On the Use of Q^2 Abstractions to Lower the Computational Cost of Derivation of Conflict Resolution Advisories in Air Traffic Control

Mei Li and and Mieczyslaw M. Kokar, *Senior Member, IEEE*

***Abstract*—This paper addresses the problem of high computational complexity of generating multi-step conflict resolution advisories in the domain of air traffic control. Since this problem is known to be NP-hard, one cannot expect algorithms that will solve every instance of the problem independently of its size. Thus the goal is to develop more efficient algorithms that will be able to analyze a wider space of possible resolution advisories, for instance, horizontal maneuvers. This paper presents a study of the use of abstraction to such a problem. However abstraction can lead to wrong decisions, e.g., to maneuvers that result in unsafe states. Such abstractions are referred to as inconsistent. To avoid this kind of problems we use the so called Q^2 abstractions, which are derived from the specifications of a problem and are guaranteed to be consistent. To assess the usability of the Q^2 approach to computing horizontal resolution advisories we analyze the impact of such abstractions on the computational cost of an exhaustive search algorithm as well as on the quality of resolution advisories found. The results show that the use of the Q^2 approach lowers the conflict resolution computation time without losing much of the quality of solutions.**

I. Introduction

In the context of air traffic control, two or more aircraft approaching each other within a close distance will create a *conflict alert* situation. Traditionally, alerts are issued based upon tracking and prediction of the aircraft positions, using quantitative algorithms (cf. [1]). When an alert is issued, the on-board Traffic-alert and Collision Avoidance System (called TCAS [2], [3]) gives the pilot maneuver instructions, known as *resolution advisories* (RA's). 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. But horizontal maneuvers are more time consuming to compute since there are many more maneuver options in the horizontal direction as compared to the vertical direction.

There are three major factors that complicate the computation of maneuvers. First, the rhythm of computing maneuvers is influenced by the frequency of measurement updates (*scans*). Second, the algorithm of the "self" aircraft needs to predict all possible situations, i.e., all possible maneuvers of the "intruder" aircraft in response to the maneuver of the "self" aircraft (combinatorial explosion). And finally, the computation must be based on the prediction of possible maneuvers for a number of steps (scans) ahead (multi-step prediction).

Although it is known that the problem of computing conflict resolution advisories is NP-hard (cf. [4]) and thus one cannot expect to find an algorithm that will solve every instance of this problem independently of its size, the question is whether the computation for limited size, practical problems of finding vertical and horizontal resolution advisories is feasible. Let us consider a scenario for an aircraft pair - the "self" and the "intruder". Let N denote the number of maneuver choices (same for both aircraft) and let L represent the prediction time, i.e., the number of look-ahead steps (measurement scans) to be considered in the prediction. In the first step (L=1) after a conflict has been detected there are N choices for a maneuver for each aircraft. Thus the aircraft pair can be in N^2 states at the end of the first maneuver. In other words, the "self" aircraft needs to compute N^2 maneuvers. In the second step (L=2), again we need to consider all possible maneuvers that both aircraft could execute in each of the N^2 states. The number of maneuver choices for the pair in the second step is again N^2. However, each of the maneuver choices needs to be computed for each of the N^2 states. This will lead to N^4 computations of maneuvers. In the third step (L=3), the same analysis can be repeated for all of the N^4 states leading to N^6 computations of maneuvers. And finally, in the $L-th$ step, there are N^{2L} maneuver computations. Thus the total number of choices for all possibilities during the time duration of L steps is:

$$N^2 + N^4 + N^6 + \ldots + N^{2L} = \frac{N^2(N^{2L}-1)}{N^2-1} \qquad (1)$$

To be close to a realistic scenario, the values of N and L need to be 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 both the horizontal and the vertical planes. These values are summarized in Table I. For horizontal maneuvers, we choose seven turns in the increments of $1^\circ/sec$. The choices for the horizontal linear acceleration are in the increments of $45knot/min$, up to $270knot/min$. For comparison, in the vertical dimension, there are nine choices for climbing up or descending in the range from 500 to $4,500ft/min$, in the increments of 500. The granularity for each of these choices could be finer, which would result in even larger N, and thus higher computational cost.

M. Li is with ARCON Corporation, Waltham, MA 02451, email: meili@coe.neu.edu.

M. Kokar is with the Department of Electrical and Computer Engineering, Norhteastern University, Boston, MA, 02115 USA, e-mail: mkokar@ece.neu.edu

Manuscript received:; revised:.

TABLE I
POSSIBLE MANEUVER CHOICES.

Maneuver Name	Dimension	Unit	Possible Values
turn left or right	horizontal	$^\circ/sec$	1,2,3...7
speed up or down	horizontal	$knot/min$	45, 90, ..., 270
climb up or descend	vertical	ft/min	500, 1000, ..., 4500

According to this table, there are $N = 27$ 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 $L = 5$ steps for prediction, the number of horizontal maneuvers that need to be computed is

$$\frac{27^2(27^{10}-1)}{27^2-1} = 2.06174 \cdot 10^{14} \quad (2)$$

Furthermore, assuming that each computation can be done within $1ms$, the computation time of an exhaustive analysis of all possible advisories would be equal to $5.73 \cdot 10^7$ hours. It is clear that the computation time for multiple-step prediction is too high to be practically implementable. Therefore, there is a need for lowering the cost of RA computation to make it applicable to horizontal RAs.

One way to achieve this goal would be by using abstraction, i.e., deriving a more coarse (abstract) representation of the original (ground) problem and then carrying out computation in the abstract space. An example of such an approach might be to lower the granularity of the partitioning of the time dimension by computing predictions and maneuvers for time intervals larger than dictated by the measurement scans. Unfortunately, this kind of simple abstraction may be too coarse and thus might result in errors - generating RAs that would lead to conflict. This kind of abstraction would be an *inconsistent abstraction* [5]. To avoid this kind of problems, we use the Q^2 approach [6] which shows how to derive *consistent* abstractions of general dynamical systems. In this approach, the system space of a dynamical system is partitioned into regions, symbols are assigned to particular regions and then inference (computation) is carried out in the abstract space. As will be shown later in this paper, the Q^2 approach had to be extended in order to be applicable to the problem of RA computation.

The rest of this paper is organized as follows. Section II contains an overview of the research that is pertinent to the problem of air traffic conflict resolution. The problem itself is formally defined in Section III. The Q^2 approach is overviewed in Section IV. Section V shows the Q^2 representation of the RA generation problem. This is followed by a description of the algorithm for computing of all possible RA's (Section VI) based upon the abstract symbolic representation developed in Section V. Then, in Section VII the Q^2 based search for RAs is compared to the search in the original solution space in terms of both computation time and quality of the found solutions. And finally, Section VIII presents conclusions and suggestions for future research in this area.

II. RELATED LITERATURE

A number of approaches [4], [7]–[25] to the problem of detection and resolution of traffic conflicts have been developed. Most of them were classified by Kuchar and Yang in [26], [27] (see also papers in the two-part special issue [28]), based upon the way the response to conflicts is determined, in the following three categories: 1) *prescribed*, 2) *force field* and 3) *optimized*.

In the *prescribed* approach [21], [22], 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 cover all possible encounter situations. Additionally, it may be difficult to certify that the system always operates as intended.

The *force field* approaches [4], [8], [13] model each aircraft as a charged particle and use modified electrostatic equations to determine conflict resolution maneuvers. The shortcoming of the force field approaches is that in some cases the computed maneuver path is difficult to realize 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 would require a high level of guidance on the flight deck and would increases the complexity of the solution beyond issuing simple heading vectors. Additionally, some solutions may include sharp-angle turns or other physically infeasible trajectories that would have to be modified to be used in operation.

An *optimized* conflict resolution approach [4], [7], [9]–[12], [14]–[20], [23], [24] selects a decision by determining which of the several avoidance options minimizes a given cost function. The existing optimized methods differ in the assumptions on the model of the dynamical 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 [7], [9]–[12], [14], [16]–[18]. 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 dynamical system model. This uncertainty is very important in long-term conflict detection. The other extreme of the use of dynamical system models is to use the *worst-case projection* [20]. However worst-case maneuvers are highly unlikely and using this model may greatly reduce the overall traffic capacity. The *probabilistic* approach [15], [19], [24], [25], [29], [30] appears to provide a reasonable balance between relying too much on an aircraft following the dynamical system model and relying too much on the assumption that an aircraft is doing worst-case maneuvers. However, there is a trade-off between the complexity of the probabilistic model and the ability to estimate probabilities rapidly. Also, in some cases 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 (c.f. [24]) does not incorporate issues such as increased fuel burn or flight time in the decision on an alert.

III. Formalization of the Problem

Based upon the literature review, we selected the optimized approach to the problem of RA generation. As the first step in this approach, we need to define the cost function and the constraints that the solutions must satisfy.

A. Cost Function

Various aspects can be considered in the selection of the cost function for the optimization problem. Some choices of maneuvers are preferred by the pilots, the controllers and the airlines. Moreover, it is usually desired to select the maneuvers that cause the least deviation from the flight plan and result in the shortest time delay. In our investigation, we define the cost function as a measure of the deviation of the 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:

$$g() = \sum_{t_k=t_1}^{t_l} \sqrt{(x^m(t_k) - x^c(t_k))^2 + (y^m(t_k) - y^c(t_k))^2} \quad (3)$$

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.

Horizontal maneuvers typically include turns, ω, and accelerations, a_v. Normally one maneuver is carried out at a time, i.e., no concurrent maneuvers [12] are executed. Therefore, a maneuver that starts at t_k and ends at $t_k + 1$ can be expressed as:

$$a(t_k) = \begin{cases} a_v(t_k), & \text{acceleration maneuver} \\ \omega(t_k)/v(t_k), & \text{turn maneuver} \end{cases} \quad (4)$$

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).

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

$$x^m(t_{k+1}) = x^m(t_k) + v_x^m(t_k) \cdot \Delta T + \frac{1}{2} \cdot a_x \cdot (\Delta T)^2 \quad (5)$$

$$y^m(t_{k+1}) = y^m(t_k) + v_y^m(t_k) \cdot \Delta T + \frac{1}{2} \cdot a_y \cdot (\Delta T)^2 \quad (6)$$

$$x^m(t_0) = x^c(t_0) \quad (7)$$

$$y^m(t_0) = y^c(t_0) \quad (8)$$

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 "intruder" aircraft is computed using the same equations using zero acceleration.

The optimization is subject to the satisfaction of three constraints. These constraints are described below.

B. PROCON, LINCON and MANCON Alerts

The cost function should be minimized under the condition that every maneuver results in a sufficient separation between two aircraft. The satisfaction of any of such danger conditions would result in the generation of a *conflict alert*. In ATC, these alerts are commonly known as PROCON, LINCON and MANCON [31].

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}:

$$(d < d_{proxim1}) \vee (d < d_{proxim2} \wedge R > R_{proxim}) \quad (9)$$

The LINCON condition is satisfied if, by linear prediction, the impact time, $LAT1$, of the aircraft pair is below the look-ahead time, $T_{look-ahead}$ (usually $40sec$). Since in this case we are considering only horizontal maneuvers, we can assume that $LAT1$ is equal to the Time of Lateral Violation ($TOLV$) [31] and get the following constraint:

$$(TOLV < T_{lookahead}) \wedge (TOLV > 0) \quad (10)$$

The predicted value of $TOLV$ is computed from the following equation [31]:

$$TOLV = \frac{1}{\triangle\dot{x}^2 + \triangle\dot{y}^2}(-(\triangle x \cdot \triangle\dot{x} + \triangle y \cdot \triangle\dot{y}) \quad (11)$$
$$-[LATQ^2(\triangle\dot{x}^2 + \triangle\dot{y}^2) - (\triangle x \cdot \triangle\dot{y} + \triangle y \cdot \triangle\dot{x})^2]^{1/2})$$

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

The MANCON conditions are tested when the turn rate ("self"), 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 [31]:

$$\omega > \omega_{thresh} \wedge [(s_1 < 0) \vee (s_2 > 0) \vee (s_3 > 0) \vee (s_4 > 0)] \quad (12)$$

$$s_1 = \triangle x \cdot \triangle\dot{x} + \triangle y \cdot \triangle\dot{y} \quad (13)$$

$$s_2 = \dot{x}_1 \cdot \dot{x}_2 + \dot{y}_1 \cdot \dot{y}_2 \quad (14)$$

$$s_3 = \triangle x \cdot \dot{x}_1 + \triangle y \cdot \dot{y}_1 \quad (15)$$

$$s_4 = -(\triangle x \cdot \dot{x}_2 + \triangle y \cdot \dot{y}_2) \quad (16)$$

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

IV. Introduction to the Q^2 Approach

As stated earlier, in our work we used the Q^2 approach to represent and reason about a dynamical system - an aircraft pair. In this section we give a brief overview of the Q^2 approach; a full description is presented in [6]. The main idea of this approach is to develop a qualitative model of a quantitative dynamical system and then use the qualitative model to analyze the behavior of the dynamical system. While the results of such analysis will be qualitative in nature,

since intervals and regions will be used instead of points, the qualitative conclusions still should be correct. Abstractions that satisfy such a requirement are called *consistent*.

For a simple example, consider the physical abstraction of "boiling temperature". While it is customary to say that the boiling temperature of water is 100 C, this is not quite true, since whether water is boiling or not also depends on the surrounding environmental pressure. If the pressure is higher than one atmosphere then boiling will not occur at 100 C. Obviously, a more precise statement is that the boiling point for water is 100 C under the pressure of 1 atmosphere. However, while adding one more variable (pressure) makes the definition of the abstraction more precise, it still does not solve the problem because it does not tell us what happens to water when it is in a vacuum or when the surrounding atmospheric pressure is two atmospheres. If we consider the Cartesian product of two variables, temperature and pressure, and a point in this space <100 C, 1 atmosphere>, then two straight lines through this point, perpendicular to the axes, would divide this space into four regions, corresponding to temperatures below/above 100 C and pressures below/above 1 atmosphere. This kind of abstraction would not be consistent since within the same region water could be boiling or not. A consistent abstraction would take into consideration the relationship between the temperature and the pressure. Such a relationship could be captured by a line in such a 2D space, which would define two regions - one for "boiling" and one for "not boiling".[1] For a higher-dimension space, the line would be replaced by a *hypersurface*.

An example of reasoning with abstractions about a dynamical system is: "Given the system input is within a given range of inputs and the initial state is within a given region of states and the input is applied over a time that is within a given time interval, then the next state will be within a given region of states and the output will be within a given range of outputs." This reasoning is correct if the predicted states and outputs always fall within the specified ranges or regions.

The Q^2 framework for reasoning with abstractions about a general dynamical system (c.f. [32]) is shown in Figure 1. In this figure, the lower row shows the (quantitative) *general dynamical system* (GDS), the top row represents the *qualitative dynamical system* (QDS) and the middle shows the *qualitative abstraction functions*, χ, that map the GDS to the QDS.

The model of the GDS can be represented by differential (or difference) equations:

$$\dot{q} = f(q(t), x(t), t) \tag{17}$$

$$w = g(q(t), t) \tag{18}$$

where q is the state, w is the output, x is the control input of the system, f is the state transition function and g is the *output function*. In the control literature [32]–[34], 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. The Cartesian products of the input, state and output variables are referred to as the *input space* **X**, *state space* **Q** and output space **W**, respectively. Additionally, time is represented by an ordered set *T*.

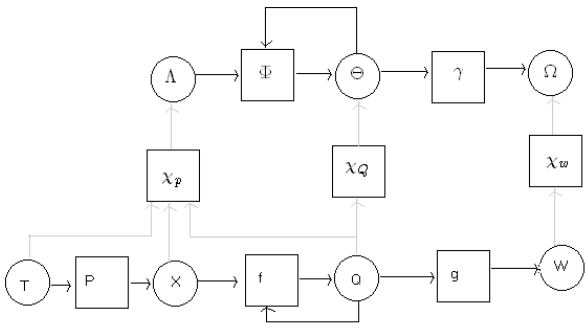

Fig. 1. Q^2: Quantitative-qualitative representation of a dynamical system.

Formally, an *abstraction* is a mapping from a real system to an abstract system which maintains certain desirable properties and throws away details [5], [35]. 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.

In the Q^2 approach, a *qualitative abstraction* consists of three abstraction functions. The first one operates on the Cartesian product of time **T**, initial state $\mathbf{Q}_0$ and input **X**. This Cartesian product is represented as **TQX**. The other two abstract current state and output. Thus formally, the *qualitative abstraction function*, χ, is defined as:

$$\chi = (\chi_{\mathbf{TQX}}, \chi_{\mathbf{Q}}, \chi_{\mathbf{W}}) \tag{19}$$

consisting of three qualitative abstraction functions:

$$\chi_{\mathbf{TQX}} : T \times \mathbf{Q} \times \mathbf{X} \to \Lambda \tag{20}$$

$$\chi_{\mathbf{Q}} : \mathbf{Q} \to \Theta \tag{21}$$

$$\chi_{\mathbf{W}} : \mathbf{W} \to \Omega \tag{22}$$

The sets Λ, Θ, Ω are called the *qualitative input events*, *qualitative states* and *qualitative outputs*, respectively. The two functions shown in the top row of Figure 1 are called the qualitative state transition function

$$\phi : \Theta \times \Lambda \to \Theta \tag{23}$$

and the qualitative output function

$$\gamma : \Theta \to \Omega \tag{24}$$

As shown in [6], a Q^2 abstraction χ is consistent if the following two conditions hold:

$$\gamma(\chi_{\mathbf{Q}}(q)) = \chi_{\mathbf{W}}(g(q)) \tag{25}$$

$$\phi(\chi_{\mathbf{Q}}(q_0), \chi_{\mathbf{TQX}}(t, q_0, x)) = \chi_{\mathbf{Q}}(f(t, q_0, x)) \tag{26}$$

Assuming that the spaces **TQX**, **Q** and **W** of a dynamical system are given, the question is how to derive the sets Λ, Θ and Ω of the QDS? The answer depends on what else is known. In [6] it was assumed that the partition of the output space **W** by *critical hypersurfaces* in that space was known.

[1] A physicist would carry this kind of reasoning further by adding more variables, but we will stop here since our goal here is to just give an idea of what the difference between consistent and inconsistent abstractions is.

In that case the critical hypersurfaces that partition the state space **Q** can be obtained as an image of the output space hypersurfaces through g^{-1}. Similarly, the critical hypersurfaces in the **TQX** space can be obtained through the inverse of the state transition function f. These hypersurfaces then delineate regions in **W**, **Q** and **TQX**, respectively. Each region is then assigned a symbol representing elements of the sets Λ, Θ and Ω, respectively.

Finally, the function ϕ can determined by selecting any point (t, q_0, x) in a given region in **TQX**, represented by the symbol $\lambda \in \Lambda$, and a region in **Q** corresponding to q_0, represented by the symbol $\theta \in \Theta$, and assigning to these two symbols the symbol θ', such that $q' = f(t, q_0, x)$ is in the region represented by the symbol θ'. A similar procedure can be used to derive the qualitative output function, γ.

V. Q^2 Representation of the Conflict Alert Resolution Problem

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

A. System Model

The joint state vector q for an aircraft pair in the $X-Y$ plane is given by Equation 27 below. The joint input vector x for the pair is given by Equation 28. And the state transition function f is given by Equation 29.

$$q(k) = [x_1(k), y_1(k), \dot{x}_1(k), \dot{y}_1(k), x_2(k), y_2(k), \dot{x}_2(k), \dot{y}_2(k)]^T \tag{27}$$

$$x(k) = [a_{x1}(k), a_{y1}(k), a_{x2}(k), a_{y2}(k)]^T \tag{28}$$

$$q(k) = f(q(k-1), x(k-1), k) = \begin{bmatrix} \mathbf{A} & \mathbf{0}(4,4) \\ \mathbf{0}(4,4) & \mathbf{A} \end{bmatrix} \cdot q(k-1) + \begin{bmatrix} \mathbf{B} & \mathbf{0}(4,2) \\ \mathbf{0}(4,2) & \mathbf{B} \end{bmatrix} \cdot x(k-1) \tag{29}$$

where $\mathbf{0}(\mathbf{4},\mathbf{4})$ is a 4 by 4 matrix of zeroes, $\mathbf{0}(\mathbf{4},\mathbf{2})$ is a 4 by 2 matrix of zeroes and $\mathbf{A}$, $\mathbf{B}$ are matrices defined below in which T represents the sampling period.

$$\mathbf{A} = \begin{bmatrix} 1 & 0 & T & 0 \\ 0 & 1 & 0 & T \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad \mathbf{B} = \begin{bmatrix} T^2/2 & 0 \\ 0 & T^2/2 \\ T & 0 \\ 0 & T \end{bmatrix} \tag{30}$$

The output vector w of the system is given by Equation (31).

$$w(k) = [x_1(k) - x_2(k), y_1(k) - y_2(k), \dot{x}_1(k) - \dot{x}_2(k), \dot{y}_1(k) - \dot{x}_2(k)]^T \tag{31}$$

The output function is given by Equation (32).

$$w(k) = \begin{bmatrix} 1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & -1 \end{bmatrix} \cdot q(k) \tag{32}$$

B. Qualitative Partitions

The space of inputs **X** of this system is four dimensional. It consists of vectors x of accelerations, as shown in Equation 28. The output space **W** of this system is four-dimensional. It consists of vectors w shown in Equation 31. The state space **Q** is eight-dimensional. It consists of state vectors q of the type shown in Equation 27. The **TQX** space is the Cartesian product of **T** $\times$ **Q** $\times$ **X**, where **T** represents time. It is a 13-dimensional space of vectors as shown in Equation 33 below.

$$[x_1(k), y_1(k), \dot{x}_1(k), \dot{y}_1(k), x_2(k), y_2(k), \dot{x}_2(k), \dot{y}_2(k), a_{x1}(k+1), a_{y1}(k+1), a_{x2}(k+1), a_{y2}(k+1), T]^T \tag{33}$$

Note that according to the Q^2 approach presented in [6], the **Q** subspace of **TQX** represents "initial states", i.e., for any time k, we have a vector $q(k) \in \mathbf{Q}$, as well as a vector $q(k-1)$ in the **Q**-subspace of **TQX**. Thus this model includes two separate spaces for state vectors. Time T shown in Equation 33 represents the length of the time interval between the time instant $k-1$ and k.

The qualitative abstraction functions can be presented in at least two ways. In [6] it was proposed to use *critical hypersurfaces* to capture partitions of the system spaces. This approach is especially appropriate for continuous time systems. In the work presented in this paper, we deal with a sampled-data system and thus we are able to represent the abstraction functions as explicit maps from appropriate subsets of the system spaces to the qualitative sets.

In the Q^2 approach presented in [6], it was assumed that the qualitative partition of the output space **W** was given and then the qualitative abstractions of the state space **Q** and of the qualitative input space **TQX** were derived from the hypersurfaces partitioning the output space through the inverses of the output function g and state transition function f, respectively. Then an automaton was constructed in such a way that qualitative inputs would drive the automaton from one qualitative state to another, while the output function would provide qualitative labels (symbols) characterizing system behavior, i.e., qualitative outputs.

In the application of the Q^2 approach to the situation of conflict alert generation discussed in this paper we are interested in the qualitative abstractions corresponding to the specific alerts described in Section III-B, i.e., PROCON, LINCON and MANCON. In order to cover all possible situations, we need to add the SAFE state to this list, i.e., the state in which none of the three alerts is generated.

The analysis of these alerts reveals that two of them - PROCON and LINCON - are defined in terms of the output variables. PROCON is defined by Equation 9. As we can see, this alert is defined purely in terms of the position variables, all of which are included in the output vector defined in Equation 31. Therefore a region in the output space - a qualitative output - corresponds to the PROCON status.

LINCON is defined in Equations 10 and 12. Again, these equations make use of only relative positions and relative velocities, which can be computed from the output variables. However, LINCON is also 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 the definition of LINCON should involve the input variables. The answer to this question has to be addressed before we can continue with partitioning the system spaces and the Q^2 approach.

From the point of view of air traffic control, 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 constant velocity assumption, the track prediction would be somewhat inaccurate. So LINCON can be fully specified in the output space.

A similar question involves MANCON. MANCON is defined by equations 12 through 16. As we can see from these equations, the MANCON condition depends on the relative positions and relative velocities of an aircraft pair. 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 specified by the equations listed above are tested only when a maneuver is detected. Thus, unlike in the LINCON case, there is no constant velocity assumed and thus accelerations need to be considered. Due to this fact and taking into consideration that the turn rate depends on the *previous state* (see [31] for turn rate computation), the conclusion is that the definition of MANCON must involve the input variables (see Equation 28). Or in other words, the MANCON condition must be defined in the **TQX** space.

In summary, to derive qualitative abstractions and define a QDS, we need to extend the approach described in [6]. More specifically, we need to develop an approach to define a QDS when some of the qualitative abstractions are defined in the output space **W**, while some others in the **TQX** space.

One might try to start with partitioning the **TQX** space and then partition the state space by mapping the partitioning of the **TQX** space through the state transition function. However, in that case the result might not be a partition, since the images of the **TQX** partitions in the state space could overlap. This problem occurs when the state transition function is not one-to-one. Our approach to developing a QDS for the conflict alert generation problem, in which some of the qualitative abstractions need to be defined in the **TQX** space, is presented in the next section.

C. Multi-step Partitioning Algorithm

To solve the problem of developing a QDS for a dynamical system for which qualitative abstractions are defined in both output and **TQX** spaces, we propose a multistep approach. First, we develop an automaton that models PROCON and LINCON. This automaton is a Moore machine [36] in which outputs (alerts) are associated with qualitative states. Then we sub-partition the **TQX** space using the MANCON condition.

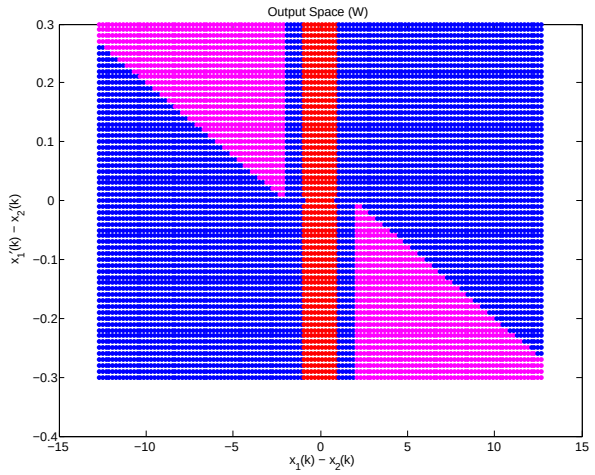

Fig. 2. Partitioning of the output space. Red dots represent ω_1 (PROCON), purple - ω_2 (LINCON), blue - ω_{34} (OTHER).

This results in a Mealy machine [36] in which alerts are associated with qualitative inputs. However, since this representation is not desirable, we convert it into an equivalent Moore machine [36]. This concludes the QDS construction process. All the steps in this process are described below.

1) Partitioning of the Output Space and Qualitative Outputs : In this step we partition the output space **W** into three regions, two of which correspond to the PROCON and LINCON alerts as defined in Section III-B, with the rest of the space assigned to the "OTHER" region, corresponding to 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 includes those points in the output space for which the PROCON alert is issued (defined by Equation 9), ω_2 corresponds to the points in which PROCON alert is not issued but LINCON alert is issued (defined by Equations 10 and 12), ω_{34} includes the "OTHER" points. These OTHER points may include MANCON and "SAFE" which are not separable at this time. The result is the abstraction function:

$$\chi_{\mathbf{W}} : \mathbf{W} \rightarrow \Omega \tag{34}$$

An example of a projection of the output space partitioning onto 2D is shown in Figure 2. The horizontal axis in this figure represents $x_1(k) - x_2(k)$ while the vertical axis represents $\dot{x}_1(k) - \dot{x}_2(k)$.

2) Partitioning of the State Space and Qualitative States : The goal of partitioning of the state space **Q** is to obtain the qualitative state abstraction function $\chi_{\mathbf{Q}}$:

$$\chi_{\mathbf{Q}} : \mathbf{Q} \rightarrow \Theta \tag{35}$$

where Θ represents the set of qualitative states. This goal can be achieved in the following steps:

1) For each state $q \in \mathbf{Q}$ compute the output associated with it by the output function g.

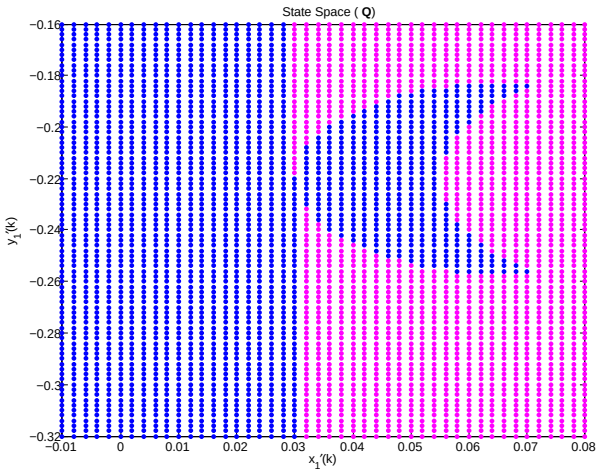

Fig. 3. Partitioning of the state space. Red is θ_1 (PROCON; not visible in this figure), purple is θ_2 (LINCON), blue is θ_{34} (OTHER).

2) Find the qualitative output corresponding to this output by the qualitative output function χ_W.
3) Find the qualitative state $\theta \in \Theta$ that is mapped to this qualitative output by the qualitative output function γ. Assign this qualitative state to q.

This algorithm can be captured by the following equation:

$$\chi_{\mathbf{Q}}(q) = \gamma^{-1}(\chi_W(g(q))) \tag{36}$$

The completion of this process requires the knowledge of the qualitative states and of the qualitative output function γ. In this case, we define the qualitative state set as consisting of three qualitative states, each corresponding to a qualitative output:

$$\Theta = \{\theta_1, \theta_2, \theta_{34}\} \tag{37}$$

The qualitative output function γ is defined in such a way that θ_1 corresponds to ω_1 (or PROCON), θ_2 corresponds to ω_2 (or LINCON), and θ_{34} corresponds to ω_{34} (or OTHER). We can represent this as an explicit listing of the assignments:

$$\gamma = \{< \theta_1, \omega_1 >, < \theta_2, \omega_2 >, < \theta_{34}, \omega_{34} >\} \tag{38}$$

An example of a projection of the state space partitions onto 2D is shown in Figure 3. The horizontal axis in this figure represents $\dot{x}_1(k)$ while the vertical axis represents $\dot{y}_1(k)$.

3) Partitioning of the Input Space and Qualitative Inputs : The qualitative input abstraction function χ_{TQX} is obtained in a similar way as the abstraction function for the states. Since at this time we are not taking into account the MANCON alerts, we denote the qualitative abstraction function as χ_{Λ_1}. We define the qualitative input set Λ_1 as consisting of three qualitative inputs corresponding to PROCON, LINCON and OTHER, respectively:

$$\Lambda_1 = \{\lambda_1, \lambda_2, \lambda_{34}\} \tag{39}$$

The meaning of these qualitative inputs is that λ_1 causes transition to θ_1, λ_2 causes transition to θ_2 and λ_{34} causes transition to θ_{34}. This defines the qualitative state transition function Φ. It is represented as an automaton as shown in

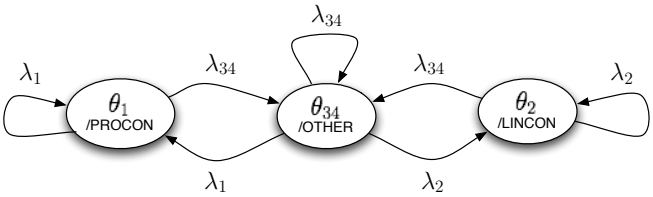

Fig. 4. Moore Machine. Qualitative state transitions without considering MANCON (level 1).

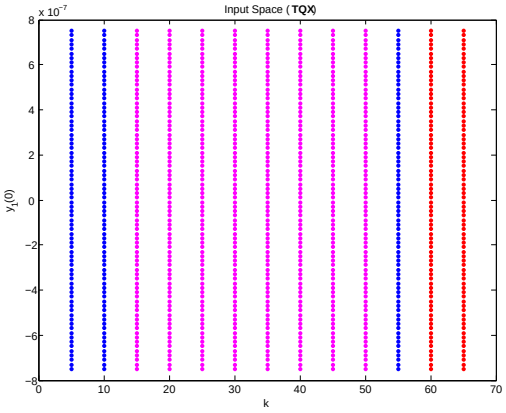

Fig. 5. Partitioning of the input space. Red is λ_1 (PROCON), purple is λ_2 (LINCON), blue is λ_{34} (OTHER).

Figure 4. This is a Moore machine in which outputs are associated with states. 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 state to the OTHER state.

The qualitative input abstraction function χ_{Λ_1} is defined by the following expression:

$$\chi_{\Lambda_1}(q_0, x, T) = \Phi^{-1}(\chi_Q(q_0), \chi_Q(f(q_0, x, T))) \tag{40}$$

An example of a projection of the input space partitions onto 2D is shown in Figure 5. The horizontal axis in this figure represents time, while the vertical axis represents the initial Y position, $y_1(0)$, of the first aircraft in the pair.

As a result, we obtain three regions in the **TQX** space corresponding to the three qualitative inputs. The regions are not necessarily connected. For instance, as can be observed in Figure 5, the region corresponding to λ_{34} is disconnected and consists of two subregions: one for $T = 5$ and 10, and another for $T = 55$.

4) Partitioning of **TQX** *with MANCON:* Now we expand the qualitative representation of this dynamical system to account for MANCON. Since MANCON is defined in terms of the variables from the input space, we need to refine the abstraction χ_{Λ_1} so that it identifies regions in the **TQX** space that correspond to MANCON. In other words, we need to sub-partition **TQX** into regions corresponding to MANCON and non-MANCON, resulting in the qualitative input set Λ^2 consisting of two qualitative inputs:

$$\Lambda^2 = \{\lambda^m, \lambda^s\} \tag{41}$$

where λ^m corresponds to MANCON and λ^s corresponds to non-MANCON. Then the two partitions (corresponding to

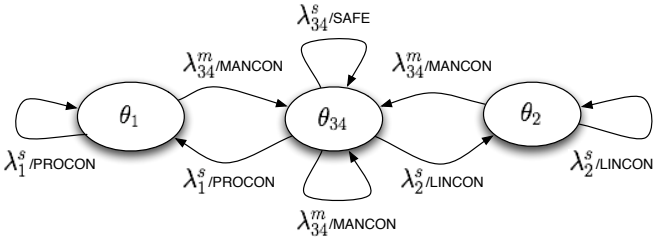

Fig. 6. Mealy Machine. Qualitative state transitions with MANCON included.

Λ^1 and Λ^2) need to be combined. In a general case, this might result into sub-partitioning of each of the partitions corresponding to PROCON, LINCON and OTHER, i.e., in a Cartesian product of the two qualitative input sets

$$\Lambda^1 \times \Lambda^2 = \{< \lambda_1, \lambda^m >, < \lambda_2, \lambda^m >, < \lambda_{34}, \lambda^m >, \\ < \lambda_1, \lambda^s >, < \lambda_2, \lambda^s >, < \lambda_{34}, \lambda^s >\} \quad (42)$$

However, in this particular example, the MANCON alert is issued only if no PROCON or LINCON is issued. Thus the pairs $< \lambda_1, \lambda^m >, < \lambda_2, \lambda^m >$ in the above Cartesian product are not needed since they do not correspond to any regions in the **TQX** space. This also implies that $< \lambda_1, \lambda^s >, < \lambda_2, \lambda^s >$ correspond to the same regions as λ_1 and λ_2, respectively. In the new representation, we label these regions as λ^s_1, λ^s_2. Finally, only the OTHER partition (represented by λ_{34}) needs to be subpartitioned by the MANCON conditions. We use the following notation for the remaining two pairs: $\lambda^m_{34} =< \lambda_{34}, \lambda^m >$ and $\lambda^s_{34} =< \lambda_{34}, \lambda^s >$. In summary, we have the qualitative inputs Λ:

$$\Lambda = \{\lambda^s_1, \lambda^s_2, \lambda^m_{34}, \lambda^s_{34}\} \quad (43)$$

where λ^s_1 corresponds to PROCON, λ^s_2 to LINCON, λ^m_{34} to MANCON and λ^s_{34} to SAFE.

Our next goal is to construct an automaton that captures the qualitative behavior of this dynamical system. Since λ^m_{34} and λ^s_{34} represent the split of λ_{34} into two qualitative inputs, we could just re-draw the automaton of Figure 4 replacing the transition labeled by λ_{34} with two transitions labeled by λ^m_{34} and λ^s_{34}, and replacing the symbols λ_1, λ_2 with the symbols λ^s_1 and λ^s_2, respectively. While this would capture the behavior of the system with respect to states and transitions, this would not capture the outputs. Since outputs are associated with states, such an automaton would not capture the output corresponding to the MANCON alert and to the SAFE condition. The outputs of MANCON and SAFE are still associated with the inputs. So in order to represent the outputs faithfully we need to use a Mealy machine representation in which outputs are associated with transitions (i.e., states and inputs), rather than with states. Towards this end, we use the Moore machine of Figure 4 as a start, but move all the outputs from the states to the transitions *to* the particular states. Moreover, we add the two new transitions caused by λ^m_{34} and λ^s_{34}. The resulting Mealy machine is shown in Figure 6.

5) Converting the Mealy Machine to a Moore Machine : While the Mealy machine of Figure 6 captures all three alerts, it is not very convenient to use for tracking system behaviors. For instance, F. Wagner [37] makes the following statement: " If the specified state machine is to be coded the

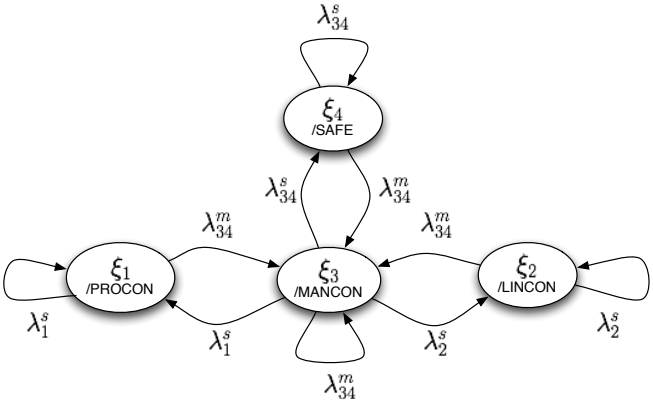

Fig. 7. Complete Moore Machine. Qualitative state transitions, with MANCON included.

model used has enormous influence on the program quality. A Moore model is very easy to code, the transition may be often implemented just by constants as initialized tables. The Mealy model opens the Pandora box: the program becomes so complex that we lose the state machine in the confusing code." However, a Mealy machine can be converted to an equivalent Moore machine, and vice verse [38]. So in the next step we convert the previously obtained Mealy machine (Figure 6) to a Moore machine. For this purpose, we use the standard procedure known in the literature (e.g., [39]).

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).

In formal terms, the conversion can be represented as a function, β, from state-input pairs of a Mealy machine to states of a Moore machine (these states are sometimes called *abstract states*). Let θ_i denote a state of the original Mealy machine, λ_j denote an input and ξ_k denote a state in the Moore machine. The Mealy-Moore conversion procedure can then be defined as:

$$\xi_k = \beta(\theta_i, \lambda_j) \quad (44)$$

The Moore machine representation of our conflict alert resolution problem is shown in Figure 7. 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. This Moore machine includes four abstract states: $\xi_1, \xi_2, \xi_3, \xi_4$. Each of these states is now associated with a single alert: ξ_1 - PROCON, ξ_2 - LINCON, ξ_3 - MANCON and ξ_4 - SAFE. The mapping from the states of the Mealy machine to the states of the Moore machine is described below.

$$\xi_1 = \beta(\theta_1, .) \quad (45)$$

$$\xi_2 = \beta(\theta_2, .) \quad (46)$$

$$\xi_3 = \beta(\theta_{34}, \lambda^m_{34}) \quad (47)$$

$$\xi_4 = \beta(\theta_{34}, \lambda^s_{34}) \tag{48}$$

The Moore machine representation is very convenient for the sake of comprehension. It is 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 the state machine from the current state to another (desired) state. However, this representation adds some complexity to the approach described in [6]. 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 had first developed for our alert resolution problem the two qualitative states - θ_1 and θ_2 (corresponding to PROCON and LINCON, respectively) - had a simple interpretation through the constraints that define these qualitative states (Equation 36). The state corresponding to OTHER, on the other hand, did not represent a clear physical status of the aircraft pair; it just represented some region that is neither associated with PROCON nor with LINCON. However, by following the two-step procedure of the derivation of the final Moore machine, i.e., first determining the concrete qualitative states defined by Equation 36 and then applying the transformation to the Moore machine procedure defined by Equations 45 through 48, we obtain a state machine representation in which all of the states have direct interpretations. Note that the Mealy machine of Figure 6 shows that two different qualitative events lead to the same qualitative state (θ_{34}). Those two qualitative events cause two different outputs. Consequently, state θ_{34} had to be split into two different qualitative states ξ_3 and ξ_4.

VI. Symbolic Reasoning for Conflict Resolution in the X-Y Plane

Now we describe the steps in the procedure of inferring conflict resolution advisories based upon the Q^2 approach presented in this paper.

1) Determine the current qualitative state.
2) Determine the desired next qualitative state.
3) Determine the sequence of qualitative inputs that results in the transition to the next qualitative state determined in step 2.
4) Select a (quantitative) value for the input vector from among the values within the region associated with the qualitative input determined in step 3.

A. Determining the Current Qualitative State

The current qualitative state is computed as follows:

1) Compute the current qualitative output using the qualitative output abstraction $\chi_{\mathbf{W}}$ (Equation 34).
2) Determine the current concrete qualitative state using the qualitative state abstraction $\chi_{\mathbf{Q}}$ defined by Equation 36.
3) Determine the current qualitative input using the qualitative input abstraction χ_{Λ_1} defined by Equation 40.
4) Determine the current qualitative state of the Moore machine using the Mealy-Moore conversion function defined in Equations 45 through 48.

B. Determining the Desired Next Abstract State

The next task 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 7. If the aircraft pair is in a 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: ξ_4=SAFE, ξ_3=MANCON, ξ_2=LINCON, ξ_1=PROCON (most urgent). This results in the following precedence ordering of the states (from the most desirable to the least desirable):

$$\xi_1 \preceq \xi_2 \preceq \xi_3 \preceq \xi_4 \tag{49}$$

The most preferred state ξ_1 may not be reachable from all the states and thus the next best state needs to be selected. The selection decision should be made by analyzing possible transitions.

C. Determining the Qualitative Input

The third task 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 during each step. The automaton map provides the choices from $\{\lambda^s_1, \lambda^s_2, \lambda^m_{34}, \lambda^s_{34}\}$. The qualitative input (actually, a series of single step inputs) that leads to the desired state is selected. This gives a coarse solution to the maneuver advisory.

D. Determining the Quantitative Input

Once the qualitative input is selected, the values of the input variables (Equation 28) need to be selected such that the input vector falls within the given qualitative input. Any set of values in this region would move the system to the desired state. However the best one will minimize the cost function defined in Equation 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.

VII. Evaluation of Usability of Q^2 Abstractions

To assess the potential utility of Q^2 abstractions in conflict alert resolution, we have implemented (in Matlab) the search algorithms for both the original search space and for a combination of the original and abstract spaces. To be able to run simulations within reasonable amounts of time we implemented a simplified version of the problem by making an assumption that the "intruder" aircraft does not perform any maneuvers, except just following the trajectory that it was running when a conflict was detected. This assumption results in a significant reduction of the number of maneuvers that need to be analyzed by the algorithms.

Theoretically, we could use just the first three steps of the algorithm described in Sections VI-A through VI-C and then randomly select a maneuver from the qualitative input region developed in step VI-C. Any maneuver from the selected qualitative input set could be chosen and each such selection

would lead to a safe state. However, such a selection might not be desirable with respect to the cost function. For this reason, we proposed to use the step described in Section VI-D in which a quantitative algorithm selects an optimal sequence of maneuvers from the qualitative partitions selected by the Q^2 algorithm.

The first steps described in Sections VI-A through VI-C are computationally easy. The step of exhaustive search of a qualitative partition, described in Section VI-D is the most time consuming operation since it is essentially the search in the original space that was analyzed is Section I. But still the advantage of this combined approach is that in this case the search space is significantly smaller. However, it is not possible to say precisely what the size of the space is. Roughly, since the whole space in the case discussed in this paper has been partitioned into four partitions, the expected size of the space is one quarter of the original size. The exact reduction, though, depends on how many of the points in the inputs fall into a given qualitative input region. In our simulations we saw, for instance, that 4, 6, 9, 11 and 15 out of total of 27 maneuvers were applicable to a single state.

To assess both the computation time and the cost function of the two approaches, we developed simulations of a number of scenarios in which aircraft pairs were moving so that they would come into a conflict and thus RA's had to be generated such that, if followed by the pilot, would move the "self" aircraft out of the danger zone. The simulated data were fed into both the Q^2 algorithm described in this paper and into an algorithm that was searching the original search space. The results then were compared in terms of both the computation times and the quality of the maneuvers generated by the two approaches.

In this section we first describe the scenarios used in the simulations. The most common encounter possibilities are included. Then the computation costs of these scenarios for both methods are compared. And finally, the maneuver quality metrics of the two methods are compared.

A. Scenarios

We simulated twenty scenarios of pairs of aircraft trajectories. Each simulation covered 30 scans. The duration of each scan was 4.6 sec. This assumption was based on the frequency of a typical scan of the radar. For each scenario, the simulated aircraft pairs were in a conflict situation at 15 scans; we call it the *approach time*. The simulated aircraft included a "self" aircraft, i.e., the aircraft from whose point of view the alerts were computed, and the "intruder" - the aircraft whose state and possible maneuvers were estimated.

The scenarios can be classified in the four types, depending on two aspects: 1. whether the planned flight path of the "self" aircraft is linear or turning, and 2. whether the velocity of the "intruder" aircraft is constant or accelerating. Table II shows parameters of four scenarios which are representative of the four scenario types. The turn rates, accelerations, starts and ends of the turns and the approach angles are all shown in Table II. Examples of simulated paths are shown Figure 8.

TABLE II
EXAMPLES OF SIMULATED SCENARIOS.

	Self (A)			Intruder (B)	
Scen	ω [$^\circ/sec$]	Turn start [scan #]	Turn end [scan #]	a [g]	Approach angle [$\circ$]
1	0	-	-	0	45
2	1	5	16	0	90
3	1	2	16	0.05	90
4	0	-	-	0.1	80

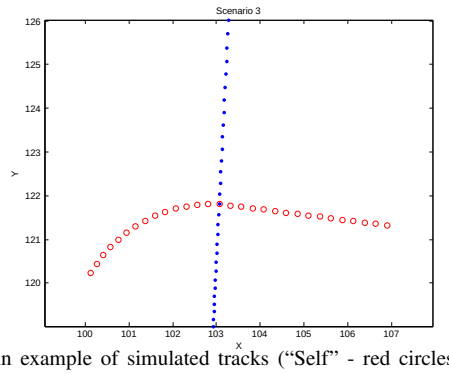

Fig. 8. An example of simulated tracks ("Self" - red circles, "Intruder" - blue dots).

B. Computed Maneuvers

Both algorithms computed conflict conditions and in case a conflict was detected, generated maneuvers that would take the "self" aircraft out of a conflict situation. In this computation, a number of look-ahead scans were considered. I.e., for each of the feasible maneuvers, all the feasible maneuvers of the "self" aircraft were considered, and so on until a predefined number L of the look-ahead steps.

Examples of maneuver sequences computed by the Q^2 algorithm for four scenarios are summarized in Table III. The first column in this table identifies the scenarios summarized in Table II. The maneuvers are labeled m_1 through m_5 for the look-ahead levels $L = 1$ through 5, respectively. For each scenario, Table III shows the steps in the maneuver sequence in terms of turns and accelerations. Moreover, for each step the number of feasible maneuvers at each L is shown. Computation of maneuvers is also shown graphically in Figures 9 and 10. Figure 9 shows maneuvers considered by the Q^2 algorithm of the "self" aircraft. Figure 10 shows the maneuvers selected by the Q^2 algorithm out of the possible maneuvers at each of the look-ahead steps.

C. Computation Time

The simulations were implemented in Matlab and thus the computation times for both algorithms were appropriately

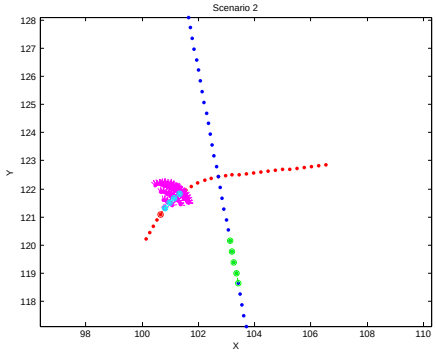

Fig. 9. Maneuvers considered by the Q^2 algorithm for Scenario 2.

TABLE III
EXAMPLES OF MANEUVERS.

Scen		m_1	m_2	m_3	m_4	m_5
1	ω [°]	2	2	2	2	2
	a [knots/min]	0	0	0	0	0
	No. of choices	5	42	354	3,010	25,973
2	ω [°]	-5	0	0	0	0
	a [knots/min]	0	135	135	90	90
	No. of choices	3	34	411	5,280	72,176
3	ω [°]	-4	0	0	0	0
	a [knots/min]	0	0	-45	0	90
	No. of choices	1	6	75	1,154	18,242
4	ω [°]	6	2	2	2	1
	a [knots/min]	0	0	0	0	0
	No. of choices	3	33	186	824	7,123

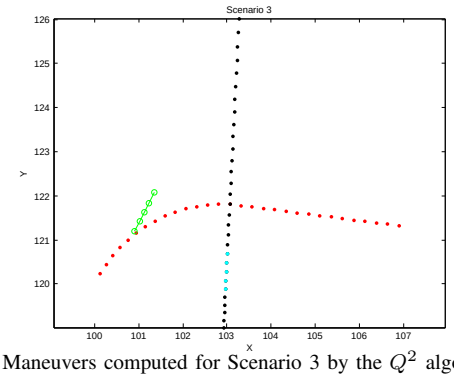

Fig. 10. Maneuvers computed for Scenario 3 by the Q^2 algorithm.

higher than if these algorithms were implemented in, say, C. However, the Matlab implementation was intended to show the relative differences in the computation times of search using the original space and the abstractions. The computer used for this task was Dell Pentium R 4, CPU 3.20GHz, 1.00GB of RAM. The PC version of the MATLAB software was version 6.5.1.

During each run of a scenario in Matlab, the computation time in seconds was recorded. The times that are needed to generate the maneuvers by the Q^2 algorithm are included in Table IV. From the table it is seen that the computation time is scenario-dependent - some scenarios cost more float point operations than others. Overall, the average cost for a 5-step computation was about 854 seconds.

The computation times of search in the original space are given in Table V. As can be seen from this table, it takes about fifteen and a half hours to compute the 5-step prediction for Scenario 4 and more than 30 hours to compute a 5-step prediction for the other three scenarios. Since scenarios 1, 2 and 3 have not finished within 30 hours, we abandoned these simulations. Comparing Tables IV and V, we can see that the search in the original space takes roughly two orders of magnitude more time than the search that uses the Q^2 approach. It is reasonable to expect that for the number of look-ahead levels more than five, the difference would be even

TABLE IV
TIME IN SECONDS OF GENERATING MANEUVERS FOR LOOK-AHEAD UP TO 5 USING THE Q^2 METHOD.

Scen	$L=1$	$L=2$	$L=3$	$L=4$	$L=5$	Q^2 total
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
4	0.095	0.233	1.094	6.5	57.4	65.3

TABLE V
COMPUTATION TIME TO GENERATE MANEUVERS FOR 5 LOOK-AHEAD STEPS USING SEARCH IN THE ORIGINAL SPACE.

Scen	L=1	L=2	L=3	L=4	L=5
1	0.219	2.157	44.235	13383	$>$ 30 hours
2	0.125	1.969	57.719	13426	$>$ 30 hours
3	0.344	2.688	54.532	13433	$>$ 30 hours
4	0.266	2.438	74.344	11293.7	15.5 hours

greater. However, we were not able to run the full-space search algorithm for more than five levels.

D. Quality of Resolution Advisories

In this investigation we defined the function $g()$, shown in Equation 3, as a measure of the quality of a sequence of maneuvers generated by a system in response to a conflict alert. A maneuver is better than another when it is less destructive to the flight plan, i.e., the derived path is closer to the flight plan than that of the other maneuver. In other words, the maneuver is better when the delay caused by the maneuver(s) is shorter, i.e., the integral of the discrepancy between the original path and the derived path over time is smaller than that for another maneuver. Therefore, the smaller the cost function, the better the performance, as long as all the constraints are satisfied.

The values of the cost function $g()$ averaged over all the maneuver steps for the four scenarios are shown in Table VI. These values represent the average deviation from the planned trajectory, in nautical miles. The second column shows the values for the Q^2 method, while column three shows the results for the full-space search. The last column shows the difference between the two methods.

As can be seen in the table, the values of the cost function for the Q^2 approach are somewhat higher. This was expected, since the Q^2 method does not analyze all of the possible maneuvers and thus the solutions provided by this method are not optimal. Note that, as described in Section VI-B, the Q^2 algorithm always picks a new qualitative input based upon the heuristic represented by the ordering function (see Equation 49). This heuristic implements a greedy policy, i.e., the next maneuver must be closer to the final (safe) state. Such a heuristic is not always optimal. For instance, in control theory the control input is not necessarily a monotonic function of time, but instead, the control signal is an overshot and then it is diminished when the system gets closer to the desired state.

However, the price for the increased speed of the computation is not that high. On the average, for the Q^2 approach, the average deviation from the planned path in each step is about 0.2 to 0.4 nautical miles, while the "best possible maneuvers" (using the full-space search method) are, on average, 0.15 nautical miles away from the flight plan. It is worth noting that the radar measurement noise is equivalent to 0.05 nautical miles, i.e., the position readings are within ± 0.05 nautical miles. The differences between the two methods are not within the error margin but still are relatively small.

TABLE VI
VALUES OF THE AVERAGE COST FUNCTION g/L (IN NAUTICAL MILES).

Scen	Q^2 method	Full-space Search method	Difference
1	0.2178	0.1584	0.0594
2	0.4264	0.1155	0.3109
3	0.3414	0.0816	0.2898
4	0.4412	0.1422	0.2990

VIII. CONCLUSIONS

First, it was shown that the problem of generating horizontal conflict resolution advisories has a prohibitively high computational cost. Three reasons for this were identified - a large size of the original solution search space, a need to analyze all possible maneuvers of the "intruder" aircraft in response to each maneuver of the "self" aircraft and a need for multi-step prediction of potential maneuvers by the aircraft. To ameliorate this problem, the use of qualitative abstractions based on the Q^2 method was proposed. In this approach, the system's output space is partitioned by hypersurfaces derived from the problem specification. Then qualitative partitions of the state and input spaces are derived by inverse mappings of the output space partitions through the output and state transition functions, respectively. Then an automaton is constructed to represent the dynamic behavior of the system. The derivation of maneuver decisions is then based on the analysis of the automaton by searching possible sequences of qualitative inputs to the automaton.

In the selected air traffic control scenario, the specification of the problem is provided by the definitions of the four alerts: PROCON, LINCON, MANCON and SAFE. In our approach we used these specifications for defining the semantics of qualitative states (a different semantics was used for a similar idea in [40]). Following the Q^2 approach, first a model of the dynamical system was provided and the problem was formulated as an optimization problem. Then qualitative partitions of the system space were analyzed. The conclusion was that PROCON and LINCON can be fully defined by qualitative partitions of the output space of the dynamical system. However, MANCON can be represented only in the Cartesian product of input, state and time. Thus the Q^2 approach had to be extended to cover the case when not all the partitions are defined fully within the output space.

To address this problem, a multi-step approach was proposed. In the first step an automaton (a Moore machine) modeling PROCON, LINCON and OTHER (that includes both SAFE and MANCON) is derived following the Q^2 approach. Then the input space (Cartesian product of input, state and time) is partitioned into qualitative inputs and another automaton is constructed. It is a Mealy machine, having outputs associated with transitions. In the next step, the Mealy machine is converted to an equivalent Moore machine, which is subsequently used for reasoning about sequences of conflict resolution advisories. The reasoning algorithm proposed in this paper is rather straight-forward, however all of the steps are intimately linked to the representation construction steps. First, the current qualitative state needs to be determined, then the desired final state (SAFE) is selected and a sequence of qualitative inputs is searched for such that satisfies some quality conditions. Finally, quantitative inputs are selected from the selected regions of the inputs.

To assess the impact of using Q^2 abstractions, a number of conflict scenarios involving two aircraft were simulated in Matlab and used for assessing the computation cost of exhaustive search with and without the use of the abstractions (to extend our approach to more aircraft one might resort to the method of decomposing the problem into pair-wise configurations proposed in [41]). Moreover, the utility of the proposed abstractions was assessed in terms of a quality metric. The conclusion was that the maneuver computation time with the use of Q^2 abstractions is orders of magnitude smaller than for the search in the original solution space. Although it was still too long for a CPU of the assumed computational speed of one millisecond per one analysis, it would be easy to satisfy by using a faster CPU.

In terms of quality of solutions found, as expected, the full-space search provided conflict resolution advisories that resulted in paths that were closer to the planned path and thus were somewhat better than the ones produced by the Q^2 approach. However, the difference between the solutions provided by the two algorithms seems to be small enough to justify the use of the Q^2 approach due to its computational feasibility.

In summary, this paper provided a novel, systematic approach to the use of abstraction to speed up the computation of multi-step conflict resolution advisories. This approach could be incorporated into practical algorithms for computing conflict resolution advisories that include horizontal maneuvers. Obviously, to make such an important decision would require further studies and more experimentation. To achieve this objective, we had to expand the Q^2 approach in significant ways. The method is generic since it is expressed in terms of general dynamical systems and automata. Thus this method could be relatively easy to apply to any scenario where problem specifications can be interpreted in terms of different qualitative partitions of output, state and input spaces of a dynamical system.

ACKNOWLEDGMENTS

The authors would like to thank all the anonymous reviewers for their detailed, insightful comments.

REFERENCES

[1] A. Polychronopoulos, M. Tsogas, A. J. Amditis, and L. Andreone, "Sensor fusion for predicting vehicles path for collision avoidance systems," *IEEE Transactions On Intelligent Transportation Systems*, vol. 8, No. 3, pp. 549–562, 2007.

[2] "Traffic Alert and Collision Avoidance System (TCAS II)," Competency Standards for TCAS II Operations and Aeronatical Knowledge Syllabus of Training,FAA, Tech. Rep.

[3] "Introduction to TCAS II," United States Department of Transportation, Federal AviationAdministration,Washington D.C, Tech. Rep., 1990.

[4] A. Inselberg, "Conflict detection and planar resolution for air traffic control," in *IEEE Intelligent Transportation Systems Conference Proceedings*. IEEE, 2001, pp. 1200–1205.

[5] F. Giunchiglia and T. Walsh, "A theory of abstraction," *Artificial Intelligence*, vol. 57, pp. 323–389, 1992.

[6] M. M. Kokar, "On Consistent Symbolic Representations of General Dynamic Systems," *IEEE Transactions on Systems, Man and Cybernetics*, vol. 25, no. 8, pp. 1231–1242, 1995.

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

[8] V. N. Duong, "FREER: Free-Route Experimental Encounter Resolution - Initial Results," in *EUROCONTROL Experimental Centre EEC*, 1997.

[9] N. Durand and J.-M. Alliot, "Optimal Resolution of En Route Conflicts," in *Laboratoire d'Optimisation Globale*, 1997.

[10] 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.

[11] W. Gerling, "Conflict Detection in Air Traffic Control," in *Institut fur Flugfuhrung*, 1994.

[12] J. Hu, M. Prandini, A. Nilim, and S. Sastry, "Optimal Coordinated Maneuvers for Three Dimentional Aircraft Conflict Resolution," in *AIAA Conf. Guidance, Navigat. Contr*, ser. 2001-4294, 2001.

[13] J. Kosecka and C. Tomlin, "Generation of Conflict Resolution Maneuvers for Air Traffic Management," Department of Electrical Engineering and Computer Sciences, Tech. Rep., 1996.

[14] J. Krozel and M. Peters, "Strategic Conflict Detection and Resolution for Free Flight," in *Proceedings of the 36th Conference on Decision & Control*, 1997.

[15] S. Mondoloni and S. Conway, "An airborne conflict resolution approach using a genetic algorithm," CSSI Inc. and NASA Langley Research Center, Tech. Rep.

[16] R. A. Paielli and H. Erzberger, "Conflict Probability Estimation for Free Flight," in *Journal of Guidance, Control and Dynamics*, 1997.

[17] W. D. Pekela and B. Hilburn, "Air Traffic Controller Stategies in Resolving Free Flight Traffic Conflicts: The Effect of Enhanced Controller Displays for Situation Awareness," Publication of Society of Automotive Engineers, Inc., Tech. Rep., 1997.

[18] R. Schild, "Rule Optimization for Airborne Aircraft Separation," Ph.D. dissertation, Institute of Econometrics, Operations Research and System Theory at the Technical University of Vienna, 1997.

[19] J. Shen, N. H. McClamroch, and E. G. Gilbert, "A Computational Approach to Conflict Detection Problems," in *Proceedings of the American Control Conference*, 1999.

[20] C. 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.

[21] B. C. Breen, "Controlled Flight Into Terrain and the Enhanced Ground Proximity Warning System," *Aerospace and Electronic Systems Magazine, IEEE*, vol. 14, no. 1, 1991.

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

[23] L. Pallottino and A. Bicchi, "On the optimal conflict resolution for air traffic control," in *IEEE Intelligent Transportation Systems Conference Proceedings*. IEEE, 2000, pp. 167–172.

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

[25] A. Visintini Lecchini, W. Glover, J. Lygeros, and J. Maciejowski, "Monte Carlo Optimization for Conflict Resolution in Air Traffic Control," *IEEE Transactions on Intelligent Transportation Systems*, vol. 7, No. 4, pp. 470–482, 2006.

[26] J. K. Kuchar and L. C. Yang, "Survey of Conflict Detection and Resolution Methods," in *AIAA Guidance, Navigation and Control Conference*, 1997.

[27] ——, "A review of Conflict Detection and Resolution Modeling Methods," *IEEE Transactions on Intelligent Transportation Systems*, vol. 1, no. 4, pp. 179–189, 2000.

[28] N. H. McClamroch and B. Sridhar, "Guest editorial: Special issue on automated air traffic control systems," *IEEE Transactions On Intelligent Transportation Systems*, vol. 2, No. 2, pp. 37–38, 2001.

[29] M. Prandini, J. Hu, J. Lygerors, and S. Sastry, "A probabilistic approach to aircraft conflict detection," *IEEE Transactions On Intelligent Transportation Systems*, vol. 1, No. 4, pp. 199–220, 2000.

[30] J. Hu, M. Maria Prandini, and S. Sastry, "Aircraft conflict prediction in the presence of a spatially correlated wind field," *IEEE Transactions On Intelligent Transportation Systems*, vol. 6, No. 3, pp. 326–340, 2005.

[31] "Common ARTS design specification," FAA, Tech. Rep., 2003.

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

[33] 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*, vol. 16, no. 2/3, pp. 155–186, 1999.

[34] J. W. S. Liu, *Real-time systems*. Upper Saddle River, N.J.: Prentice Hall, 2000.

[35] P. Tabuada, G. Pappas, and P. Lima, "Compositional Abstractions of Hybrid Control Systems," in *Proceedings of the 40th IEEE Conference on Decision and Control, 2001.*, 2001, pp. 352 – 357.

[36] Z. Bavel, *Introduction to the theory of automata*. Reston, A Prentice-Hall Company, 1983.

[37] F. Wagner, "StateWorks: Moore or Mealy model?" http://www.stateworks.com/technology/TN10-Moore-Or-Mealy-Model/, 2005.

[38] J. E. Hopcroft and J. D. Ullman, *Introduction to automata theory, languages and computation*. Addison-Wesley, 1979.

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

[40] C. Tomlin, I. Mitchell, and R. Ghosh, "Safety verification of conflict resolution maneuvers," *IEEE Transactions On Intelligent Transportation Systems*, vol. 2, No. 2, pp. 110–120, 2001.

[41] K. Treleaven and Z.-H. Mao, "Conflict resolution and traffic complexity of multiple intersecting flows of aircraft," *IEEE Transactions On Intelligent Transportation Systems*, vol. 9, No. 4, pp. 633–643, 2008.

Mei Li received the Ph.D. in Electrical and Computer Engineering from Northeastern University in 2007. She works for ARCON Corporation in the field of Air Traffic Control. Her areas of interest include sensor fusion, radar to ADS-B registration, IMM tracker performance, safety function performance and multilateration integration with the STARS automation system.

Mieczyslaw M. Kokar (M'86 – SM'03) is with the Department of Electrical and Computer Engineering of Northeastern University in Boston, USA. He holds an M.S. (1969) and a Ph.D. (1973) degree in computer and system engineering, both from the Wroclaw University of Technology. His technical research interests include information fusion, ontology-based information processing, cognitive radio and self-controlling software. Professor Kokar teaches various graduate courses in software engineering, formal methods and artificial intelligence.