Anticipatory Runway Incursion Prevention Systems

Kai SHI††††, Member, Yuichi GOTO†, Zhiliang ZHU††, and Jingde CHENG†, Nonmembers

SUMMARY Avoiding runway incursions is a significant challenge and a top priority in aviation. Due to all causes of runway incursions belong to human factors, runway incursion prevention systems should remove human from the system operation loop as much as possible. Although current runway incursion prevention systems have made big progress on how to obtain accurate and sufficient information of aircraft/vehicles, they cannot predict and detect runway incursions as early as experienced air traffic controllers by using the same surveillance information, and cannot give explicit instructions and/or suggestions to prevent runway incursions like real air traffic controllers either. In one word, human still plays an important position in current runway incursion prevention systems. In order to remove human factors from the system operation loop as much as possible, this paper proposes a new type of runway incursion prevention system based on logic-based reasoning. The system predicts and detects runway incursions, then gives explicit instructions and/or suggestions to pilots/drivers to avoid runway incursions/collisions. The features of the system include long-range prediction of incidents, explicit instructions and/or suggestions to pilots/drivers, and flexible model for different policies and airports. To evaluate our system, we built a simulation system, and evaluated our system using both real historical scenarios and conventional fictional scenarios. The evaluation showed that our system is effective at providing earlier prediction of incidents than current systems, giving explicit instructions and/or suggestions for handling the incidents effectively, and customizing for specific policies and airports using flexible model.

key words: runway incursion prevention, logic-based reasoning, anticipatory computing, anticipatory reasoning-reacting system

1. Introduction

Runway incursions are occurrences at an aerodrome involving the incorrect presence of an aircraft, vehicle or person on the protected area of a surface designated for the landing and take-off of aircraft [1]. Runway incursions have the potential to cause serious accidents with significant loss of life. The world’s deadliest aviation accident was the result of a runway incursion [2]. In the United States, an average of three runway incursions occur daily [2]. Avoiding runway incursions has been a top ten priority for the National Transportation Safety Board for over a decade [3].

Ideal runway incursion prevention systems should replicate the abilities of experienced air traffic controllers. The causes leading to runway incursions include pilot deviations, operational errors/deviations, and vehicle/pedestrian deviation [3]. All the causes belong to human factors. Traditionally, to prevent runway incursions, both pilots and controllers rely on visual cues, occasional communications by radio, and their memories [4]. In recent years, people invented several modern devices and systems to decrease the chance of runway incursions [4]. Because all causes of runway incursions belong to human factors, runway incursion prevention systems require removing human from the system operation loop as much as possible [3]. Therefore, ideal runway incursion prevention systems should replicate the abilities of experienced air traffic controllers, i.e., the system should not only predict and detect a runway incursion but also give explicit instructions and/or suggestions to the pilots/drivers to avoid the runway incursion/collision.

Although several runway incursion prevention systems have been developed, current systems are only assistant tools for human, and still far from ideal. First, current systems cannot always predict/detect a runway incursion as early as an experienced air traffic controller predict/detect by using the same surveillance information, although they have made big progress on obtaining accurate and sufficient information of aircraft/vehicles. Second, current systems lack the ability to give explicit instructions and/or suggestions to prevent runway incursions. A runway incursion prevention system should have such an ability, because (1) both the air traffic controllers and pilots/drivers need time to make decisions about instructions to prevent the incursion, although alert of incursion is given, and (2) at a critical moment, human may make mistakes due to overstrain, but machine do not. Third, current systems do not consider the special conditions of a certain airport and different areas of that airport. In contrast, experienced air traffic controllers can use such information for better prediction/detection.

In order to remove human factors from the system operation loop as much as possible, this paper proposes a new type of runway incursion prevention system, named “anticipatory runway incursion prevention system”, which can predict and detect runway incursions, and then give explicit instructions and/or suggestions to pilots/drivers to avoid the runway incursion/collision. Our system provides: (1) long-range prediction: to predict runway incursions earlier than current runway incursion prevention systems for reserving more time to handle the incidents, (2) explicit instructions and/or suggestions: not only to give alerts to air traffic controllers/pilots/drivers, but also to give explicit instructions and/or suggestions to pilots/drivers to avoid the incursions/collisions, and (3) flexible model: to use flexible customized anticipatory model for different air traffic control.
policies and airports. Our system uses logic-based reasoning [5, 6] for prediction and decision making. The core of the system architecture is anticipatory reasoning-reacting system [7]. To evaluate our system, we built a simulation system, and evaluated our system using both real historical scenarios and conventional fictional scenarios. The evaluation showed that (1) our system uses flexible model, which can be customized for different airports and air traffic control policies, (2) our system could provide earlier prediction of incidents than current systems, and (3) our system could give explicit instructions and/or suggestions for handling the incidents effectively.

2. Problems of Current Runway Incursion Prevention Systems

A number of systems for runway incursion avoidance and detection were proposed, such as runway incursion prevention system (RIPS) [8], airport surface detection equipment - model X (ASDE-X) [9], runway incursion monitoring and collision avoidance system (RIMCAS) [10], runway status light system (RWSL) [11], airport movement area safety system (AMASS) [12], and advanced surface movement guidance and control system (A-SMGCS) [13, 14].

Current runway incursion prevention systems focus on situational awareness, collision prediction, and runway incursion detection. Figure 1 shows a general architecture of current runway incursion prevention system [3]. Traffic information system and traffic surveillance sensor system collect data from a variety of sources, then generate and provide information of aircraft and vehicles, such as location, speed, as well as identification information. Situation assessment system detects an ongoing runway incursion or predicts a forthcoming collision based on the information of aircraft and vehicles, then gives alerts and warning to human machine interface system and airport traffic signal system. Human machine interface system gives traffic situation, alerts, and warnings about runway incursions or collisions. Besides, air traffic controllers also use the human machine interface system as an input device. Airport traffic signal system gives alerts, warnings, and ATC-commands to pilots/drivers by using lights or sound through different systems deployed in the airport. In current systems, alerts and warnings are generated by the situation assessment system, and all ATC-commands are generated by air traffic controllers.

We investigated and analyzed the problems of current runway incursion prevention systems, then specified what these problems are, and why those problems should be solved or why these problems arise.

- Tardy prediction/detection
  Although current systems can obtain accurate and sufficient information of aircraft/vehicles, they cannot always predict/detect a runway incursion as early as an experienced air traffic controller predict/detect by using the same surveillance information. There are two reasons arising tardy prediction/detection. First, current systems focus on detecting an ongoing runway incursion, or predicting a forthcoming collision [3], but not focus on predicting a forthcoming runway incursion. Due to the high speed of aircraft, when a runway incursion has occurred, there may not be sufficient time to manage the incident. Second, current systems use inflexible algorithms/models, and cannot flexibly utilize the empirical knowledge of air traffic controllers and the concrete conditions of the airport [15]. For example, in RIMCAS, only two aircraft simultaneously enter a “critical circle”, an alert can be triggered [15]. A related incident will be shown in Sect. 4.9 later. In contrast, a concentrated experienced air traffic controller could utilize the layout of runways to predict that runway incursion long before the two aircraft enter the “critical circle”.
- Lack of explicit instructions and/or suggestions
  Current systems only give alerts, but cannot give explicit instructions and/or suggestions to pilots/drivers to avoid the incursions/collisions. Although air traffic controllers/pilots/drivers may be aware of an ongoing/forthcoming incident, they have to make correct decisions to manage the incidents in very limited spans of time. To make the decisions also consumes time, besides, in an extreme emergency, people tend to relay on instinct instead of rationality.
- High frequency of missed alerts and false alerts
  Several current systems suffer high frequency of missed alerts and false alerts [16]. Missed alerts and false alerts is a direct result of erroneous or missing traffic data [8]. Besides, inflexible algorithms/models may lose their effectiveness for some special situations and airports [15]. Furthermore, numerical calculation approaches cannot distinguish between a severe accident and a trivial incident, which causes high frequency of unnecessary alerts.
- Inflexible algorithms/models
  Most of current system use inflexible algorithms/
models to predict/detect incursions/collisions. Inflexible algorithms/models cannot adapt to the needs of customization and variability, i.e., they cannot flexibly utilize the empirical knowledge of air traffic controllers and the concrete conditions of the airports. Besides, airport operations are complex and vary for different policies and different airports (even in different areas of one airport). Therefore, we should not use a single general algorithm/approach for different policies and airports (even in different areas of one airport). Furthermore, as we discussed above, inflexible algorithms/models could cause both tardy prediction/detection and missed/false alerts. Although some approaches based on artificial intelligence may use flexible models, such as [17], they did not show their effectiveness and efficiency, as well as practical use in real systems.

In this paper, our aim is not to solve all these problems completely, but only to focus on (1) earlier prediction/detection of runway incursions/collisions than current systems by using the same processed information of aircraft/vehicles, such as position, speed, and acceleration, (2) facility to give explicit instructions and/or suggestions to avoid the incursions/collisions, and (3) flexible models. Although it is important for runway incursion prevention systems to get accurate and sufficient surveillance information of aircraft/vehicles, such as position, speed, and acceleration, out of noisy surveillance data in time, the problems about how to get accurate and sufficient surveillance information in time and how to work under missed/false surveillance information are beyond the scope of this paper. On the other hand, almost all current systems focus on aircraft/vehicles surveillance, and have made big progress on this topic.

3. Anticipatory Runway Incursion Prevention Systems

This paper proposes a new type of runway incursion prevention system, named “anticipatory runway incursion prevention system (ARIPS for short)”, which is a runway incursion prevention system not only predict and detect runway incursions/collisions as early as an experienced air traffic controller does, but also give explicit instructions and/or suggestions to pilots/drivers to avoid runway incursions/collisions. Due to the problem of inflexible algorithms/models and its consequences, an ARIPS uses flexible models, which can be customized for different air traffic control policies and airports.

We analyzed the system requirements of the ARIPS as follows.

R1 The ARIPS should predict/detect a runway incursion/collision as early as an experienced air traffic controller does.

R2 The ARIPS should give alerts to both air traffic controllers and related pilots/drivers. Besides, the ARIPS should give alerts with different emergency levels, such as advisory, watch, and warning to specify an alert from trivial to severe. System users can choose to turn off trivial alerts to focus on the critical incidents.

R3 Besides alerts, the ARIPS should also give explicit instructions and/or suggestions to pilots/drivers to avoid the incursions/collisions.

R4 The ARIPS should be a general one, which can be deployed in different airports with trivial modification or configuration.

R5 The ARIPS should use flexible models, which can be customized for specific air traffic control policies and airports, as well as different areas of a specific airport. Besides, the models of ARIPS should express the knowledge of air traffic control in an explicit way, thus the domain experts can customize, modify, and examine these models easily.

R6 The ARIPS should have low frequency of missed alerts, false alerts, false instructions and/or false suggestions.

4. An Implementation Based on Anticipatory Reasoning-Reacting System

4.1 Overview

An ARIPS contains several sub-systems and several components shown in Fig. 2, while in this paper, our aim is not to improve current traffic information systems, traffic surveillance sensor systems, human machine interface systems, or airport traffic signal systems, but rather only focus on the core components of ARIPS used for prediction/detection of runway incursions/collisions and decision-making about instructions and/or suggestions. From view point of system architecture, there are two main differences between the architecture of current systems[3] shown in Fig. 1 and that of ARIPS shown in Fig. 2. First, the situation assessment sys-
tem of current systems focus on both providing traffic situation and alerts/warnings about incursions/collisions, while ARIPS’s predictor only focus on give prediction/detection of incursions/collisions. Second, ARIPS has a component for decision-making about instructions, but current systems do not have. For other components, there is no essential difference between ARIPS and current systems. In ARIPS, alerts, instructions, and/or suggestions are generated by the system, indeed the core components, but not human. Based on the instructions and/or suggestions, air traffic controllers can also give ATC-commands.

4.2 Technical Approach

To satisfy requirements R4 and R5, we adopted logic-based reasoning [7] as the mechanism to predict/detect runway incursions/collisions and to make decisions about instructions and/or suggestions for avoiding the incursions/collisions. The basic idea of logic-based reasoning method used in ARRSs is to explicitly separate the underlying logical system, reasoning/computing mechanism, and empirical knowledge in any prediction/decision making, such that both underlying logical system and empirical knowledge can be revised/replaced/customized in various predictions/decision making processes performed by an area-independent, task-independent, general-purpose reasoning mechanism [5], [6].

For system architecture, we adopted anticipatory reasoning-reacting system (ARRS)[7] as the core of ARIPS. An ARRS uses logical reasoning to predict and make decisions [7]. Moreover, an ARRS consists of traditional reactive systems and core components including predictor, decision maker, and filter. Such architecture is fit to embrace other different reactive systems such as traffic information systems, traffic surveillance sensor systems, human machine interface systems, and airport traffic signal systems. Due to space limitations, this paper only briefly presents mechanisms of prediction and decision making of ARIPS based on ARRS. Refer to [18], [19] for the architecture and working process of ARRSs.

4.3 System Architecture

Figure 2 shows an architecture of ARRS-based ARIPS, which includes the following components: traffic information system, traffic surveillance sensor system, filter, predictor, decision maker, databases (LTDB, ETDB, and RDB), human machine interface system, and airport traffic signal system. Traffic information system provides local traffic information for the participating users [3]. Traffic surveillance sensor system provides information about the environment, which is usually preprocessed into a compact meaningful representation by the sensors [3]. These sensors can be part of an airport infrastructure or part of an aircraft and/or other mobile units [3]. Filter filters out the trivial information of aircraft/vehicles and generates the qualitative information of aircraft/vehicles as current situation for predic-

tor and decision maker. Predictor receives current situation from the filter, then outputs predictions about runway incursions, collisions, etc. Decision maker receives predictions from the predictor, current situation from the filter, and aircraft’s/vehicles’ real time status from the traffic surveillance sensor system, then outputs instructions and/or suggestions for avoiding runway incursions or collisions. LTDB is a logical theorem database, which stores fragments of logical systems [7]. ETDB is an empirical theory database storing anticipatory model, including world model, predictive model, and behavioral model. RDB is a database storing transform rules, calculative rules, interesting formula definitions, and interesting terms. Human machine interface system gives the alerts about runway incursions or collisions, as well as the instructions and/or suggestions for avoiding runway incursions or collisions to both pilots/drivers and air traffic controllers. Besides, air traffic controllers also use the human machine interface system as an input device. Airport traffic signal system gives instructions to pilots/drivers by using lights or sound through different systems deployed in the airport.

4.4 Filtering

Because the ARIPS uses logic-based reasoning method to predict and make decisions, the predictor and decision maker only need qualitative information of aircraft/vehicles. The most useful information for ARIPS is aircraft’s/vehicles’ locations, as well as their statuses, such as whether an aircraft is accelerating.

The filter consists of a set of filter modules, while each module filters out a certain kind of sensory information. The filter’s locating module filters out qualitative location information of aircraft/vehicles, including past locations and current locations. We divide an airport including its surrounding airspace into different regions, and name each region with a unique name, shown in Fig. 3. The locating module represents each region as a polygon, i.e., a set of vertices. Both traffic information system and traffic surveillance sensor system could provide the location of an aircraft/vehicle by a point, i.e., a pair of coordinate. The locating module calculates which region each air-
craft/vehicle locates, by calculating whether a point is inside a given polygon. The locating module maintains two hash maps: PastLocation<aircraft/vehicle identifier, locating region> and CurrentLocation<aircraft/vehicle identifier, locating region>. When an aircraft/vehicle move from one region to another one, the locating module uses its location stored in CurrentLocation to overwrite its location in PastLocation, then use the new current location to overwrite its location in CurrentLocation. To filter out aircraft’s status information, the filter’s status module gets acceleration of aircraft from traffic surveillance sensor system, e.g., an aircraft’s accelerometer could provide the aircraft’s acceleration. If an aircraft’s acceleration is greater than a given value, the filter modules will not always output filtered information, the filter’s status module gets acceleration of the aircraft in CurrentLocation. To filter out aircraft’s status Location, then use the new current location to overwrite its location in PastLocation to another one, the locating module uses its location.

4.5 Predicting

The purpose of prediction of ARIPS is to find out “which aircraft/vehicles will cause runway incursions/collisions?”, “which aircraft will be affected by the runway incursions/collisions?”, and “where the runway incursions/collisions will occur?”

Figure 4 shows the data flow diagram of the predictor. First, formula generator translates the filtered current situation to logical formulas according to transformation rules, which base on the predicate dictionary. Second, the forward reasoning engine gets input of these logical formulas, reasoning. Third, the formula chooser chooses predictions according to the interesting formula definitions and interesting terms from all formulas reasoned out by forward reasoning engine, and transfers them to decision maker and human machine interface system.

Because the ARIPS uses logic-based reasoning method to predict and make decisions, we need to choose a logical basis to underlie temporal reasoning and deontic reasoning. Therefore, we chose temporal deontic relevant logics, [20] as the logic basis to both predict and make decisions.

A world model is a set of empirical theories represented by logical formulas in the target domain except empirical theories related with time and behavior [18]. The world model has two functions: to represent the static conditions of the airport, such as topology of runways and taxiways, and to represent essential empirical knowledge except related with time and behavior, such as a piece of knowledge for judging whether an aircraft is taking off.

A predicate vocabulary specifies a vocabulary to represent the status of the real world, shown in Table 1. By using the predicate vocabulary, we can represent the airport diagram as a set of logical formulas, and represent the current situation as a set of logical formulas. We use Fig. 3 as an example to show how to represent the airport. First, we divide an airport including its surrounding airspace into different regions, and name each region with a unique name, i.e., a constant symbol, shown in Fig. 3. Second, we use a formula constructed by a predicate and the constant symbol to represent what is that region, e.g., Runway(18L) representing region 18L is a runway, Way(18L,D) representing that it is a section of way, and A,D representing that it is a section of way.

![Fig. 4 Predictor’s data flow diagram.](image)

Table 1 Predicate vocabulary.

| Formula             | Meaning                                                   |
|---------------------|-----------------------------------------------------------|
| Aircraft(o)         | o is an aircraft.                                         |
| Vehicle(o)          | o is a vehicle.                                           |
| Other(o)            | o is an object cannot communicate with air traffic controllers. |
| Runway(r)           | r is a runway.                                            |
| Taxiway(r)          | r is a taxiway.                                           |
| Intersection(r)     | r is an intersection.                                     |
| Way(r)              | r is a section of way.                                    |
| Approach(r)         | r is an area region on extension line of a runway, from the runway threshold to 2000 meters far away from the runway threshold. |
| HotSpot(r)          | r is a hot spot, which is defined as a location on an airport movement area with a history of potential risk of collision or runway incursion. |
| Active(r)           | Runway r is occupied for taking off or landing.            |
| Accelerate(o)       | o is notability accelerating.                             |
| At(o,r)             | o is in region r.                                         |
| RHyg(o,r)           | o is causing a runway incursion on runway r.              |
| Collision(o1,o2)    | o1 and o2 collide on the runway.                          |
| TakeOff(From o,r)   | Aircraft o is taking off from runway r.                   |
| LandOn(o,r)         | Aircraft o is landing on runway r.                        |
| OccupyIntersection | Aircraft o occupies two intersecting runways.             |
| (o,r1,r2)           | Runway r1 and r2 cross each other.                        |
| Intersecting(r1,r2) | Region r1 connects region r2.                             |
| C(r1,r2)            | Runway(r2) ∧ C(r1,r2) ∧ C(r2,r1) ∧ Way(r2)               |
| C2(r1,r2,r3)        | Runway(r3) ∧ C(r1,r2) ∧ C(r2,r3) ∧ C(r3,r2) ∧ Way(r2)     |
| C3(r1,r2,r3)        | Runway(r3) ∧ Approach(r2) ∧ C(r1,r2) ∧ C(r2,r3) ∧ Way(r2) |
is an intersection. Third, we use a formula constructed by a predicate and constant symbols to represent the relationship of different regions, e.g., $C(18L, A_{18L} D)$ representing region 18L and region $A_{18L} D$ connect with each other, and $C(2, A_{2} D, A_{18L} D, 18L)$ representing that region 18L is a runway, $A_{18L} D$ is a section of way, region $A_{2} D$ connects with region $A_{18L} D$, and region $A_{18L} D$ connects with region 18L. These formulas represent the topology construction of the airport, which include all geographic information that the system needs. To transform the situation information from the filter into logical formulas is also based on the predicate dictionary. Figure 5 shows how to transform situation information represented by strings into logical formulas. “$P$” is a temporal operator, while $PA$ meaning “it has been the case at least once in the past up to now that $A$”. Next step is to determine the common knowledge involving in the target problem and to represent the knowledge as conditionals. Specifically, the knowledge in world model does not include knowledge related with prediction and behavior. For example, “if an aircraft is accelerating on a runway, then that aircraft is taking off” can be written as: $\forall o \forall r (At(o, r) \land \text{Runway}(r) \land \text{Aircraft}(o) \land \text{Accelerate}(o) \Rightarrow \text{TakeOffFrom}(o, r))$ (WM1). “If a aircraft is taking off from a runway, then that runway is active” is written as $\forall o \forall r (\text{TakeOffFrom}(o, r) \Rightarrow \text{Active}(r))$ (WM2). “If a aircraft will land on a runway, then that runway is active” is written as $\forall o \forall r (\text{LandOn}(o, r)) \Rightarrow \text{Active}(r)$ (WM3). “$F$” is a temporal operator, while $FA$ meaning “it will be the case at least once in the future form now that $A$”. “If a aircraft is taking off from a runway besides that runway intersects with another runway, then that aircraft is occupying the two intersecting runways” is written as $\forall o \forall r_1 \forall r_2 (\text{TakeOffFrom}(o, r_1) \land \text{Intersecting}(r_1, r_2) \Rightarrow \text{OccupyingIntersection}(o, r_1, r_2))$ (WM4).

A predictive model is a set of empirical theories, which are represented by logical formulas and related with time in a target domain of the system [18]. The predictive model represents the predictive knowledge used to make prediction. Predictive knowledge is time-related. A piece of predictive knowledge specifies, that when a certain state (both past and current) occurred, some following state will be true in the future. In ARRSs, predictive knowledge is a set of conditionals whose consequent of the future is true if and only if the antecedent holds. Because for any prediction, both the predicted thing and its truth must be unknown before the completion of that prediction, the conclusion should not include the knowledge what we have known. Besides, the premises and conclusion should be relevant. Furthermore, we are only interested in certain kinds of predictions, while in ARIPS, they are mainly collisions, runway incursions, and other predictions, which can help to predict collisions and runway incursions. Here are some examples: $\forall o \forall r_1 \forall r_2 \forall r_3 (C(2, r_1, r_2) \land \text{H}(At(o, r_1)) \land \text{At}(o, r_2) \Rightarrow F(\text{At}(o, r_3)))$ (PM1), $\forall o \forall r (\text{At}(o, r) \land \text{Active}(r) \Rightarrow F(\text{Rilby}(o, r)))$ (PM2), $\forall o \forall r_1 \forall r_2 \forall r_3 (C(r_1, r_2, r_3) \land H(At(o, r_1)) \land \text{At}(o, r_2) \Rightarrow F(\text{LandOn}(o, r_3)))$ (PM3), $\forall o \forall o_2 \forall r (\text{TakeOffFrom}(o_1, r) \land F(\text{LandOn}(o_2, r)) \Rightarrow F(\text{Rilby}(o_2, r)))$ (PM4), $\forall o_1 \forall o_2 \forall r_1 \forall r_2 (\text{OccupiedIntersection}(o_1, r_1, r_2) \land \text{TakeOffFrom}(o_2, r_2) \Rightarrow F(\text{Rilby}(o_2, r_1)))$ (PM5), $\forall o_1 \forall o_2 \forall r (\text{TakeOffFrom}(o_1, r) \land \text{TakeOffFrom}(o_2, r) \Rightarrow F(\text{Collision}(o_1, o_2)))$ (PM6). PM1 means when an aircraft has crossed the hold line of a runway, that aircraft will arrive at that runway. PM1 is valid because aircraft cannot draw back, when an aircraft reach one end of a way, the aircraft will arrive at the other end. Because the above formulas are easy to understand, we do not explain them one by one.

The forward reasoning is performed by a forward reasoning engine, which is a program to automatically draw new conclusions by repeatedly applying inference rules to given premises and obtained conclusions until some previously specified conditions are satisfied [21]. In ARIPS, we chose FreeEnCal [21] as the forward reasoning engine.

The formula chooser chooses predictions from all formulas reasoned out by forward reasoning engine, and transfers them to decision maker and other components such as human machine interface systems. We call predictions interesting formulas (IF), which is defined as follows [19]: If $A$ is an interesting term, then $A$ is an IF; If $A$ is an interesting term, then $\sim A$ is an IF; If $A$ or $B$ is IFs, then $U(A, B)$ is an IF; If $A$ is an IF, then $\forall xA$ and $\exists xA$ are IFs; If $A$ is an IF, then $\Phi A$ is an IF, where $\Phi$ is one of $\{G, F\}$. The interesting terms are $\text{Rilby}$, $\text{Collision}$, $\text{TakeOffFrom}$, $\text{LandOn}$, $\text{OccupyIntersection}$, etc.

Alert generator gets the prediction and generates alerts with different emergency levels, such as $\text{Rilby}$ and $F(\text{Collision})$ having highest emergency level, and $F(\text{Rilby})$ taking second place.

4.6 Decision Making

There are two phases to choose instructions and/or suggestions for avoiding the runway incursions/collisions: first phase is to make qualitative decision by logic-based reasoning, and second phase is to refine the decision by quantitative calculation. Figure 6 shows the data flow diagram of the decision maker.

Qualitative decision: The progress of qualitative decision making by logic-based reasoning is similar with prediction by logic-based reasoning, where the main difference is to use behavioral model instead of predictive model. The purpose of qualitative decision is to find out “which
actions must be taken?” and “which actions should be taken?” Thus, the result of qualitative decision is a set of candidates of next actions, which are labeled as “obligatory”/“permitted” and/or priorities. A behavioral model is a set of empirical theories that are represented by logical formulas and related with behavior in a target domain of the system [18]. The behavioral model specifies that, when a certain event or state occurs, which actions must/should be taken, as well as when a prediction about certain event or state is made, which anticipatory actions must/should be taken. The behavioral model is assembled by conditionals, which can be used to get these results of qualitative decision. Here are some examples:

\[ \forall o \forall r (F(LandOn(o, r)) \land F(Riby(o, r)) \Rightarrow O(GoAround(o))) \]

\[ (BM1) \]

\[ \forall o_1 \forall o_2 (TakeOff(o_1, r)) \land F(Riby(o_2, r)) \Rightarrow O(Evade(o_1)) \]

\[ (BM2) \]

\[ \forall o \forall r (A(t(o, r)) \land F(Riby(o, r)) \Rightarrow O(Hold(o))) \]

\[ (BM3) \]

\[ \forall o_1 \forall o_2 (F(LandOn(o_1, r)) \land F(Riby(o_2, r)) \Rightarrow O(GoAround(o_1))) \]

\[ (BM4) \]

\[ \forall o_1 \forall r_1 \forall r_2 (TakeOff(o_1, r_1) \land F(Riby(o_1, r_2)) \Rightarrow O(Evade(o_1))) \]

\[ (BM5) \]

“○” is a deontic operator, while OA meaning “it is obligatory that A”. “GoAround(o)” is a go-around instruction that an aborted landing of an aircraft o that is on final approach. “Hold(o)” is a hold-in-position instruction that a taxiing aircraft/vehicle stops going forward, stays in the current position. “Evade(o)” is an instruction that an aborted taking off of an aircraft o as well as “see and void” a potential collision. To choose candidates of actions, the interesting formulas (IF) is defined as follows: If A is an interesting term, then A is an IF; If A is an interesting term, then \( \neg A \) is an IF; If A is an IF, then \( \forall A \) and \( \exists A \) are IFs; If A is an IF, then \( \phi A \) is an IF, where \( \phi \) is one of \{O, P\}. The interesting terms are GoAround, Evade, Hold, etc.

**Quantitative calculation:** In order to decide an instruction for real-time operation, quantitative calculation is necessary. For example, if the speed of aircraft o is greater than V1, then aircraft o cannot execute “Evade(o)”, because V1 is the maximum speed in the takeoff at which the pilot can take the first action (e.g., apply brakes, reduce thrust, deploy speed brakes) to stop the airplane within the accelerate-stop distance.

### 4.7 Databases

In our current implementation of ARIPS, the databases LTDB, ETDB, and RDB just store the corresponding data for initializing predictor and decision maker, while these databases do not participate in the phase of prediction/detection and decision-making. In the initial phase of ARIPS, predictor reads transformation rules, interesting formula definitions, and interesting terms from RDB, reads fragment of logical system from LTDB, and reads world model and predictive model from ETDB, while decision maker reads transformation rules, interesting formula definitions, interesting terms, and calculative rules from RDB, reads fragment of logical system from LTDB, and reads world model and behavioral model from ETDB. Both predictor and decision maker stores these data in their working space for the sake of efficiency. When any data in RDB, LTDB, and ETDB changes in run time, predictor and decision maker will updates their corresponding data.

### 4.8 Ad Hoc Methods for Efficiency

Traditionally, logic-based reasoning may be not so efficient and do not satisfy the requirement of time restriction for runway incursion prevention. Especially, the current FreeEnCal does not handle high degree of logic connectives/operators [21] efficiently.

To improve the performance of ARIPS, we adopted some ad hoc methods. First, the formula generator generates the conjunctions of premises, whose degree of “∧” greater than or equal to 3, then inputs these formulas as redundancy formulas into FreeEnCal. Second, we try to ensure the degree of “∧” of empirical conditional in anticipatory model is less than 3. Third, we use multi-processing/multi-threading of FreeEnCal in both predictor and decision maker. When the situations of aircraft/vehicle change, predictor/decision maker creates a new process/thread of FreeEnCal for reasoning. Therefore, there may be several forward reasoning process/thread running simultaneously in predictor or decision maker. Multi-process/thread of FreeEnCal may deduce redundant results. However, ARIPS just output these redundant predictions, instructions, and/or suggestions one by one, which do not affect the correctness of alerts, instructions, and/or suggestions.

### 4.9 System Mechanism

In order to explain the mechanism of ARIPS, we use a real incident scenario to show the working process of ARIPS. The incident occurred on 18 June 2010 at Zurich airport [15], shown in Fig. 7. At 12:00:30 (UTC) the aircraft THA971 received clearance to taxi to the take-off position on runway 16. At 12:01:31 the aircraft BCI937 received
clearance to taxi to the take-off position on runway 28. The aircraft BAW713 was ready to depart at holding position point B to the north of the threshold of runway 28. At 12:02:26 the crew of THA971 received clearance to take off from runway 16; they acknowledged it immediately and initiated the take-off roll. At 12:02:31 the crew of BCI937 initiated their take-off roll on runway 28 (Ideally, air traffic controller could notice the runway incursion at this moment). At 12:02:47, the crew of BAW713 informed the air traffic controller could notice the runway incursion at this moment. The air traffic controller instructed the crew of BCI937 to abort take-off. At 12:02:50 the air traffic controller instructed the crew of BCI937 to abort take-off. The crew obeyed this instruction and vacated runway 28 on taxiway A4. The crew of THA971 continued their take-off and flight to their destination. At 12:03:01, the RIMCAS generates a stage 2 alert.

How ARIPS works under this scenario is as follows. In the initial phase of ARIPS, predictor and decision maker reads the anticipatory model and other data from LTDB, ETDB, and RDB, as shown in Sect. 4.7. The anticipatory model contained all empirical theorems shown in Sect. 4. Besides, the world model also contained static conditions of the airport, such as Runway(28), Runway(16), and Intersecting(28, 16). After initialization, the filter continuously receives each aircraft’s location, speed, and acceleration from traffic information system and traffic surveillance sensor system, and filter out the qualitative information of aircraft for predictor and decision maker. At 12:02:31, the filter filtered out Aircraft(THA971), Aircraft(BCI937), At(THA971, 16), At(BCI937, 28), Accelerate(THA971), and Accelerate(BCI937). Once filter updates information, predictor uses this information for prediction, while decision maker uses this information for decision-making. From (1) above facts at 12:02:31, (2) above static conditions of the airport from model, (3) empirical theorems \( \forall o \forall r \lnot (\text{At}(o, r) \land \text{Runway}(r) \land \text{Aircraft}(o) \land \text{Accelerate}(o) \Rightarrow \text{TakeOffFrom}(o, r)) \) (WM1) and \( \forall o \forall r_1 \forall r_2 \forall t_1 \forall t_2 (\text{TakeOffFrom}(o, r_1) \land \text{Intersecting}(r_1, r_2) \Rightarrow \text{OccupiedIntersection}(o, r_1, r_2)) \) (WM4) of world model, and (4) empirical theorem \( \forall t_1 \forall t_2 \forall o \forall r_1 \forall r_2 (\text{OccupiedIntersection}(o, r_1, r_2) \land \text{TakeOffFrom}(o, r_2) \Rightarrow F(\text{Rlby}(o_2, r_1))) \) (PM5) of predictive model, based on temporal deontic relevant logics, the predictor deduced predictions/detections \( \text{TakeOffFrom}(BCI937, 28), \text{TakeOffFrom}(THA971, 16), \) and \( F(\text{Rlby}(BCI937, 16)) \), then sent these predictions/detections to decision maker and human machine interface system. Then human machine interface system gave alert \( F(\text{Rlby}(BCI937, 16)) \), i.e., “BCI937 will cause a runway incursion on runway 16”, to both air traffic controller and pilot of BCI937. From the above predictions/detections and empirical theorems \( \forall o_1 \forall o_2 \forall r_1 (\text{TakeOffFrom}(o_1, r) \land F(\text{Rlby}(o_2, r)) \Rightarrow \text{O(Evade}(o_1))) \) (BM2) and \( \forall o_1 \forall r_1 \forall r_2 (\text{TakeOffFrom}(o_1, r_1) \land F(\text{Rlby}(o_1, r_2)) \Rightarrow \text{O(Evade}(o_1))) \) (BM5), based on temporal deontic relevant logics, the decision maker deduced instructions \( \text{O(Evade}(BCI937)) \) and \( \text{O(Evade}(THA971)) \). Because the speed of either BCI937 or THA971 was less than V1, according to the calculative rule, the final instructions were “BCI937 abort take-off” and “THA971 abort take-off”, while decision maker gave these instructions to both human machine interface system and airport traffic signal system. Human machine interface system gave these instructions to air traffic controller and pilots of BCI937 and THA971, while airport traffic signal system also gave signal instruction to both BCI937 and THA971 for aborting take-off.

We compare ARIPS with the RIMCAS deployed in Zurich airport. In the real incident, when potential runway incursion occurred 30 seconds, the RIMCAS generated a stage 2 alert. In contrast, in our simulation experiments shown in Sect. 5, the ARIPS only use 4.9–5.0 seconds to give alert of prediction of runway incursion as well as the instructions to prevent the runway incursion.

5. Simulation Experiments

The purpose of simulation experiments is to show the performance, correctness, and generality of ARIPS. Because the difference between current RIPSs and ARIPS is the core components of ARIPS, we only implemented and tested the core components of ARIPS.

To prepare input data and to simulate outside of the core components of ARIPS, we also built a simulation program to simulate real dynamic airports. The simulation program has following features: (1) to simulate any existing airports or user-define airports, (2) user-define aircraft/vehicles routes and speeds (supporting speed change), (3) tools for user-define airports and aircraft/vehicles routes, (4) aware of collision and display that, (5) interface for ARIPS getting information of aircraft/vehicles, including location, speed, and acceleration, and (6) graphic interface to display the dynamic airport (aircraft taxiing, aircraft taking off and landing), which is similar with ASDE-X. All programs ran on a PC with Intel Core i7-860 Processor (2.8 GHz, 4 cores, 8 threads), 4 Gbyte memory, and Scientific Linux release 6.2 for x86_64 (Linux kernel is 2.6.32-220). All programs are written in Java and running on OpenJDK Runtime Environment (IcedTea6 1.10.4), except FreeEnCal in C++. Because
the purpose of this work is not about how to get surveillance information of aircraft/vehicles, we used real surveillance information about the position, speed and acceleration of aircraft from the three historical incidents and four test scenarios [8] as the basic experimental data instead of raw sensory data. Besides, we also prepared variations of above scenarios as experimental data.

To evaluate the performance of ARIPS, we compared the execution time of ARIPS with current RIPSs and/or human using the same surveillance information from the three historical incidents and four test scenarios. The execution time of current RIPSs and/or human is obtained directly from related documents. Figure 8 shows the meaning of execution time. For human, the total time of prediction/detection is time from the incident occurs to the ATC controller considers the incident or gives the alert about the incident, and the total time of decision-making for instructions is from the incident occurs to the ATC controller gives the ATC-commands for handling the incident. For current RIPSs, the total time of prediction/detection is time from the incident occurs to the RIPS generates alert about the incidents. For ARIPS, the total time of prediction/detection is the sum of the lead time for updating surveillance information and the time for predicting/detecting the incursion/collision using the surveillance information, and the total time of decision-making for instructions and/or suggestions is the sum of the lead time for updating surveillance information, the time for prediction/detection, and the time for generating the decision. In the experiments, we did not simulate the process of generating surveillance information, but used surveillance information as input data of ARIPS directly. To compensate the time for updating surveillance information, we introduced the lead time of updating surveillance information, and added the lead time to the execution time of ARIPS additionally. Table 2 shows the surveillance performance of some traffic surveillance sensor systems [3].

According to this table, the experimental lead times of ADS-B, SMR(i), and MLAT are 0.25–2 seconds, 0.25–1 seconds, 0.25–0.5 seconds correspondingly. In the experiments, we adopted worst lead time, 2 seconds.

A summary of experimental results are as follows. Because we did each scenario experiment at least five times, our result showed a range of execution time of ARIPS.

- **Scenario A**: Boston Logan International Airport (BOS), Nov. 24, 2010 [22]
  Type: Departure/Taxi
  In the real incident: The ATC controller use 9 seconds to react, and 2 more seconds to give instructions. The runway incursion prevention system did not work.
  ARIPS’s performance: Using 3.4–4.0 seconds to give alert “F(RIby(JB1264,15R))” and instructions “O(Evade(JBU417))” and “O(Hold(JBU1264))”.
  \[T1 = T3 + T4 = 2.0 + 4.0 = 6.0\]
  \[T2 = T3 + T5 = 2.0 + 4.0 = 6.0\]

- **Scenario B**: Charlotte Douglas International Airport (CLT), May 29, 2009 [23]
  Type: Departure/Taxi
  In the real incident: After the incident occurred, 14 seconds later, ASDE-X alert was given.
  ARIPS’s performance: Using 2.6–3.8 seconds to give alert “F(RIby(N409DR,18L))” and instructions “O(Evade(JIA390))” and “O(Hold(N409DR))”.
  \[T1 = T3 + T4 = 2.0 + 3.8 = 5.8\]
  \[T2 = T3 + T5 = 2.0 + 3.8 = 5.8\]

- **Scenario C**: Zurich Airport (LSZH), June 18, 2010 [15]
  - the incident shown in Sect. 4.9.
  Type: Departure/Departure on intersecting runways
  In the real incident: After the incident occurred, 16 seconds later, the crew of BWA713 reported the runway incursion to ATC Controllers. After the incident occurred, 19 seconds later, ATC controller instructed BC1937 to abort take-off. After the incident occurred, 30 seconds later, the RIMCAS system generates a stage 2 alert.
  ARIPS’s performance: Using 4.9–5.0 seconds to give alert “F(RIby(BC1937,18L))” and instructions “O(Evade(BC1937))” and “O(Evade(THA971))”.
  \[T1 = T3 + T4 = 2.0 + 5.0 = 7.0\]
  \[T2 = T3 + T5 = 2.0 + 5.0 = 7.0\]

- **Scenario D**: Test scenario 1 on Dallas-Fort Worth International Airport (DFW), NASA, 2002 [8]
  Type: Arrival/Taxi
  Performance of NASA’s Runway incursion prevention system (RIPS): After the incident occurred, 7 seconds
later, runway incursion advisory and alerting system (RIAAS) gave runway traffic alert.

**ARIPS’s performance:** Using 4.4–4.7 seconds to give alert “F(RIby(Taxi, 35L))” and instructions “O(Hold(Taxi))” and “O(Evade(Arrival))”.

\[
T1 = T3 + T4 = 2.0 + 4.7 = 6.7 \\
T2 = T3 + T5 = 2.0 + 4.7 = 6.7
\]

- **Scenario E:** Test scenario 2 on DFW, NASA, 2002 [8]
  - **Type:** Departure/Taxi
  - **Performance of RIPS:** After the incident occurred, 15 seconds later, RIAAS gave runway traffic alert.

**ARIPS’s performance:** Using 4.4–4.7 seconds to give alert “F(RIby(Taxi, 35L))” and instructions “O(Evade(Departure))” and “O(Hold(Taxi))”.

\[
T1 = T3 + T4 = 2.0 + 4.7 = 6.7 \\
T2 = T3 + T5 = 2.0 + 4.7 = 6.7
\]

- **Scenario F:** Test scenario 3 on DFW, NASA, 2002 [8]
  - **Type:** Taxi/Departure
  - **Performance of RIPS:** After the incident occurred, 10 seconds later, RIAAS gave runway traffic alert.

**ARIPS’s performance:** Using 4.1–4.5 seconds to give alert “F(RIby(Taxi, 35L))” and instructions “O(Evade(Departure))” and “O(Hold(Taxi))”.

\[
T1 = T3 + T4 = 2.0 + 4.5 = 6.5 \\
T2 = T3 + T5 = 2.0 + 4.5 = 6.5
\]

- **Scenario G:** Test scenario 4 on DFW, NASA, 2002 [8]
  - **Type:** Arrival/Departure
  - **Performance of RIPS:** After the incident occurred, 13 seconds later, RIAAS gave runway traffic alert.

**ARIPS’s performance:** Using 2.5–2.6 seconds to give alert “F(RIby(Arrival, 35L))”, while using 4.4–4.9 seconds to give instructions “O(GoAround(Arrival))” and “O(Evade(Departure))”.

\[
T1 = T3 + T4 = 2.0 + 2.6 = 4.6 \\
T2 = T3 + T5 = 2.0 + 4.9 = 6.9
\]

Table 3 compares current RIPS and/or human with ARIPS in the total time of prediction/detection and total time of decision making for instructions. We adopted worst lead time of current traffic surveillance sensor systems (2 seconds) and worst execution time of ARIPS. We could conclude that ARIPS provided earlier prediction of incidents and earlier decision-making for instructions than current RIPSs and/or humans in all scenarios.

The following contents explain the reason why ARIPS could predict earlier than conventional RIPSs in the scenarios of three real incidents based on the same sensory data. Conventional RIPSs use inflexible algorithms/models, thus they cannot flexibly utilize the empirical knowledge of air traffic controllers and the concrete condition of a certain airport or certain area of that airport. For example, in the real incident occurred on LSZH, after potential runway incursion occurred, 30 seconds later, the RIMCAS generated a stage 2 alert, because “every second, the speed and directional vector are determined from the current position by calculation. In the process, the directional vector is continuously projected forward. The speed must be higher than 12 meters per second. In order to recognize the problem of two aircraft crossing on two different runways, a circular area with a diameter of 400 meters was laid around the intersection of runways 16/28. If, on the basis of the calculated projections, two aircraft simultaneously enter this ‘critical circle’ a Stage 2 alert is triggered” [15]. In contrast, ARIPS use flexible models, which include the empirical knowledge about the runway intersection such as WM4 and PMS, and the fact that runway 28 and runway 16 cross each other. Therefore, ARIPS could work like real air traffic controllers more than conventional RIPSs, thus ARIPS could provide earlier alerts.

We evaluated the correctness of predictions/detections and decision about instructions and/or suggestions of ARIPS using scenario A–G, five variations for each test scenarios, and combination of test scenarios of NASA, i.e., three aircraft/vehicles involved in a runway incursion simultaneously. The result of prediction/detection and decision about instructions and/or suggestions of ARIPS for scenario A–G have been given in the results of performance experiments. In all experiments, predictions/detections are correct, and all instructions are effective to avoid the incursion/collisions.

We evaluated the generality of ARIPS using different scenarios, different airports, and flexible control policy. Scenario A–G and five variations for each test scenarios provide the empirical data about different airports and different incident scenarios. For flexible control policy, we designed following experiment. We assumed air traffic controllers of DFW decided to use the parallel runway 35L for landing and runway 35C for take-off. Therefore, we added \( \forall \text{or}(\text{TakeOffFrom}(o, r) \land \text{LandingRunway}(r) \Rightarrow \text{RunwayConfusion}(o)) \), \( \forall \text{or}(\text{TakeOfRunway}(r) \land \text{LandOn}(o, r) \Rightarrow \text{RunwayConfusion}(o)) \), and \( \text{LandingRunway} \)
(35L), and \textit{TakeOffRunway}(35C) in world model, added \(\forall o r r(\text{RunwayConfusion}(o) \land \text{TakeOff}(\text{From}(o, r)) \Rightarrow P(\text{AbortTakeOff}(o)))\) and \(\forall o r r(\text{RunwayConfusion}(o) \land F(\text{LandOn}(o, r)) \Rightarrow P(\text{GoAround}(o)))\) \(("P"\) is a deontic operator, while \(PA\) means “it is permitted that \(A\)” in behavioral model, added \textit{RunwayConfusion} as interesting term of predictor, and added \textit{AbortTakeOff} as interesting term of decision maker. \textit{Scenario}: The departure aircraft was taking off from runway 35L. \textit{ARIPS’s performance}: Using 4.6–4.8 seconds to give alert \textit{“RunwayConfusion(Departure)"}, and instructions \textit{“P(AbortTakeOff(\text{Departure}))”}.

In our experiments, scenario A–G and five variations for each test scenarios showed that our system could deal with different airports and different incident scenarios based on flexible models. The experiment about flexible control policy showed that our system could change control policy on run time by changing behavioral model.

Simulation experiments showed that: (1) our system provided earlier prediction of incidents than current systems in the experimental scenarios, (2) our system could give explicit instructions and/or suggestions for handling the incidents, and (3) our system uses flexible model, which can be customized for specific air traffic control policies and airports. However, our current implementation cannot solve the problems of tardy, false or missed alerts caused by erroneous or missing sensory data.

6. Comparison with Related Work

There are some applications using artificial intelligence for runway safety, such as [17]. The logic-based reasoning approach for prediction and decision-making used in ARRSs is different from other methods in the following points. First, the logic-based reasoning approach used in ARRSs is to explicitly separate the underlying logical system, reasoning/computing mechanism, and empirical knowledge in any prediction/decision making. Second, the underlying logical system used in ARRSs belongs to the family of strong relevant (relevance) logics [24]. Detailed explain of the differences refer to [5], [6].

Kitajima et al. proposed a decision maker in an ARRS for terminal radar control [25]. The differences between our work and the work of Kitajima et al. are (1) runway incursion and terminal radar control are totally two different problems, and (2) Kitajima et al. only presented a decision maker, not a whole system.

Besides ARRS for terminal radar control, there are also other ARRSs. The most important feature of logic-based reasoning approach used in ARRSs is generality. That means we can apply logic-based reasoning approach to any areas for predicting or decision making, by changing the logical system and empirical knowledge. In ARRSs, the empirical knowledge is the anticipatory model. Therefore, in terms of logic-based reasoning approach, the major differences of different ARRSs are different logical systems and/or anticipatory models.

7. Concluding Remarks

This paper analyzed the problems of current runway incursion systems, and proposed a new type of runway incursion prevention system based on logic-based reasoning, which can predict and detect runway incursions, then give explicit instructions and/or suggestions to pilots/drivers to avoid runway incursions/collisions. Simulation experiments showed that our system has long-range prediction of runway incursions, effective explicit instructions and/or suggestions, and flexible model for different aircraft and air traffic control policies. However, our current implementation cannot solve the problems of tardy, false or missed alerts caused by erroneous or missing sensory data.

Our future work will consider such factors such as missed/false surveillance information, extreme weather, as well as their consequences. The evaluation scenarios in this paper are not adequate enough, thus more test scenarios are needed for a more complete test. Besides, we will consider use our system for other runway safety problems, such as runway excursion and runway confusion.

References

[1] International Civil Aviation Organization, Manual on the prevention of runway incursions, First ed., ICAO, 2007. Doc 9870 AN/463.
[2] Federal Aviation Administration, Pilot’s handbook of aeronautical knowledge — Appendix 1, runway Incursion Avoidance, FAA, 2012.
[3] J. Schönefeld and D. Möller, “Runway incursion prevention systems: A review of runway incursion avoidance and alerting system approaches,” Progress in Aerospace Sciences, vol. 51, pp. 31–49, 2012.
[4] S. Young and D. Jones, “Runway incursion prevention: A technology solution,” Proc. 54th Annual International Air Safety Seminar, pp. 221–238, Athens, Greece, 2001.
[5] J. Cheng, “Adaptive prediction by anticipatory reasoning based on temporal relevant logic,” Proc. 8th Int. Conf. Hybrid Intelligent Systems, pp. 410–416, 2008.
[6] J. Cheng, “Adaptive decision making by reasoning based on relevant logics,” Proc. Computational Intelligent: Foundations and Applications, 9th International FLINS Conference, pp. 541–546, 2010.
[7] J. Cheng, “Temporal relevant logic as the logical basis of anticipatory reasoning-reacting systems,” Proc. Computing Anticipatory Systems: CASYS — 6th International Conference, AIP Conference Proceedings, pp. 362–375, Liege, Belgium, 2004.
[8] R. Cassell, C. Evers, J. Esche, and B. Sleep, “NASA runway incursion prevention system (RIPS) Dallas-Fort Worth demonstration performance analysis,” Tech. Rep. CR-2002-216777, 2002.
[9] J. Gertz and R. Grappel, “Surveillance improvement algorithms for airport surface detection equipment model X (ASDE-X) at Dallas-Fort Worth airport,” Tech. Rep. ATC-333, MIT Lincoln Laboratory, 2007.
[10] Zurich Airport, “2011 runway safety report,” 2011.
[11] Federal Aviation Administration Airport Engineering Division, “Runway status lights system,” Tech. Rep. Engineering Brief 64D, FAA, 2011.
[12] Federal Aviation Administration, “Fact sheet — Airport movement area safety system,” http://www.faa.gov/news/fact_sheets/news_story.cfm?newsId=6302, accessed Feb. 11, 2013.
[13] International Civil Aviation Organization, Advanced surface movement guidance and control systems (A-SMGCS), manual, First ed.,
Jingde Cheng is a professor of computer science at Graduate School of Science and Engineering, Saitama University in Japan. He received the degree of Bachelor of Engineering in computer science from Tsinghua University in China in 1982, and the Master of Engineering degree and the Doctor of Engineering degree, both in computer science from Kyushu University in Japan, in 1986 and 1989 respectively. Before he joined Saitama University in 1999, he was a research associate (1989–1991), an associate professor (1991–1996), and a professor (1996–1999) at Kyushu University. His current research interests include relevant and ampliative reasoning, relevant logic and its applications, epistemic programming paradigm for scientific discovery, autonomous evolution of knowledge-based systems, anticipatory reasoning-reacting systems, persistent computing, and information security engineering environment. He is a senior member of ACM, and a member of IEEE-CS, IEEE-SMC, IEEE, and IPSJ.

Zhiliang Zhu is a professor of software engineering at Software College, Northeastern University in China. He received the degree of Bachelor of Communications and Information System from the Northeastern University, the degree of Master degree Communications and Information System, and degree of Doctor of Engineering in Computer Software and Theory from Northeastern University, in 1984, 1989, and 2006, respectively. His research interests include information integrate, complexity software system, network coding and communication security, chaos-based digital communications, applications of complex-network theories, and cryptography. He is a Fellow of the China Institute of Communications (CIC), a senior member of the Chinese Institute of Electronics (CIE), a member of IEEE and a member of Software Engineering Education Instruction Committee under the Ministry of Education and an Expert of SOA Standards Working Group of Ministry of Industry and Information Technology.

Yuichi Goto is an assistant professor of computer science at Graduate School of Science and Engineering, Saitama University in Japan. He received the degree of Bachelor of Engineering in computer science, the degree of Master of Engineering in computer science, and the degree of Doctor of Engineering in computer science from Saitama University in 2001, 2003, and 2005 respectively. His current research interests include relevant reasoning and its applications, automated theorem finding, anticipatory reasoning-reacting systems, and Web services engineering. He is a member of ACM, IEEE-CS, IPSJ, and JSAI.

Kai Shi is a Ph.D. student of computer science at Graduate School of Science and Engineering, Saitama University in Japan. He is also a lecturer of software engineering at Software College, Northeastern University in China. He received the degree of Bachelor of Engineering in computer science and the degree of Master of Engineering in computer science from Northeastern University in China in 2004 and 2007, respectively. His current research interests include relevant logic and its applications, anticipatory reasoning-reacting systems, and safety engineering.