Typically, the directional antenna transmits interrogations on 1030 MHz at varying power levels in each of four 90° azimuth segments. Transponder replies are received on 1090 MHz and are sent to the TCAS computer unit. The directional antenna permits the partitioning of replies to reduce synchronous garbling. An omni-directional transmitting and receiving antenna mounted on the bottom of the aircraft provides range and altitude data on targets that are below the TCAS aircraft.

Displays:

Traffic Advisory Display: The traffic advisory (TA) display depicts the position of the traffic relative to the TCAS aircraft to assist the pilot in visually acquiring intruding aircraft. This display can be either a dedicated TCAS display or a joint-use weather radar and traffic display. Alternatively, in some aircraft the TA display will be an electronic flight instrument system (EFIS) or a

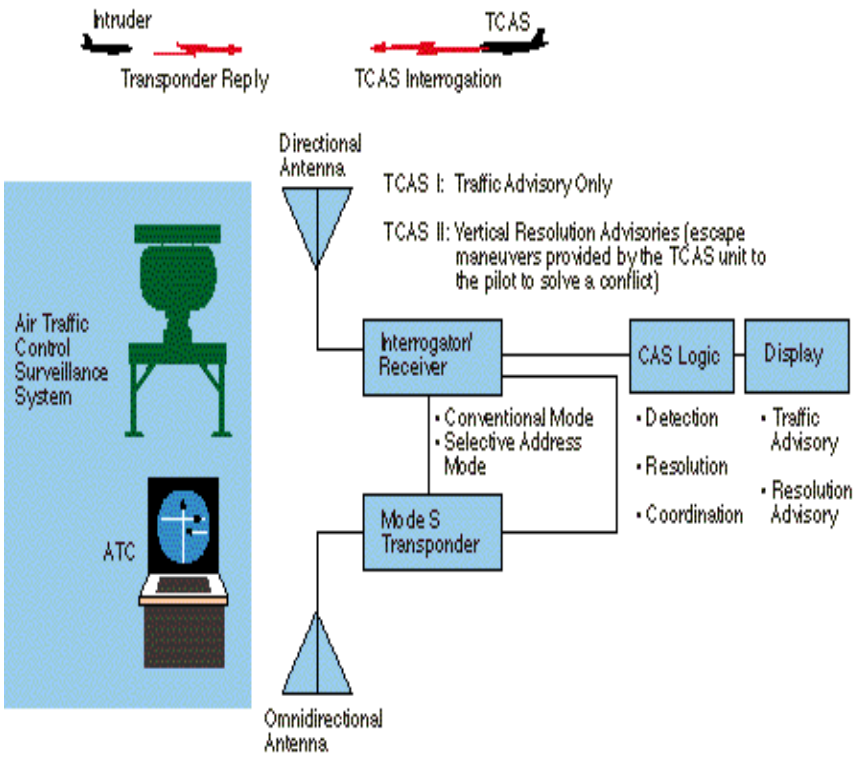

Figure 1.1: TCAS components.

flat-panel-display which combines traffic and resolution advisory information on the same scope face.

Resolution Advisory Display: The resolution advisory (RA) display is a standard Vertical Speed Indicator (VSI), modified to indicate the vertical rate that must be achieved to maintain safe separation from threatening aircraft. The RA display contains segmented red and green eyebrow lights around the vertical speed scale.

Normally, there are two RA displays, one for the Captain and one for the First Officer. In some cases, the TA and RA displays are combined, e.g., traffic information is shown in the center portion of an electronically displayed VSI.

Aural Annunciation: Displayed traffic and resolution advisories are supplemented by synthetic voice advisories generated by the TCAS Computer. The words "Traffic, Traffic" are annunciated at the time of the traffic advisory which directs the pilot to look at the TA display to locate the intruding aircraft. If the encounter does not resolve itself, a resolution advisory is annunciated, e.g., "Climb, Climb, Climb." At this point the pilot adjusts or maintains the vertical rate of the aircraft according to the advisory. This action is reflected on the display by showing that the VSI needle is out of the red segments.

Below we explain some terminology used in the description of TCAS.

Mode A: A Mode A transponder can encode a number into the reply signal. This code is a four digit octal number. "1200" is an example of a Mode A code for the decimal number 640.

Mode C: A Mode C transponder can encode its altitude into the reply signal. This code is known as the "Grey Code", and it encodes 100 ft. increments into 12 bits. Note that Mode C transponders can also encode Mode A, and that ground radar (i.e., radar on the ground,

nearby aircraft. For aircraft with Mode C or S transponders, the TCAS II display can generate an RA, which commands vertical maneuver (climb/descend) to avoid nearby co-altitude traffic. For aircraft with Mode S transponders and TCAS II equipment, RAs will be coordinated between aircraft (e.g., the two TCAS II processors will cooperatively agree to send one aircraft in a climb and the other in a descent.) Note: aircraft equipped with TCAS II must have Mode S transponders installed.

TCAS III: Attempted to use the TCAS directional antenna to assign a bearing to other aircraft, and thus be able to generate a horizontal maneuver (e.g. turn left or right). TCAS III has been judged by the industry to be infeasible due to limitations in the accuracy of the TCAS directional antennas. The directional antennas were judged not to be accurate enough to generate an accurate horizontal-plane position, and thus an accurate horizontal resolution.

TCAS IV: Uses additional information encoded by the target aircraft in the transponder reply (i.e., the target aircraft encodes its own position into the transponder signal) to generate a horizontal resolution to be included in an RA. This requires the target aircraft to have some data link capability at a minimum. In addition, some reliable source of position (e.g., GPS) is needed on the target aircraft in order for it to be encoded.

TCAS IV can use Mode S data link capability to encode position information into TCAS replies. TCAS IV development is still underway, but it is not likely to be fielded any time soon, as there are still technical and institutional issues that need to be resolved.

Also, new trends in data link capabilities, such as Automatic Dependent Surveillance Broadcast (ADSB), have popped up recently and have pointed out a need to re-evaluate whether a data link system dedicated to collision avoidance, such as TCAS IV, should be incorporated into a more generic system of air-to-air data link. This kind of data link capability could then be used in other on-board applications, e.g., monitoring of civilian aircraft by military aircraft.

1.3.3 TCAS Vertical Resolution Advisory Selection

The existing TCAS II uses the following logic (cf. [2]) for RA selection. When a threat (conflict alert) is declared, a two-step process is used to select an RA. The first step is to select the sense direction - upward or downward - of the resolution advisory. Based on the range and altitude of the intruder, the CAS (collision avoidance system) logic models the intruder's path to the CPA. The CPA is defined as the vertical plane on which the collision would happen. Figure 1.2 shows the paths that would result if own aircraft climbing or descending to resolve the encounter. The CAS logic computes the predicted vertical separation for each of the two cases and, in the case shown in Figure 1.3, selects the "downward" sense because it provides the greater vertical separation.

However, the CAS logic includes additional criteria in selecting the direction of maneuver. In those cases where an altitude crossing by the threat or the TCAS aircraft is projected, the CAS logic will pick the sense direction that avoids altitude crossing if it can maintain the desired amount of vertical separation at the CPA. The desired amount of separation, referred to as ALIM, varies from 400 feet to 740 feet, depending on own aircraft's altitude

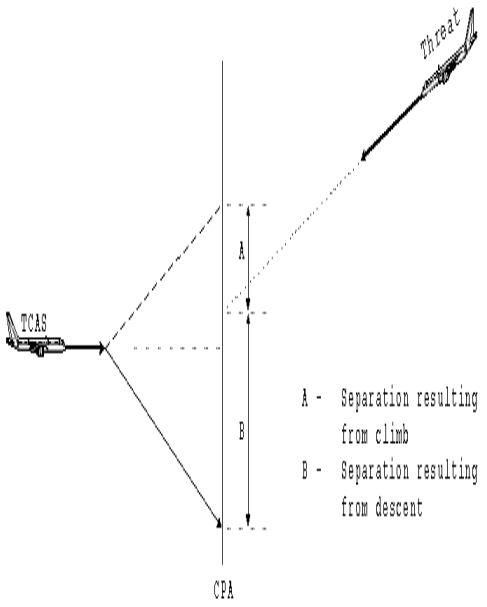

Figure 1.2: RA sense selection

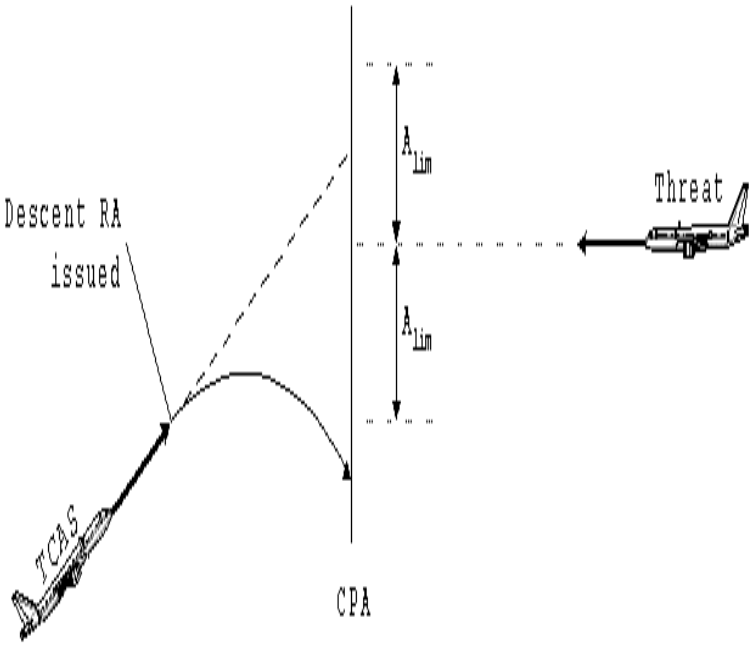

Figure 1.3: Non-crossing RA sense selection

regime. An example is shown in Figure 1.3, where the upward sense is selected because this selection guarantees the ALIM requirement and avoids altitude crossing. However, if ALIM cannot be achieved, a crossing RA (meaning the maneuver makes paths cross at different times) will be issued.

The second step in selecting an RA is to select the strength of the advisory. The least disruptive (meaning the least distance away from the original flight plan) vertical rate maneuver that will still achieve safe separation is selected. Advisory strength will be continuously evaluated and modified if necessary during the course of the encounter.

After the TCAS aircraft has chosen an RA, occasionally a threat aircraft maneuvers vertically in a manner that thwarts the RA. If the threat is not equipped with TCAS, the own aircraft will be advised either to increase its

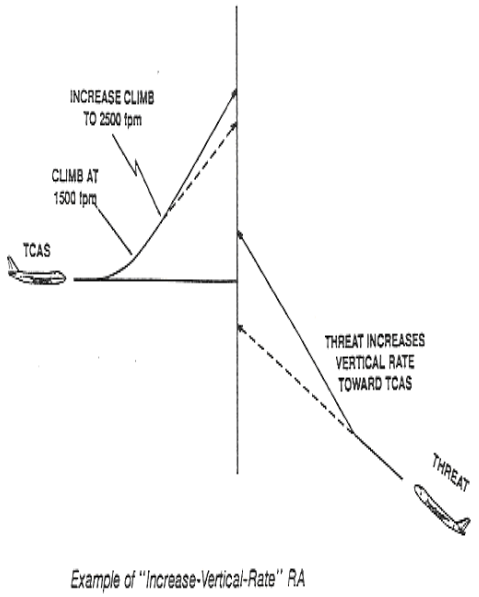

Figure 1.4: "Increase vertical rate" maneuver

vertical rate from 1500 fpm to 2500 fpm or to reverse sense. Examples of "increase-vertical-rate" and "sense-reversal" RAs are shown in Figure 1.4 and Figure 1.5, respectively. For threat aircraft equipped with TCAS II, an unexpected vertical maneuver is handled only by an "increase-vertical-rate" advisory, as "sense-reversal" advisories are not permitted in this case.

Due to aircraft climb performance limitations at high altitude or in the landing configuration, the CAS logic may inhibit a "climb" or "increase climb" advisory during an encounter. These limitations are provided to the CAS logic from an on-board store of geographic information. When such a limitation is identified, the CAS logic will choose a viable alternative RA.

TCAS is able to handle multi-aircraft situations in various ways. First, TCAS will attempt to resolve the situation with a single RA, if it can main-

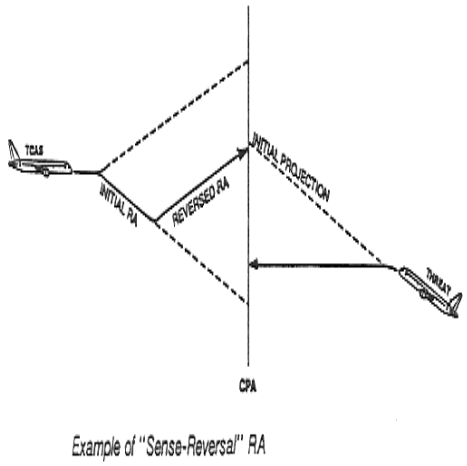

Figure 1.5: "Sense reversal" maneuver

tain safe separation from each of the threat aircraft. Second, it will generate a multiple-maneuver RA that is a sequence of basic maneuvers. The CAS logic avoids "contradictory" climb and descent maneuvers, i.e., such that are not necessary for avoiding collision. The extreme case is when there is no feasible solution; then the decision is to maintain the current altitude.

It should also be noted that all "increase-descent" RAs are inhibited below 1450 AGL and all "descend" RAs are inhibited below 1000 AGL, so that the aircraft flight profile will not fall below the Standard Glide Path (the landing angle above the runway). All RAs are inhibited below 500 AGL.

After the CPA is passed and the range between the TCAS aircraft and the threat aircraft begins to increase, the resolution advisory is canceled and pilots return to their assigned altitude or to their original vertical rate (if climbing or descending).

1.3.4 TCAS Horizontal RA

In TCAS III, the horizontal RA capability is made possible through the use of estimates of the miss distance, i.e., the distance in the horizontal plane between an intruder and a host aircraft at the time of the closest approach. An accurate estimate of an intruder's miss distance offers the capability to issue a horizontal RA, which instructs the host aircraft to turn in the horizontal plane to escape a possible collision (cf. [15]). Depending on the method chosen to calculate the miss distance, five parameters must be known. For the TCAS III method, the five parameters are: range (distance between the intruder and the host aircraft), range rate, bearing (angle between the paths of the two aircraft), bearing rate, and speed of the host aircraft. With these parameters, the miss distance m can be calculated as

$$m = r^2\omega/v, \tag{1.1}$$

where r is the measured relative range between the host and intruder aircraft, ω is the estimated intruder bearing rate, and v is the magnitude of the relative velocity between the two aircraft (which is calculated from three parameters: range rate, bearing of the intruder, and speed of the host aircraft).

An intruder that penetrates the threat boundary may be filtered out by Miss Distance Filtering (MDF) in order to avoid the generation of too many nuisance RAs. The CAS logic selects the appropriate RA based on a comparison of the expected increase in aircraft separation that would result from each valid RA type: climb, descend, turn left, or turn right.

Although TCAS antenna configurations caused errors which were just too large to support accurate MDF and horizontal RAs, the use of Mode S data link and GPS data in TCAS IV promises to provide the data accu-

racies required for horizontal functions as well as for improved vertical RA performance. TCAS IV is under development and a lot of research has been contributed to this direction.

1.4 Design Considerations

In the TCAS systems currently in use, RA's concern maneuvers in the altitude direction, such as to climb or to descend, and at what rate. Having horizontal maneuvers as RA's would provide more efficient usage of the air space, and thus would allow for the accommodation of more aircraft safely in the controlled air space.

When aircraft come into encounter situations, based on the level of coordination among participating aircraft, conflict resolution methods can be classified as non-cooperative and cooperative (cf. [37]). In the non-cooperative case, the aircraft involved in the encounter do not exchange information on their intentions and do not trust one another at all, hence the worst case approach is adopted. The two-aircraft non-cooperative conflict resolution problem can be formulated (cf. [72]) as each aircraft playing a zero-sum non-cooperative game against disturbances that model the uncertainty in the other aircraft intentions, with the value function being the aircraft distance. In the cooperative conflict resolution case, the current positions and intentions of the aircraft are assumed to be perfectly known to a supervising central controller. Each aircraft completely trusts the central controller (and hence all the other aircraft), and follows its advice. The cooperative conflict resolution problem is typically formulated as an optimization problem, where the flight plans of all the aircraft are designed so as to avoid conflicts while minimizing a certain cost function (cf. [23]). In

between the extremes of non-cooperative and cooperative conflict resolution there is the probabilistic conflict resolution approach. In this approach, each aircraft position is assumed to be distributed according to some probabilistic law, which models the presence of disturbances affecting the aircraft motion as well as the partial confidence of each aircraft in the available information on the intentions of the other aircraft (cf. [55, 38]).

Although most of the commercial aircraft are equipped to be cooperative, a conflict resolution system can not rely solely on this assumption. In the case where data link fails or data exchange is interfered, the resolution system is supposed to deal with a conflict situation without knowing the intention of the intruder aircraft. Thus the system should be required to handle both cooperative and non-cooperative aircraft.

Typically, conflicts are resolved by resorting to three different actions: turn, climb/descend, and accelerate/decelerate, which affect the aircraft heading, altitude, and speed, respectively. Resolution strategies can be one of these actions or a combination of them. Unless absolutely necessary, an aircraft takes one maneuver at a time, instead of doing two maneuvers or three in combination, because that is the simplest and easiest for the pilot to follow. Among these maneuver choices, cost and efficacy of each are different. It is known that climb/descend is the most efficient action for resolving short-term conflicts. On the other hand, excessive changes of altitude are likely to cause discomfort to passengers. Also, vertical maneuvers are not much compatible with the current vertically layered structure of the airspace. Speed changes cause more fuel consumption and have to be considered with caution. The maneuvers must direct the aircraft away from the conflict situation, but should also include a plan to direct aircraft back to the flight path after conflict is solved. Overall, maneuvers have to

favor shorter travel distance, less fuel consumption, passenger comfort, ease of handling for the pilot, less deviation from flight path, and minimum time delay.

With so may factors to consider, we need to investigate the current methods to handle them. The next chapter provides a background study on how conflict detection is modeled and how maneuvers are determined in recent research.

Chapter 2

Literature Review

Several approaches [20, 22, 23, 25, 26, 31, 37, 38, 53, 55, 58, 65, 67, 72] to the problem of detection and resolution of traffic conflicts have been developed and described. James K. Kuchar and Lee C. Yang in "Survey of Conflict Detection and Resolution Modeling Methods" (cf. [39, 40]) give a comprehensive overview of recent work and the applicability in an advanced Air Traffic Management (ATM) Environment comprising free flight traffic conflict situations.

2.1 Categorization

Conflict resolution can be categorized (cf. [39]) by the way the responses to conflicts are determined. Three categories are identified:

- Prescribed
- Force *fi*eld.
- Optimized

In the prescribed approach [11, 24], conflict resolution maneuvers are determined in advance, based on a set of procedures. For example, the Ground Proximity Warning System (GPWS) issues a standard "Pull Up" warning when a conflict with the terrain exists. GPWS does not perform additional computation to determine an optimal escape maneuver.

The shortcoming of the prescribed approach is that these models can be complex and require a large number of rules to completely cover all possible encounter situations. Additionally, it may be difficult to certify that the system always operates as intended, or the system may in fact not use the best strategy in resolving conflicts.

Force field [22, 37] approaches model each aircraft as a charged particle and use modified electrostatic equations to determine conflict resolution maneuvers.

While the force field methods seem attractive in the sense that a conflict resolution solution is continuously available using relatively simple equations, the shortcoming is that some cases exist that the computed maneuvers are difficult to be used in operation. For example, a solution from a force field model may require that an aircraft continually make a series of gradual turns and speed changes. This requires a high level of guidance on the flight deck and increases complexity beyond issuing simple heading vectors. Additionally, some solutions may include cusps or other physically infeasible trajectories that must be modified to be used.

Optimized [23, 38, 72] conflict resolution derives a decision for determining which of several avoidance options minimizes a given cost function. Several sub-categories exist that differ in the way the decision is derived:

- Rule based - the situation is compared against a set of pre-defined

rules to determine the course of action

- Optimal Control Theory approach (OCT) - cost functions and constraints are defined and an optimal solution is determined that minimizes the cost functions.

- Game theory based - the problem is modeled as a zero-sum non-cooperative dynamic game and a solution is found through negotiation among the game players.

- Genetic Algorithms [53](GA) based - genetic algorithms are used for searching for a solution.

The existing optimized methods are different in terms of the assumptions on the model of the dynamic system being used. The main difference is in the method that the current state is projected into the future. This dictates how conflicts are managed. The Nominal projection method is the most straightforward method, it gives a first order estimate of where and how conflicts will occur. Nominal projections, however, do not account for the possibility that an aircraft does not behave as predicted by the dynamic model. This uncertainty is very important in long-term conflict detection.

The other extreme of the dynamic model is to use worst-case projection. However worst-case maneuvers are highly unlikely, using this model may greatly reduce the overall traffic capacity. The probabilistic approach [80] appears to provide a reasonable balance between relying too much on an aircraft following the dynamic model vs. relying too much on the assumption an aircraft doing worst-case maneuvers. However, there is a tradeoff between the complexity of the probabilistic model and the ability to estimate probabilities rapidly. Also, the resolution maneuvers used to develop

the alerting logic are based on the immediate problem of avoiding a conflict and do not consider the additional maneuvering required to return to the original flight path. Thus the maneuver selection logic [80] does not incorporate issues such as increased fuel burn or flight time in the decision on alert.

2.2 Selected Papers and Comments

From the previous section it appears that the category of Optimized approaches should be the main focus. The category of Prescribed resolution maneuvers can not offer the capability to determine an optimal escape maneuver. The category of Force field approaches ignores to consider the physical character of a flight and maneuvers, and causes trajectories that are not physically realizable in the sense that it does not take into consideration the fact that this is an airplane with crew, passengers and cargo on board, all of which require some constraints to be satisfied regarding the possible maneuvers.

Within the most promising Optimized category, different approaches distinguish themselves in the method by which the current state is projected into the future. Below is a list of these different approaches with the example papers.

It is worth mentioning that the Traffic Alert and Collision Avoidance System (TCAS), the Ground Proximity Warning System (GPWS) and the Parallel Runway Monitor (PRM) are the systems that are actually used. Others are just research models being investigated.

1. Traffic Alert and Collision Avoidance System (TCAS) [2]

The TCAS system is a collision avoidance system currently used on board of transport passenger aircraft and is working as an ATC independent safety-net to detect and avoid short term conflicts. The TCAS system has been reviewed in Section 1.3.

Problem: The current system only gives vertical solutions, while horizontal maneuvers are preferred for both passenger comfort and the variety of possible solutions.

2. Durand et al., "Optimal Resolution of En Route Conflicts." [23]

   An automatic conflict solver and its implementation in an Air Traffic Simulator, with statistical results on real traffic over France, is presented. The solver takes into account speed uncertainties and allows aircraft to fly on direct routes, solves every conflict on a pre-loaded day, and gives each aircraft its requested flight level (altitude) and departure time. The conflict resolution problem is highly combinatorial involving n aircraft and cannot be optimally solved using classical mathematical optimization techniques. Therefore an optimization approach using genetic algorithms was used.

   The "cost function" approach used in [23] is very similar to the starting point of the problem formulation in this thesis and thus will be used in the solution proposed in this thesis.

3. Yang & Kuchar, "Prototype Conflict Alerting System for Free Flight." [80]

   A prototype alerting system for a conceptual free *fl*ight [1] environment

[1] A *free flight* [25] is defined as a safe and efficient flight operating capability under instrument flight rules in which the operators have the freedom to select their path and

is discussed. The alerting logic is based on a probabilistic model of aircraft sensor and trajectory uncertainties that need not be Gaussian distributions. Monte Carlo simulations are used over a range of encounter situations to estimate conflict probability as a function of intruder position, heading and speed, as determined through a datalink between the aircraft. The probability of conflict along potential avoidance trajectories is used to indicate whether adequate space is available to resolve the conflict.

Problems identified by the authors:

(a) There is a tradeoff between the complexity of the probabilistic model and the ability to estimate probabilities rapidly.

(b) The resolution maneuvers used to develop the alerting logic are based on the immediate problem of avoiding a conflict and do not consider the additional maneuvering required to return to the original flight path. Thus the logic does not incorporate issues such as increased fuel burn or flight time in the decision to alert.

(c) Centralized traffic management issues have been ignored. Because, as assumed in the free flight concept, pilots have initial responsibility for traffic separation, ground controllers could have difficulty when suddenly presented with a conflict that was not resolved by the flight crews.

Despite the complexity, this paper gives a very clear explanation of the conflict alert problem and its resolution procedure. The solution is intuitive and straight forward. However, due to the high computa-

speed in real time.

tional complexity of Monte Carlo simulations this approach cannot be used to solve the problem addressed in this thesis.

4. Paielli & Erzberger, "Conflict Probability Estimation for Free Flight." [55]

   The probability of conflict between aircraft along a predicted trajectory is estimated. The trajectory prediction errors are modeled as normally distributed and the two error covariances for an aircraft pair are combined into a single, equivalent covariance of the relative position. A coordinate transformation is then used to derive an analytical solution. Numerical examples and a Monte Carlo validation are presented.

   Problems: This approach is similar to [80], so all the problems identified in [80] apply to this approach, too. Also, the methods are intended to "assist rather than replace human air traffic controllers. That is, they are intended to provide automated advisories for the controllers, but not to make the ultimate decisions. The conflict probability cannot be reduced to zero".

   Despite the problems listed above, this paper gives a very good formulation of trajectory prediction using two error covariances. The method of trajectory prediction will be used in this thesis.

5. Tomlin et al., "Conflict Resolution for Air Traffic Management: a Case Study in Multi-Agent Hybrid Systems." [3]

   A conflict resolution architecture for multi-agent hybrid systems with emphasis on Air Traffic Management Systems (ATMS) is presented. In such systems, conflicts arise in the form of potential collisions which

are resolved locally by inter-agent coordination. To allow optimization of agent objectives, inter-agent coordination is minimized by non cooperative conflict resolution methods based on game theory. If non-cooperative methods are not successful, cooperative methods are used. Examples of potential resolution maneuvers in such an environment are given.

Problems: The paper uses the method of "artificial potential field" from robotic path planning to produce conflict-free maneuvers for given scenarios. As a result of the method used, the conflict is sometimes avoided by aircraft transitioning to a "circle" mode, i.e., a circular path, or a loop. This solution did not consider the real maneuver style of commercial aircraft, and did not consider increased fuel burn or flight time.

6. Kosecka et al., "Generation of Conflict Resolution Maneuvers for Air Traffic Management." [37]

   Based on the proposed predefined coordination maneuvers (Tomlin et al., [3]) an extension was proposed. A distributed motion planning algorithm based on potential and vortex fields is used. The algorithm does not always guarantee flyable trajectories, but the results can serve as a qualitative prototype which can be approximated using combinations of straight lines and arcs.

   Problems: This is similar to [3], so all the problems in [3] exist.

7. Gent et al. "Free Flight with Airborne Separation Assurance." [25]

   Human in the loop simulations regarding conflict resolution under a proposed Free Flight Environment are described. The algorithm for

simulating human behavior was developed and integrated with a cockpit system display.

Problems: The paper's focus is to probe for relevant human factors issues. It does not solve the problem addressed in this thesis, i.e., it does not propose an algorithm for an automated maneuver solution under conflicts.

8. Gerling, "Conflict Detection in Air Traffic Control." [26]

   The prediction of horizontal conflicts between aircraft on the basis of surveillance data, taking planned course changes into account, is described. Assuming standard turn maneuvers, the predicted flight paths and the horizontal distances are calculated as a function of time. In addition, the variation of ground speed in case of a known constant wind component is considered. The resulting distance function is transferred into a measure of threat reflecting basic conflict characteristics such as severity, urgency, dynamics and duration.

   This paper gives details as to how to calculate the amount of distance from flight path in horizontal level when aircraft is taking planned course changes. A similar method of defining the distance between the planned path and the maneuver path will be used in this thesis.

2.3 Summary

From the literature review provided above it seems clear that in order to solve the conflict resolution problem addressed in this thesis one could attempt to use an approach in which a cost function would be defined and a best solution would be chosen based upon the cost function. With stan-

dard turn maneuvers, the predicted flight paths and the horizontal distances would be calculated as a function of time [26]. Thus the value of the integral of distances over time would give the measure of how "disruptive" the maneuver is. This would provide the basis of defining cost function like in [23]. Based on the cost function, the least disruptive maneuver which ensures the conflict situation to be solved would be selected.

The above cost function is based on the knowledge of both aircraft positions and velocities, and it is based on the assumption that the intruder aircraft is not doing any evasive maneuvers. For each step of the prediction, the optimal maneuver is obtained by assuming the intruder aircraft will continue with its velocity for the next scan, and the self aircraft is only changing course according to the selected maneuver. In a more practical case, the intruder aircraft will likely maneuver once it detects a conflict situation. It is necessary to anticipate this maneuver. There are at least two ways to handle this. One, even though the maneuver is unknown, it is possible to estimate intruder aircraft's optimal maneuver using a similar cost function approach. Then the maneuver selection procedure should include such an estimate and revise/update the resolution advisory. As a result, this increases the complexity of the problem. The other method is to consider all the possible maneuvers by the intruder aircraft and anticipate all of them in the calculation. Under each possible maneuver by the intruder aircraft, the self aircraft maneuver guidance can be computed accordingly. If the number of maneuver choices is N for each aircraft, the total number of maneuver possibilities is N^2 for two (not doubled, but squared!). For this research, we limit our effort to search for maneuvers of the own aircraft by predicting L steps ahead, without considering possible evasive maneuvers of the intruder. Even so, the problem is complicated enough. We show the

complexity analysis in section 3.5, after we formally formulate the problem.

Chapter 3

Mathematical Formulation of a Quantitative Method

3.1 Vertical vs. Horizontal Maneuvers

The current Traffic Alert and Collision Avoidance System, called TCAS, was born in 1981. The final version, version 7, has been successfully developed and used since 1997. TCAS II provides Traffic Advisories (TA's) of potential threats and it also provides pilots with resolution advisories (RA's) when needed. The system determines the course of each aircraft – climbing, descending, or flying straight at the same level. Then it issues a RA advising the pilots to execute an evasive maneuver to avoid the other aircraft, such as "Climb" or "Descend". If both planes are equipped with TCAS II, then the two computers offer de-conflicting RA's.

The RA gives maneuvers such as to climb or to descend. It is mostly in the altitude direction. TCAS is designed to select the RA strength that is the least disruptive to the existing flight path, while still providing ALIM

feet of separation. In the case where vertical maneuvers are too difficult (there are aircraft climb performance limitations at high altitude) to provide adequate vertical separation, horizontal maneuvers should be considered. Also, when the altitude information of the aircraft is not provided (non-altitude reporting intruders), vertical RA's cannot be generated. In such cases co-altitude is assumed and horizontal maneuvers should be selected.

3.2 Components of the Goal Function of the Optimization Problem

The aim for the general ATC conflict alert problem is to find the optimal maneuver, either horizontal or vertical, that will resolve the conflict situation. From the end of Chapter 1, where design issues were discussed, it is obvious that some choices of maneuvers are preferred by the pilots, the controllers and the airlines. It is usually desired to select the least destructive maneuver from the flight plan, the shortest time delay and the simplest maneuver for the pilot to carry. These provide the guidelines for defining the optimization problem. To select the best maneuver among a list of candidate maneuvers it is necessary to define an optimization criterion, or a goal function that assesses the quality of a selected maneuver. For vertical maneuver RA in TCAS it is obvious that the minimum path change is preferred. To quantify this, denote the path of aircraft A in the vertical plane (2D plane yet to be defined) as curve C_1, and denote its maneuver path as M_1. The path change can then be measured by the area between the two curves. Since the aircraft always return to their original flight paths, the area between the two curves can be calculated. Denote the path of aircraft B in the vertical plane as curve C_2, and M_2, if maneuver is applied for it. The area between the

two paths can then be calculated. If return maneuvers are to be included in the computation, the goal function can be defined as the sum of these areas for all the aircraft in the encounter space. The goal function should be minimized under the condition that every possible maneuver has to create sufficient separation between the two aircraft.

For horizontal maneuvers, the areas can be defined on the horizontal plane in the same manner.

In addition to path difference calculations, often fuel consumption and passenger comfort during evasive maneuvers are considered in selecting RA's. Finally, the number of maneuver orders is desired to be minimum for the sake of pilots. We first incorporate all these elements in the goal function and later make assumptions to simplify them.

3.3 Optimization Problem Considerations

The goal function should be minimized under the condition that every possible maneuver has to create sufficient separation between two aircraft. For two aircraft, there is one separation constraint between them. For N aircraft one needs to consider all pair-wise constraints and thus in such a case there are $N(N-1)/2$ separation constraints in total. For our research, the simplest case with a pair ($N = 2$) of aircraft is considered. The optimization problem is under the constraints of maintaining separation, i.e., avoiding the conflict alerts of three types – PROCON, LINCON and MANCON - described below.

3.3.1 Mathematical Definitions of PROCON, LINCON and MANCON

The PROCON, LINCON and MANCON alerts are defined in [5].

Mathematically, the PROCON alert is declared if the distance, d, between two aircraft is smaller than a threshold, $d_{proxim1}$, or if the distance is smaller than a larger threshold, $d_{proxim2}$, and the two aircraft are approaching each other at a rate R larger than a speed threshold R_{proxim}:

$$(d < d_{proxim1}) \vee (d < d_{proxim2} \wedge R > R_{proxim}) \tag{3.1}$$

The LINCON condition is satisfied if, by linear prediction, the impact time, $IMPT$, of the aircraft pair is below the lookahead time, $T_{lookahead}$, (a threshold of 40 seconds):

$$(IMPT < T_{lookahead}) \wedge (IMPT > 0) \tag{3.2}$$

where $IMPT$ is the time at which the two aircraft are predicted to come into a lateral conflict or the Time of Co-Altitude (TOCA), which ever is earlier. Since in the example of this thesis we are considering only horizontal maneuvers, we can assume that $IMPT$ is equal to the Time of Lateral Violation ($TOLV$), whose predicted value is computed from the following equation [5]:

$TOLV =$

$$\frac{-(\triangle x \cdot \triangle \dot{x} + \triangle y \cdot \triangle \dot{y}) - [LATQ^2(\triangle \dot{x}^2 + \triangle \dot{y}^2) - (\triangle x \cdot \triangle \dot{y} + \triangle y \cdot \triangle \dot{x})^2]^{1/2}}{\triangle \dot{x}^2 + \triangle \dot{y}^2} \tag{3.3}$$

and $LATQ$ is a system parameter (e.g., 2.0 nmi), $\triangle \dot{x}$, $\triangle \dot{y}$ are relative velocities in x, y directions, $\triangle x$, $\triangle y$ are relative positions in x, y directions.

MANCON conditions are tested when the turn rate, ω, is larger than a threshold, ω_{thresh}. The MANCON conditions are satisfied if one aircraft is turning into the other aircraft and will cause a conflict situation under the "turning model" prediction [5]:

$$\omega > \omega_{thresh} \wedge [(s_1 < 0) \vee (s_2 > 0) \vee (s_3 > 0) \vee (s_4 > 0)] \tag{3.4}$$

where

$$s_1 = \triangle x \cdot \triangle \dot{x} + \triangle y \cdot \triangle \dot{y} \tag{3.5}$$

$$s_2 = \dot{x}_1 \cdot \dot{x}_2 + \dot{y}_1 \cdot \dot{y}_2 \tag{3.6}$$

$$s_3 = \triangle x \cdot \dot{x}_1 + \triangle y \cdot \dot{y}_1 \tag{3.7}$$

$$s_4 = -(\triangle x \cdot \dot{x}_2 + \triangle y \cdot \dot{y}_2) \tag{3.8}$$

and $\triangle \dot{x}$, $\triangle \dot{y}$ are relative velocities in x, y directions, $\triangle x$, $\triangle y$ are relative positions in x, y directions, x_i, y_i, $\dot{x}_i, \dot{y}_i$ are position and velocity of aircraft i $(i = 1, 2)$.

PROCON is the highest priority alert since it is issued when two aircraft are already in close proximity. LINCON is less severe since it is based upon the prediction within 30 seconds in the future. MANCON is even less severe than LINCON since it is based on the prediction for 60 seconds in the future, including the estimation of the trajectory of own aircraft due to its maneuver. If none of the three conditions are satisfied, the aircraft are in a safe status.

While it would be desirable that none of the conditions for LINCON alert or MANCON alert are satisfied during a maneuver, this is rather difficult to achieve. When an aircraft pair is in some form of conflict alert, the maneuver may or may not be able to take the aircraft out of conflict alert immediately.

It is more likely the aircraft pair continues to be in conflict for a few scans, with more severe conflicts turning into less severe conflicts, and eventually turning into a safe status.

3.3.2 Mathematical Formulation of the Goal Function

Based upon the literature review, we selected the Optimized approach to the problem of RA generation. Towards this aim, we need to define the cost function and the constraints that the solution must satisfy. Taking into account the components of the goal function discussed in Section 3.2, we define the cost function as a measure of deviation of flight trajectory from the original flight plan during the course of the conflict avoiding maneuver. We limit this research to horizontal maneuvers and ignore the vertical maneuvers.

Assuming the prediction time (number of steps) is l, the cost of the maneuver can be expressed as:

$$g() = \sum_{t_k=t_1}^{t_l} \sqrt{(x^m(t_k) - x^c(t_k))^2 + (y^m(t_k) - y^c(t_k))^2} + w_2 \cdot |a_v| + w_3 \cdot |a_\omega| + w_4 \cdot |\dot{z}^{(m)}(t_k)| \tag{3.9}$$

where the first item is the deviation of flight, the second term and the third term (with w_2 and w_3 as weight coefficients) are fuel consumption proportional to speed changes and turns, and the fourth term (with weight coefficient w_4) [44] is associated with passenger discomfort. However, not considering vertical maneuvers, the fourth term can be ignored. Since the fuel consumption rate depends on the weight of the aircraft, the type of the aircraft, the design of the engine and the type of fuel in the engine, it is very complicated and not within the scope of this research. We will ignore the

second term and third term too. Therefore, the cost function is:

$$g() = \sum_{t_k=t_1}^{t_l} \sqrt{(x^m(t_k) - x^c(t_k))^2 + (y^m(t_k) - y^c(t_k))^2} \qquad (3.10)$$

where t_k - time, $(x^m(t_k), y^m(t_k))$ - a point on the X, Y trajectory of the maneuver at time t_k, and $(x^c(t_k), y^c(t_k))$ - a point on the trajectory of the planned flight path at time t_k. The function $g()$ computes the discrepancy between the two paths (planned and maneuver) during the maneuver interval $[t_1, t_l]$. The planned trajectory is assumed to be known. The maneuver trajectory is computed by the RA determination algorithm.

3.4 Optimization and Possible Maneuvers

A concrete maneuver command must be given in the form of the values of climbing rate, e.g., in feet per minute (fpm), descending rate (fpm), turn rate (degree per second) to the left or right, speed up or speed down rate (fraction of the gravity constant g). All possible maneuver choices are listed in Table 3.1 in Section 1.2. These choices are based upon [4].

The smallest interval of the vertical rate is 500 fpm. The choices are $\pm$500 fpm, $\pm$1000 fpm, ..., $\pm$4500 fpm. The largest feasible vertical rate is $\pm$4500.

The smallest interval of turn rate is 1 deg/sec. The choices are $1, 2, \ldots, 7$ deg/sec for either left turn or right turn. The largest feasible turn rate is 7 deg/sec.

The smallest interval of speed up rate (acceleration) is 45 knots/minute, which is approximately 0.04g. The choices are $0.036g$ $0.073g$ $0.11g$ $0.146g$ $0.183g$ $0.22g$. The value of $0.22g$ is the maximum acceleration feasible.

We see that horizontal maneuvers typically include turns, ω, and accelerations, a_v. Normally one maneuver is carried out at a time, i.e., no concurrent maneuvers [31] are executed. Therefore, a maneuver that starts at t_k and ends at t_{k+1} can be expressed as:

$$a(t_k) = \begin{cases} a_v(t_k), & \text{acceleration maneuver} \\ \omega(t_k)/v(t_k), & \text{turn maneuver} \end{cases} \tag{3.11}$$

The optimization problem is then to find a sequence of maneuvers $a(t_1)$, $a(t_2),\ldots,a(t_{l-1})$ that minimizes the cost function (3.10).

Since this optimization includes multi-step prediction, the trajectories of the (own) aircraft are computed according to the following motion equations.

$$\begin{aligned} x^m(t_{k+1}) &= x^m(t_k) + v_x^m(t_k)\cdot\Delta T + \frac{1}{2}\cdot a_x\cdot(\Delta T)^2 \\ y^m(t_{k+1}) &= y^m(t_k) + v_y^m(t_k)\cdot\Delta T + \frac{1}{2}\cdot a_y\cdot(\Delta T)^2 \end{aligned} \tag{3.12}$$

and

$$\begin{aligned} x^m(t_0) &= x^c(t_0) \\ y^m(t_0) &= y^c(t_0) \end{aligned} \tag{3.13}$$

where a_x, a_y, v_x, v_y are projections of a and v on the X and Y coordinates, respectively. The trajectory of the other aircraft is computed using the same equations using zero acceleration.

The optimization is subject to the satisfaction of three constraints defined in section 3.3.1 by equations 3.1, 3.2 and 3.4, respectively. The constraint PROCON must evaluate to false, i.e., at any time $t \in [t_0, t_0+T]$ the minimum separation constraint must be satisfied (eq. (3.1) $\equiv$ false). Moreover, at the end of the maneuver time, the aircraft pair is out of conflict situation, i.e., the conflict constraints eq. (3.2) $\equiv$ false and eq. (3.4) $\equiv$ false are maintained.

The maneuvering has to be performed within the time interval that

ua04.9T(s)3186.7()3239(he)318617serie(s)3186.7ofemaaneuvr(s)3186.7ha04.9a(s)313239(os)312662(b)-26.

Maneuver Name	Dimension	Unit	Possible Values
turn left or right	horizontal	degree/second	1,2,3...7
speed up or down	horizontal	knot/minute	45, 90, ..., 270
climb up or descend	vertical	foot/minute	500, 1000, ..., 4500

Table 3.1: Possible maneuver choices

steps are computed, there are N_1 choices for each choice computed in up to step $N_2 - 1$. Thus the total number of choices for all possibilities during the time duration of N_2 steps is: $N_1 + N_1^2 + N_1^3 + ... + N_1^{N_2}$. Therefore, the time complexity of this problem is in the order of:

$$O(\frac{N_1^{N_2+1} - 1}{N_1 - 1}) \cdot O(1) = O(N_1^{N_2+1}) \tag{3.14}$$

To give an intuitive understanding of the complexity of this problem we compute the number of possible maneuvers that need to be considered for the values of N_1 and N_2 that are chosen based upon the characteristics of the typical sensor, i.e., on a typical scan duration, as well as on a reasonable maneuver granularity in the horizontal plane. These values are summarized in Table 3.1. For horizontal maneuvers, seven turns in the increments of one degree per second have been considered. The choices for horizontal linear acceleration are in the increments of 45 knots per minute, up to 270 knots. For comparison, in the vertical dimension, there are nine choices for climbing up or descending in the range from 500 to 4,500 feet per minute, in the increments of 500. The granularity for each of these choices could be finer, which would result in even higher values of N_1, N_2, and as a net result, in a higher computational complexity.

According to this table, there are $N_1 = 27$ possible horizontal maneuvers:

seven left turns, seven right turns, no maneuver, six positive accelerations and six negative accelerations. Since we are assuming only one maneuver at a time, the case of no turn is the same as zero acceleration. Assuming $N_2 = 6$ steps for prediction, the number of horizontal maneuvers that need to be computed is

$$\frac{27^7 - 1}{27 - 1} = 4.02 \cdot 10^8$$

Furthermore, assuming that each computation can be done within 10ms, the computation time of such a problem would be equal to $4.02 \cdot 10^6\ sec = 1,117$ hours. In case $N_2 = 10$ was taken, this would be on the order of 10^{10} hours of computation.

It is clear that the time complexity of multiple-step prediction is too high to be practically implementable. Therefore, there is a need for lowering the complexity of RA generation without sacrificing the accuracy of maneuver advisories. One way to achieve this goal would be by lowering the granularity of partitioning the various variables into intervals; for instance by recomputing predictions at larger time intervals. Unfortunately, this might result in errors - making maneuvers that might lead to conflict. What is really needed is a non-uniform resolution - dense quantization in some (critical) regions and sparse partition in some other (not so important) regions. In other words, there is a need for "logical", rather than just uniform partition of the space of the dynamical system. But the notion of "logical partitioning" would first had to be formalized. This is exactly what the Q^2 approach [36] is meant to achieve - logical (consistent) partitioning, s.t., symbols are assigned to partitions and decisions in the symbolic space never cause inconsistencies, i.e., while the decision is not precise, it always falls within the boundaries of the right partitions. In the research

described in this thesis we applied the main principles of the Q^2 approach to the problem of RA generation.

The prediction of the aircraft trajectory can be modeled by state transition functions. The maneuvers behave like the input control signals that influence the state. The task to find the optimal input is to search within all the input possibilities, compute the future state from the current state, check for sufficient separation between aircraft, compute the cost function for each valid possibility, and select the one with the lowest cost function. The trajectory of the aircraft depends, not only on the value of the input, but also on the state the system is in, and on the time interval at which a given input is applied. As a result, the dimension of the search is large: two for control input in X and Y, four for initial state in X and Y, one for time. Thus for one aircraft alone, the search space is seven-dimensional. For two aircraft as a pair, the search space is thirteen-dimensional.

For our ATC problem, we will try to find a quantized version of the input, state, and output. The goal will be to select a resolution that allows us to preserve the consistency of the logical reasoning about conflict resolution. The process that we will apply is the one of abstraction. In this process we will select some special values in the input, state and output spaces of the dynamical system, which we will treat as symbolic inputs, i.e., such that will carry some meaning. More detail of this idea is given in chapter 4.

Chapter 4

Q^2 Approach

In the previous chapters the quantitative approach to the problem of detection and resolution of conflict alerts has been analyzed. The problem was formulated as an optimization problem. The analysis of complexity resulted in the conclusion that the problem is not computationally tractable within the available time limits. For this reason, we now turn our attention to qualitative (symbolic) methods, instead. Symbolic reasoning is prefered for many tasks because (cf. [6]) people relate better to reasoning based on alphanumeric symbols; planning can be performed more easily; symbolic reasoning can be used to achieve goals expressed in a high-level symbolic language; certain complex problems become computationally tractable when converted to symbolic representation [61]. This approach has demonstrated partial success in maneuver detection (cf. [46]), where qualitative reasoning about a quantitative dynamic system was used. In this thesis we investigate the possibility of using symbolic reasoning for the problem of conflict resolution formulated in Chapter 3. Similarly as in [46], our system will be modeled as a dynamic system.

Some of the symbols and notations are listed below.

$q(t)$, $u(t)$ – state and control input, respectively

$\mathbf{W}$ – output set,

$\mathbf{Q}$ – state set,

$\mathbf{U}$ – input set

X – position vector (in the state space), with components X_x and X_y

V – velocity vector (in the state space)

$\triangle X$ – position difference, a vector (in the output space)

$\mathbf{x}_1$, $\mathbf{x}_2$ – position vectors of aircraft 1,2

$\mathbf{a}_1$, $\mathbf{a}_2$ – acceleration for aircraft 1,2

P – input process function

F – global state transition function

f, g – state transition function, output function

χ – abstraction function

S, Σ – GDS and QDS

$\mathbf{TQU}$ – the input space

Λ – qualitative inputs (events),

Θ – qualitative states,

Ω – qualitative output

4.1 Introduction to General Dynamic Systems and Q^2 Symbolic Reasoning

The integration of symbolic (qualitative) and quantitative representation involves several tasks (cf. [36]):

1. Determine the region (qualitative output) to which the output of the system belongs given a region (qualitative state) to which the current state belongs.

2. Determine to which of the regions in the state space the state will belong (next qualitative state) given the region in the Cartesian product of input, initial state and time sets.

3. Determine the class of input processes that would cause the system to switch to a particular region in the state space.

It is important to know how to partition a quantitative dynamic system so that the results of reasoning within the qualitative structure (i.e., answers to the above three questions) always hold in the underlying quantitative dynamic system. In the formalization in [36], the quantitative structure is represented using a general dynamic system (GDS). The qualitative counterpart, called Qualitative Dynamic System (QDS), is represented using a finite-state automaton structure. The two structures are related through abstraction functions called qualitative abstractions of dynamic systems. The abstraction functions take the duty of mapping the GDS onto the QDS. Interpretation of the QDS is achieved by the inverse of the abstraction function. The entire quantitative/qualitative structure is referred to as the Q^2 structure.

A typical approach to translating quantitative variables into qualitative symbols is to partition the quantitative state variables with some critical values of the quantitative variables. The result is partitions of the space of the dynamic system in hyper boxes. When the value of the quantitative state variable crosses a critical value, a qualitative event occurs. These events can be monitored and used as inputs to a finite state automaton,

which provides a qualitative model of the continuous plant. The automaton switches to a new state and generates qualitative output associated with this state. Various approaches to mapping quantitative systems to qualitative systems have been investigated, mainly in the AI community (cf. [73, 43, 49, 63, 68, 29, 50, 60, 78]).

In the hyperbox representation, a qualitative event is triggered by a transition between boxes, with each event causing a corresponding transition between states in the automaton. A hyperbox representation constrains the critical values of one state variable independent of the values of the other state variables. In contrast to the hyperbox approach, the Q^2 approach [36] allows the critical values for a quantitative variable to be functions of the other quantitative variables, which results in partitions that in general have more complex shapes than hyperboxes. Reasoning with boxes is not consistent since such a representation of a dynamic system cannot guarantee that its results will hold in the underlying quantitative system. This means that if the symbolic reasoner derives a conclusion that the next state will be within a given box, it may or may not be true. It is important to understand (cf. [36]) that the boxes approach is not optimal, and that whenever we use it, we are making a tradeoff between the consistency of reasoning and the lack of precise knowledge.

Similar results can be obtained by identifying regions in the spaces of a general dynamical system, instead of considering boundaries between regions. Describing points in the regions is easier when the boundaries are hard to describe, and thus it is used in this paper.

For our problem, it is necessary to find the consistent partitioning of three system spaces, i.e., the output space, the state space, and the input space (Cartesian product of the input set, initial state space and time). This

partitioning allows us to construct qualitative reasoning systems whose inferences are guaranteed to hold in the underlying (deterministic) continuous dynamic system [36].

4.1.1 General Dynamic System

Our quantitative plant is modeled by a General Dynamic System (c.f. [52]). The model can be represented in differential equation form as:

$$\dot{q} = f(q(t), u(t), t) \tag{4.1}$$

$$w = g(q(t), t) \tag{4.2}$$

where q is the state, w is the output and u is the control input of the system. In the control literature, this set of equations is generally referred to as the plant model, where the function f determines how the system state q varies in response to the input (control signals), g determines how the output signal is generated as a function of state. A plant whose characteristics don't change with time t is called a time-invariant system. [47, 19] The time set in the definition of a general dynamic system must have an order relation on it. The input process consists of time functions together with inner splicing on them, and the state transition function has the following properties [52]: consistency, semigroup, and causality.

The definition of a general dynamic system is constructed in such a way as to reflect some general characteristics of all dynamic systems. The splicing property of the input processes gives the flexibility of choosing any control strategy for the given system at any point in time. The consistency property of this function ensures that state transitions are not instantaneous. The semigroup property means that the system state at any particular time, t,

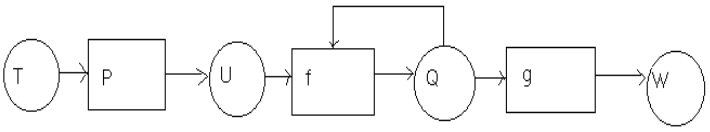

Figure 4.1: The structure of a general dynamic system.

can be computed either directly for the whole time interval $[t_0, t)$, or indirectly by performing two computations: for $[t_0, t_1)$ and $[t_1, t)$. More generally, it allows us to decompose the computation of a state into finitely many steps. The causality property means that the value of the new state depends only on the input function restricted to the interval $[t_0, t)$ (in addition to the previous state).

A model of a general dynamic system includes an input set $\mathbf{U}$, an output set $\mathbf{W}$, a time set T with an order relation on it, a state set $\mathbf{Q}$, input process functions, P, a global state transition function F and an output function, g. From now on, we will refer to the input, state, and output sets as input space, state space and output space, respectively.

The anatomy of a general dynamic system is represented in Figure 4.1. The circles in this figure represent sets: T - time set, $\mathbf{U}$ - input set, $\mathbf{Q}$ - state set, and $\mathbf{W}$ - output set, respectively. The rectangles represent functions that appear in the definition of a general dynamic system: P - input process functions, f - local state transition function, and g - output function. The state transition function has two incoming arrows: from $\mathbf{U}$ - the input at current time, and from $\mathbf{Q}$ - the state at current time.

4.1.2 Q^2 Representation and Consistency of Symbolic Reasoning

An abstraction is the mapping from a real system to an abstract system which maintains certain desirable properties and throws away details [28] [7, 8, 71, 27] [12]. In general, a mapping from a set S to a finite set I is called a qualitative abstraction function if it is total and many to one. A typical example of such an abstraction function is the function

$$sign : W \rightarrow \{-1, 0, 1\} \tag{4.3}$$

whose value is determined by the sign of the output W of the system (it is -1 for negative values of W, 1 for positive, and 0 for $W = 0$).

For the purpose of this research, we propose to use abstraction to represent an infinite-state quantitative dynamic system with a finite state automaton. The representation is provably correlated with the underlying dynamic system. This means that the conclusions about the state transitions and outputs of the system in the qualitative representation hold in the quantitative dynamic system .

The Q^2 qualitative abstraction approach given in [36] differs from other methods of qualitative abstractions in the fact that it makes qualitative inputs a function of not only quantitative inputs but also of quantitative states and time. The idea of treating subsets of $T \times \mathbf{Q} \times \mathbf{U}$ as the qualitative input allows the qualitative dynamic system to be a consistent abstraction of the general dynamic system plant. The argument and proof are in [36].

Qualitative reasoning in the AI literature [59, 77, 69, 21, 17, 62, 13, 70, 41, 64, 51, 66] uses the term "abstraction function" and "qualitative dynamic system" in a somewhat different way than we use it in this paper. In the literature about qualitative reasoning, direct qualitative counterparts of

quantitative concepts are dealt with: qualitative inputs, qualitative states, qualitative outputs, and qualitative time. We, following the work in [36], interpret qualitative inputs (events) as subsets of the Cartesian product of quantitative states, inputs and time. This removes the main difficulty to constructing representation in which qualitative reasoning about a quantitative dynamic system is consistent. Obviously we cannot abstract just qualitative inputs (meaning input values from an interval of possible values), because the qualitative behavior which includes both the trajectory of the system's state and the output, depend not only on the value of the input, but also on the state the system is in. Plus, this behavior depends also on the time interval at which a given input is applied. In order to be able to predict the system's behavior, we need all these three pieces of information: the current state, the input, and the time interval at which the input is applied.

The relation between the GDS and the corresponding QDS is shown in Figure 4.2. As shown in this figure, the lower row indicates how the GDS operates. The middle row gives the abstraction functions. Let χ be the abstraction function, where

$$\chi = (\chi_{\mathbf{TQU}}, \chi_{\mathbf{Q}}, \chi_{\mathbf{W}}) \tag{4.4}$$

consisting the three qualitative abstraction functions:

$$\chi_{\mathbf{TQU}} : T \times \mathbf{Q} \times \mathbf{U} \to \Lambda$$

$$\chi_{\mathbf{Q}} : \mathbf{Q} \to \Theta$$

$$\chi_{\mathbf{W}} : \mathbf{W} \to \Omega.$$

Λ, Θ, Ω are the qualitative input event set, qualitative state set, and qualitative output set, respectively. The top row of Figure 4.2 indicates how the

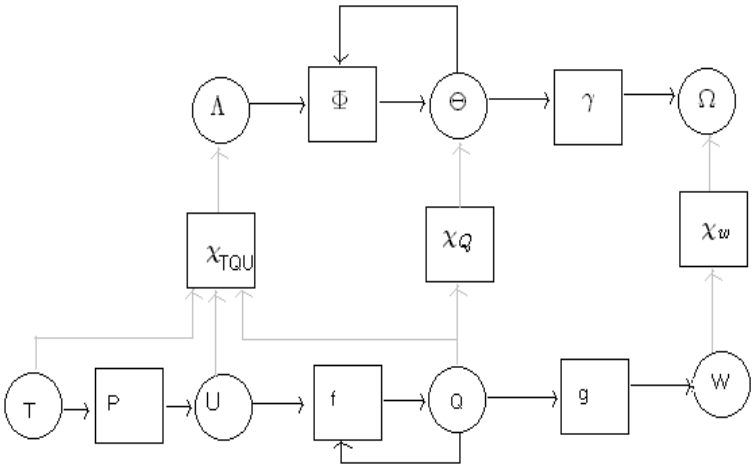

Figure 4.2: Q^2:Quantitative/qualitative representation of a dynamic system.

QDS operates, where ϕ is the qualitative state transition function

$$\phi : \Theta \times \Lambda \rightarrow \Theta,$$

and γ is the qualitative output function

$$\gamma : \Theta \rightarrow \Omega.$$

Definition 1 [36] Let S, Σ and χ represent a GDS, a QDS, and an abstraction function de*fi*ned as in previous sections. The pair (Σ, χ) is a consistent representational structure of a dynamic system S if the following consistency postulates are satis*fi*ed $\forall q, q_0, u, t$:

1.

$$\gamma(\chi_{\mathbf{Q}}(q)) = \chi_{\mathbf{W}}(g(q))$$

2.

$$\phi(\chi_{\mathbf{Q}}(q_0), \chi_{\mathbf{TQU}}(t, q_0, u)) = \chi_{\mathbf{Q}}(f(t, q_0, u)).$$

4.1.3 Qualitative Regions

So far we have not made any assumptions about the functions f, g and about the topology of the initial partitioning of the output space $\mathbf{W}$. Without such presumptions we can not make any statements about the form of the derived partitions. For practical applications, the assumption of continuity of functions can be made such as: functions f, g are continuous, except for a finite number of points. If the dynamic system is a single-output system, we can assume the partitioning of this variable into intervals. The types of constraints on the system spaces and of the consequences of such constraints (like the above assumptions) on the partitioning of the spaces of a *GDS* is a subject of topology. Any deeper analysis of topological properties of dynamic systems is beyond the scope of this thesis. However, we make these comments for the sake of completeness. For instance, in our problem, the most relevant questions are whether by following the approach proposed here one can arrive at a fine number of partitions and whether qualitative regions are connected. For instance, qualitative classes of the state space $\mathbf{Q}$, obtained through the application of g^{-1} to open intervals in $\mathbf{W}$, are regions in $\mathbf{Q}$ which are open and connected (cf. [36]). This is because inverse images of open and connected sets through a continuous function are also open and connected. Similarly, the application of f^{-1} to these regions results in a collection of open and connected regions in $T \times \mathbf{Q} \times \mathbf{U}$. Depending on the

type of the functions f and g, there might be only a few regions or an infinite number. Interesting questions include the following: what are the classes of dynamic systems for which these regions are contiguous? When would we have only a finite number of them? And many others. As we stated above, such questions are beyond the scope of this thesis.

All of the system's spaces can be subdivided into regions. The behavior of the system can be represented as a trajectory in each of the spaces: input process plus initial state trajectory, state space trajectory, and output space trajectory. Each of these trajectories may cross the region boundaries. The event of crossing a boundary in one of the system's spaces coincides with the crossing of the respective critical boundary in all system's spaces. Qualitative behaviors can be characterized by the trajectories, i.e., by series of qualitative inputs (events) in Λ, qualitative states in Θ, or qualitative output in Ω.

4.2 Application of Q^2 to Conflict Alert Resolution

Now we apply the Q^2 approach to our problem of conflict alert resolution. First we consider two aircraft and refer to them as an aircraft pair. We represent two aircraft as one dynamic system. The variables of this dynamic system are as follows.

The output space of the pair is a vector space span over the output variable $\mathbf{W}$:

$$\mathbf{W} = (\mathbf{x_1} - \mathbf{x_2}, \mathbf{v_1} - \mathbf{v_2}) \tag{4.5}$$

where $\mathbf{x_1}, \mathbf{x_2}$ are the position vectors of the two aircraft and $\mathbf{v_1}, \mathbf{v_2}$ are

their velocities. It is composed of the difference of position vectors, and the difference of velocity vectors.

State space of the pair is a vector space span over the state variable $\mathbf{Q}$:

$$\mathbf{Q} = (\mathbf{x_1}, \mathbf{x_2}, \mathbf{v_1}, \mathbf{v_2}). \tag{4.6}$$

It is composed of individual position vectors and velocity vectors of each aircraft in the pair. This can be very high dimensional if $\mathbf{x_1}$ alone is vector of 3-D.

Input space of the pair is a vector space span over the variables consisting of the previous state vector, accelerations and the time interval.

$$\mathbf{TQU} = (\mathbf{Q_0}, \mathbf{a_1}, \mathbf{a_2}, T) \tag{4.7}$$

where $\mathbf{a_1}, \mathbf{a_2}$ are the acceleration vectors of the two aircraft and they are the results of the maneuvers by each of the aircraft in the pair.

The partitions of these spaces are discussed below. The details of this dynamic system such as its state transition function and output equation are given in next chapter.

4.2.1 Partitioning of System Spaces in Conflict Alert Situation

From the point of view of alert resolution, the physical status of an aircraft pair would be either "in conflict" or "no conflict" (safe). The most intuitive way of partitioning the output space is to split it into two regions - "in-conflict" and "no conflict". However, in air traffic control, different types of conflict alerts are considered, indicating different levels of emergency. When two aircraft are very close in proximity, or in close proximity with fast approaching rate, PROximity CONflict alert (PROCON) is issued. This re-

quires the pilot to maneuver immediately to ensure safety. If no PROCON alert has been detected, linear prediction of the aircraft position for 30 seconds ahead is computed to decide whether LINCON alert should be issued. If no PROCON or LINCON alert has been detected, MANeuver prediction (IMM filter maneuver mode) of the aircraft position in the next 60 seconds is computed to decide whether a MANCON alert should be issued. If no PROCON, LINCON or MANCON has been detected, the aircraft pair is in "no conflict" status. The partitioning of quantitative output space is given by an algorithm based on industry standards. The mathematical definitions are provided in section 3.3.1.

When the spaces are partitioned into regions according to the definitions of PROCON, LINCON and MANCON, the regions can be described either by the boundaries (distinguished points and hypersurfaces) that enclose the regions, or by groups of points in the regions. When the boundaries are difficult to express in equations, the characteristic functions are used.

The approach in [36] was to partition the output space by distinguished points first, map back the distinguished points to the state space (the result is hypersurfaces in the state space), then map back the hypersurfaces to the input space. For our conflict alert problem the boundaries are difficult to describe, so the groups of points in the regions are used. Also, some challenges surface for our ATC problem, thus some of these partitioning steps can be different.

PROCON, by definition, depends on the relative position and relative speed. It is defined using variables from the output space. Therefore a region in the output space corresponds to the PROCON status.

LINCON also depends on the relative position and relative velocity. Additionally, it is based on the assumption that the acceleration is zero (linear

prediction means constant velocity is assumed for each aircraft). Since the acceleration variable belongs in the input space, the question is whether LINCON should be defined in the input space.

A similar question involves MANCON. MANCON depends on the relative position and relative velocity. It also depends on the individual (self aircraft) position and velocity, plus it depends on the turn rate (or cross track deviation). But the MANCON conditions are tested only when a maneuver is detected. Should then MANCON be defined based upon acceleration, and thus only the input space should be partitioned so that a portion of the input space corresponds to MANCON?

The answers to the above questions have to be addressed before we can continue with partitioning the system spaces and the Q^2 approach. From the air traffic control point of view, even though the linear prediction in LINCON is based upon the assumption that the acceleration is zero, the actual acceleration of the aircraft is not measured or computed to support this assumption. Neither is the aircraft controlled by the pilot to fly at zero acceleration. When the aircraft is flying at varying speed, LINCON conditions are still checked and LINCON alerts are generated whenever the LINCON conditions are satisfied. It is understood that when the actual flight has a discrepancy with the model assumption, the track prediction would be somewhat inaccurate. So LINCON is merely a way of identifying a portion of the space as representing the LINCON alarm. Other parts of the space can be classified as safe or as some other alerts. So LINCON can be fully specified in the output space.

MANCON depends on the relative position and relative velocity, the individual (self aircraft) position and velocity and the turn rate. Some of these are variables in the state space, some variables can be computed from

the state space variables, some (e.g. turn rate) depend on the previous state (see [5] for turn rate computation). Thus MANCON partitioning is defined in the input space.

From this point on, the approach in (cf. [36]) and the proposed approach for this ATC problem diverge. By the end of section 4.2, the partitioning mechanism for this ATC problem will be constructed in a somewhat different manner. In the approach in [36], the partitioning starts from the output space. When a partitioning of the output space is mapped back to the state space, a partitioning of the state space is obtained. For the ATC problem the partitioning of the system spaces imposed by MANCON has to be done in the input space. One might try to partition the state space by mapping the partitioning of the input space through the state transition function. However, in that case the result might not be a partition, since the images of the input partitions could overlap. This would mean that the images of the partition of the input space would not result in a partition of the state space. This problem occurs when the function is not one-to-one.

4.2.2 Partitioning of Output Space and Qualitative Outputs

As stated earlier, only PROCON and LINCON alerts are defined in the output space, thus the output space is divided into three regions - two regions corresponding to the PROCON and LINCON alerts, and "OTHER" region, corresponding the situations where none of the two alerts are issued. To formalize the problem we introduce the following symbols to label the regions of the output space (qualitative outputs):

$$\Omega = \{\omega_1, \omega_2, \omega_{34}\},$$

where

ω_1 - PROCON alert issued,

ω_2: PROCON not issued, LINCON alert issued,

ω_{34}: OTHER. This may include MANCON and "safe" which are not separable at this time.

An example of a projection of the output partitioning onto 2D is shown in Figure 4.3.

4.2.3 Partitioning of State Space and Qualitative States

The mapping of the partitions of the output space through the inverse of the output function defines a partition of the state space. The "functional" property of the inverse of the map, guarantees that the disjoint property in the output space also holds in the state space. The state space is partitioned into regions corresponding to PROCON and LINCON alerts, as well as OTHER (similar to the output space partitioning). These three regions in the state space can be represented by three symbols in the qualitative state space:

$$\Theta = \{\theta_1, \theta_2, \theta_{34}\} \tag{4.8}$$

An example of a projection of the state space partitions onto 2D is shown in Figure 4.4.

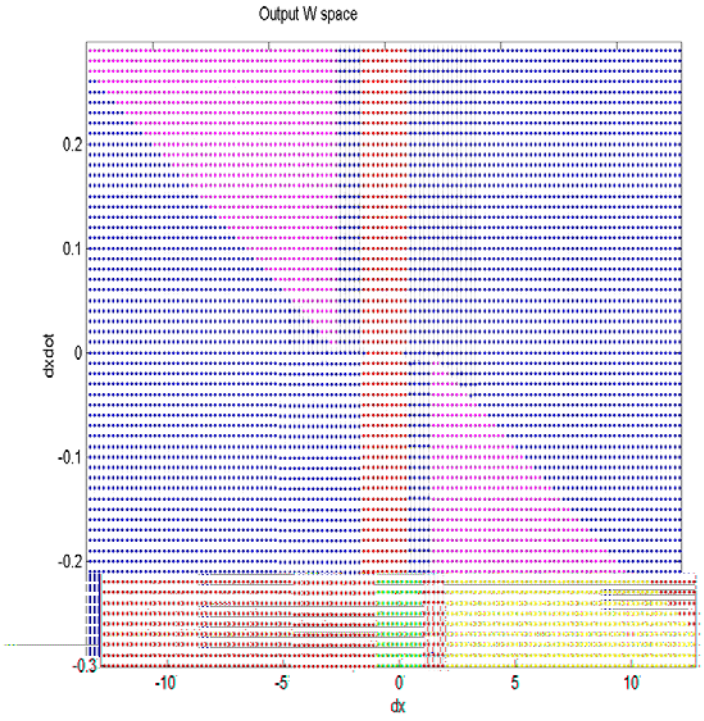

Figure 4.3: Partitioning of the output space. Red is ω_1 (PROCON), purple is ω_2 (LINCON), blue is ω_{34} (OTHER)

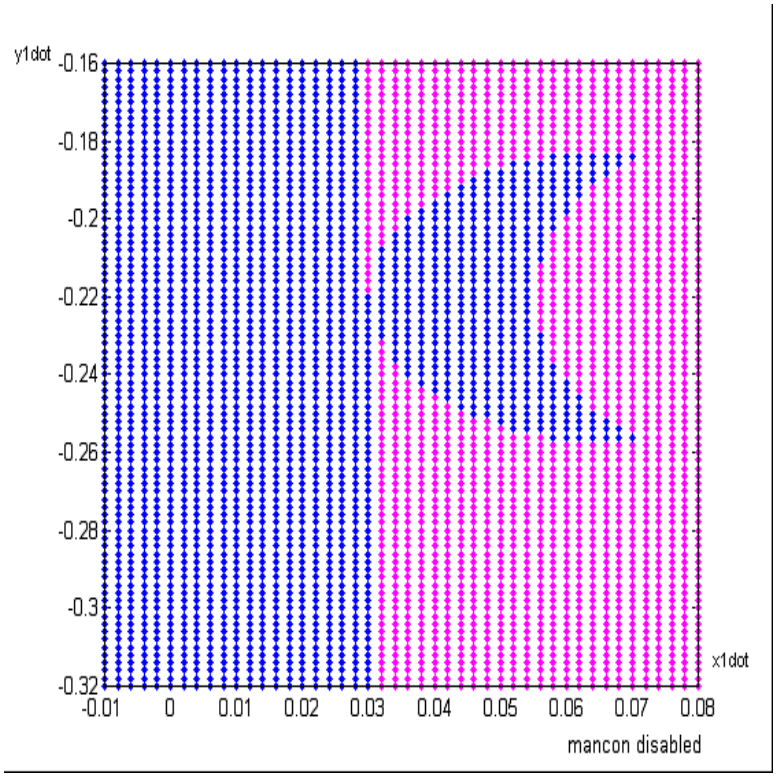

Figure 4.4: Partitioning of the state space. Red is θ_1 (PROCON; not visible in this figure), purple is θ_2 (LINCON), blue is θ_{34} (OTHER).

4.2.4 Partitioning of Input Space and Qualitative Inputs

Mapping from the state partitions

$$\Theta = \{\theta_1, \theta_2, \theta_{34}\}$$

to the input space, we obtain three regions (not necessarily connected) in the input space.

$$f(\mathbf{U}_i) = \mathbf{Q}_i, \ i = 1, 2, 34. \tag{4.9}$$

We call it "level 1" partitioning, since we have not introduced MANCON, as yet. The three regions (level 1) in the input space can be represented by three symbols in the qualitative space:

$$\Lambda_1 = \{\lambda_1, \lambda_2, \lambda_{34}\} \tag{4.10}$$

where λ_1 corresponds to PROCON, λ_2 corresponds to LINCON and λ_{34} corresponds to OTHER.

An example of a projection of the input space partitions onto 2D is shown in Figure 4.5.

4.2.5 A Moore Machine Representation of the Dynamical System

At this point, since MANCON is not considered, the output space, state space and input space partitions can be defined in exactly the same way as in [36]. A Moore machine (automaton) can represent the qualitative state transitions, as shown in Figure 4.6. The states of this automaton are qualitative states defined in the previous section. Transitions are caused by qualitative inputs. Since this is a Moore machine, the outputs are associated with the states of the automaton.

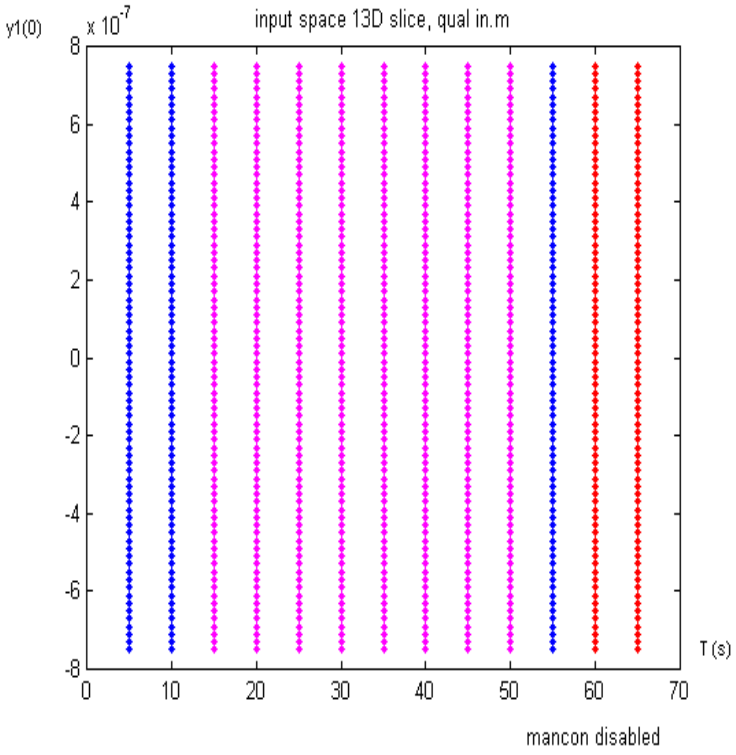

Figure 4.5: Partitioning of the input space. Red is λ_1 (PROCON), purple is λ_2 (LINCON), blue is λ_{34} (OTHER).

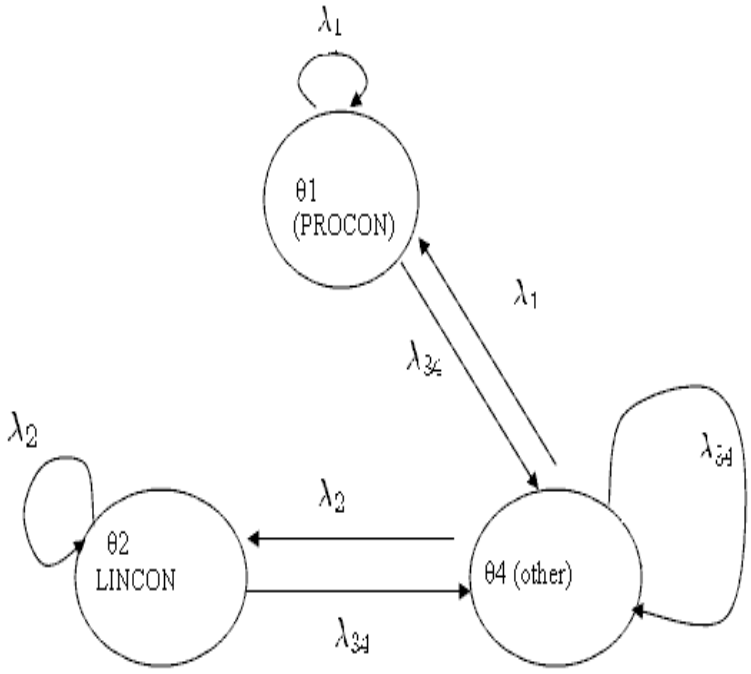

Figure 4.6: Moore Machine. Qualitative state transitions without considering MANCON (level 1).

From this automaton, we can see that when the aircraft pair is in a conflict state, say PROCON, it can transition to OTHER by applying the input λ_{34}. λ_{34} defines a class of maneuvers, each of which takes the aircraft pair from either the PROCON or the LINCON conflict situation to the OTHER situation.

4.2.6 Partitioning of Input Space with MANCON Included

The Moore machine representation in Figure 4.6 captures two types of alert - PROCON and LINCON. MANCON is defined in terms of the input-space

variables. So now we need to expand the representation of Figure 4.6 to account for MANCON. Since inputs define transitions, in order to achieve this goal, we need to first split some of the qualitative inputs into two or more, depending on the subpartitioning of the input space, and then associate outputs (alerts) with transitions, rather than with states. Outputs are associated with transitions in a Mealy machine. Therefore, our next step is to develop a Mealy machine representation of this dynamical system. Towards this goal, we first sub-partition the input space. In the general case, this means that each of the PROCON and LINCON partitions of the input space is subpartitioned by the MANCON partitioning, meaning that the resulting partition is a Cartesian product of the two partitions. However, in this particular example, MANCON is issued only if no PROCON or LINCON is issued. This means that only the OTHER partition needs to be subpartitioned by MANCON. As a result, we partition OTHER into two regions labeled by:

$$\Lambda^2 = \{\lambda^m, \lambda^s\} \tag{4.11}$$

where λ^m corresponds to MANCON and λ^s corresponds to SAFE.

From the cross product $\Lambda_1 \times \Lambda^2$, the only possible (allowed) combinations are : $< \lambda_1, \lambda^s >$, $< \lambda_2, \lambda^s >$,$< \lambda_{34}, \lambda^m >$ and $< \lambda_{34}, \lambda^s >$. These four combinations are denoted as

$$\Lambda = \{\lambda_1^s, \lambda_2^s, \lambda_{34}^m, \lambda_{34}^s\} \tag{4.12}$$

Based upon the partitioning of the input space, a Mealy machine automaton can be constructed as shown in Figure 4.7. The outputs that used to be associated with the states in the Moore machine have been moved to the transitions, and some of the transitions have been due to the splitting

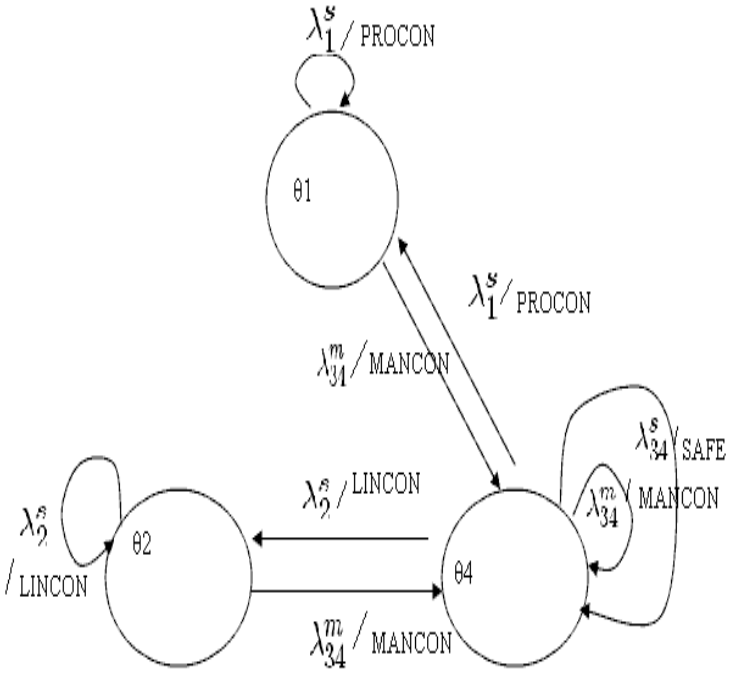

Figure 4.7: Mealy Machine. Qualitative state transitions with MANCON included (level 2).

of the input state partitions. Consequently, multiple transitions may exist for any given two states.

4.2.7 Converting the Mealy Machine to a Moore Machine

While the Mealy machine of Figure 4.7 captures all three alerts, it is not very convenient to use [74]. A Moore model is very easy to code – "the transition may be often implemented just by constants as initialized tables". Whereas the program for a Mealy model becomes so complex that one often loses the state machine in the confusing code. Note that in order to determine what

qualitative state the system is in one has to keep track of the history of transitions until the current time. From this point of view, the Moore machine representation is better. It is known, however, that a Mealy machine can be converted to a Moore machine, and vice versa.

So now we convert the previously obtained Mealy machine (Figure 4.7) to a Moore machine. The procedure is standard and is given in many articles such as [45].

1. For each transition, move the output associated with the transition "forward" into the next state, i.e., associate the output with the state.

2. If this results in a state with two different outputs, then "split" that state into as many states as there are different outputs.

3. The "next" states for the created states are the same next states as for the original state (i.e., same as for the state that was split in the previous step).

A Moore machine representation of our conflict alert resolution problem is shown in Figure 4.8. As we can see, an additional state has been created as a result of splitting the θ_{34} qualitative state. So now there are no multiple transitions between two states and thus the state can be recognized directly. This Moore machine includes four abstract states: $\xi_1, \xi_2, \xi_3, \xi_4$. The mapping between the concrete states of the Mealy machine to the abstract states of the Moore machine is described below.

Let θ_i denote a state of the Mealy machine (we sometimes call it "original state"), λ_j denote qualitative input and ξ_k denote a state in the Moore machine (we sometimes call it "abstract state"). For any state θ_i and any qualitative input λ_j the Moore-Mealy conversion procedure is unique (cf.

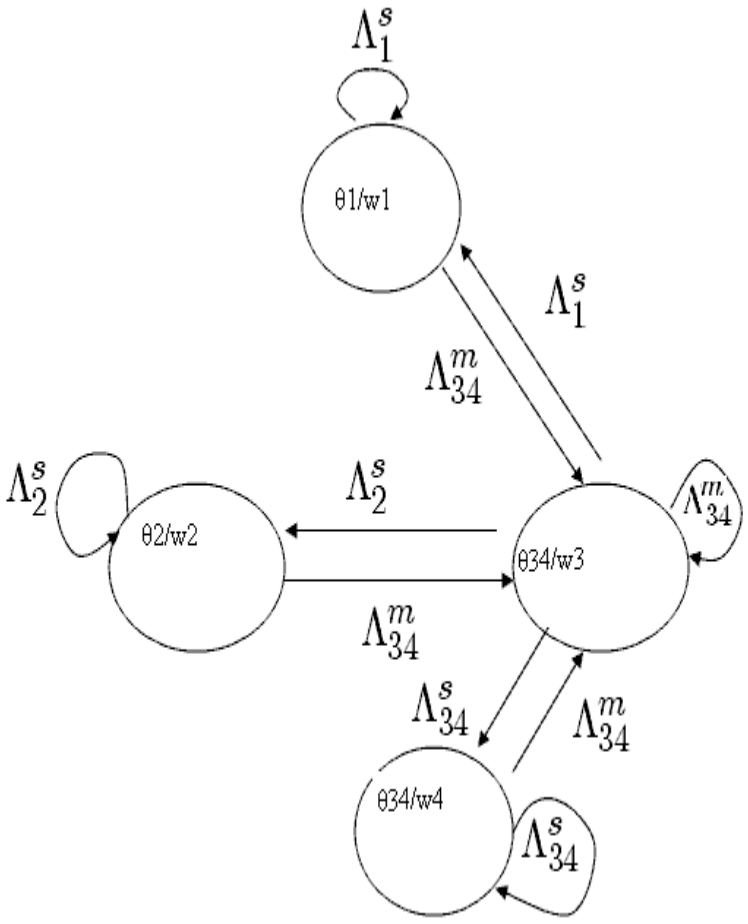

Figure 4.8: Complete Moore Machine. Qualitative state transitions, with MANCON included.

[45]), i.e., there is a function β defined by this conversion procedure

$$\xi_k = \beta(\theta_i, \lambda_j) \tag{4.13}$$

For instance, for the problem discussed in Chapter 4, we know that

$$\xi_1 = \beta(\theta_1, .)$$

$$\xi_2 = \beta(\theta_2, .)$$

$$\xi_3 = \beta(\theta_{34}, \lambda_{34}^m)$$

$$\xi_4 = \beta(\theta_{34}, \lambda_{34}^s)$$

This representation is very convenient for the sake of comprehension. One does not need to trace transitions in order to determine the current state. It is also easy to understand how to search for desired plans of control actions - one just needs to find a sequence of qualitative inputs such that they drive from the current state to another (desired) state. However, this representation adds some complexity to the approach described in [36]. This complexity is due to the fact that the newly created states are abstract, i.e., they don't have constraints associated with them directly. This means that an algorithm for state identification needs to be developed.

For instance, in the representation we have developed so far for our alert resolution problem the two qualitative states - θ_1 and θ_2 (corresponding to PROCON and LINCON, respectively) - have a simple interpretation through the constraints that define these qualitative states. The state corresponding to OTHER, on the other hand, does not represent a clear physical status of the aircraft pair yet; it just represents some region that is neither associated with PROCON nor with LINCON. From the Mealy machine of Figure 4.7, we can see that the outputs were different when the inputs were different,

even with the same state θ_{34} . Thus this state was split into two abstract states.

4.3 Pseudo Code for the Level 1 Partitioning Algorithm

In Table 4.3.1, Table 4.3.2 and Table 4.3.3 we give the partitioning algorithms of the conflict alert problem spaces (output space, state space and input space, respectively) without taking MANCON into consideration. With only PROCON, LINCON and OTHER, the partitioning is straight-forward and it just follows the framework described in [36].

First, the partitioning is done in the output space (see Figure 4.9). Then the inverse of the output function maps the partitions of the output space to the partitions of the state space. Then the inverse of the state transition function maps the partitions of the state space to the partitions of the input space.

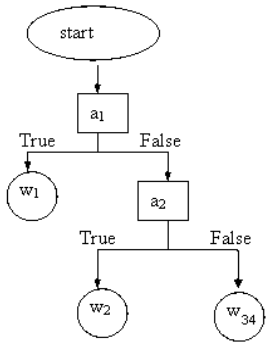

Figure 4.9: Flowchart of the partitioning logic of the output space for the conflict resolution problem

4.3.1 Algorithm (P_Ω) for Partitioning the Output Space for the Conflict Resolution Problem

Notation:

X_x, X_y	: position; $x \in X_x, y \in X_y$
V_x, V_y	: velocity; $\dot{x} \in V_x, \dot{y} \in V_y$
$\triangle X_x$	: position difference in x direction; $\triangle x = x_1 - x_2, \triangle x \in \triangle X_x$
$\triangle X_y$	: position difference in y direction; $\triangle y = y_1 - y_2, \triangle y \in \triangle X_y$
$\triangle V_x$	: velocity difference in x direction; $\triangle \dot{x} = \dot{x}_1 - \dot{x}_2, \triangle \dot{x} \in \triangle V_x$
$\triangle V_y$	: velocity difference in y direction; $\triangle \dot{y} = \dot{y}_1 - \dot{y}_2, \triangle \dot{y} \in \triangle V_y$
$\triangle X$	: $\triangle X_x \times \triangle X_y$
$\triangle V$	: $\triangle V_x \times \triangle V_y$
$\triangle X \times \triangle V$	: output space
Ω	: $\{\omega_1, \omega_2, \omega_{34}\}$ - qualitative outputs
P_Ω	: $\triangle X \times \triangle V \rightarrow \Omega$
d	: $d = \sqrt{(\triangle x)^2 + (\triangle y)^2}$, distance between two aircraft
R	: $R = \sqrt{(\triangle \dot{x})^2 + (\triangle \dot{y})^2}$, approach rate of the two aircraft
$LAT1$	: time of coming into conflict by linear prediction; computed from $\triangle x, \triangle y, \triangle \dot{x}, \triangle \dot{y}$,
$\delta x, \delta y$	: resolution in $\triangle x$ and $\triangle y$
$\delta \dot{x}, \delta \dot{y}$	: resolution in $\triangle \dot{x}$ and $\triangle \dot{y}$

$[\triangle x_l, \triangle x_h]$: valid range (low and high bound) of $\triangle x$
$[\triangle y_l, \triangle y_h]$: valid range (low and high bound) of $\triangle y$
$[\triangle \dot{x}_l, \triangle \dot{x}_h]$: valid range (low and high bound) of $\triangle \dot{x}$
$[\triangle \dot{y}_l, \triangle \dot{y}_h]$: valid range (low and high bound) of $\triangle \dot{y}$

System parameters:
$d_{proxim1}$: threshold value in distance
$d_{proxim2}$: threshold value in distance
R_{proxim} : threshold value in speed
$T_{lookahead}$: look ahead time for predicting positions
d_{safe} : threshold value in distance

Constraints:
c_1 : $(d < d_{proxim1}) \vee (d < d_{proxim2} \wedge R > R_{proxim})$
: meaning the two aircraft are very close, or close and approaching;
c_2 : $(LAT1 < T_{lookahead}) \wedge (LAT1 > 0)$
: meaning by linear prediction, will be in conflict within 40 seconds;

System parameter values used in this computation:
$d_{proxim1}$: 0.75nmi
$d_{proxim2}$: 1 nmi
R_{proxim} : 0.0096 nmi/sec
$T_{lookahead}$: 40 seconds
d_{safe} : 4 nmi

Input: $\delta x, \delta y, [\triangle x_l, \triangle x_h], [\triangle y_l, \triangle y_h], \delta \dot{x}, \delta \dot{y}, [\triangle \dot{x}_l, \triangle \dot{x}_h], [\triangle \dot{y}_l, \Delta \dot{y}_h], \Omega$

Output: $P_\Omega : \triangle X \times \triangle V \rightarrow \Omega$

begin{program}
for $\triangle x = \triangle x_l : \delta x : \triangle x_h,\ \triangle y = \triangle y_l : \delta y : \triangle y_h,$
$\triangle \dot{x} = \triangle \dot{x}_l : \delta \dot{x} : \triangle \dot{x}_h,\ \triangle \dot{y} = \triangle \dot{y}_l : \delta \dot{y} : \triangle \dot{y}_h$
if(c_1)
$P_\Omega(\triangle x, \triangle y, \triangle \dot{x}, \triangle \dot{y}) \leftarrow \omega_1$
else if(c_2)
$P_\Omega(\triangle x, \triangle y, \triangle \dot{x}, \triangle \dot{y}) \leftarrow \omega_2$
else
$P_\Omega(\triangle x, \triangle y, \triangle \dot{x}, \triangle \dot{y}) \leftarrow \omega_{34}$

end(c_2)
end(c_1)
end {for }
end{program}.

4.3.2 Algorithm (P_Θ) for Partitioning the State Space

Notation:

$\mathbf{x_1}, \mathbf{x_2}$	: position vector for aircraft 1 and 2
$\mathbf{v_1}, \mathbf{v_2}$	: velocity for aircraft 1 and 2
$\mathbf{x_1} \times \mathbf{v_1} \times \mathbf{x_2} \times \mathbf{v_2}$	: state space
Θ	: $\{\theta_1, \theta_2, \theta_{34}\}$ - qualitative states
P_Θ	: $\mathbf{x_1} \times \mathbf{v_1} \times \mathbf{x_2} \times \mathbf{v_2} \rightarrow \Theta$
$\delta x, \delta y$	: resolution in x and y ;
$\delta \dot{x}, \delta \dot{y}$	: resolution in $\dot{x}$ and $\dot{y}$;
$[x_l, x_h], [y_l, y_h]$	: valid range (low and high bound) of position
$[\dot{x}_l, \dot{x}_h], [\dot{y}_l, \dot{y}_h]$	: valid range (low and high bound) of velocity
$\triangle X \times \triangle V$	: output space (see Section 4.3.1)
g	: $\mathbf{x_1} \times \mathbf{v_1} \times \mathbf{x_2} \times \mathbf{v_2} \rightarrow \triangle X \times \triangle V$
	: output function (see Chapter 5)

Input: $\delta x, \delta y, [x_l, x_h], [y_l, y_h], \delta \dot{x}, \delta \dot{y}, [\dot{x}_l, \dot{x}_h], [\dot{y}_l, \dot{y}_h], g, \Theta$

Output: $P_\Theta : \mathbf{x_1} \times \mathbf{v_1} \times \mathbf{x_2} \times \mathbf{v_2} \rightarrow \Theta$

begin{program}
for $x_1 = x_l : \delta x : x_h$, $y_1 = y_l : \delta y : y_h$, $x_2 = x_l : \delta x : x_h$, $y_2 = y_l : \delta y : y_h$,
$\dot{x}_1 = \dot{x}_l : \delta \dot{x} : \dot{x}_h$, $\dot{y}_1 = \dot{y}_l : \delta \dot{y} : \dot{y}_h$, $\dot{x}_2 = \dot{x}_l : \delta \dot{x} : \dot{x}_h$, $\dot{y}_2 = \dot{y}_l : \delta \dot{y} : \dot{y}_h$
$\omega \leftarrow P_\Omega(g(x_1, y_1, \dot{x}_1, \dot{y}_1, x_2, y_2, \dot{x}_2, \dot{y}_2))$
for i=[1,2,34]
if($\omega == \omega_i$)
$P_\Theta(x_1, y_1, \dot{x}_1, \dot{y}_1, x_2, y_2, \dot{x}_2, \dot{y}_2) \leftarrow \theta_i$
exit forloop(i)
end if
end for (i)
end {for}
end{program}.

4.3.3 Algorithm (P_{Λ_1}) for Level 1 Partitioning of the Input Space

Notation:

$\mathbf{Q}$	: $\mathbf{x_1} \times \mathbf{v_1} \times \mathbf{x_2} \times \mathbf{v_2}$, state space (see section 4.3.2)
$\mathbf{Q}^0$	: initial state
$\mathbf{a}_1 = a_{x1} \times a_{y1}$	: acceleration for aircraft 1
$\mathbf{a}_2 = a_{x2} \times a_{y2}$	: acceleration for aircraft 2
T	: time
$\mathbf{Q}^0 \times \mathbf{a_1} \times \mathbf{a_2} \times \mathbf{T}$	: **TQX** (input space), see Equation 4.7
Λ_1	: $\{\lambda_1, \lambda_2, \lambda_{34}\}$ - qualitative inputs
P_{Λ_1}	: $\mathbf{Q}^0 \times \mathbf{a}_1 \times \mathbf{a}_2 \times T \rightarrow \Lambda_1$
$x_1^0, x_2^0, y_1^0, y_2^0$	: initial position x and y coordinates of aircraft 1 and 2
$\dot{x}_1^0, \dot{x}_2^0, \dot{y}_1^0, \dot{y}_2^0,$	: initial velocities x and y components of aircraft 1 and 2
$a_{x1}, a_{y1}, a_{x2}, a_{y2}$	: acceleration x and y components of aircraft 1 and 2
δT	: resolution in T
$\delta x, \delta y$	: resolution in x and y
$\delta\dot{x}, \delta\dot{y}$	: resolution in $\dot{x}$ and $\dot{y}$
$\delta a_x, \delta a_y$	: resolution in a_x and a_y
$[T_l, T_h]$	: valid range (low and high bound) of T
$[x_l, x_h], [y_l, y_h]$	: valid range (low and high bound) of position
$[\dot{x}_l, \dot{x}_h], [\dot{y}_l, \dot{y}_h]$	: valid range (low and high bound) of velocity
$[a_{xl}, a_{xh}], [a_{yl}, a_{yh}]$	: valid range (low and high bound) of accelerations
f	: $\mathbf{Q}_0 \times \mathbf{a}_1 \times \mathbf{a}_2 \times T \rightarrow \mathbf{x_1} \times \mathbf{v_1} \times \mathbf{x_2} \times \mathbf{v_2}$ - state transition function

Input: $\delta T, [T_l, T_h], \delta x, \delta y, [x_l, x_h], [y_l, y_h], \delta\dot{x}, \delta\dot{y}, [\dot{x}_l, \dot{x}_h], [\dot{y}_l, \dot{y}_h], \delta a_x, \delta a_y, [a_{xl}, a_{xh}], [a_{yl}, a_{yh}], f, \Lambda_1$

Output: $P_{\Lambda_1} : \mathbf{Q}^0 \times \mathbf{a}_1 \times \mathbf{a}_2 \times T \rightarrow \Lambda_1$

begin{program}
for $x_1^0 = x_l : \delta x : x_h, x_2^0 = x_l : \delta x : x_h, y_1^0 = y_l : \delta y : y_h, y_2^0 = y_l : \delta y : y_h$
$\dot{x}_1^0 = \dot{x}_l : \delta\dot{x} : \dot{x}_h, \dot{x}_2^0 = \dot{x}_l : \delta\dot{x} : \dot{x}_h, \dot{y}_1^0 = \dot{y}_l : \delta\dot{y} : \dot{y}_h, \dot{y}_2^0 = \dot{y}_l : \delta\dot{y} : \dot{y}_h,$
$a_{x1} = a_{xl} : \delta a_x : a_{xh}, a_{y1} = a_{yl} : \delta a_y : a_{yh}, a_{x2} = a_{xl} : \delta a_x : a_{xh}, a_{y2} = a_{yl} : \delta a_y : a_{yh}$
$T = T_l : \delta T : T_h$
$\theta = P_\Theta(f(x_1^0, x_2^0, y_1^0, y_2^0, \dot{x}_1^0, \dot{x}_2^0, \dot{y}_1^0, \dot{y}_2^0, a_{x1}, a_{y1}, a_{x2}, a_{y2}, T))$
for k=[1,2,34]
if($\theta == \theta_k$)

$P_{\Lambda_1}(x_1^0, x_2^0, y_1^0, y_2^0, \dot{x}_1^0, \dot{x}_2^0, \dot{y}_1^0, \dot{y}_2^0, a_{x1}, a_{y1}, a_{x2}, a_{y2}, T) \leftarrow \lambda_k$
exit forloop(k)
end if
end for (k)
end {for }
end{program}.

4.4 Pseudo Code for the Level 2 Partitioning Algorithm (MANCON included)

In 4.3.3, MANCON was not considered. Now we present the algorithm for partitioning the input space, including the MANCON alerts.

4.4.1 Algorithm (P_{Λ^2}) for the Level 2 Partitioning of the Input Space

Notation:

$\mathbf{x_1}, \mathbf{x_2}$	: position of aircraft 1 and 2
$\mathbf{v_1}, \mathbf{v_2}$	,v

$[T_l, T_h]$	: valid range (low and high bound) of T
$[x_l, x_h], [y_l, y_h]$	: valid range (low and high bound) of position
$[\dot{x}_l, \dot{x}_h], [\dot{y}_l, \dot{y}_h]$	: valid range (low and high bound) of velocity
$[a_{xl}, a_{xh}], [a_{yl}, a_{yh}]$	: valid range (low and high bound) of accelerations
f	: $Q_0 \times \mathbf{a_1} \times \mathbf{a_2} \times \mathbf{T} \rightarrow \mathbf{x_1} \times \mathbf{v_1} \times \mathbf{x_2} \times \mathbf{v_2}$ - state transition function

Constraints:

c_3	: $\omega_{thresh} \wedge [(s_1 < 0) \vee (s_2 > 0) \vee (s_3 > 0) \vee (s_4 > 0)]$ meaning one aircraft is turning into the other aircraft and will cause a conflict situation under the turning model prediction (s_1,s_2,s_3,s_4 are from eq.3.4)

Input:	$\delta T, [T_l, T_h], \delta x, \delta y, [x_l, x_h], [y_l, y_h], \delta\dot{x}, \delta\dot{y}, [\dot{x}_l, \dot{x}_h], [\dot{y}_l, \dot{y}_h], \delta a_x, \delta a_y, [a_{xl}, a_{xh}], [a_{yl}, a_{yh}], f, \Lambda_1, \Lambda^2$
Output:	$P_\Lambda : \mathbf{Q^0} \times \mathbf{a_1} \times \mathbf{a_2} \times \mathbf{T} \rightarrow \mathbf{\Lambda}$

begin{program}
for $x_1^0 = x_l : \delta x : x_h, x_2^0 = x_l : \delta x : x_h, y_1^0 = y_l : \delta y : y_h, y_2^0 = y_l : \delta y : y_h$
$\dot{x}_1^0 = \dot{x}_l : \delta\dot{x} : \dot{x}_h, \dot{x}_2^0 = \dot{x}_l : \delta\dot{x} : \dot{x}_h, \dot{y}_1^0 = \dot{y}_l : \delta\dot{y} : \dot{y}_h, \dot{y}_2^0 = \dot{y}_l : \delta\dot{y} : \dot{y}_h,$
$a_{x1} = a_{xl} : \delta a_x : a_{xh}, a_{y1} = a_{yl} : \delta a_y : a_{yh}, a_{x2} = a_{xl} : \delta a_x : a_{xh}, a_{y2} = a_{yl} : \delta a_y : a_{yh}$
$T = T_l : \delta T : T_h$
θ= $P_\Theta(f(x_1^0, x_2^0, y_1^0, y_2^0, \dot{x}_1^0, \dot{x}_2^0, \dot{y}_1^0, \dot{y}_2^0, a_{x1}, a_{y1}, a_{x2}, a_{y2}, T))$
for k=[1,2,34]
if($\theta == \theta_k$)
$P_{\Lambda_1}(x_1^0, x_2^0, y_1^0, y_2^0, \dot{x}_1^0, \dot{x}_2^0, \dot{y}_1^0, \dot{y}_2^0, a_{x1}, a_{y1}, a_{x2}, a_{y2}, T) \leftarrow \lambda_k$
exit forloop(k)
end if
end for (k)
$if(k == 34)$
$if(c3)$
$P_{\Lambda^2}(x_1^0, x_2^0, y_1^0, y_2^0, \dot{x}_1^0, \dot{x}_2^0, \dot{y}_1^0, \dot{y}_2^0, a_{x1}, a_{y1}, a_{x2}, a_{y2}, T) \leftarrow \lambda^m$
$P_\Lambda(x_1^0, x_2^0, y_1^0, y_2^0, \dot{x}_1^0, \dot{x}_2^0, \dot{y}_1^0, \dot{y}_2^0, a_{x1}, a_{y1}, a_{x2}, a_{y2}, T) \leftarrow \lambda_{34}^m$
else
$P_{\Lambda^2}(x_1^0, x_2^0, y_1^0, y_2^0, \dot{x}_1^0, \dot{x}_2^0, \dot{y}_1^0, \dot{y}_2^0, a_{x1}, a_{y1}, a_{x2}, a_{y2}, T) \leftarrow \lambda^s$
$P_\Lambda(x_1^0, x_2^0, y_1^0, y_2^0, \dot{x}_1^0, \dot{x}_2^0, \dot{y}_1^0, \dot{y}_2^0, a_{x1}, a_{y1}, a_{x2}, a_{y2}, T) \leftarrow \lambda_{34}^s$
end $(c3)$

else (i.e., $k == 1 or 2$)
$P_\Lambda(x_1^0, x_2^0, y_1^0, y_2^0, \dot{x}_1^0, \dot{x}_2^0, \dot{y}_1^0, \dot{y}_2^0, a_{x1}, a_{y1}, a_{x2}, a_{y2}, T) \leftarrow \lambda_k^s$
end
end {for }
end{program}.

4.5 Consistency of Partitioning by Multiple Criteria

The partitioning problem described in the previous section is different from the problem addressed in [36]. In particular, it adds two levels of complexity to the approach in [36] in two ways. Each of these complexities requires some additional investigations.

1. Unlike [36], it partitions the system spaces using multiple criteria. In the first step, it uses LINCON and PROCON. To derive the partitions of the state space and of the input state we used the same algorithm as in [36].

   Now the question is - should we first derive two partitions of the output space using PROCON and LINCON, then map them back to the state space and then overlay the two partitions to obtain a final refined partitioning of the state space? Or should we overlay the two partitions of the output space obtaining a combined partitioning of the output space and then map it back to the state space? Intuitively the two approaches should give the same result. But to be sure we need to formulate and prove an appropriate theorem. A similar question can be asked of the mapping of the partitioning of the state space back to the input space.

2. The second question arrives due to the fact that in our conflict alert

resolution problem we first constructed a Moore machine using PROCON and LINCON, then we constructed a Mealy machine that also incorporates MANCON, then we converted it to a Moore machine by defining some abstract states. The approach in [36], on the other hand, only constructs a Moore machine.

The question is whether the final Moore machine representation of the dynamical system under consideration guarantees consistency. In other words, the question is whether reasoning with a such constructed Mealy machine is sound. Again, we could have formulated and proved an appropriate theorem to address this question. Instead, we base a conclusion that the constructed abstraction is consistent on the fact that Moore and Mealy machines are equivalent.

To formulate and prove the theorem about the equality of partitions provided by the two ways outlined above we need to introduce some notation. Suppose LINCON results in the partitioning of W into two subsets:

$$P^L = \{L_0, L_1\} \tag{4.14}$$

where L_0 is the area with LINCON not satisfied, L_1 is the area with LINCON satisfied, i.e., LINCON alert. Since P^L is a partition, it satisfies the following:

$$L_0 \cap L_1 = \emptyset; \tag{4.15}$$

$$L_0 \cup L_1 = W. \tag{4.16}$$

Suppose PROCON results in the partitioning of the output space into two subsets:

$$P^P = \{P_0, P_1\} \tag{4.17}$$

where P_0 is the area with PROCON not satisfied, P_1 is the area with PROCON satisfied, i.e., PROCON alert. Again, the following must hold:

$$P_0 \cap P_1 = \emptyset; \tag{4.18}$$

$$P_0 \cup P_1 = W. \tag{4.19}$$

The partitions of the state space generated by the inverse of the W partition by P^P and P^L, respectively, can be represented as follows

$$g_P^{-1}(W) = g^{-1}(P_0) \cup g^{-1}(P_1) \tag{4.20}$$

$$g_L^{-1}(W) = g^{-1}(L_0) \cup g^{-1}(L_1) \tag{4.21}$$

By superposing the two partitions we obtain the following partitioning of the state space Q

$$g_{P\cup L}^{-1}(W) = (g^{-1}(P_0) \cup g^{-1}(P_1)) \cap (g^{-1}(L_0) \cup g^{-1}(L_1)) \tag{4.22}$$

This is equal to

$$\begin{aligned} g_{P\cup L}^{-1}(W) \ = \qquad & \\ g^{-1}(P_0) \cap g^{-1}(L_0) \cup g^{-1}(P_0) \cap g^{-1}(L_1) & \\ \cup\ g^{-1}(P_1) \cap g^{-1}(L_0) \cup g^{-1}(P_1) \cap g^{-1}(L_1) & \end{aligned} \tag{4.23}$$

On the other hand, we can first superpose the two partitions of the output space and obtain the following partitioning

$$W = P_0 \cap L_0 \cup P_0 \cap L_1 \cup P_1 \cap L_0 \cup P_1 \cap L_1 \tag{4.24}$$

Through the inverse of g, we obtain the following partitioning

$$g_{P\cap L}^{-1}(W) = g^{-1}(P_0 \cap L_0) \cup g^{-1}(P_0 \cap L_1) \cup g^{-1}(P_1 \cap L_0) \cup g^{-1}(P_1 \cap L_1) \tag{4.25}$$

Theorem 1 Partitions of the state space generated by $g^{-1}_{P \cup L}(W)$ and $g^{-1}_{P \cap L}(W)$ are equal.

$$(i) \quad g^{-1}(P_0 \cap L_0) = g^{-1}(P_0) \cap g^{-1}(L_0)$$
$$(ii) \quad g^{-1}(P_0 \cap L_1) = g^{-1}(P_0) \cap g^{-1}(L_1)$$
$$(iii) \quad g^{-1}(P_1 \cap L_0) = g^{-1}(P_1) \cap g^{-1}(L_0)$$
$$(iv) \quad g^{-1}(P_1 \cap L_1) = g^{-1}(P_1) \cap g^{-1}(L_1)$$

The proof of this theorem follows from the fact that $f^{-1}(A \cap B) = f^{-1}(A) \cap f^{-1}(B)$.

Chapter 5

Symbolic Reasoning for Conflict Resolution in the X-Y Plane

This chapter explains how symbolic reasoning is applied to the conflict alert resolution problem described in Chapter 3. Our goal is to find a sequence of maneuver advisories on the $X - Y$ plane in a conflict alert situation, so that if the advisories are executed, the aircraft will transit from an unsafe state to the SAFE state. The algorithm will provide these advisories within real-time constraints.

The movement of each aircraft separately is modeled by a dynamic equation - the state transition function. In order to consider the pair of aircraft together, we develop a model in which the dynamic system (the pair) is described by a single state transition function and an output function, in which the joint state vector, the joint output vector, and the joint input vector are used. In the previous chapter, the detailed dynamical system state transition and output functions were not given in detail. They will be provided in the following three sections. This will be followed by the

verification of the dynamic-system properties of the constructed model.

First, some of the symbols and notations are listed below.

$\mathbf{Q}$ – state set

$\mathbf{U}$ – input set (consists of acceleration vectors)

$\hat{\mathbf{A}}$, $\hat{\mathbf{B}}$ – state transition matrices

$\mathbf{a_i}$ – acceleration vectors of the aircraft i

5.1 State Transition Function

The joint state vector for an aircraft pair in the $X-Y$ plane is given by Equation 5.1 below. The input vector is given by Equation 5.2. And the state transition function is given by Equation 5.3.

$$\mathbf{Q} = \begin{bmatrix} x_1 \\ y_1 \\ \dot{x}_1 \\ \dot{y}_1 \\ x_2 \\ y_2 \\ \dot{x}_2 \\ \dot{y}_2 \end{bmatrix} \tag{5.1}$$

$$\mathbf{U} = \begin{bmatrix} a_{x1}(k) \\ a_{y1}(k) \\ a_{x2}(k) \\ a_{y2}(k) \end{bmatrix} \tag{5.2}$$

$$\mathbf{Q}(k) = f(\mathbf{Q}(k-1), \mathbf{U}(k-1), T) =$$

$$\begin{bmatrix} \hat{\mathbf{A}} & \mathbf{0}(4,4) \\ \mathbf{0}(4,4) & \hat{\mathbf{A}} \end{bmatrix} \mathbf{Q}(k-1) + \begin{bmatrix} \hat{\mathbf{B}} & \mathbf{0}(4,2) \\ \mathbf{0}(4,2) & \hat{\mathbf{B}} \end{bmatrix} \cdot \mathbf{U}(k-1) \tag{5.3}$$

where $\mathbf{0}(\mathbf{4},\mathbf{4})$ is a four by four matrix of zeroes, $\mathbf{0}(\mathbf{4},\mathbf{2})$ is a four by two matrix of zeroes and $\hat{\mathbf{A}}$, $\hat{\mathbf{B}}$ are matrices defined [34, 32] below.

$$\hat{\mathbf{A}} = \begin{bmatrix} 1 & 0 & T & 0 \\ 0 & 1 & 0 & T \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{5.4}$$

$$\hat{\mathbf{B}} = \begin{bmatrix} T^2/2 & 0 \\ 0 & T^2/2 \\ T & 0 \\ 0 & T \end{bmatrix} \tag{5.5}$$

The state transition functions for each aircraft $(i = 1, 2)$ are shown next.

$$\begin{bmatrix} x_i(k) \\ y_i(k) \\ \dot{x}_i(k) \\ \dot{y}_i(k) \end{bmatrix} = \hat{\mathbf{A}} \begin{bmatrix} x_i(k-1) \\ y_i(k-1) \\ \dot{x}_i(k-1) \\ \dot{y}_i(k-1) \end{bmatrix} + \hat{\mathbf{B}} \cdot \mathbf{a_i}(k-1) \tag{5.6}$$

$$\mathbf{a_i} = \begin{bmatrix} a_{xi}(k-1) \\ a_{yi}(k-1) \end{bmatrix} \tag{5.7}$$

5.2 Output Function

The output of the system is given by Equation 5.8

$$\begin{bmatrix} x_1 - x_2 \\ y_1 - y_2 \\ \dot{x}_1 - \dot{x}_2 \\ \dot{y}_1 - \dot{x}_2 \end{bmatrix} \tag{5.8}$$

The output function is given by Equation 5.9.

$$\begin{bmatrix} x_1 - x_2 \\ y_1 - y_2 \\ \dot{x}_1 - \dot{x}_2 \\ \dot{y}_1 - \dot{x}_2 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & -1 \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ y_1 \\ \dot{x}_1 \\ \dot{y}_1 \\ x_2 \\ y_2 \\ \dot{x}_2 \\ \dot{y}_2 \end{bmatrix} \tag{5.9}$$

5.3 Input Space

The input space of the aircraft pair is 13-dimensional. It consists of the initial state vector at $k-1$, accelerations at k, and time interval T, as shown in Equation 5.10.

$$\begin{bmatrix} x_1(k-1) \\ y_1(k-1) \\ \dot{x}_1(k-1) \\ \dot{y}_1(k-1) \\ x_2(k-1) \\ y_2(k-1) \\ \dot{x}_2(k-1) \\ \dot{y}_2(k-1) \\ a_{x1}(k) \\ a_{y1}(k) \\ a_{x2}(k) \\ a_{y2}(k) \\ T \end{bmatrix} \tag{5.10}$$

5.4 Verification of Dynamic System Properties

The state transition function has to satisfy some properties such as consistency, semigroup and causality (c.f. [52]). Consistency and causality properties are inherent in the definition of the system provided by Equations 5.1 through 5.10. We provide the formulation of the semigroup property, without proof, below.

Theorem 2 The state transition function f of our ATC problem (Eq. 5.3) satisfies the following property (semigroup property):

$$f(\mathbf{U}(k-2), \mathbf{Q}(k-2), 2) = f(\mathbf{U}(k-1), f(\mathbf{U}(k-2), \mathbf{Q}(k-2), 1), 1)$$

where $\mathbf{U}$ is the input set and $\mathbf{Q}$ is the state set.

5.5 Reasoning Procedures

If the ATC problem could be framed using the conventional Q^2 method, the automaton constructed according to the Q^2 approach [36] would define transitions from one conflict alert status to another conflict alert status with the appropriate input selection. Given this automaton, as for example in Figure 4.6, first the qualitative state that the aircraft pair is currently in would have to be determined based upon the quantitative state estimation and qualitative state partitioning (see Section 4.2.3). Then a qualitative input would have to be selected by analyzing the possible state transitions of the automaton. For instance, if two aircraft are in the LINCON alert status (state θ_2), then the aircraft would have to transit to a less emergent conflict alert status state, i.e., to state θ_4, which in this case includes both the SAFE state and the MANCON state. Then, with the input space partitions that were defined previously (see Section 4.2.1), it is only necessary to select a maneuver within the region that corresponds to the qualitative input λ_4.

However, as described in Chapter 4, our ATC problem does not fall in the frame of the original Q^2 approach. As explained in Chapter 4, the partitioning of the output space is in fact not a simple task. When MANCON is considered, the input space is partitioned into PROCON, LINCON, MANCON and SAFE ($\lambda_1^s, \lambda_2^s, \lambda_{34}^m, \Lambda_{34}^s$) by using the cross product of two levels of partitioning. The original state space is constructed into an abstract state space where the input becomes a part of it. This creates a complication in the reasoning as compared to the conventional Q^2 approach. The complication comes from the fact that now it is more difficult to determine the current qualitative state of the aircraft pair. To determine the current qualitative state, the procedure outlined in Section 4.2.7 should be followed.

Pseudo code for the algorithm will be developed and presented in Section 5.5.1.

5.5.1 Pseudo Code for Determining Current Abstract State

First we give an outline of the procedure for determining the qualitative abstract state that the aircraft pair is in. An initial outline of this procedure was first given in Section 4.2.7.

λ_i, ξ_2 means that this is the abstract state that was constructed from the concrete state θ_2 with any λ_i, ξ_3 means that this is the abstract state that was constructed from the concrete state θ_{34} by applying the input λ_{34}^m, ξ_4 means that this is the abstract state that was constructed from the concrete state θ_{34} by applying the input λ_{34}^s.

The question is whether the procedure for determining the abstract state is unique. In other words, we need to prove that the determination of abstract state is unique, i.e., the unique combination of θ_i and λ_j would create a unique abstract state ξ_k.

5.5.2 Determining Desired Next Abstract State

The second thing for Q^2 reasoning is to determine the qualitative abstract state that the aircraft pair should be in at the next scan. This can be done by looking up the automaton map, shown in Figure 4.8. If the aircraft pair is in conflict state, a new state should be selected such that it is no more emergent than the current state. The emergency levels from low to high are: SAFE, MANCON, LINCON, PROCON (most urgent), i.e., ξ_4 (built from θ_{34} and λ_{34}^s) is more preferred than ξ_3 (built from θ_{34} and λ_{34}^m), ξ_3 is more preferred than ξ_2, ξ_2 is more preferred than ξ_1. This can be formalized by defining a preference function

$$b : \Xi \rightarrow \Xi \tag{5.11}$$

For the conflict resolution problem, this function is defined as:

$$\begin{aligned} b(\xi_1) &= \xi_3 \\ b(\xi_2) &= \xi_3 \\ b(\xi_3) &= \xi_4 \end{aligned} \tag{5.12}$$

$$b(\xi_4) = \xi_4$$

(Note the first equation is not $b(\xi_1) = \xi_2$.)

Since some of the preferred states are not reachable from the current state, the final decision should be made by analyzing possible transitions (Section 5.5.3).

5.5.3 Determining Qualitative Input

The third thing for Q^2 reasoning is to find the qualitative input that will result in a transition to the selected desired state. This can be achieved by selecting a qualitative input that leads to a less emergent state. The automaton map provides the choices from $(\lambda_1^s, \lambda_2^s, \lambda_{34}^m, \lambda_{34}^s)$. The qualitative input that leads to the desired state is selected. This gives a coarse solution to the maneuver advisory.

Once the qualitative input is selected, the values of the input variables need to be selected that fall within the given qualitative input. Any set of values in this region would put the system to the desired state. However the best one should be searched for using a goal function defined in Chapter 3. Since the search is only within a subregion of the input space, it is less time consuming than a full-space search would be. Also, the search is in the subregion that guarantees the desired existence of a solution.

In Section 5.5.4 we show an algorithm for selecting qualitative state. This algorithm provides a solution for a one-step RA. This algorithm will be utilized in the process of planning a full RA solution (multiple steps).

5.5.4 Algorithm for Q^2 Reasoning (selection of qualitative input)

% Reasoning algorithm P_{RA}

Notation:

X_x, X_y	: position; $x \in X_x, y \in X_y$
V_x, V_y	: velocity; $\dot{x} \in V_x, \dot{y} \in V_y$
X	: $X_x \times X_y$, position
V	: $V_x \times V_y$, velocity vector
$\mathbf{x}_1, \mathbf{x}_2$	: positions for aircraft 1 and 2,
$\mathbf{v}_1, \mathbf{v}_2$	: velocity vectors for aircraft 1 and 2
$a_{x1}, a_{y1}, a_{x2}, a_{y2}$	: accelerations of aircraft 1 and 2
$\mathbf{a}_1 = [a_{x1}, a_{y1}]$	: acceleration vector for aircraft 1
$\mathbf{a}_2 = [a_{x2}, a_{y2}]$	: acceleration vector for aircraft 2
T	: time interval between consecutive state transitions
Λ	: qualitative inputs, $\lambda_k \in \Lambda$
Θ	: qualitative states in the Mealy machine, $\theta_m \in \Theta$
Ω	: qualitative outputs, $\omega_n \in \Omega$
Ξ	: qualitative abstract states in the Moore machine, $\xi_j \in \Xi$
P_Λ	: partitioning algorithm of input space
P_Θ	: partitioning algorithm of state space
P_Ω	: partitioning algorithm of output space
ϕ	: $\Lambda \times \Theta \rightarrow \Theta$ - qualitative state transition function
ξ'_j	: next abstract state
b	: preference function on abstract states

Input: $\mathbf{x}_1, \mathbf{v}_1, \mathbf{x}_2, \mathbf{v}_2, \mathbf{a}_1, \mathbf{a}_2$

Output: RA

begin{program}

$\omega_n = P_\Omega(\mathbf{x}_1, \mathbf{v}_1, \mathbf{x}_2, \mathbf{v}_2)$

$\theta_m = P_\Theta(\mathbf{x}_1, \mathbf{v}_1, \mathbf{x}_2, \mathbf{v}_2)$

$\lambda_k = P_\Lambda(\mathbf{x}_1, \mathbf{v}_1, \mathbf{x}_2, \mathbf{v}_2, \mathbf{a}_1, \mathbf{a}_2)$

$\xi_j = \beta(\lambda_k, \theta_m)$ %see eq. 4.13

if $(\xi_j \neq \xi_4)$

$\xi'_j = b(\xi_j)$

for l = [1,2,3,4]

if $\phi_{moore}(\xi_j, \lambda_l) == \xi'_j$

$RA \leftarrow \lambda_l$

exit for

end if

end for

end if

end{program}.

5.6 Reasoning with Multiple Step Prediction

In the case the aircraft pair is in conflict and the next available RA can only maneuver the pair into a less dangerous state, but not into the final SAFE state, multiple steps of maneuver are needed. At each scan, the positions and velocities of the aircraft pair are updated, the qualitative abstract state is updated, the desired next state is updated, and thus the RA (maneuver)

is updated.

5.6.1 Pseudo code for Reasoning with Multiple Step Prediction

% Reasoning algorithm for multiple RA's MRA

begin{program}

get initial $(\mathbf{x}_1(0), \mathbf{v}_1(0), \mathbf{x}_2(0), \mathbf{v}_2(0), \mathbf{a}_1(0), \mathbf{a}_2(0))$ at $T0$

initiate j=-1

for $i = 0, 1, 2, ...9$, (t=T0+i*4.7)

$\omega_n = P_\Omega(\mathbf{x}_1(i), \mathbf{v}_1(i), \mathbf{x}_2(i), \mathbf{v}_2(i))$

$\theta_m = P_\Theta(\mathbf{x}_1(i), \mathbf{v}_1(i), \mathbf{x}_2(i), \mathbf{v}_2(i))$

$\lambda_k = P_\Lambda(\mathbf{x}_1(i), \mathbf{v}_1(i), \mathbf{x}_2(i), \mathbf{v}_2(i), \mathbf{a}_1(i), \mathbf{a}_2(i))$

$\xi_j = \beta(\lambda_k, \theta_m)$ %see eq. 4.13

if $(\xi_j \neq \xi_4)$

$\xi'_j = b(\xi_j)$ (use the preference function)

for l = [1,2,3,4]

if $\phi_{moore}(\xi_j, \lambda_l) == \xi'_j$

$MRA(i) \leftarrow \lambda_l$

exit for (l)

end if

end for

Given λ_l, compute $(\mathbf{x}_1(i+1), \mathbf{v}_1(i+1), \mathbf{x}_2(i+1), \mathbf{v}_2(i+1),$

$\mathbf{a}_1(i+1), \mathbf{a}_2(i+1))$ at time index $i+1$ using Kinetic equations

else (already safe)

exit for (i)

end if

end (for)

print out all computed $MRA(i)$ for $i = 1, 2, ...$

optimize the MRA solution

end{program}.

Chapter 6

Comparison of Methods

We have already shown two approaches to obtain resolution advisory in a conflict alert situation. The traditional quantitative approach is described in Chapter 3, and the qualitative approach is formulated in Chapter 4 and Chapter 5. There is a third approach that attempts to take the benefits of both. The third approach will be described in Section 6.1.3. To compare these three methods, the costs and the benefits need to be evaluated.

6.1 Computation Cost

The complexity of the method has a direct effect on the computation time, and thus influences whether a method can be applied in real-time. In a conflict alert situation, it is vital to provide resolution advisory as soon as possible, so that the pilot has time to react and maneuver.

6.1.1 Qualitative Method

The qualitative method described in Chapters 4 and 5 is expected to be less computation expensive than the traditional quantitative method in ATC. For an aircraft pair, given its current positions and velocities, the time it

takes to give a maneuver advisory using the qualitative method consists of the following components:

1. Time to decide its current conflict status (ξ_1, ξ_2, ξ_3, or ξ_4). According to Matlab profiler, the time for this is t_1=0.047 second.

2. Time to decide its next conflict status as where it wants to be (ξ_1, ξ_2, ξ_3, or ξ_4). This time is very small as compared to the previous item and thus can be ignored.

3. Time to search the automaton for a corresponding input ($\lambda_1, \lambda_2, \lambda_3$ or λ_4). This time is also very small and thus can be ignored.

4. Time to convert the input space partition λ_i into a concrete quantitative input (in 13D) and save it. According to Matlab profiler, the time for this is t_4=0.062 second. However, this needs to be repeated for every quantitative input in this partition.

5. Time to select the best a_{x1}, a_{y1} and a_{x2}, a_{y2} from the results in the previous step, using the cost function calculation. According to Matlab profiler, the time for this is t_5=0.204 second.

Thus the total computation cost (assuming automaton is built already) is $t = t_1 + t_4 \cdot N_a + t_5$, where N_a is the number of quantitative input sets (it depends on the resolution of the input variables).

Using $t_4 = 0.062$ and $N_a = 7^5$, we get $t = 1.04 \cdot 10^3$ seconds.

6.1.2 ATC Quantitative Method

The traditional quantitative method (Chapter 3) is a point to point search through all possible accelerations for each scan. The following computation cost is incurred:

1. Time to decide if the state is currently under conflict. According to Matlab profiler, the time for this is tt_1=0.063 second.

2. Time to compute the next state under given (a_{x1}, a_{y1}), to see if it is within constraints. According to Matlab profiler, the time for this is tt_2=0.031 second. This needs to be repeated for each (a_{x1}, a_{y1}) point in the entire search space.

3. Time to save this set of (a_{x1}, a_{y1}) if constraints are satisfied. Same as before, the time for this is tt_3=0.016 second. This also needs to be repeated for all valid (a_{x1}, a_{y1}) accelerations.

4. Time to compute the cost function. According to Matlab profiler, the time for this is tt_4=0.204 second.

Thus the total computation cost is $tt = tt_1 + tt_2 \cdot N_e + tt_3 \cdot N_s + tt_4$, where N_e is the number of quantitative inputs in the entire search space (it depends on the resolution of the input variables); N_s is the number of the quantitative inputs that satisfy the constraints (it can be less than N_e).

Using $tt2 = 0.031$, $N_e = 27^5$, $tt3 = 0.016$ and $N_s = 27^4 \times 13$ (so that it is less than N_e), we get $t = 5.55 \cdot 10^5$ seconds.

6.1.3 Middle Ground Method

The qualitative method divides the output space into four partitions. The question is whether it is optimal for computation. After qualitative reasoning, the fine search within each region is computationally costly, and its complexity depends on the size of that particular region. The smaller the region, the less costly this search will be. If the output space is divided into more partitions, then each region will be (on average) smaller. However, at

the other extreme, more partitions in the output space will increase the complexity of the symbolic reasoning, and will increase the computation cost on the qualitative side. Time for performing step 1 in the qualitative method (see Section 6.1.1) will increase, since the number of constraints to check for the partitioning will increase. Time in items 2 and 3 in the qualitative method can not be ignored. Yet time in item 5 may be smaller. Thus there is a tradeoff between the qualitative reasoning and the quantitative search.

In order to know if the computation time can be minimized by using another n (the number of partitions of the output space), it is necessary to estimate the time needed for symbolic reasoning when the number of symbols is large. It is also necessary to note that even though each region is on average smaller, the number of regions that can transit to a given state may be more than one, i.e., there are more than one paths going into the state that is most desired. In fact, the number of desired states can be more than one if the partitions of the state space increase. If this is the case, the reasoning becomes very complicated and then there is no good reason for dividing the output space further into more (smaller) partitions.

In our analysis of this problem, we assume that the time in item 1 is linearly proportional to the number of qualitative outputs, m, and that the linearity constant is K. We also assume that time in items 2 and 3 can still be ignored if $m < 100$. Moreover, we assume that time in item 4 depends linearly (coefficient equal to J) on the number of quantitative input choices, N_a. And at last, we assume that time in item 5 is linearly proportional (constant H) to the number of quantitative input choices, N_a. This can be summarized as follows.

$$t_1 = K \cdot m,$$

$$t_2 = 0,\ m < 100$$

$$t_3 = 0,\ m < 100$$

$$t_4 = J \cdot N_a,$$

$$t_5 = H \cdot N_a.$$

We also use the estimated values of time (Sections 6.1.1 and 6.1.2) for $m = 4, N_a = (27/m)^5$ to calibrate the parameters K, J and H. Using $t_1 = 0.047,\ t_4 = 0.062 \cdot N_a,\ t_5 = 0.204$, we get $K = 0.047/4,\ J = 0.062,\ H = 0.204/(27/4)^5$. Thus

$$t = 0.047/4 \cdot m + 0.062 \cdot (27/m)^5 + \frac{0.204}{((27/4)^5)} \cdot (27/m)^5.$$

Solving the minimization of t over m, we obtain $m = 26.897$. Rounding it up to the nearest interval, we obtain $m = 27$. The conclusion then is that in order to optimize the computation time of the qualitative method for this particular problem, the output space would have to be partitioned into 27 qualitative regions.

6.1.4 Time Distribution of the Methods

As another method to evaluate the performance of different approaches, we introduce the concept of "average resolution time". Resolution time is defined as the time from the time of declaring a conflict alert to the time of the output of a valid RA (resolution advisory). We expected there would be an improvement in the resolution time of the qualitative method over the quantitative method.

The magnitude of this difference will depend on various aspects of the scenario and of the speed of the computer used. If a very small number of look-ahead step is used for prediction and if the computer is very fast,

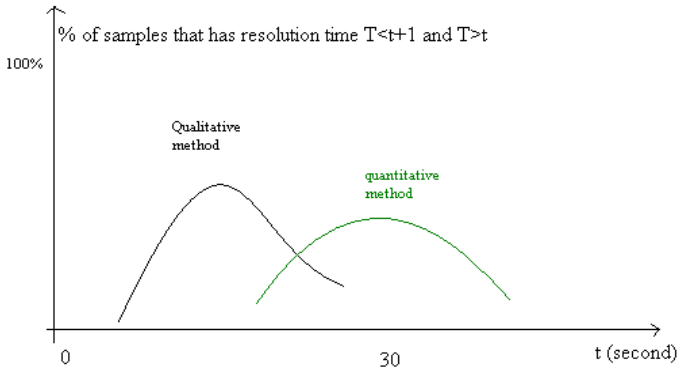

Figure 6.1: Expected resolution time distribution curve

the difference might even be unnoticeable. But if a five-step-ahead RA is computed, while the intruder's maneuver intention is randomly generated, the resolution time is expected to have a more significant improvement using the qualitative method. The computation time also depends on the conflict scenario, the geometry of the conflict, the approaching speed of the pair, the "hostility" of the intruder and others. In some situations five steps of RA to solve the conflict are not necessary. The ideal RA is to plan as many steps as necessary until a conflict free situation is achieved and the flight returns to the planned trajectory. The time from the issuing of the conflict alert to the end of the last step of RA generation can be accumulated from a large sample of randomly generated scenarios. The resolution time distribution curve was expected to be like the one in Figure 6.1. In this figure, most conflicts are solved in a shorter time by the qualitative method (black curve) than by the quantitative method (green curve). This is the expected advantage that had to be demonstrated in this research. Since resolution is scenario dependent,

these curve needs to be generated and only compared under the same type of scenarios.

Due to a very long computation time needed to simulate a quantitative solution, we have limited our experiments $L = 3$ look-ahead prediction steps. The resolution time has been collected and the distribution for 20 scenarios. The resulting distribution of resolution time is shown in Figure 6.2. This curve is essentially similar in nature to the expected curve of Figure 6.1. It is important to note that the difference between the two curves will be much larger if we use $L = 5$ or more (refer to table 7.5 and table 7.6 in section 7.3 for details). Also, this curve will be much smoother if we run 100 to 200 scenarios.

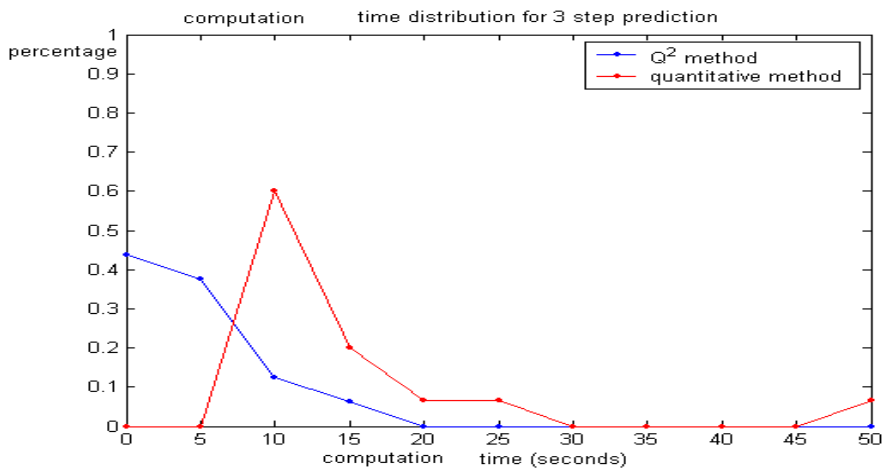

Figure 6.2: Comparison of resolution time distribution curves for Q^2 method and quantitative method

6.2 Quality of Maneuvers

While computation time is critical to the generation of RA's, the quality of maneuvers (quality of RA's) is also very important. Our goal was to develop a method for generating RA's that could be computed in real time. But the maneuvers would still need to be close in the quality to the maneuvers generated by the quantitative method. To evaluate the quality of maneuvers, we use the cost function g (Equation 3.10) defined in Chapter 3. The maneuver is better when it is less destructive to the flight plan, i.e., when the flight path due to the maneuver is closer to the original flight plan than those caused by other maneuvers. The maneuver is also better when the delay caused by the maneuver is shorter. Therefore, the smaller the cost function, the better the performance, provided the constraints being all satisfied.

The results of the evaluating experiments are shown in the next chap-

ter. Here we give an example of the comparison of costs for the qualitative method vs. the quantitative method. In that example (see scenario 4 in the next chapter), the value of the cost function g for the optimal maneuver path derived by the Q^2 method using $L = 5$ prediction steps was equal to $g = 2.206$. The quantitative method, on the other hand, resulted in $g = 0.711$. The qualitative method paid a small price for its faster computation in terms of a slightly higher g. This means the path computed by the qualitative method is somewhat worse than the one computed by the quantitative method.

To assess the significance of the difference in the costs we need to understand the meaning of the quantitative value of the cost function. The cost function is expressed in the unit of nautical miles. However, it is spread over a number of integration intervals. In this case, the integration (summation) was performed over five steps (look-ahead predictions). So the average deviation from the optimal path in each step is about 0.2 to 0.4 nautical miles. On the other hand, the best possible maneuvers derived by the quantitative method) are on average 0.15 nautical miles away from the flight plan. Thus the qualitative method gave advisories that are from 0.05 to 0.25 nautical miles away from the quantitative method. Note that the radar measurement noise is 0.05 nautical miles, i.e., 0.05 nautical miles of position difference (or less) is not detectable by the radar. In conclusion, although the difference between the best RA's given by the qualitative method are not below the noise level, they are also not that significant, either. This difference is well compensated by the very large improvement in the computation time.

Chapter 7

Results

In this chapter we describe the scenarios used for the evaluation the proposed approach and the results of evaluation.

7.1 Scenarios

Four scenarios have been developed and simulated. These four scenarios have been devised in such a way as to reflect different types of common encounter scenarios.

1. **Scenario 1:** Aircraft A (self) in linear flight, constant velocity; aircraft B (intruder) also in linear flight, constant velocity. The approach angle, approach time, simulation time are given in Table 7.1. The plot of this particular scenario is shown in Figure 7.1.

2. **Scenario 2:** Aircraft A (self) in turning flight with constant turn rate; aircraft B (intruder) in linear flight, constant velocity. The approach angle, approach time, simulation time, the value of the turn rate, start and end time of the turn and left turn or right turn indicator are given in Table 7.2. The plot of this particular scenario is shown in Figure 7.2.

3. **Scenario 3:** Aircraft A (self)in turning flight with constant turn rate; aircraft B (intruder) in linear flight, accelerating. The approach angle, approach time, simulation time, the value of the turn rate of A, start and end time of the turn, left turn or right turn indicator and the acceleration of B are given in Table 7.3. The plot of this scenario is shown in Figure 7.3.

4. **Scenario 4:** Aircraft A (self) in linear flight, constant velocity; aircraft B (intruder) also in linear flight, but accelerating. The approach angle, approach time, simulation time, and acceleration of B are given in Table 7.4. The plot of this scenario is shown in Figure 7.4.

scenario 1	value
approach time	15 scan
simulation time	30 scans
angle of approach	45 degrees
acceleration for A	0
acceleration for B	0

Table 7.1: Scenario 1 parameters.

scenario 2	value
approach time	15 scan
simulation time	30 scans
angle of approach	90 degrees
turn rate for A	1.0 deg per second
turn start time for A	5 scan
turn end time for A	16 scan
right turn indicator A	1, right turn
acceleration for B	0

Table 7.2: Scenario 2 parameters.

scenario 3	value
approach time	15 scan
simulation time	30 scans
angle of approach	90 degrees
turn rate for A	1.0 deg per second
turn start time for A	2 scan
turn end time for A	16 scan
right turn indicator A	1, right turn
acceleration for B	0.05G

Table 7.3: Scenario 3 parameters.

scenario 4	value
approach time	15 scan
simulation time	30 scans
angle of approach	80 degrees
acceleration for A	0
acceleration for B	0.1G

Table 7.4: Scenario 4 parameters.

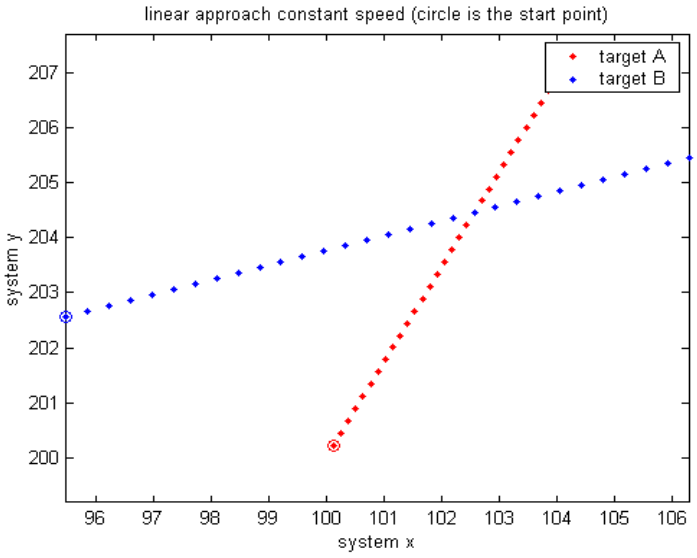

Figure 7.1: Scenario 1

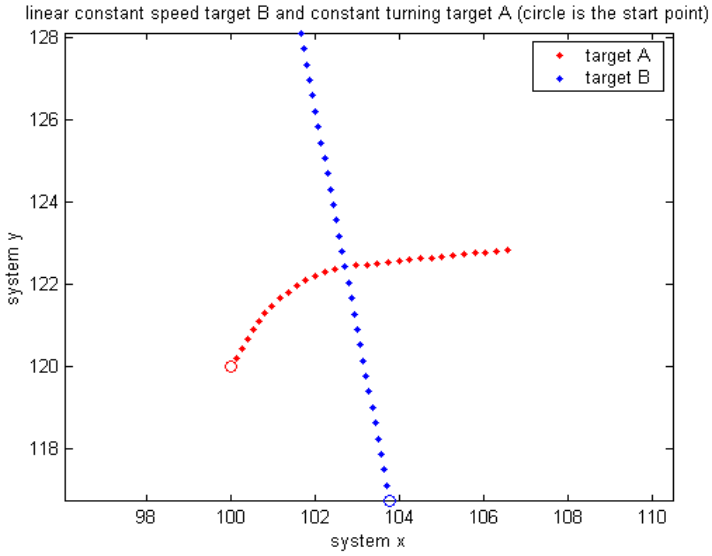

Figure 7.2: Scenario 2

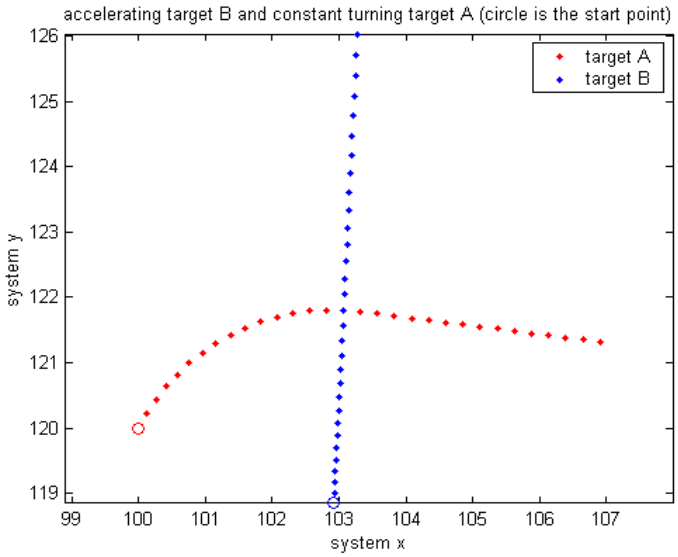

Figure 7.3: Scenario 3

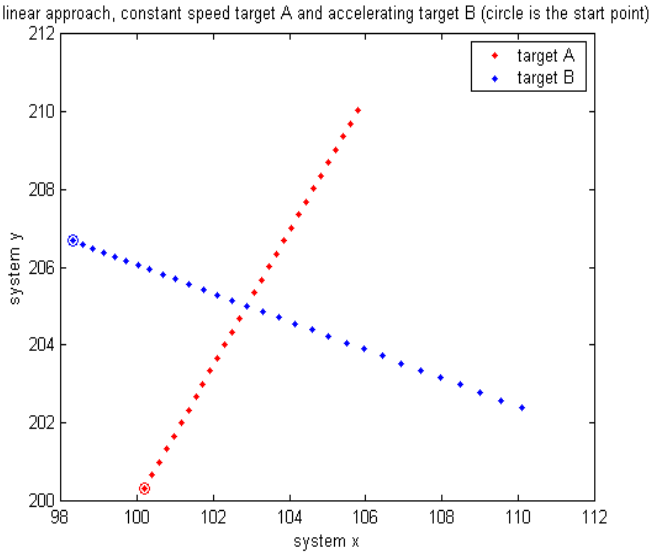

Figure 7.4: Scenario 4

7.2 Computed Maneuvers

These scenarios were simulated, then data were saved and fed into our Q^2 reasoning algorithm. The resulting maneuvers are described below. Maneuvers in two scenarios are shown in Figure 7.5 and 7.6 just to give a direct view of the solutions. These figures illustrate the correctness of the algorithm.

Below we list the best maneuvers generated by the qualitative method for each of the four scenarios. For each maneuver step, the number of possible choices is also provided.

Best maneuver for Scenario 1:

1. $m1$: Turn right at $\omega = 2^o$ per sec, 5 choices of maneuver
2. $m2$: Turn right at $\omega = 2^o$ per sec, 42 choices of maneuver
3. $m3$: Turn right at $\omega = 2^o$ per sec, 354 choices of maneuver

4. $m4$: turn right at $\omega = 2^o$ per sec, 3010 choices of maneuver

5. $m5$: Turn right at $\omega = 2^o$ per sec, 25973 choices of maneuver (535.45 seconds to reach this step)

Best maneuver for Scenario 2:

1. $m1$: Turn left at $\omega = -5^o$ per sec, 3 choices of maneuver

2. $m2$: Accelerate at $a = 135 knots/min$, 34 choices of maneuver

3. $m3$: Accelerate at $a = 135 knots/min$, 411 choices of maneuvers

4. $m4$: Accelerate at $a = 90 knots/min$, 5280 choices of maneuver

5. $m5$: Accelerate at $a = 90 knots/min$, 72176 choices of maneuver (2619.3 seconds to reach this step)

Best maneuver for Scenario 3:

1. $m1$: Turn left at $\omega = -4^o$ per sec, 1 choices of maneuver

2. $m2$: No maneuver, coasting at constant speed, 6 choices of maneuver

3. $m3$: Decelerate at $a = -45 knots/min$, 75 choices of maneuver

4. $m4$: No maneuver, coasting at constant speed, 1154 choices of maneuver

5. $m5$: Accelerate at $a = 90 knots/min$, 18242 choices of maneuver (198.516 seconds to reach this step)

Best maneuver for Scenario 4:

1. $m1$: turn right at $\omega = 6^o$ per sec, 3 choices of maneuver

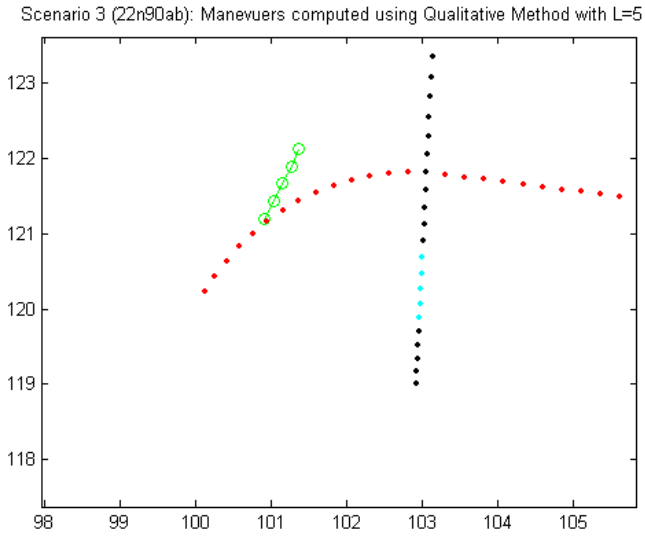

Figure 7.5: Scenario 3 conflict alert resolution (in green)

2. $m2$: turn right at $\omega = 2^o$ per sec, 33 choices of maneuver
3. $m3$: turn right at $\omega = 2^o$ per sec, 186 choices of maneuver
4. $m4$: turn right at $\omega = 2^o$ per sec, 824 choices of maneuver
5. $m5$: turn right at $\omega = 1^o$ per sec, 7123 choices of maneuver (65.344 seconds to reach this step)
6. $m6$: turn right at $\omega = 1^o$ per sec, 112938 choices of maneuver (6036.8 seconds to reach this step)

7.3 Computation Time

During each run in Matlab [35], the computation time in seconds was recorded. The computer used for this task was a Dell Pentium R 4 CPU 3.20GHz, 1.00GB of RAM. The PC version of the MATLAB software was version

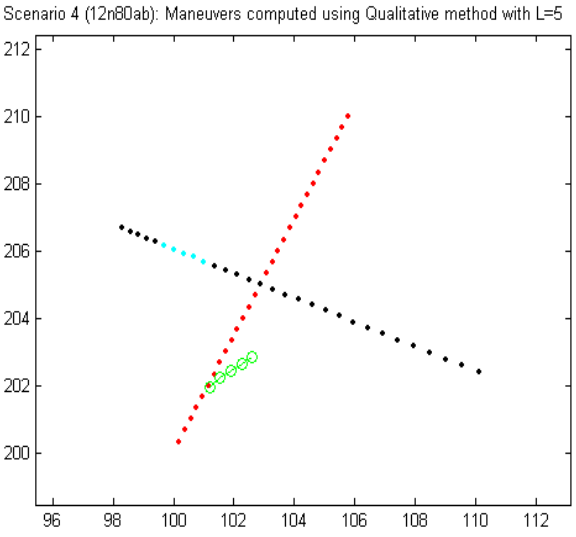

Figure 7.6: Scenario 4 conflict alert resolution (in green)

6.5.1. The times that are needed to generate the maneuvers are included in table 7.5. From the table it can be seen that the computation is scenario dependent – some scenarios cost more float point operations than others. Overall, the average cost for a 5-step computation is about 500 seconds.

To put the computation times of the qualitative algorithm in perspective, we implemented the quantitative method so that the two approaches could be compared. The quantitative method is the straight forward search in the maneuver space. Basically, all possible maneuvers are considered and the cost function is computed for every possibility. In the end, the maneuver path with the lowest cost is selected. Figures 7.7 to 7.8 show the extensive selections of maneuvers for each encounter scenario. The computation cost of the quantitative method is collected and given in table 7.6. In theory, the number of maneuver choices for a 5-step computation using traditional quantitative method is about 14 millions – hundreds of times higher than

scen	Step 1	Step 2	Step 3	Step 4	Step 5	Q2 method total	ave -rage
1	0.110	0.422	2.390	24.6	507.9	535.4	
2	0.078	0.156	2.016	31.4	2585.4	2619.3	
3	0.047	0.094	0.437	6.4	191.5	198.5	500
4	0.095	0.233	1.094	6.5	57.4	65.3	

Table 7.5: Time in seconds that is needed to generate each maneuver upto 5 steps using Q^2 method

	Step 1	Step 2	Step 3	Step 4	Step 5	total
Scenario 3	0.141	1.921	62.360	13923.1	329627.3	343614.8
Scenario 4	0.266	2.172	71.906	11219.4	44488.8	55782.5

Table 7.6: Time in seconds that is needed to generate L steps of maneuvers using the quantitative method for Scenario 3 and Scenario 4

that of the qualitative method.

The maneuvers for Scenarios 3 and 4 have been computed using the quantitative method. Results for Scenario 4 are shown in Figure 7.9. It took about fifteen and a half hours to compute the 5-step prediction for Scenario 4 and more than 95 hours to compute 5-step prediction for Scenario 3. Therefore we did not perform the computations for the other two scenarios. According to our estimates, it would take 15 to 150 hours to compute maneuvers for the 5-step prediction using the quantitative method. It would be 100 to 1000 times longer than the time used by the qualitative method at $L = 5$. The saving would be even more significant at $L = 7$. However, such a comparison is not provided since it is impossible to carry the simulation of the quantitative method at $L = 7$ using a PC.

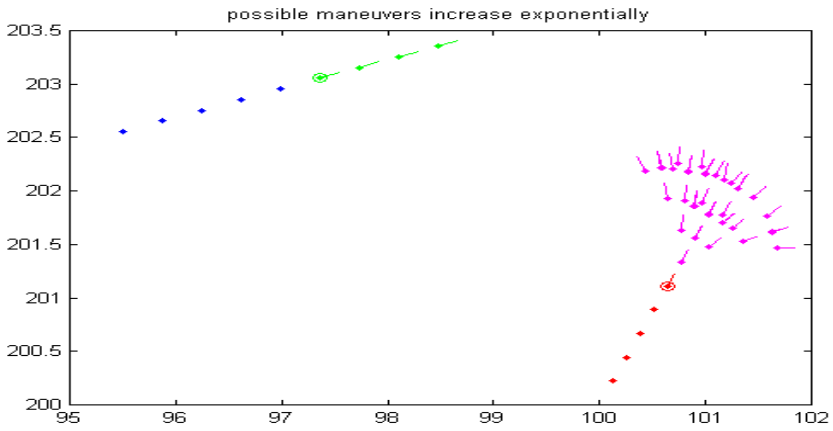

Figure 7.7: Scenario 1: possible maneuvers (in purple) before constraints are applied.

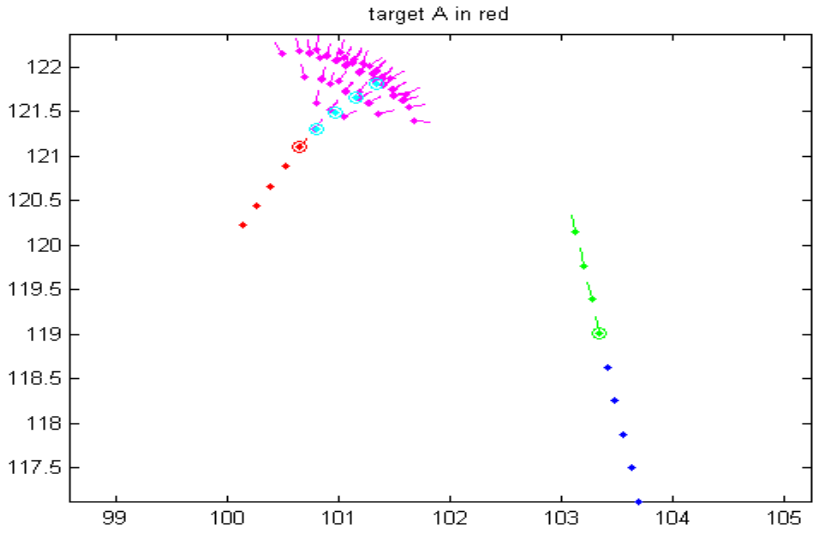

Figure 7.8: Scenario 2: possible maneuvers (in purple) before constraints are applied.

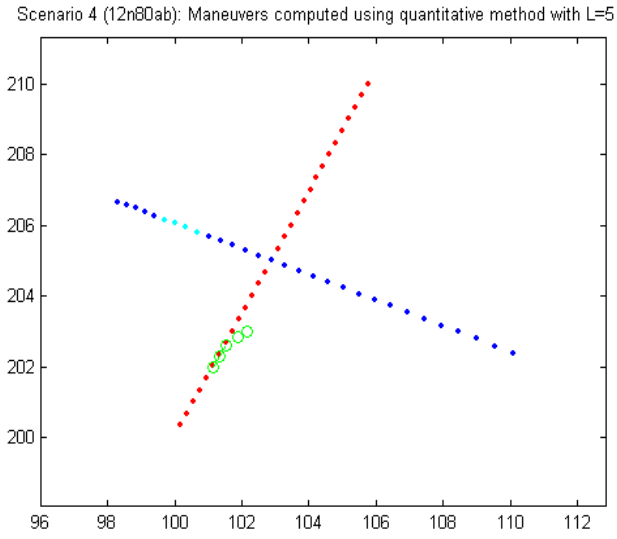

Figure 7.9: Scenario 4: conflict alert resolution (in green) using quantitative method

7.4 Evaluation of Maneuver Quality

As discussed earlier, we used the cost function, g, to evaluate the quality of the maneuvers. The values of the cost function, g, for the chosen optimal maneuver paths using both methods are included in Table 7.7. We can see that the qualitative method computes faster, but it pays a small price for this, i.e., it gives a slightly higher g.

Scenario	Total Q2 method	Quantitative method
1	1.089	
2	2.132	
3	1.707	
4	2.206	0.711

Table 7.7: Minimum cost function g for both methods

Chapter 8

Generalization of the Q^2 Approach

Even though only an example problem from the air traffic control field is used in this research, the method and the algorithms developed in this research may be applicable to a wider range of problems ([75] etc.) dealing with dynamical systems. We start this chapter by looking at the different types of systems. Then we provide algorithms that are applicable to some cases of linear dynamic systems.

8.1 Types of General Dynamic Systems

8.1.1 Static and Dynamic Systems

There are two basic types of system: static and dynamic. In a static system, the current outputs are based solely on the instantaneous values of the current inputs. An example of a static system is a resistor hooked up to a current source:

$$V(t) = R \cdot i(t). \tag{8.1}$$

At any given moment, the voltage across the resistor (the output) depends only on the value of the current running through it (the input). The current

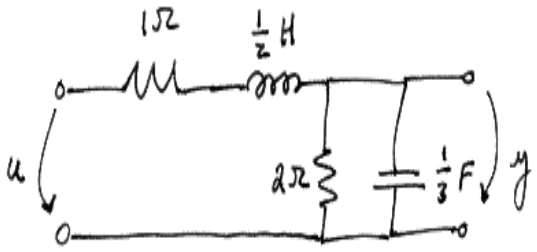

Figure 8.1: RLC circuit: 2nd order

at any time t is simply multiplied by the constant value describing the resistance R to give the voltage V.

When a system's output depends on both the present and the past input, it is said to be a dynamical system. Take the RLC circuit shown in Figure 8.1 as an example. The current can be described by:

$$\frac{dy^2(t)}{d^2t} + \frac{7}{2}\frac{dy(t)}{dt} + 9y(t) = 6u(t). \tag{8.2}$$

8.1.2 Linear and Non-linear Systems

Dynamical systems are not limited to electrical circuits. Any system whose output depends on current and past inputs is a valid dynamical system. The mathematical models used to describe the swinging of a clock pendulum, the flow of water in a pipe, or the number of fish each spring in a lake are examples of dynamical systems.

Unfortunately, nonlinear dynamics are not fully understood and the best we can do is simulate the real world with [16] linear or low-order approximations. To be more precise, linear behavior is simulated locally, at a point or along a small interval in space-time, and then the results are extrapolated to the general domain. This extrapolation relies on knowledge about the

direction of linearity (say, directional derivatives at a point on a surface) and knowledge of the nonlinear behavior. For example, a clock pendulum problem (shown by Eq.8.3) is non-linear.

$$\frac{d^2\theta}{dt^2} + \frac{mgl}{I}\sin\theta = 0 \tag{8.3}$$

But this differential equation can be approximated by an equation obtained though the Taylor series expansion. Near the equilibrium point, the higher orders of the sin function can be ignored. Thus this system can be viewed as (approximated by) a linear system.

$$\frac{d^2\theta}{dt^2} + \frac{mgl}{I}\cdot\theta = 0 \tag{8.4}$$

8.1.3 Example for Testing

The algorithms developed for the ATC problem have been generalized to some cases of linear systems. An example of such linear dynamical system is shown in Figure 8.1.

$$\begin{bmatrix} \dot{x} \\ \dot{y} \end{bmatrix} = \begin{bmatrix} -7/2 & -9 \\ 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} 6 \\ 0 \end{bmatrix} \cdot u(t) \tag{8.5}$$

We take the output function to be $W = x - 3.0$. We define the critical point at $W = 0$, i.e., we define the qualitative output ω_1 if $W < 0$, the qualitative output ω_2 if $W = 0$ and the qualitative output ω_3 if $W > 0$.

8.2 Generalization of Partitioning and Reasoning

First we specify the generalization of partitioning and then the generalization of reasoning in the next two sections. We provide the pseudo code for the algorithms below. Then we select two example problems:

1. a simple circuit example

2. a simplified ATC problem

and use them to test our generalized approach.

Some symbols and notations used in this chapter are listed below:

$\mathbf{Q}$ – state

$\mathbf{W}$ – output

$\mathbf{U}$ – input

$\mathbf{U} \times \mathbf{T} \times \mathbf{Q0}$ – input space

Θ – qualitative state space (θ_i)

Ω – qualitative output space (ω_i)

Λ – qualitative input space

Ξ – qualitative states in the final Moore machine

P_i – general partition algorithm of input

P_h – general partition algorithm of state

P_g – general partition algorithm of output

$\bar{A}$ – constraints

$\hat{\mathbf{A}}$, $\hat{\mathbf{B}}$ – state transition matrices

8.2.1 Pseudo Code for Partitioning the Output Space of Linear Dynamic Systems

If f in Eq. (4.1) and g are linear, or if they can be linearized, then Eq. (4.1) can be represented in linear matrix form like Eq. (8.6). The inverse mappings of f and g can then be obtained using the inverse of the matrices $\hat{\mathbf{A}}$ and $\hat{\mathbf{B}}$ (provided their determinants are not zero). The pseudo code for the partitioning of the output space for a general case is given below.

$$\mathbf{Q}(k) = \hat{\mathbf{A}} \cdot \mathbf{Q}(k-1) + \hat{\mathbf{B}} \cdot \mathbf{U}(k-1). \tag{8.6}$$

Generalized algorithms for output space partitioning:
% Partitioning algorithm P_g

Notations:

L	: dimension of the output space
$W(1)$, $W(2)$,..., $W(L)$	: variable of the output space
$W(1) \times W(2) \times ... \times W(L)$	: output space
k	: number of functions
m	: number of thresholds
l	: number of constraints
n	: number of qualitative outputs
Ω	: qualitative outputs $\{\omega_1, \omega_2, ...\omega_n\}$
z_i	: functions $z_i = f_i(W(1), W(2), ..., W(L))$ where $i = 1, 2, ...k$
d_j	: thresholds where $j = 1, 2, ...m$
RO	: relation operators $RO \in \{<, <=, =, >, \geq, \}$
$Wmin(i)$	: lower bound of the output variable W(i) where $i = 1, 2, ...L$
$Wmax(i)$	: upper bound of the output variable W(i) where $i = 1, 2, ...L$
$Wstep(i)$	: resolution of the output variable W(i) where $i = 1, 2, ...L$

Constraints:

$C = \{c_1, c_2, ..., c_l\}$ set of constraints, where $c_s = (z_i \ RO \ d_j)$
$\bar{A} = \{\bar{a}_1, \bar{a}_2, ..., \bar{a}_{n-1}\}$ set of ordered constraints,
where $\bar{A} =_{DNF}^{CNF} (C)$ and $\bar{a}_1 \prec \bar{a}_2 \prec ... \prec \bar{a}_{n-1}$
DNF - Disjunctive Normal Form
CNF - Conjunctive Normal Form

Input:

$L, n, \bar{A}, Wmin(.), Wmax(.), Wstep(.)$

Output:

$P_g : W(1) \times W(2) \times ... \times W(L) \rightarrow \Omega$

```
begin{program}
  for all W(1 to L)=Wmin(1 to L):Wstep(1 to L):Wmax(1 to L)
    for i=1...n-1
        if(ā_i)
            P_g(.) ← w_i
            break
        end(ā_i)
    end {for }
    P_g(.) ← w_n
  end all W
end{program}.
```

Note that the constraints in $\bar{A}$ depend on the specific problem. For the ATC problem, they depend on how "PROCON","LINCON"and "MANCON" are defined. For a circuit problem, they may depend on functions of the current and/or the voltage being greater or smaller than some thresholds. For a heat conducting problem, they may depend on functions of the temperature difference being greater or smaller than thresholds. For each specific problem, these constraints should be known.

8.2.2 Pseudo Code for Partitioning the State Space of Linear Dynamic System

If the problem at hand is such that the constaints $\bar{A}$ depend only on the output variables, then partitions of the state space can be obtained by inverse-

mapping of the output space partitions. For the type of problems where this type of simple inverse mappings exist, the pseudo-code is provided below.

Generalized algorithms for state space partitioning:
% Partitioning algorithm P_h

Notations:

Ls	: dimension of the state space
$Q(1), Q(2), ...Q(Ls)$	: variables of the state space
$Q(1) \times Q(2) \times ... \times Q(Ls)$	: state space
$\triangle Q, \triangle S$	: variables of the output space
Θ	: $\{\theta_1, \theta_2, ...\theta_n\}$
P_g	: $W(1) \times W(2) \times ... \times W(L) \rightarrow \Omega$
g	: output function, $(W(1), W(2), ...W(L))$ $=g(Q(1), Q(2), ...Q(Ls))$
$Qmin(i)$	: lower bound of the state variable Q(i) where $i = 1, 2, ...Ls$
$Qmax(i)$	: upper bound of the state variable Q(i) where $i = 1, 2, ...Ls$
$Qstep(i)$	: resolution of the state variable Q(i) where $i = 1, 2, ...Ls$

Input:

$Ls, g, P_g, Qmin(.), Qmax(.), Qstep(.)$

Output:

$P_h : Q(1) \times Q(2) \times ... \times Q(Ls) \rightarrow \Theta$

```
begin{program}
 for all Q(1 to Ls)=Qmin(1 to Ls):Qstep(1 to Ls):Qmax(1 to Ls),
       ω ← P_g(g(.))
             for i=[1,2...n]
                if(ω == ω_i)
                 P_h(.) ← θ_i
                 exit forloop(i)
                end if
             end for (i)
 end {for Q}
end{program}.
```

For problems where the constraints depend on the state variables, the state

space can be partitioned directly like the partitioning of the output space in page 117. The pseudo-code is not included in order to avoid redundancy. For some problems, the constraints depend on the input variables, then the input space can be partitioned directly (although the Q^2 approach seem unnecessary now). There may be cases where the constraints depend on both the input and the output variables (such as our ATC problem), or constraints depend on the input, state and the output. In that case the algorithms are more complicated so more steps would be required to partition the state space. We provide some discussion of these issues on page 121.

8.2.3 Pseudo Code for Partitioning the Input Space of Linear Dynamical Systems

If partitions of the state space are already obtained using the method discussed in the previous subsection, then the partitions of the input space can be obtained by the inverse-mapping of the state space partitions. For the problems where this type of simple inverse mappings exist, the pseudo-code is provided below.

Generalized algorithms for input space partitioning:
% Partitioning algorithm P_i

Notations:

Lo	: number of input variables (exclude initial state and time)
$U(1), U(2), ...U(Lo)$	: input variables
$U \times \mathbf{T} \times Q0$	: input space
Λ	: $\{\lambda_1, \lambda_2, ...\lambda_n\}$
P_h	: $Q(1) \times Q(2) \times ... \times Q(Ls) \rightarrow \Theta$
f	: state transition function, $(Q) = f(U, \mathbf{T}, Q0)$
$Umin(i)$	: lower bound of the input variable U(i) where $i = 1, 2, ...Lo$
$Umax(i)$	: upper bound of the input variable U(i) where $i = 1, 2, ...Lo$

$Ustep(i)$: resolution of the input variable U(i)
where $i = 1, 2, ...Lo$
$[t_l, t_h]$: the valid range (low and high bound) of T
δt : resolution of T

Input:

$Lo, f, P_h, Umin(.), Umax(.), Ustep(.),$
$Qmin(.), Qmax(.), Qstep(.), [t_l, t_h], \delta t$

Output:

$P_i : U \times \mathbf{T} \times Q0 \rightarrow \Lambda$

```
begin{program}
 for all U(1 to Lo)=Umin(1 to Lo):Ustep(1 to Lo):Umax(1 to Lo),
     Q0(1 to Ls)=Qmin(1 to Ls):Qstep(1 to Ls):Qmax(1 to Ls), t = t_l : δt : t_h,
      θ ← P_h(f(.))
            for i=[1,2...n]
               if(θ == θ_i)
                P_i(.) ← λ_i
                exit forloop(i)
               end if
            end for (i)
 end {for}
end{program}.
```

8.2.4 Partitioning Complicated Cases of Linear Dynamical Systems

As discussed earlier, the constraints may depend on the output variables, state variables or input variables, or on a combination of all of them. If we look at all the possibilities, the constrants $\bar{A}$ could depend on:

1. the output variables
2. the state variables
3. the input variables
4. the output and input variables

5. the output and state variables

6. the state and input variables

7. the output, state and input variables

For case 1 to 3, the spaces can be partitioned directly, as mentioned earlier. For case 4, it is exactly like our ATC example. The general approach for all linear dynamic problems is the same as the approach in our example (Chapter 4). Before we lay down the pseudo-code of a general approach for this case, we briefly discuss cases 5 and 7.

If the constraits depend on the output and state (case 5) variables, we know that the output is obtained from the state using the output function. If we expand the output function and include the involved state variables in an expanded output space, then the constraints depend only on the expanded output variables, therefore case 5 becomes case 1.

Similarly, for case 7, since the output is obtained from the state using the output function, if we expand the output function and include the involved state variables in an expanded output space, then the constraints depend on the expanded output variables and the input. Therefore, case 7 becomes case 4.

As a result of the above analysis, we need to address cases 4 and 6. For a general linear dynamic system where constrains depend on the output and input variables, the inverse mapping of partitions do not apply easily. Let us start with the case where constraints $\bar{A}$ depend on both output and input variables (case 4).

Let P_i be the partitioning of input space obtained from inverse mapping through output partitioning and state partitioning. Let $P_{g'}$ be partitioning

of the input space directly. Then the resultant partitioning is the superposition of the two. Our ATC problem is such a case. Note that $P_{g'}$ is the direct partitioning algorithm of the input space.

For case 6, where the constraints depend on the state and input variables, a similar approach can be followed by starting from the state space. Let Λ_1 be level 1 input partitions obtained from the inverse mapping of partitions of the state space, i.e., $\Lambda_1 = \{\lambda_1, \lambda_2, ...\lambda_n\}$. Let Λ^2 be level 2 partitions obtained by partitioning the input space directly , i.e., $\Lambda^2 = \{\lambda^1, \lambda^2, ...\lambda^m\}$. Then $\Lambda_j^k = \beta(\Lambda_j, \Lambda^k)$ can be obtained by superposing the two sub-partitionings. The pseudo-code is not included in order to avoid redundancy.

8.2.5 Generalization of Reasoning

For those cases (out of the seven cases discussed in the previous section) that the output variables are not involved in the constraints, there is no need to use Q^2 approach because the problems are simpler than the general format presented here. Therefore, only cases 1,4,5,7 correspond to problems where the output variables are in the constraints, and only these cases need to be discussed. Also, it was discussed earlier that case 5 becomes case 1 after some extension of the output space, case 7 becomes case 4 after some expansion. As a result, we only need to discuss case 1 and 4.

Fortunately, case 1 is the simpliest - we have shown the reasoning for the ATC problem (without MANCON) for this case. Case 4 is more complicated but we have also shown the reasoning for the ATC (with MANCON) problem for this case in Section 5.5. As a result, they all come down to reasoning within a Moore Machine framework. The pseudo-code for the reasoning is given below.

% Reasoning algorithm for general linear system – P_{rg}

Notation:

$Q(1), ...Q(Ls)$	: variables in the state space
T	: time interval between consecutive state transitions
Λ	: qualitative inputs, $\lambda_k \in \Lambda$
Ω	: qualitative outputs, $\omega_n \in \Omega$
Ξ	: qualitative states in the final Moore machine, $\xi_j \in \Xi$
P_Λ	: partitioning algorithm of input space
P_Ξ	: partitioning algorithm of state space
P_Ω	: partitioning algorithm of output space
ϕ	: $\Lambda \times \Xi \rightarrow \Xi$ - qualitative state transition function
ξ'_j	: next state
b	: preference function on states
ξ_m	: the most desired (goal) states
n	: total number of qualitative states

Input: $Q(1)...Q(Ls)$, - current state
P_Ξ, n, b, ξ_m, ϕ

Output: rg

begin{program}

$\xi_j = P_\Xi(Q(1), Q(2), ...Q(Ls))$

if $(\xi_j \neq \xi_m)$

$\xi'_j = b(\xi_j)$

for l = [1,2,...n]

if $\phi_{moore}(\lambda_l, \xi_j) == \xi'_j$

$rg \leftarrow \lambda_l$

exit for

end if

end for

end if

end{program}.

8.3 Testing of Generalized Approach

For testing purposes, we implemented the general linear system partitioning algorithm discussed above. As the input to this algorithm, the following information is given in a text file.

1. number of output variables,
2. number of state variables,
3. number of input variables,
4. number of ordered constraints,
5. number of functions used in constraints,
6. number of thresholds,
7. the ranges and resolution sizes of each variable

The text file is read in to load the necessary parameters and functions before the partitioning algorithms start. Also, the state transition function, f, and the output function, g, are given for the test problem. The partitioning

algorithms can work with any valid text input file and f and g functions for a general linear dynamic system.

8.3.1 The Circuit Problem

A very simple case (Q is the state, W is the output, and $A1$ is the input) is used. The linear dynamic system used as a test example, which was given in eq. 8.5, can be transformed to :

$$\begin{bmatrix} x(k+1) \\ y(k+1) \end{bmatrix} = \begin{bmatrix} -7/2 * T + 1 & -9 * T \\ T & 1 \end{bmatrix} \cdot \begin{bmatrix} x(k) \\ y(k) \end{bmatrix} + \begin{bmatrix} 6 * T \\ 0 \end{bmatrix} \cdot u(k) \tag{8.7}$$

Output function:

$$w = x - 3.0 \tag{8.8}$$

Ordered constraints:

$$\begin{aligned} &\bar{a}1 : w < 0; \\ &\bar{a}2 : w = 0; \end{aligned} \tag{8.9}$$

This example system was processed successfully by the generalized partitioning and reasoning algorithms. Projections of the partitions of the input space onto a 2-D plane and the partitions of the state space are shown in Figures 8.2 and 8.3.

The reasoning procedure takes the initial state ($x(0) = 0.0$; $y(0) = -13.0$). Then it computes the input partition which will make the state transition to another state. For this example, it is desired that $\omega = \omega_1$ transition to $\omega = \omega_3$. The input partition to achieve this task is λ_3. From

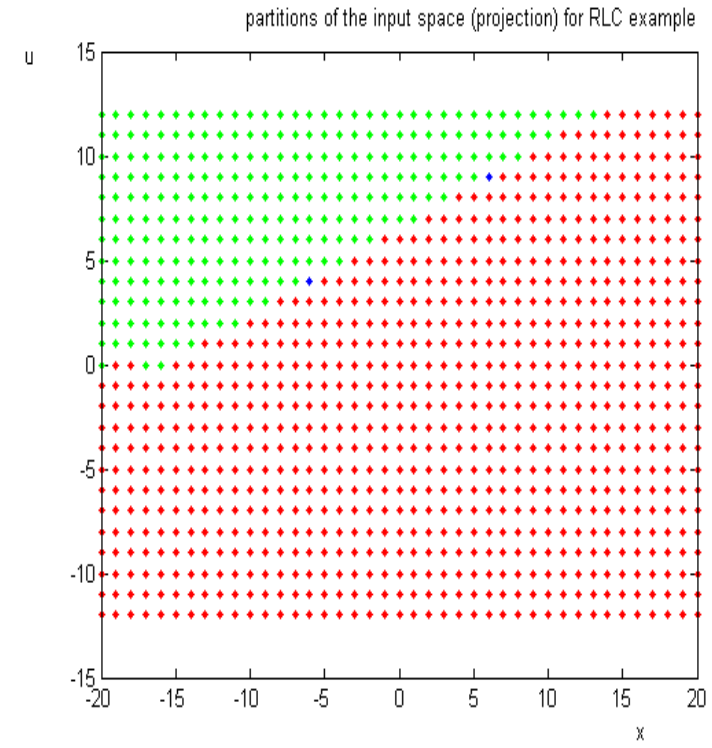

Figure 8.2: Projection of partitions of the input space onto a 2-D plane for the RLC circuit example

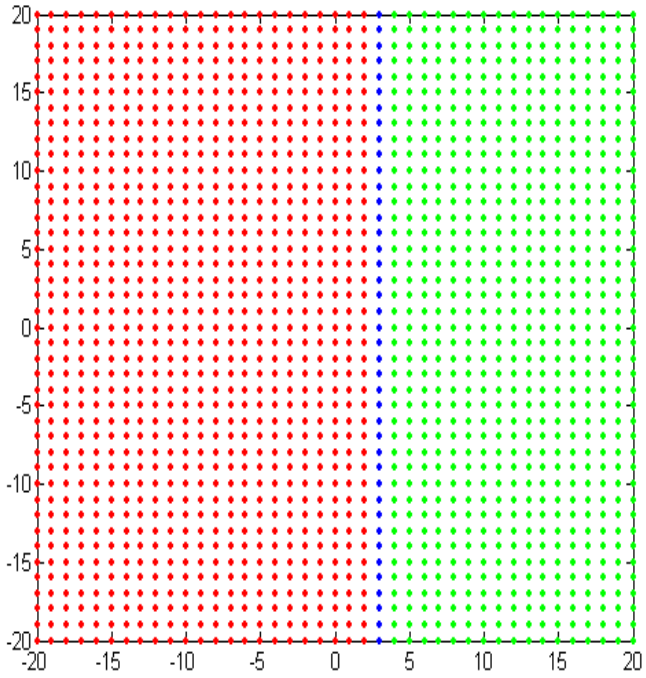

Figure 8.3: Partitions of the state space for the RLC circuit example

our reasoning algorithm implementation, the input u that belongs to this partition is $u = -12$. For this case the algorithm exits on the first solution. It does not attempt to obtain the optimal solution since our goal in this section is only to test the implementation of Q^2 reasoning of the general linear system.

8.3.2 The ATC problem

We have already solved the ATC problem. In this section we just show the result of a test of the implementation of general linear system against the ATC example. Compared to the circuit example, the dimensionality is much higher. The partitioning algorithms successfully generated partitions of the input space and partitions of the state space. Projections onto 2-D of those partitions are shown in figure 8.4 and figure 8.5.

The reasoning procedure takes the initial state ($Q(1) = 101$; $Q(2) = 121$; $Q(3) = 0.035$; $Q(4) = 0.059$; $Q(5) = 103.5$; $Q(6) = 119.8$; $Q(7) = -0.007$; $Q(8) = 0.07$). Then it computes the input partition which will cause the state transition to a safer state. For this example, $\omega = \omega_2$ is required to transition to $\omega = \omega_3$. The input partition to achieve this task is λ_3. From our reasoning algorithm, the input variables that belong to this partition is $A(1) = -0.007, A(2) = -0.006$. The algorithm exits on the first solution. It does not attempt to obtain the optimal solution since our purpose of this experiment is only to test the implementation of a general linear system.

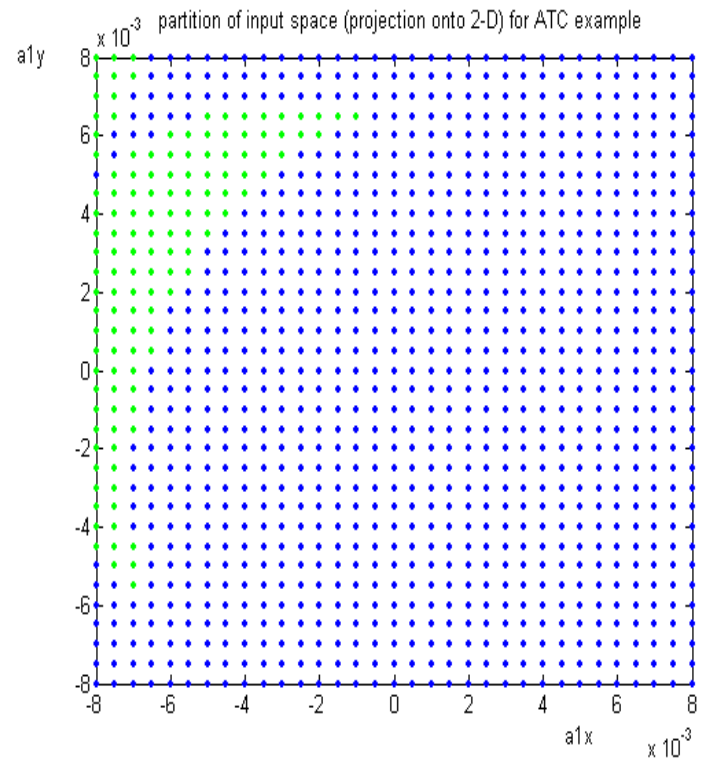

Figure 8.4: Projection of the partitions in the input space onto a 2-D plane for the ATC example (without MANCON)

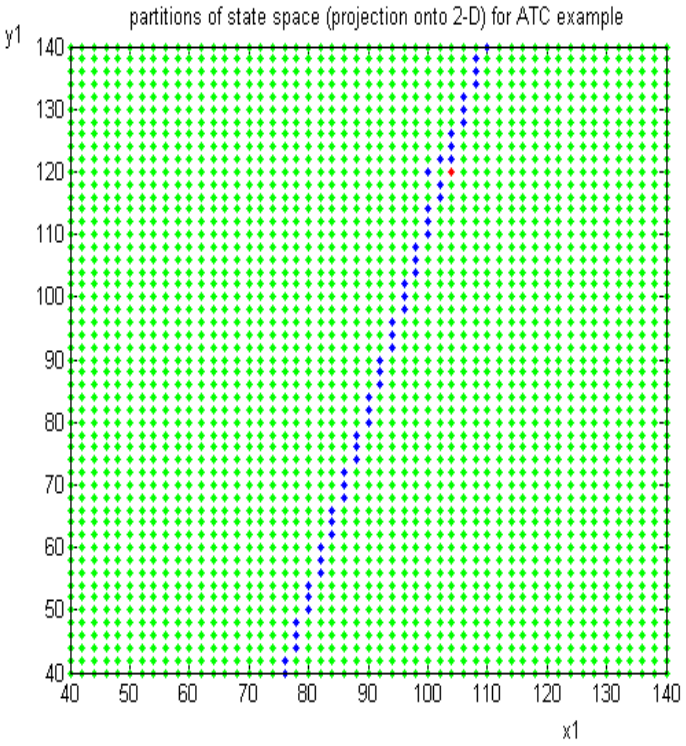

Figure 8.5: Partitions in the state space for the ATC example (without MANCON)

Chapter 9

Conclusion

Through this thesis research, it has been shown that the symbolic reasoning approach can be used in the application to the air traffic conflict and resolution problem. Moreover, the Q^2 approach has been expanded and generalized to linear dynamic systems. Examples are tested against the generalized partitioning and reasoning.

9.1 Contributions

1. Application of qualitative reasoning about dynamic systems in a realistic scenario. The major contribution of this thesis is to show that qualitative reasoning about dynamic systems in a real world application brings measurable benefits over traditional quantitative approaches. The scenario from the air traffic control field has been selected for this purpose.

2. Modifications to the Q^2 approach. The air traffic scenario picked for the thesis does not lend itself to the straight-forward application of the Q^2 approach. Therefore, the Q^2 approach had to be extended, including conversions between Mealy and Moore automata.

3. Hypersurface partition of the output space. One of the contributions of this thesis is the extension of the Q^2 approach by partitioning the output space by hypersurfaces, rather than by "landmark points", as was done in [36]. Such an extension is needed for dealing with more complex problems in which the output space is higher-dimensional and thus partitions cannot be just intervals.

4. Qualitative partitions with multiple criteria. Combining partitions derived from partitioning the output space first with those derived through partitioning the input space first have been investigated and appropriate algorithms have been developed.

5. An algorithm for partitioning dynamic system spaces and for deriving qualitative abstractions have been developed.

6. Qualitative reasoning algorithms. Qualitative reasoning algorithms about dynamic systems have been developed and implemented. Consistency and complexity results have been provided.

7. The middle ground approach to reasoning about dynamic systems. A hybrid approach (quantitative and qualitative) has been proposed to address issues of computational efficiency of reasoning.

8. Generalization of the Q^2 approach to linear systems has been achieved. Both partitioning and reasoning algorithms have been implemented and tested against two example problems successfully.

9.2 Possible Future Research

1. Consider more specific models for the intruder aircraft, like including maneuvers, instead of just nominal projection. This would make the

inference more precise, but at the same time this would increase the complexity significantly. Therefore an in-depth study of this problem is needed.

2. Expand the cost function so that whenever possible, a single maneuver would be used, instead of multiple small maneuvers.

3. Investigate the appropriateness of the Q^2 approach in the context of noise. This would require the incorporation of an uncertainty in the reasoning process.

4. Expand the Q^2 approach to more complicated systems, not just linear systems.

Bibliography

[1] Traffic Alert and Collision Avoidance System (TCAS II). Technical report, Competency Standards for TCAS II Operations and Aeronatical Knowledge Syllabus of Training,FAA.

[2] Introduction to TCAS II. Technical report, United States Department of Transportation, Federal AviationAdministration,Washington D.C, 1990.

[3] TELSACS - Annex 1. Technical report, TELSACS project programme, 1996.

[4] Common ARTS Computer Program Functional Specification (CPFS). Technical report, FAA, 2003.

[5] Common ARTS Design Specification. Technical report, FAA, 2003.

[6] J.S Albus. Outline for a Theory of Intelligence. IEEE Trans.Systems Man Cybernet, 1991.

[7] T. Ball and O Kupferman. An Abstraction-Refinement Framework for Multi-Agent Systems. 21st Annual IEEE Symposium on Logic in Computer Science, 2006, pages 379 – 388, 2006.

[8] P. Benjamin, M. Erraguntla, D. Delen, and R. Mayer. Simulation Modeling at Multiple Levels of Abstraction. Simulation Conference Proceedings, 1:391 – 398, 1998.

[9] T. J. Biggerstaff. Fixing Some Transformation Problems. 14th IEEE International Conference on Automated Software Engineering, 1999., pages 148 – 157, 1999.

[10] N. Bredeche, Shi Zhongzhi, and J.-D. Zucker. Perceptual Learning and Abstraction in Machine Learning. Proceedings of the Second IEEE International Conference on Cognitive Informatics, 2003., pages 18 – 25, 2003.

[11] B.C. Breen. Controlled Flight Into Terrain and the enhanced Ground Proximity Warning system. Aerospace and Electronic Systems Magazine, IEEE, 14(1), 1991.

[12] M Brockmeyer. Automatic Abstractions of Real-time Specifications. Fifth IEEE International Symposim on High Assurance Systems Engineering, 2000., 1:147 – 158, 2000.

[13] R.E. Bryant. Symbolic Simulation-techniques and Applications. Proceedings of 27th ACM/IEEE Design Automation Conference, 1990., pages 517–521, 1990.

[14] S. Buhne, G. Halmans, K. Pohl, M. Weber, H. Kleinwechter, and T. Wierczoch. Defining Requirements at Different Levels of Abstraction. Proceedings of 12th IEEE International Requirements Engineering Conference, 2004., pages 346 – 347, 2004.

[15] Douglas W. Burgess, Sylvia I. Altman, and M.L Wood. TCAS: Maneuvering Aircraft in the Horizontal Plane. The Lincoln Laboratory Journal, 7(2):295–311, 1994.

[16] Chi-Tsong. Chen. Linear System Theory and Design. Oxford Univerity Press, New York Oxford, 1999.

[17] C. Combastel, S. Gentil, and J.-P. and Rognon. A Symbolic Reasoning Approach to Fault Detection and Isolation Applied to Electrical Machines. Proceedings of the 1998 IEEE International Conference on Control Applications, 1998., 1:475–479, 1998.

[18] J.-C. Delvenne and V.D. Blondel. Complexity of Control on Finite Automata. IEEE Transactions on Automatic Control, 51:977 – 986, 2006.

[19] Z. J. Deng, J. W. S. Liu, L. Zhang, M. Seri, and A. Frei. An Open Environment for Real-time Applications. Real-Time Systems Journal, 16(2/3):155–186, 1999.

[20] G. Dowek, A Geser, and C Munoz. Tactical Conflict Detection and Resolution in a 3D Airspace. In 4th USA/Europe Air Traffic Management R and D Seminar, 2001.

[21] D. Dubois, A. Hadj-Ali, and H. Prade. Incoherence Detection and Approximate Solving of Equations Using Fuzzy Qualitative Reasoning. The Ninth IEEE International Conference on Fuzzy Systems, 2000. FUZZ IEEE 2000., 1:203–208, 2000.

[22] V. N. Duong. FREER: Free-Route Experimental Encounter Resolution - Initial Results. In EUROCONTROL Experimental Centre EEC, 1997.

[23] Nicolas Durand and Jean-Marc Alliot. Optimal Resolution of En Route Conflicts. In Laboratoire d'Optimisation Globale, 1997.

[24] FAA. Precision Runway Monitor Demonstration Report. www.tc.faa.gov/acb300/techreports/RD-91-5.pdf. Document DOT/FAA/RD-91/5, February, 1991.

[25] R.N.H.W van Gent, J.M. Hoekstra, and R.C.J. Ruigrok. Free Flight with Airborne Separation Assurance. http://www.cami.jccbi.gov/AAM-500/freeflight/rvg_doc.pdf, 1997.

[26] W. Gerling. Conflict Detection in Air Traffic Control. In Institut f r Flugf hrung, 1994.

[27] P Godefroid. Generalized Model Checking. 12th International Symposium on Temporal Representation and Reasoning, pages 3 – 4, 2005.

[28] F. Guinchiglia and T. Walsh. A Theory of Abstraction. Artificial Intelligence, 1992.

[29] G. Gupta and E. Pontelli. A Constraint-based Approach for Specification and Verification of Real-time Systems. Proceedings of the 18th IEEE Real-Time Systems Symposium, 1997., pages 230 – 239, 1997.

[30] W.H. Harman. TCAS: A System for Preventing Midair Collision. The Lincoln Laboratory Journal, 2(3):437–458, 1989.

[31] Jianghai Hu, M. Prandini, A. Nilim, and S. Sastry. Optimal Coordinated Maneuvers for Three Dimentional Aircraft Conflict Resolution. In AIAA Conf. Guidance, Navigat. Contr, 2001-4294, 2001.

[32] Inseok Hwang, Jesse Hwang, and Clarie Tomlin. Flight-Mode-Based Aircraft Conflict Detection using a Residual-Mean Interacting Multiple Model Algorithm. Technical report, Hybrid Systems Laboratory, Departmetn of Aeronautics and Astronautics, Stanford University.

[33] R. Jota, J. Martins, A. Rito-Silva, and J. Pereira. Experimenting with a Flexible Awareness Management Abstraction for Virtual Collaboration Spaces. Proceedings of Symposium on Applications and the Internet, 2003., pages 56 – 64, 2003.

[34] Keith Kastella and Mark Biscuso. Tracking Algorithm for Air Traffic Control Applications. Air Traffic Control Quarterly, 3(1), 1996.

[35] M. Keulers and Beckers. W. Efficient Data Storage Handling in Matlab. IEEE Symposium on Computer-Aided Control System Design, CACSD, pages 9 – 14, 1992.

[36] M.M. Kokar. On Consistent Symbolic Representations of General Dynamic Systems. IEEE Transactions on systems, man and cybernetics, 25(8):–, 1995.

[37] Jana Kosecka and Claire Tomlin. Generation of Conflict Resolution Maneuvers for Air Traffic Management. In Department of Electrical Engineering and Computer Sciences, 1996.

[38] Jimmy Krozel and Mark Peters. Strategic Conflict Detection and Resolution for Free Flight. In Proceedings of the 36th Conference on Decision & Control, 1997.

[39] J. K. Kuchar and L. C. Yang. Survey of Conflict Detection and Resolution Methods. In AIAA Guidance, Navigation and Control Conference, 1997.

[40] J. K. Kuchar and L. C. Yang. A review of Conflict Detection and Resolution Modeling Methods. IEEE Transactions on Intelligent Transportation SYstems, 1(4):179–189, 2000.

[41] P. Kungas and M. Matskin. Symbolic Negotiation in Linear Logic with Coalition Formation. IEEE/WIC/ACM International Conference on Intelligent Agent Technology, 2006., pages 298 – 305, 2006.

[42] R. Kurki-Suonio and T. Mikkonen. Abstractions of Distributed Cooperation, their Refinement and Implementation. Proceedings of International Symposium on Software Engineering for Parallel and Distributed Systems, 1998., pages 94 – 102, 1998.

[43] Hsuan-Shih Lee. Minimizing Fuzzy Finite Automata. The Ninth IEEE International Conference on Fuzzy Systems, 2000., 1:65 – 70, 2000.

[44] Lissys Limited. In-Flight Performance. http://www.lissys.demon.co.uk/pug/c09.html.

[45] Charles Lin. Finite State Machines with Output. www.cs.umd.edu/class/spring2003/cmsc311/Notes/Seq/fsm.html. University of Maryland, Computer Science.

[46] S.P. Linder and M.M. Kokar. q^2 Symbolic Reasoning about Noisy Dynamic Systems. Journal of Intelligent and Robotic Systems, 24:295–311, 1999.

[47] J. W. S. Liu. Real-time Systems. Prentice Hall, Upper Saddle River, N.J., 2000.

[48] X. Liu, H. Yang, H Zedan, and A. Cau. Speed and Scale Up Software Reengineering with Abstraction Patterns and Rules. Proceedings of International Symposium on Principles of Software Evolution, 2000., pages 90 – 99, 2000.

[49] J. Lygeros, K.H. Johansson, S.N. Simic, Jun Zhang, and S. Sastry. Continuity and Invariance in Hybrid Automata. Proceedings of the 40th IEEE Conference on Decision and Control, 2001., 1:340 – 345, 2001.

[50] J. Lygeros, K.H. Johansson, S.N. Simic, Jun Zhang, and S. Sastry. Continuity and Invariance in Hybrid Automata. Proceedings of the 40th IEEE Conference on Decision and Control, 2001, 1:340 – 345, 2001.

[51] A. Meisels and D. Mintz. Image Interpretation by Symbolic Reasoning. The Sixteenth Conference of Electrical and Electronics Engineers in Israel, 1989., pages 1 – 3, 1989.

[52] M.D. Mesarovic and Y Takahara. Abstract Systems Theory. Springer Verlag, 1989.

[53] S Mondoloni and S. Conway. An Airborne Conflict Resolution Approach using a Genetic Algorithm.

[54] T. Moor, J. M. Davoren, and J. Raisch. Strategic Refinements in Abstraction Based Supervisory Control of Hybrid Systems. Proceedings of Sixth International Workshop on Discrete Event Systems, 2002., pages 329 – 334, 2002.

[55] Russel A. Paielli and Heinz Erzberger. Conflict Probability Estimation for Free Flight. In Journal of Guidance, Control and Dynamics, 1997.

[56] G. J. Pappas and S. Simic. Consistent Abstractions of Affine Control Systems.

[62] M. Rao, T.-S. Jiang, and J.J.-P. Tsai. A New Method to Design Intelligent Control Systems. Proceedings of the IEEE National Aerospace and Electronics Conference, 1988. NAECON 1988., 2:408 – 413, 1988.

[63] K. Rohloff and S. Lafortune. On the Computational Complexity of the Verification of Modular Discrete-event Systems. Proceedings of the 41st IEEE Conference on Decision and Control, 2002, 1:16 – 21, 2002.

[64] H. Sawamura and K. Kiyozuka. A Hybrid Reasoning System with Diagrams and Sentences. Proceedings of IEEE International Symposium on Visual Languages, 2000., pages 73 – 74, 2000.

[65] Raoul Schild. Rule Optimization for Airborne Aircraft Separation. PhD thesis, Institute of Econometrics, Operations Research and System Theory at the Technical University of Vienna, 1997.

[66] D.G Schwartz. Connection Between Fuzzy Quantifiers and the Classical Modalities. NAFIPS/IFIS/NASA '94. Proceedings of the First International Joint Conference of the North American Fuzzy Information Processing Society Biannual Conference., pages 310 – 314, 1994.

[67] Jinglai Shen, N. H. McClamroch, and E.G. Gilbert. A Computational Approach to Conflict Detection Problems. In Proceedings of the American Control Conference, 1999.

[68] S. Sivasundaram and S. Vassilyev. Automata Dynamics Preserved Under Homomorphism: Connectness, Reachability, Optimality. IEEE International Conference on Systems, Man, and Cybernetics, 1998., 2:1462 – 1466, 1998.

[69] M Stevenson, Pin-chan Du, and Ming Rao. An Expert System for selecting Viscosity Models and Prediction. IEEE Pacific Rim Conference on Communications, Computers and Signal Processing, 2:730–733, 1993.

[70] M. Stevenson, Q. Wang, and M. Rao. An Intelligent Approach to Conceptual Design Automation of Chemical Processes. IEEE Pacific Rim Conference on Communications, Computers and Signal Processing, 1993., 2:650 – 653, 1993.

[71] P. Tabuada, G.J. Pappas, and P Lima. Compositional Abstractions of Hybrid Control Systems. Proceedings of the 40th IEEE Conference on Decision and Control, 2001., 1:352 – 357, 2001.

[72] Claire Tomlin, G.J. Pappas, and S. Sastry. Conflict Resolution for Air Traffic Management: A Study in Multi-Agent Hybrid Systems. In Department of Electrical Engineering and Computer Sciences, 1997. Versions of this paper has been presented at the 1997 IEEE COnference on Decision and COntrol, San Diego, and the first USA / Europe ATM Seminar, Eurocontrol, Paris, 1997.

[73] C. Unsal, P. Kachroo, and J.S. Bay. Simulation Study of Learning Automata Games in Automated Highway Systems. IEEE Conference on Intelligent Transportation System, 1997, pages 936 – 941, 1997.

[74] F. Wagner. Moore or Mealy model? http://www.stateworks.com/active/download/TN10-Moore-Or-Mealy-Model.pdf.

[75] Yingjie Wang, Li min Jia, and Yong Qin. Application of Entity Automata to Railway Transportation System Modeling and Simulation. Proceedings of IEEE Intelligent Transportation Systems, 2003., 2:1463 – 1466, 2003.

[76] Oliver Watkins and John Lygeros. Safety Relevant Operational Cases in Air Traffic Management. Technical report, University of Cambridge, UK, 2002.

[77] E.U. Weber and Coskunoglu O. Descriptive and Prescriptive Models of Decisionmaking. IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS, 20(2):310–317, 1990.

[78] Xian Xu. P Systems and Finite Automata. First International Conference on Complex, Intelligent and Software Intensive Systems. CISIS 2007., pages 135 – 138, 2007.

[79] T. Yairi, K. Hori, and S. Nakasuka. Unified Criterion for State and Action Abstraction in Autonomous Agent. Proceedings of IEEE International Conference on Systems, Man, and Cybernetics, 1999., 5:165 – 170, 1999.

[80] L. C. Yang and J. K. Kuchar. Prototype Conflict Alerting System for Free Flight. In AIAA Journal of Guidance, Control and Dynamics, 1999.