Verification of an Air-Traffic Control System with Probabilistic Real-time Model-checking

Tran Thi Bich Hanh and Dang Van Hung

April 2007

UNU-IIST Report No. 355

$\boxed{\mathcal{R}}$

The United Nations University

UNU-IIST

International Institute for Software Technology

P.O. Box 3058
Macao

Verification of an Air-Traffic Control System with Probabilistic Real-time Model-checking

Tran Thi Bich Hanh and Dang Van Hung

Abstract

In this paper we present an approach to safety verification of Air-Traffic Control Systems (ATC) using probabilistic real-time model-checking. Namely we use ATC as a case study for the formal analysis of real time systems whose performance depends on uncertain or probabilistic behaviors. We construct a probabilistic timed automata model for ATC by extending the Operator Choice Model (OCM), a model of operators' behaviors in ATC system, with timed and probabilistic assumptions. Some properties of this system are then expressed in probabilistic real time computation tree logic (PCTL). The verification results produced by using the tool PRISM illustrate the uses of this approach to better understand the human errors in real time systems.

Tran Thi Bich Hanh is a fellow at UNU-IIST from September 01, 2006 to June 01, 2007. She got a BSc degree in Computer science at the University of Natural Sciences, Vietnam National University, Hochiminh City in 2003. She is currently a lecturer at Department of Software Engineering, University of Natural Sciences, Vietnam National University, Hochiminh City. Her research interests include Software Engineering (Formal Methods), Internet/Web technologies, Computer Vision, and Geography Information System (GIS). E-mail: ttbhanh@fit.hcmuns.edu.vn

Dang Van Hung is a research fellow for the research project "Theories and Design Methods for Realtime Systems" since October 1995 being on leave of absence from Institute of Information Technology, Nghia Do, Cau Giay, Hanoi, Vietnam. He has a PhD (equivalent) degree in Computer Science (concurrent systems) in 1988, Computer and Automation Research Institute (SZTAKI), Hungarian Academy of Sciences, Budapest, Hungary, and a BSc degree in Mathematics (numerical methods) in 1977, Hanoi University, Hanoi, Vietnam. His research interests include Formal Techniques of Programming, Concurrent and Distributed Computing, Design Techniques for Real-Time systems. E-mail: dvh@iist.unu.edu

Contents

1 Introduction 1

2 Air-Traffic Control Systems 2
2.1 Overview of the system 2
2.2 Formulation of the Modeling and Verification Problem 3

3 Abstract Probabilistic Timed Model 4
3.1 Probabilistic Timed Automata 4
3.2 Probabilistic real time Computation Tree Logic 6
3.3 Air-Traffic Control Model 6

4 PRISM Model for ATC and Verification Results 9
4.1 PRISM Model for ATC 9
4.2 Experimental Verification Results 10
4.2.1 Conflict free 10
4.2.2 Deadline effect 12

5 Conclusion 18

1 Introduction

Ensuring the safety for Air-Traffic Control systems (ATC) has become a crucial issue due to air traffic increasing growth year after year. The main task in ATC is to keep a safe separation distance between aircraft and to manage the flow of air traffic. Despite the aids of automated parts in the system, ATC is heavily dependent upon the capabilities of human operators. Some accidents in ATC were characterized by "human errors" with the causal factors involving perception, memory, decision making, communication and team resource management [12]. Therefore, formal analysis is becoming central for safety verification for ATC. Formal methods [3] are based on the use of mathematics notations with precise semantics to specify and verify complex systems. The aim is to remove ambiguities in system specifications, and develop rigorous reasoning about the properties of these specifications. These techniques have been applied successfully in many application domains. However safety verification for ATC contains real-time constraints and probabilistic data because of uncertainties in the environment, such as weather conditions and human decision errors etc. These factors encourage us to carry out a formal analysis for ATC characterized with timing and probabilistic properties.

A formal analysis of ATC was proposed by Cerone et al. [1]. This paper presents an approach to classifying and analyzing human error based on patterns of recurring behavior. A model with the consideration of task failures in operators' decision making was formalized using CSP (Hoare's Communicating Sequential Processes [6]). Some task failures based on top-down decomposition of the operator's behavior were specified as formulae in the linear temporal logic [11]. Then model checking techniques [2] were used to verify the completeness of the decomposition. The formal analysis proved that the initial decomposition based on the psychological analysis of the results of the experiments conducted using an ATC simulator, was not complete. This contributed to improve the cognitive model and define a new task failure decomposition, which was formally proven to be complete. However due to the difficulty of model-checking analysis caused by state explosion with the explicit representation of time in temporal logic, the analysis only considered an abstract model in which the separation distance between aircraft is not explicitly modeled with time. In our opinion, such an abstract model cannot capture the realistic erroneous behaviors in ATC.

This paper presents an approach to the safety verification of ATC systems. Our approach is illustrated with a simple conflict-detection and resolution task. We construct a probabilistic timed automata model [9] by extending the model proposed in [1] with timed and probabilistic assumptions. Some real-time and probabilistic properties of this system are then specified as formulae in the probabilistic real time computation tree logic [4]. We then use the tool PRISM for symbolic model checking for probabilistic timed automata [10] to verify and analyze these properties.

The paper proceeds as follows. An overview of the operator's behaviors in air-traffic control system is described in the next section. In Section 3, we recall the necessary concepts of probabilistic timed automata and probabilistic real time computation tree logic, illustrating how they can be used to construct the abstract model. The probabilistic model checking results are then presented in Section 4. Finally, Section 5 concludes the paper.

2 Air-Traffic Control Systems

2.1 Overview of the system

In this paper we use the case study presented in [1] that concerns a formal analysis of erroneous operator behavior in a conflict detection and resolution task for an air traffic control (ATC) simulator for our verification approach. The case study is based on the Operator Choice Model (OCM), which was designed to provide a means of simulating realistic operators' behaviors in complex real-world ATC systems. The OCM includes both "correct" and "incorrect" decisions, in order to record how mistakes can arise, propagate and get corrected through simulation runs.

The basic specification of the OCM is given by the statecharts in Figure 1. The operator's task is decomposed into certain key activities each of which corresponds to a state in the figure, and which are typically expected to occur sequentially. The goal of the task is to resolve all the conflicts appearing in the system before they violate separation (i.e., aircraft get closer to one another than their required separation distances), while not introducing any new conflict.

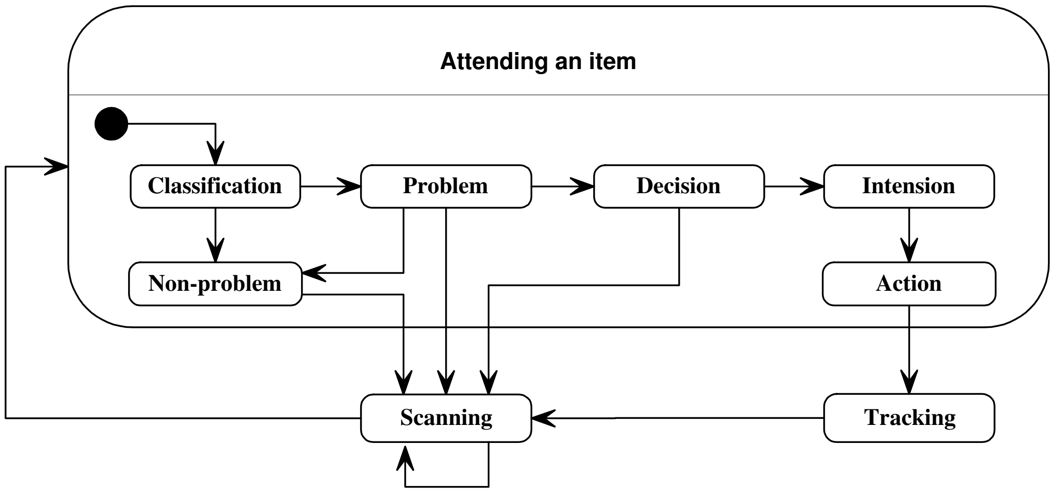

Figure 1: The Operator Choice Model

We describe those key activities as follows.

Scanning: The operator begins in the Scanning state to monitor the movement of aircraft on a screen, looking for pairs of aircraft that have potential to violate separation. There can be a number of potential problems to which the operator might attend. Thus, the Scanning process represents a method for selecting one for the operator's intention. The transition from the Scanning back to itself shows that the operator may not notice any pair needed to be attended to on the interface.

Classification: Once attending to a pair, the operator determines whether it is a problem requiring action or not.

Non-problem: If the attended pair is classified as a non-problem, the operator returns to Scanning to look for another pair to attend to.

Problem: If the attended pair is classified as a problem , the operator, while examining possible solutions, may reclassify the pair as a non-conflict one; or he/she may proceed to decide an action on that pair. The operator may also defer deciding an action and return to scanning because he/she may think that the conflict has a too low priority to require immediate attention.

Decision: The operator develops an appropriate plan to solve the problem. After action plan is decided he can go immediately on to intention or return to scanning.

Intention: The operator selects a pair on which he/she intends to perform action, in believing that this is the problem although this might not be the case.

Action: The operator implements the plan of action on the selected pair of aircraft, which may be different from the one that was intended to be selected.

Tracking: After the action is taken, the environment will record the result of the action, and update any possible changes in state that occur for each pair. It is assumed that: (1) the number of aircraft in the system is fixed; (2) at any time a pair can be in conflict or not; and (3) the conflict status of a pair can change only as the result of the operator performing an action on one or the other of the aircraft in that pair. Once the environment has been updated, the OCM continues to run from the Scanning state.

2.2 Formulation of the Modeling and Verification Problem

The OCM described above provides a basis for quantitative risk analysis of complex, interleaved tasks. It is considered as a framework for describing the operator's behavior in ATC. However it still lacks a lot of information that the real ATC offers. In practice, the performance of ATC depends very much on how much time it takes for the operator at each state of the process to solve the problem as well as the time duration related to aircraft's speeds and distances to the point where a separation violation starts if no action is taken. Moreover the choice made by the operator at every state in the process is a probabilistic choice. Therefore it is more reasonable and efficient to characterize such an erroneous behavior as the cause of a separation violation related to time and probability.

To capture stochastic behaviors of ATC, we present an extended model for ATC using Probabilistic Timed Automata [9]. We extend the abstract model to include our assumption that it takes the operator some time to perform each activity. We will use our guess to assign probabilistic data to operator's choices at each state in the model, and will change the data during the verification with model checking to analyze their effects on the performance of the system. However, the precise data for the probability and timing models are specific to the domain of application, and should be specified using realistic statistic data from domain experts.

3 Abstract Probabilistic Timed Model

In this section we recall the concepts of probabilistic timed automata [9, 8] and probabilistic real time computation tree logic [4]. We also show how to use them to construct the abstract model of the ATC.

3.1 Probabilistic Timed Automata

Probability distribution and Markov Decision Processes. A discrete probability distribution over a finite set S is a function $\mu : S \rightarrow [0, 1]$ such that $\sum_{s \in S} \mu(s) = 1$. The set of all probability distributions over S is denoted by $Dist(S)$.

A (labeled) Markov decision process $\mathcal{M}$ is a tuple $(S, s_0, \mathcal{L}, Steps)$ where:

- S is a finite set of states,
- $s_0 \in S$ is an initial state,
- $\mathcal{L} : S \rightarrow 2^{AP}$ is a function assigning to each state a set of atomic propositions which are true in that state,
- $Steps : S \rightarrow 2^{Dist(S)}$ is a function assigning to each state $s \in S$ a finite, non-empty set of discrete probability distributions on S.

A path ω of $\mathcal{M}$ is a finite or infinite sequence of states:

$$\omega = s_0 \xrightarrow{\mu_0} s_1 \xrightarrow{\mu_1} s_2 \xrightarrow{\mu_2} \ldots$$

where $s_i \in S, \mu_i \in Steps(s_i)$ and $\mu_i(s_{i+1}) > 0$. The probability measure Pr over a finite path $\omega = s_0 s_1 ... s_n$ is defined as $Pr(\omega) = 1$ if $n = 0$ and $Pr(\omega) = \prod_{i=0}^{n-1} \mu_i(s_{i+1})$ otherwise. For a finite path ω, let $last(\omega)$ and $\omega(k)$, respectively, denote the last and the k^{th} state in ω where k is less than the length of ω. An *adversary* of a Markov decision process $\mathcal{M}$ is a function mapping every finite path ω of $\mathcal{M}$ to a distribution μ on S such that $\mu \in Steps(last(\omega))$. Let Adv be a set of all adversaries in $\mathcal{M}$ and $Path^{adv}$ be a set of all paths over the adversary $adv \in Adv$.

Clocks, clock valuations and zones. Let $\mathbb{R}$ be a time domain of non-negative reals, and C be a finite set of clocks which take values from the time domain $\mathbb{R}$. A clock valuation is a function $v : C \to \mathbb{R}$ that assigns a real value to each clock. Let $\mathbb{R}^C$ denote the set of all clock valuations. For a set of clocks $X \subseteq C$ we use $v[X := 0]$ to denote the clock valuation obtained from v by assigning 0 to all of the clocks in X. For $t \in \mathbb{R}$, the clock valuation $v + t$ denotes the time increment for v by assigning $v(x) + t$ to each clock $x \in C$.

A zone is a conjunction of atomic constraints of the form $x \sim c$ for $x \in C, \sim \in \{\leq, =, \geq\}$, and $c \in \mathbb{N}$. Clock valuation v satisfies the zone ζ, written $v \models \zeta$, if and only if ζ evaluates to true after substituting each clock $x \in C$ with the corresponding clock value from v. Let $\mathcal{Z}_C$ denote the set of all zones over C.

Probabilistic timed automata. A probabilistic timed automaton is a tuple

$$\mathcal{A} = (S, s_0, C, \Sigma, inv, prob)$$

where:

- S is a finite set of states,
- $s_0 \in S$ is an initial state of A,
- C is a finite set of clocks,
- Σ is a finite set of events,
- $inv : S \to \mathcal{Z}_C$ is a function mapping each state to an invariant condition,
- $prob \subseteq S \times \mathcal{Z}_C \times \Sigma \times Dist(S \times 2^C)$ is the probabilistic edge relations.

The semantics of a probabilistic timed automaton $\mathcal{A}$ can be expressed in terms of an infinite-state Markov decision process whose states are pairs (s, v)'s where $s \in S$ and v is a clock interpretation such that $v \models inv(s)$. The initial state is (s_0, θ) where we have $\theta(x) = 0 \ \forall x \in C$. There are two types of transitions from a state (s, v). (1) A state can change due to elapse of time while the invariant condition $inv(s)$ is satisfied. (2) A tuple $(s, \zeta, \sigma, p) \in prob$ represents a discrete transition from the state s which is enabled by ζ to the state s' on event σ with the probability $p(s', \lambda)$ where λ is the set of resetting clocks.

Let $\mathcal{A}_i = (S_i, s_{i_0}, C_i, \Sigma_i, inv_i, prob_i)$ for $i \in \{1, 2\}$ and assume that $C_1 \cap C_2 = \emptyset$. The parallel composition of two probabilistic timed automata $\mathcal{A}_1$ and $\mathcal{A}_2$ is the probabilistic timed automaton $\mathcal{A}_1 \| \mathcal{A}_2 = (S_1 \times S_2, (s_{1_0}, s_{2_0}), C_1 \cup C_2, \Sigma_1 \cup \Sigma_2, inv, prob)$ where $inv(s_1, s_2) = inv_1(s_1) \wedge inv_2(s_2)$ for all $(s_1, s_2) \in S$. Tuple $((s_1, s_2), \zeta, \sigma, p) \in prob$ if and only if one of the following conditions holds:

- $\sigma \in \Sigma_1 \backslash \Sigma_2$ and there exists $(s_1, \zeta, \sigma, p_1) \in prob_1$ such that $p = p_1.\mu(s_2, \emptyset)$;

- $\sigma \in \Sigma_2 \backslash \Sigma_1$ and there exists $(s_2, \zeta, \sigma, p_2) \in prob_2$ such that $p = \mu(s_1, \emptyset).p_2$;
- $\sigma \in \Sigma_1 \cap \Sigma_2$ and there exists $(s_1, \zeta_1, \sigma, p_1) \in prob_1$ and $(s_2, \zeta_2, \sigma, p_2) \in prob_2$ such that $\zeta = \zeta_1 \wedge \zeta_2$ and $p = p_1.p_2$

where $\mu(s', \lambda)$ is a distribution defined as $\mu(s', \lambda)(s, \rho) = 1$ iff $s = s'$ and $\rho = \lambda$ for all $(s', \lambda) \in S_i \times 2^{C_i}$, and where for any $s_1 \in S_1, s_2 \in S_2, \lambda_1 \subseteq C_1$ and $\lambda_2 \subseteq C_2$: $p_1.p_2((s_1, s_2), \lambda_1 \cup \lambda_2) = p_1(s_1, \lambda_1).p_2(s_2, \lambda_2)$.

3.2 Probabilistic real time Computation Tree Logic

Probabilistic real time Computation Tree Logic (PCTL) is a logic for specifying and reasoning about real-time and probabilistic behaviors of a system. A formula in PCTL is defined by the following grammar:

$$\begin{aligned} \varphi &::= a \mid \neg\varphi \mid \varphi \wedge \varphi \mid P_{\bowtie\lambda}[\phi] \\ \phi &::= X\varphi \mid \varphi\, U_{\leq t}\, \varphi \mid \varphi\, W_{\leq t}\, \varphi \end{aligned}$$

where φ is a state formula, ϕ is a path formula, $\bowtie = \{<, \leq, >, \geq\}$, $\lambda \in [0, 1]$ and $t \in \mathbb{N}$.

The satisfaction relations $s \models_{\mathcal{A}} f$ and $\omega \models_{\mathcal{A}} f$, which intuitively mean that the state formula f is true in the state s and the path ω satisfies the path formula f in the probabilistic time automaton $\mathcal{A}$, are defined as follows:

$$\begin{array}{lllll} s & \models_{\mathcal{A}} & a & \text{iff} & a \in L(s) \\ s & \models_{\mathcal{A}} & \neg\varphi & \text{iff} & \text{not } s \models_{\mathcal{A}} \varphi \\ s & \models_{\mathcal{A}} & \varphi 1 \wedge \varphi 2 & \text{iff} & s \models_{\mathcal{A}} \varphi 1 \text{ and } s \models_{\mathcal{A}} \varphi 2 \\ s & \models_{\mathcal{A}} & P_{\bowtie\lambda}[\phi] & \text{iff} & \forall adv \in Adv.\ Pr(\{\omega \mid \omega \in Path_{\mathcal{A}}^{adv}\ \&\ \omega \models_{\mathcal{A}} \phi\}) \bowtie \lambda \\ \omega & \models_{\mathcal{A}} & X\ \varphi & \text{iff} & \omega(1) \models_{\mathcal{A}} \varphi \\ \omega & \models_{\mathcal{A}} & \varphi 1\ U_{\leq t}\ \varphi 2 & \text{iff} & \exists\ 0 \leq i \leq t.\ \omega(i) \models_{\mathcal{A}} \varphi 2 \text{ and } \forall\ 0 \leq j < i.\ \omega(j) \models_{\mathcal{A}} \varphi 1 \\ \omega & \models_{\mathcal{A}} & \varphi 1\ W_{\leq t}\ \varphi 2 & \text{iff} & \omega \models_{\mathcal{A}} \varphi 1\ U_{\leq t}\ \varphi 2 \text{ or } \forall\ i \geq 0.\ \omega(i) \models_{\mathcal{A}} \varphi 1 \end{array}$$

A probabilistic timed automaton $\mathcal{A}$ satisfies a property expressed as a state formula f if and only if its initial state satisfies f.

$$\models_{\mathcal{A}} f \equiv s_0 \models_{\mathcal{A}} f$$

3.3 Air-Traffic Control Model

In this section, we present our ideas and methods for modeling the ATC system.

Modeling ideas. We extend the OCM described in previous section with time and probability. The ATC system which was modeled as a parallel compositions of a number of separate processes [1] in CSP is now integrated in one probabilistic timed automaton for easy to verify when considering verification under several air-traffic scenarios with different number of pairs of aircraft. We use an array c of N Boolean variables to record the real state of N pairs of aircraft. For $k \in [1, N]$, the expression '$c[k] = true$' indicates that the k^{th} pair is in conflict and '$c[k] = false$' that it is not in conflict. The results of the operator's action are expressed as the compositions of all possible changes in real states of all pairs which are updated synchronously.

We assume that the operator has to spend some time to complete each activity of his task represented as a state in the system. The time taken for conflict resolution is the sum of the durations of individual classifications plus some amount for issuing the clearance. The probability that the operator makes a right choice at every state is maintained at a high value (0.99). The probability that overlook task failures is a low value (0.01). The variables expressing time and probability data in the system are shown below.

Variable	**Description**	**Value(s)**
N	Number of pairs in the system	$1, 2, 3$ or 6
ts	Duration of scanning	$0.1s$
tc	Duration of classification	$0.1s$
tp	Duration of reclassification	tc
td	Duration of decision	$0.25s$
tin	Duration of intention	$0.1s$
ta	Duration of action	$N * tc + td$
p_s_s	Probability to not attend to any item	0.01
p_s_spk	Probability to attend an item is varied depending on the real status of every pair, in general uniformed to others' scanning probabilities	$(1 - p\_s\_s)/N$
p_cpk_low	Probability to misclassify a non-conflict to a problem and vise-versa	0.01
p_cpk_high	Probability to correctly classify a pair regarding its real state	$1 - p\_cpk\_low$
p_ppk_s	Probability to return scanning when the pair was classified as a problem	0.01
p_ppk_low	Probability to reclassify a conflict to a non-problem or proceed to decide a plan to solve a non-conflict	0.01
p_ppk_high	Probability to proceed to decide a plan to solve a conflict or reclassify a non-conflict to a non-problem	$1 - p\_ppk\_low$ $-p\_ppk\_s$
p_dpk_s	Probability to return scanning without action plan	0.01
p_dpk_ipk	Probability to take intention on one pair before performing an action	$1 - p\_dpk\_s$
$p\_adverse\_pk$	Probability for the action which makes a non-conflict to be in conflict	0.01
$p\_unnecessary\_pk$	Probability for the unnecessary action on the non-conflict and causes no effect on this pair	0.01
$p\_noeffect\_n\_pk$	Probability for the action which is taken on another pair pk' and causes no effect on the non-conflict pk	$1 - p\_adverse\_pk$ $-p\_unnecessary\_pk$
$p\_resolved\_pk$	Probability for the action which resolves the conflict into non-conflict	0.99
$p\_unresolved\_pk$	Probability for the action which can not resolve the conflict	0.005
$p\_noeffect\_c\_pk$	Probability for the action which is taken on another pair pk' and causes no effect on the conflict pk	$1 - p\_resolved\_pk$ $-p\_unresolved\_pk$

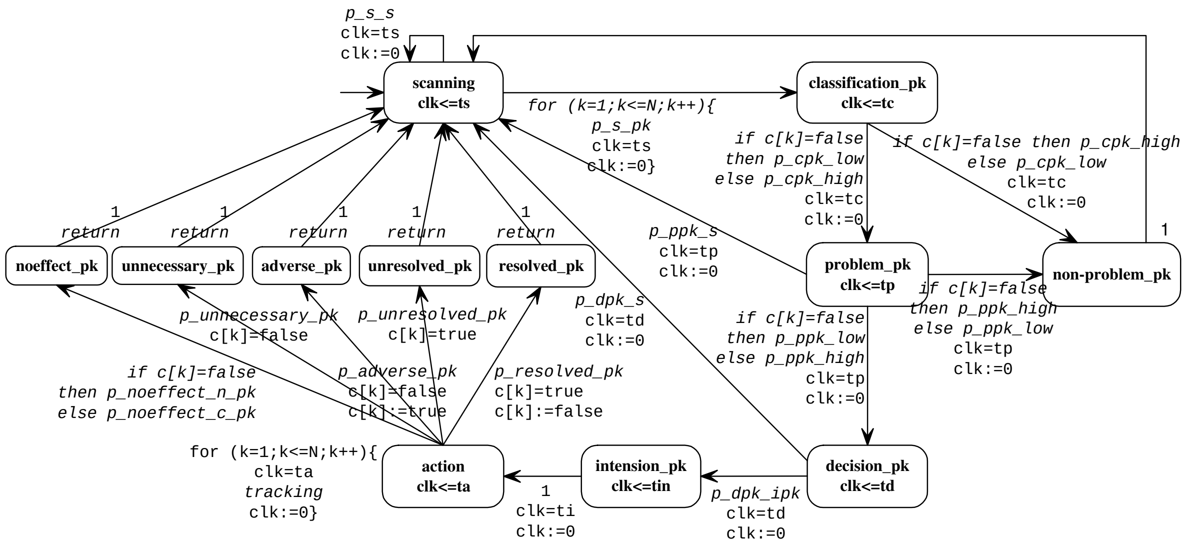

Figure 2: OCM Probabilistic timed automaton

ATC model. The probabilistic timed automaton representing OCM is shown in Figures 2. The OCM starts at the *scanning* state which represents the activity in which the operator is looking for one of N pairs of aircraft which requires attention. Once a pair is selected, the operator starts to classify it and, if it is classified as a problem, to develop a plan to solve the problem. At this point, there are two cases depending on the value of the guard "$c[k]$" on the pair. Each case corresponds to a different value of the guard and has the "opposite" probabilistic distribution to the other. We combine them to simplify the figure. After this activity, the operator might return to the Scanning state if it is appropriate to do so. The real statuses $c[k]$ of all pairs will be updated synchronously with labeled transitions *tracking* right after the operator finishes his action to resolve a problem if any. For each pair of aircraft there are two cases with three possibilities to change the real status of $c[k]$. For the k^{th} pair which is being in conflict, there are possibilities to resolve it (*resolved_pk*) or not (*unresolved_pk*) when the operator has taken an action on this pair; and another possibility is when the operator has taken an action on the different k'^{th} pair which causes no effect on the k^{th} pair (*noeffect_pk*). Similarly for the k^{th} pair which is being in non-conflict, there is a possibility to make it to be in conflict (*adverse_pk*) when the operator has taken an action on this pair; the other possibilities are when the k^{th} pair remains in non-conflict and the operator has taken an action on the pair (*unnecessary_pk*), and when the k^{th} pair remains in non-conflict and the operator has taken an action on a different pair (*noeffect_pk*). After tracking action's results and status for all pairs, the OCM returns to its initial state to start a new cycle.

4 PRISM Model for ATC and Verification Results

4.1 PRISM Model for ATC

In this section, we give a brief explanation of our model in PRISM, a PRobabilistIc Symbolic Model checker for automatic verification and formal analysis of probabilistic systems [5]. PRISM supports three types of probabilistic models [7]: discrete-time Markov chains (DTMCs) for systems with simple probabilistic behavior and no concurrency, Markov decision processes (MDPs) for systems with non-determinism and probability, and continuous-time Markov chains (CTMCs) for systems with determinism and continuous-time. PRISM also allows models to be augmented with costs and rewards which associate real values with certain states and certain transitions of the model. This enables one to reason not only about the probability but also about a wide range of quantitative measures related to the model behavior, e.g. “expected time”, “expected energy consumption”, and “expected number of messages lost”.

Models are specified using the PRISM modeling language [13], a simple, state-based language based on the Reactive Modules formalism. A model is a parallel composition of a number of modules which can interact with one another. Each module contains a set of finite states and its behaviors by all possible transitions between its states. Properties of the model are then expressed in the PRISM property specification language [13] which is based on the two probabilistic temporal logics PCTL (probabilistic computation tree logic) for DTMCs and MDPs, and CSL (continuous stochastic logic) for CTMCs. Then these properties can be verified automatically by running model checking program in PRISM.

Since the PRISM modeling language does not support array, to model the system as introduced in the previous section in PRISM, we keep the translation from our automaton model into PRISM straightforward but divide it into a parallel compositions of at least 2 modules: a common Scanner module and a Tracker module for each pair of aircraft. A common Scanner represents the operator’s behaviors when looking for and attending to one pair of aircraft in the system. Each pair of aircraft will need one specific Tracker for tracking the result and the pair’s real status whenever the operator performs an action. Scanner and Tracker will run sequentially. With *module renaming* feature supported by PRISM, we just need to describe one module Tracker for the first pair of aircraft and then duplicate it to a number of instances for other pairs. The array c for tracking real status of all pairs is now expressed as the local variables ck of Trackers in which k is renamed to be a specific pair’s index. For instance, $c1$ is for the first pair, and $c2$ is for the second pair etc. All the Trackers are kept synchronized at every state.

The Scanner module has a data variable x with values from 1 to $5 * N + 3$ to represent the set of local states of Scanner process, where N is the number of pairs of aircraft in the system, 5 is the number of local states for each pair of aircraft, and 3 is the number of extra states for common activities which are not associated with any specific pair. For example, ‘$x = 5 * N + 1$’ indicates that the operator does not have attention to any pair of aircraft in the system. When x is assigned a value from $5 * (k - 1) + 1$ to $5 * (k - 1) + 5$, it means that the operator is currently attending to the k^{th} pair and processing some activities on it. The timing behaviors of the model are expressed as discrete transitions. Each *time* transition is considered as one time unit. All the modules will be synchronized with these *time* transitions by using commands which are labeled with the same *time* action to make time progressing simultaneously. We use DTMCs for our modeling formalism since the model contains probability and no non-determinism, e.g at any state there exists only one discrete probabilistic distribution over the countable state-space of the system.

The fragment of PRISM code for the model is shown in Figure 3 and the detailed specifications are given in the Appendix. When performing model checking in PRISM we keep the number of pairs fixed (i.e., 1, 2, 3 or 6), but vary the number of conflicts in them to observe the effect of "workload" on the operator's performance. We also vary the scanning probabilities between conflict and non-conflict pairs to investigate their relationships to the performance of the system.

4.2 Experimental Verification Results

In this section, some experimental verification results for the case study under several different air-traffic scenarios are presented with detailed analysis.

4.2.1 Conflict free

Our first concern is to resolve all conflicts appearing in the system while not creating any new one, e.g "With probability 1, eventually there have been no conflict in the system."

The property can be expressed in PTCL as follows:

$$P_{\geq 1}[true\ U\ resolved\_all]$$

where $resolved\_all := \bigwedge_{i=1}^{N}(c_i = false)$

Model checking results for all configurations show that this property holds in all states. However, this property is satisfied because time is not taken into account in the model (the time for satisfying *resolved_all* is unbounded), and the number of pairs is fixed. Therefore it is always the case that all conflicts in the system are resolved eventually, and the probability for resolving all conflicts is equal to 1. In practice, the time for resolving should be bounded, and the model checking results this case will be shown below.

```
probabilistic // DTMC

const int N = 1; // number of pairs
const int k = 0; // pair's index starting at 0

module Scanner
  x : [1..5*N+3] init 5*N+1; // local states: 5*N+1:not attending, 5*k+1..5*k+5: attending the (k+1)^th pair
 clk : [0..MAX_TIME] init 0; // local clock

 // scan
 [time] (x=5*N+1)&(clk<ts) -> (clk'=min(clk+1, t_scan));
 [] (x=5*N+1)&(clk>=ts) -> p_s_s:(x'=5*N+1)&(clk'=0)
                          + p_s_spk:(x'=5*k+1)&(clk'=0);

  // classify - spk
  [time] (x=5*k+1)&(clk<tc) -> (clkp'=min(clk+1, tc));
  [] (x=5*k+1)&(clk>=tc) -> ((ck=true)?p_cpk_low:p_cpk_high):(x'=5*k+2)&(clk'=0)
                          + ((ck=true)?p_cpk_high:p_cpk_low):(x'=5*k+3)&(clk'=0);
 ...
 // action - a
 [time] (x=5*N+2)&(clk<ta) -> (clk'=min(clk+1, ta));
 [tracking] (x=5*N+2)&(clk>=ta) -> (x'=5*N+3)&(clk'=0);

 // return scanning
 [return] (x=5*N+3) -> (x'=5*N+1);
endmodule

module Tracker
  sk: [0..5] init 0;// local states (0:scanning, 1:unresolved, 2:resolved, 3:noeffect, 4:unnecessary, 5:adverse)
  ck: bool init cinit; // pair's status (false:non-conflict, true:conflict)

  // unnecessary_pk + adverse_pk + noeffect_n_pk
  [tracking] (sk=0)&(c1=false)&(x=5*N+3) -> p_unnecessary_pk:(sk'=4)&(ck'=false)
                          + p_adverse_pk:(sk'=5)&(ck'=true)
                          + p_noeffect_n_pk:(sk'=3)&(ck'=false);
  ...
  [return] (sk>0) -> (sk'=0);
endmodule
```

Figure 3: PRISM code representing the model

```
module Timer
  clk : [0..DEADLINE];
  [time] (clk<DEADLINE) -> (clk'=min(clk+1, DEADLINE));
  [] (clk>=DEADLINE) -> (clk'=DEADLINE);
endmodule
```

Figure 4: PRISM code representing Timer

4.2.2 Deadline effect

Since the expected time to resolve all the problems with probability 1 is infinite, we now investigate the probability that the system reaches a conflict-free state by a given deadline. To check these properties, we add an extra clock (Figure 4) which is synchronized with other modules via *time* labeled commands to calculate the exact time elapsing (not the number of discrete steps) and prevent progress once the clock reaches a certain deadline.

Therefore, we want to check the property:

"is the probability for the system to have no conflict within time T greater than λ?"

The property can be expressed in PTCL as follows:

$$P_{\geq\lambda}[true\ U_{\leq T}\ resolved\_all]$$

where $resolved\_all := \bigwedge_{i=1}^{N}(c_i = false)$

The verification results by PRISM for different scenarios are as follows.

Work load effect

To observe and analyze the effect that the workload of the operator, represented as the number of pairs of aircraft, has on the performance of the operator we vary the number N of pairs of aircraft from 1, 3 and 6. Assume that at the beginning there is only one real conflict in the system ($C = 1$). The operator will randomly pick up one pair to attend to in Scanner process. The model checking results in Figure 5 show that the more aircraft appearing in the system, the less effective the operator can handle the situation. He should need more time to observe the system, which leads the probability of solving all conflicts to be lower within a given period of time.

In the next experiment, we keep the number of pair fixed (N=3) but vary the number of conflicts in them (C=1, 2, or 3). Figure 6 unexpectedly shows that the probability of solving all problems is higher in the case with more conflicts than in the case with less conflicts. This explains the

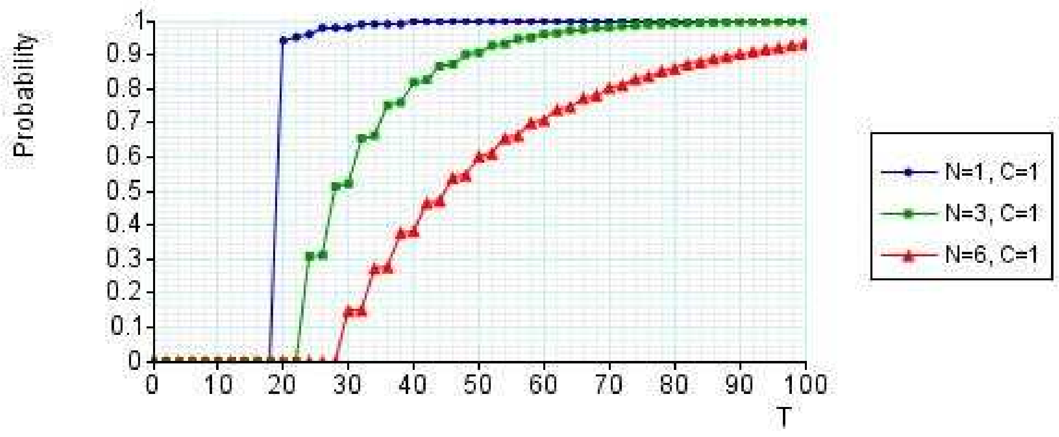

Figure 5: Probability of resolving all conflicts - Varying number of non-conflicts

effect of operator's errors on the performance of the system. The operator may misclassify a non-conflict as a conflict and try to take action to resolve it, or he may waste time in attending to non-conflicts other than to conflicts.

Misclassification

To make the above experiment more informative, we calculate the probabilities for the operator to have misclassification within an interval of time $[0, T]$ for the cases where there are 3 pairs of aircraft and vary the number of conflicts in them. The properties for misclassifying a conflict pair as a non-problem and vise-versa within time T are expressed by the following formulas respectively.

- $misclass\_conflict\_p_k := (c_k = true)\ U_{\leq T}\ non\_problem\_p_k$, where $non\_problem\_p_k := (x = 5 \times k + 2)$,
- $misclass\_nonconflict\_p_k := (c_k = false)\ U_{\leq T}\ problem\_p_k$, where $problem\_p_k := (x = 5 \times k + 3)$.

Figures 7 and 8 show that the more conflicts there are in the system, the lower probabilities for the operator to misclassify a conflict pair as a non-problem and vise-versa. However the operator will increase his tendency to misclassify a non-conflict pair as a problem when time passes.

Scanning effect

To observe and analyze the effect of operator's attention to conflicts in the Scanner process on the performance of the system, we consider the scenario in which there are 4 aircraft. These

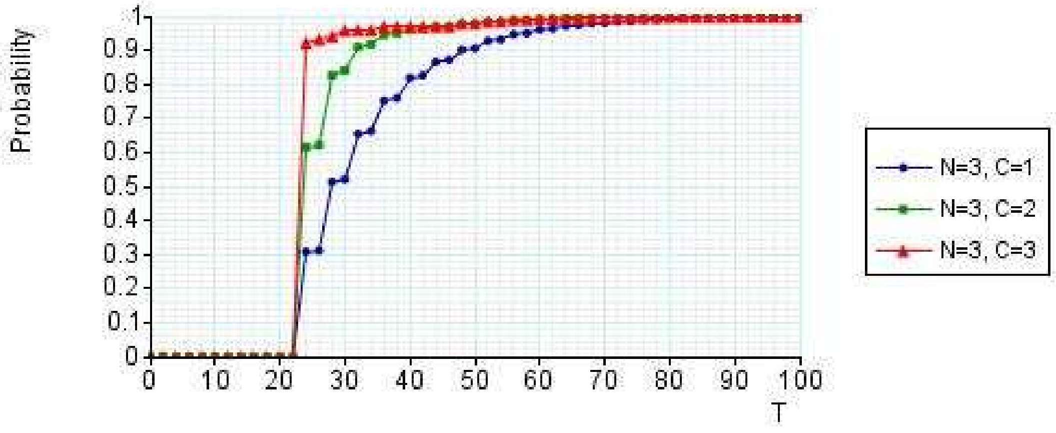

Figure 6: Probability of resolving all conflicts - Varying number of conflicts

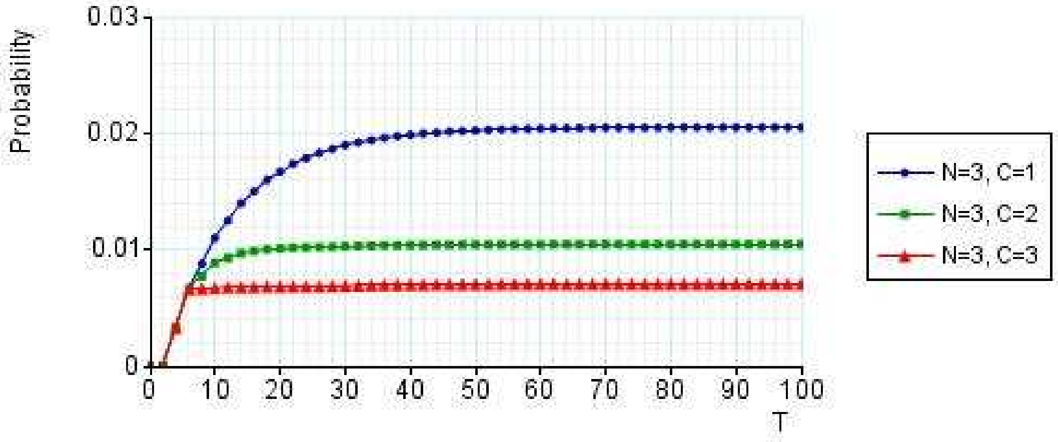

Figure 7: Probability of misclassifying a conflict pair as a non-problem

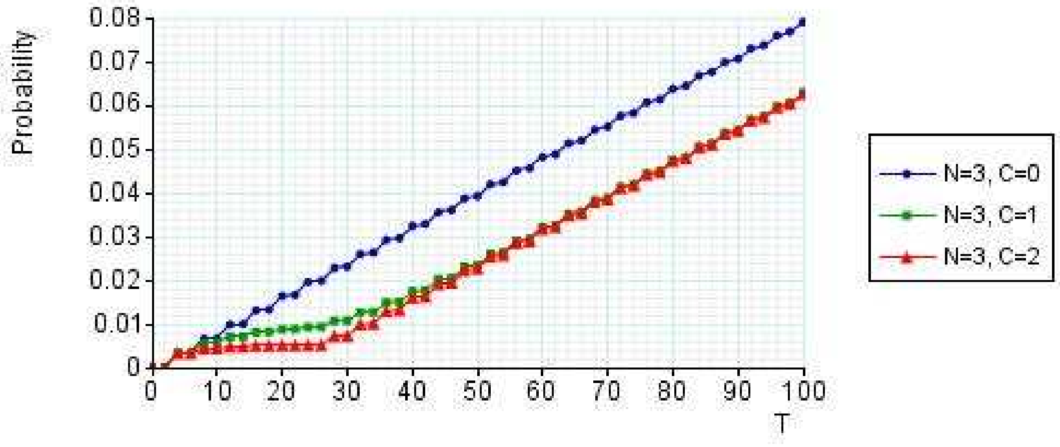

Figure 8: Probability of misclassifying a non-conflict pair as a problem

aircraft are divided into 2 separate groups, each group is formed by one pair of aircraft. Assume that any action which is taken on any aircraft in one group does not make any effect to the pairs containing an aircraft in the other group. This scenario represents a case where there are only 2 pairs of aircraft needed to monitor. We vary the number of conflicts in these 2 pairs as well as the probabilities of choosing each pair to attend to in the Scanner process. We assume that the operator will give much more attention to conflicts (pc) than to non-conflicts (pn) in the case there is 1 non-conflict and 1 conflict in the system. The results in Figure 9 show that the lower the probability of choosing non-conflict, i.e. the higher the probability of choosing conflict, the higher the probability to resolve the problem. Again, as discussed in the previous experiment, this explains the effect of operator's errors on the performance of the system when losing attention to the right pair to attend.

In the case there are only 2 conflicts in the system, the changes in probabilities of scanning each pair do not make any effect on the system, as shown in Figure 10.

Expected Time

To make the above two experiments more informative (more detailed) we associate the following reward structure (Figure 11) with the model when considering $0.05s$ as one unit of time to calculate the maximum expected time the operator should take to resolved all problem in the system with 2 pairs.

Model checking results in Figure 12 show that the operator will need more time to resolve the problem if he pays more attention to the non-conflict pair. However when both pairs are conflict, it does not matter which one is selected first.

Task failure

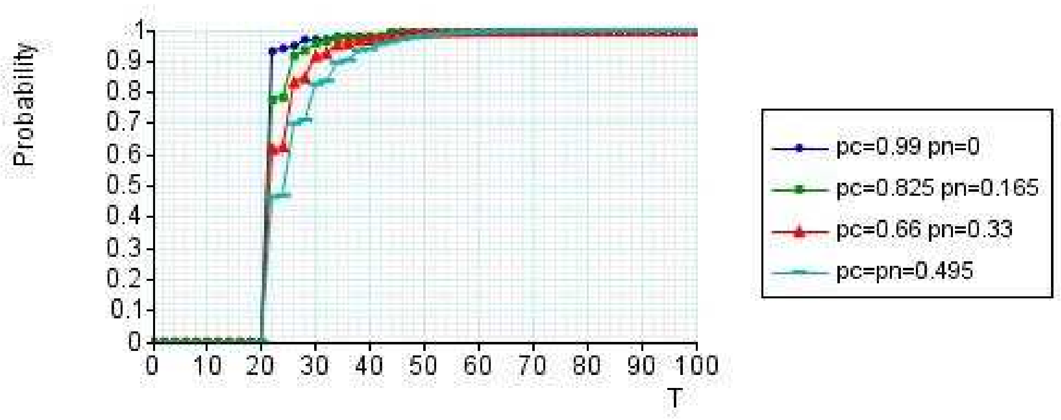

Figure 9: Probability of resolving all conflicts - 1 conflict and 1 non-conflict

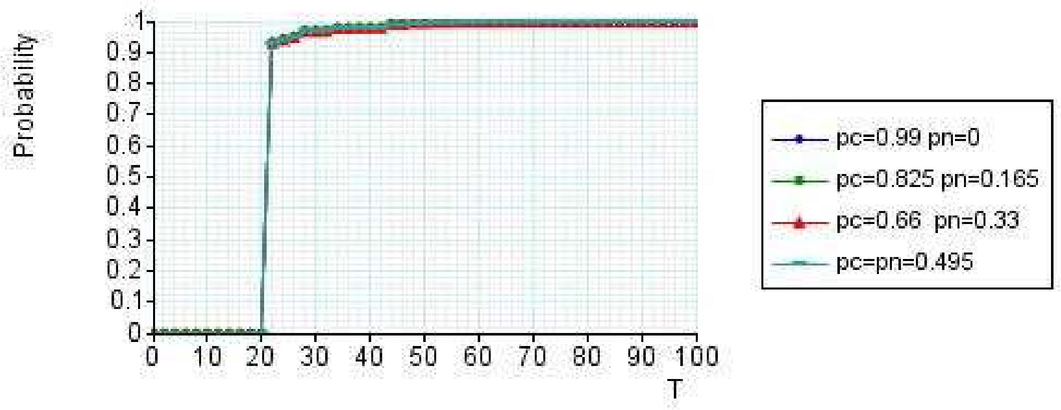

Figure 10: Probability of resolving all conflicts - 2 conflicts

```
rewards
  [time] true : 0.05;
endrewards
```

Figure 11: PRISM code representing Rewards

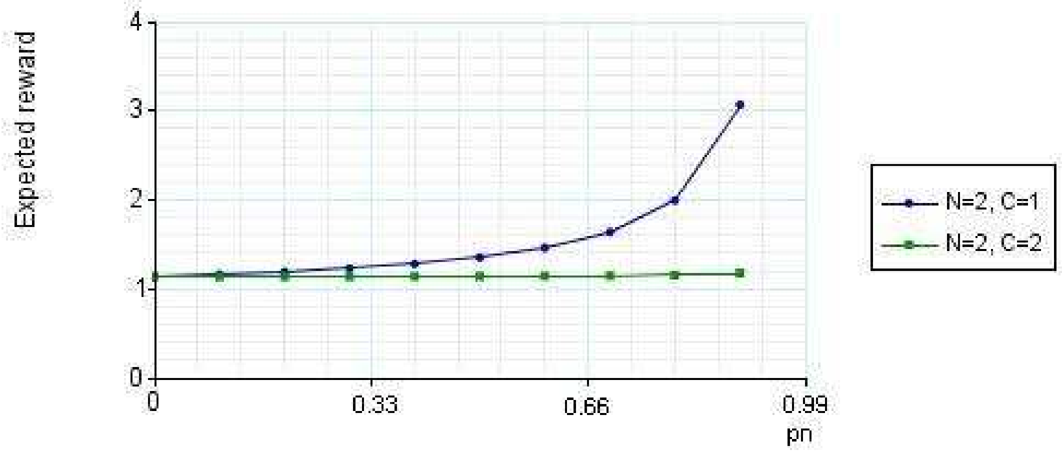

Figure 12: Expected time to resolve all conflicts

The probabilities of some operator's task failures within a certain time T are calculated. We consider some following task failures:

1. "The operator can not resolve all conflicts within time T",
2. "Some operator's action produces adverse situations within time T",
3. "The operator does not pay attention to a certain conflict within time T",
4. "The operator does not have intention to response to a certain conflict within time T".

In PCTL these are expressed by the following formulas respectively:

1. Let $resolved\_all_T$ be $(true\ U_{\leq T}\ resolved\_all)$, where $resolved\_all := \bigwedge_{i=1}^{N}(c_i = false)$. Then, $non\_resolved_T := \neg\ resolved\_all_T$,
2. $conflict\_created_T := (true\ U_{\leq T}\ \bigvee_{i=1}^{N} adverse\_p_i)$, where $adverse\_p_i := (s_i = 5)$,
3. $non\_scan\_pk_T := \neg((c_k = true)\ U_{\leq T}\ classification\_p_k)$, where $classification\_p_k := (x = 5 \times k + 1)$,
4. $non\_response\_pk_T := \neg((c_k = true)\ U_{\leq T}\ intension\_p_k)$, where $intension\_p_k := (x = 5 \times k + 5)$.

We run the model checker for the cases where the number of pairs varies among $N = 1, 3, 6$ and there is only 1 conflict ($C = 1$). At first we calculate the expected time units R within which the operator causes the task failure, then we use the values of R together with the deadlines T

to produce the corresponding probability (P). Table 1 shows that when there are more non-conflict pairs in the system, the operator may cause reversed situations earlier. The operator has to spend more time to make the right decision but the probability for task failure increases.

N	1	3	6
$R_{adverse}$	501,952.54	500,644.38	498,648.07
$R_{resolved\_all}$	20.50	33.71	54.48
$P_{non\_resolved_T} = 1 - P_{resolved\_all_T}$	0.059	0.324	0.345
R_{scan_p1}	2.02	10.10	22.23
$P_{non\_scan\_p1_T} = 1 - P_{scan\_p1_T}$	0.01	0.304	0.342
$R_{response\_p1}$	11.29	39.95	116.75
$P_{non\_response\_p1_T} = 1 - P_{response\_p1_T}$	0.05	0.052	0.01

Table 1: Expected time and Probability of task failures

5 Conclusion

We have presented the air traffic control system as a case study for safety verification with PRISM model-checking tool. Our model of ATC with probabilistic timed automata is simple enough for analysis but is still close to the real-world. Although the probabilities in the model are artificial, the verification results for the system are still useful for the system analysis. We have presented our experimental results using our assumptions about time and probability data. It is proved that our approach can help to better understand the human errors in safety-critical interactive systems.

To simplify the model we just considered the time the operator spends at each state of the process and made the assumption about the status of each pair of aircraft regardless of aircraft's speeds and time distances from the point where separation violation starts. For future work, we will extend our model so that it can capture the realistic behavior of aircraft regarding aircraft's speeds and time duration before separation violations occur in order to precisely characterize operator's errors.

From this case study, we got some experiences in using the probabilistic model checker PRISM. With the familiarity of the tool and experiences learned from this case study, it will easier for us to apply this verification techniques for other industrial case studies. We found that the description language of PRISM is simple but provides an intuitive way to construct complex systems. However, the lack of array data structure in the language makes it inflexible. It will be better if PRISM supports array and iteration structure in order to shorthand code and avoid mistakes when explicitly modeling different codes with similar behaviors. It will help to increase the automatic level of model checking since we just need one model and run model checking for all configurations through setting values for parameters which are expressed as undefined constants.

Acknowledgements

The authors would like to thank Antonio Cerone for the helpful discussion that initiates this work, and his valuable comments on the early draft of this report which lead to its significant improvement.

References

[1] A. Cerone, P.A. Lindsay, and S.Connelly. Formal Analysis of Human-computer Interaction using Model-checking. In *Proc. 3rd IEEE International Conference on Software Engineering and Formal Methods (SEFM'05)*, pages 352–362, IEEE Computer Society Press, 2005.

[2] E. Clarke, O. Grumberg, and D. Peled. Model Checking. MIT Press, 1999.

[3] E. Clarke and J. Wing. Formal methods: state of the art and future directions. In *ACM Computing Surveys*, ACM Press, 1996.

[4] H. Hansson and B. Jonsson. A Logic for Reasoning about Time and Reliability. *Formal Aspects of Computing*, 6(5):512–535, 1994.

[5] A. Hinton, M. Kwiatkowska, G. Norman, and D. Parker. PRISM: A Tool for Automatic Verification of Probabilistic Systems. In *Proc. 12th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'06)*, volume 3920 of LNCS, pages 441-444, Springer, 2006.

[6] C. Hoare. Communicating Sequential Processes. In *International Series in Computer Science*, Prentice Hall, 1985.

[7] M. Kwiatkowska. Model Checking for Probability and Time: From Theory to Practice. In *Proc. 18th IEEE Symposium on Logic in Computer Science (LICS'03)*, pages 351–360, IEEE Computer Society Press, 2003.

[8] M. Kwiatkowska, G. Norman, D. Parker and J. Sproston. Performance Analysis of Probabilistic Timed Automata using Digital Clocks. In *Formal Methods in System Design*, vol. 29, pages 33-78, Springer, 2006.

[9] M. Kwiatkowska, G. Norman, R. Segala, and J. Sproston. Automatic verification of real-time systems with discrete probability distributions. Theoretical Computer Science, 282(1):101–150, 2002.

[10] M. Kwiatkowska, G. Norman, J. Sproston, F. Wang. Symbolic model checking for probabilistic timed automata. In *Joint Conference on Formal Modelling and Analysis of Timed Systems (FORMATS) and Formal Techniques in Real-Time and Fault Tolerant Systems (FTRTFT)*, LNCS, vol. 3253, pages 293–308, Springer, 2004.

[11] Z. Manna and A. Pnueli. The Temporal Logic of Reactive and Concurrent Systems - specification. Springer-Verlag, 1992.

[12] S.T. Shorrock, B. Kirwan. Development and application of a human error identification tool for air traffic control. In *Applied Ergonomics*, vol. 33, no. 4, pp. 319-336(18), 2002.

[13] PRISM 3.1 Manual. *www.cs.bham.ac.uk/dxp/prism/doc/manual.pdf.*

Appendix

PRISM Code representing Air-Traffic Control System

```
probabilistic // DTMC

////////////////////////////////////////////////////////////////////////////////
// probabilities to make transitions from scanning to scanning (s) and classification the $k^{th}$ pair (cpk)
const double p_s_s = 0.01;
const double p_s_spk = (1 - p_s_s)/N;

// probabilities for classification
const double p_cpk_low = 0.01;
const double p_cpk_high = 1-p_cpk_low;

// probabilities to make transitions from problem to scan (s), non-problem (npk) and decision (dpk)
const double p_ppk_s = 0.01;
const double p_ppk_low = 0.01;
const double p_ppk_high = 1-p_ppk_s-p_ppk_low;

// probabilities to make transitions from decision to scan (s) and intension (ipk)
const double p_dpk_s = 0.01;
const double p_dpk_ipk = 1-p_dpk_s;

////////////////////////////////////////////////////////////////////////////////
// time durations at each state
const int ts = 2; // duration of scanning
const int tc = 2; // duration of classification
const int tp = tc; // duration of problem
const int td = 5; // duration of decision
const int tin = 2; // duration of intension
const int ta = N*tc+td; // duration of action

////////////////////////////////////////////////////////////////////////////////
const int MAX_TIME = 100;
const int DEADLINE = 100;
const int N = 1; // number of pairs
const int k = 0; // pair index starting at 0


module Scanner

  x:[1..5*N+3] init 5*N+1; // local states (5*N+1:not attending, 5*N+2:action, 5*N+3:tracking,
  // 5*k+1:classification pair $k^{th}$, 5*k+2:non-problem, 5*k+3:problem, 5*k+4:decision, 5*k+5:intension)

  clk:[0..MAX_TIME] init 0; // local clock
  // scan
  [time] (x=5*N+1)&(clk<ts) -> (clk'=min(clk+1, ts));
  [] (x=5*N+1)&(clk>=ts) -> p_s_s:(x'=5*N+1)&(clk'=0)
  + p_s_spk:(x'=5*k+1)&(clks'=0);

  // classify - spk
  [time] (x=5*k+1)&(clk<tc) -> (clk'=min(clk+1, tc));
  [] (x=5*k+1)&(clk>=tc) -> ((ck=true)? p_cpk_low:p_cpk_high):(x'=5*k+2)&(clk'=0)
  + ((ck=true)?p_cpk_high:p_cpk_low):(x'=5*k+3)&(clk'=0);

  // non-problem - npk
  [] (x=5*k+2) -> (x'=5*N+1)&(clk'=0);

  // problem - cpk
  [time] (x=5*k+3)&(clk<tp) -> (clk'=min(clk+1, tp));
  [] (x=5*k+3)&(clk>=tp) -> p_ppk_s:(x'=5*N+1)&(clk'=0)
  + ((ck=true)?p_ppk_low:p_ppk_high):(x'=5*k+2)&(clk'=0)
  + ((ck=true)?p_ppk_high:p_ppk_low):(x'=5*k+4)&(clk'=0);

  // decide - dpk
  [time] (x=5*k+4)&(clk<td) -> (clk'=min(clk+1, td));
  [] (x=5*k+4)&(clk>=td) -> p_dpk_s:(x'=5*N+1)&(clk'=0)
  + p_dpk_ipk:(x'=5*k+5)&(clk'=0);

  // intention - ipk
  [time] (x=5*k+5)&(clk<tin) -> (clk'=min(clk+1, tin));
  [] (x=5*k+5)&(clk>=tin) -> (x'=5*N+2)&(clk'=0);

  // action - a
  [time] (x=5*N+2)&(clk<ta) -> (clk'=min(clk+1, ta));
  [tracking] (x=5*N+2)&(clk>=ta) -> (x'=5*N+3)&(clk'=0);

  // return scanning
  [return] (x=5*N+3) -> (x'=5*N+1);
endmodule


/////////////////////////////////////////////////////////////////////////////////////
// probabilities to make transitions to unnecessary, adverse and noeffect states
const double p_unnecessary_pk = 0.01;
const double p_adverse_pk = 0.01;
const double p_noeffect_n_pk = 1-p_unnecessary_pk-p_adverse_pk;

// probabilities to make transitions to resolved, unresolved and noeffect states
const double p_resolved_pk = 0.99;
const double p_unresolved_pk = (1-p_resolved_pk)/2;
const double p_noeffect_c_pk = p_unresolved_pk;

// initial status of the pair
const bool cinit = true;
```

```
module Tracker
  sk:[0..5] init 0;// local states (0:scanning, 1:unresolved, 2:resolved, 3:noeffect, 4:unnecessary, 5:adverse)
  ck:bool init cinit; // pair's status (false:non-conflict, true:conflict)

  // unnecessary_pk + adverse_pk + noeffect_n_pk
  [tracking] (sk=0)&(ck=false)&(x=5*N+3) -> p_unnecessary_pk:(sk'=4)&(ck'=false)
  + p_adverse_pk:(sk'=5)&(ck'=true)
  + p_noeffect_n_pk:(sk'=3)&(ck'=false);

  // resolved_pk + unresolved_pk + noeffect_c_pk
  [tracking] (sk=0)&(ck=true)&(x=5*N+3) -> p_resolved_pk:(sk'=2)&(ck'=false)
  + p_unresolved_pk:(sk'=1)&(ck'=true)
  + p_noeffect_c_pk:(sk'=3)&(ck'=true);

  // return to scanning
  [return] (sk>0) -> (sk'=0);
endmodule

/////////////////////////////////////////////////////////////////////////////////////////


module Timer
  clk : [0..DEADLINE];

  [time] (clk<DEADLINE) -> (clk'=min(clk+1, DEADLINE));
  [] (clk>=DEADLINE) -> (clk'=DEADLINE);
endmodule

/////////////////////////////////////////////////////////////////////////////////////////


rewards
  [time] true : 0.05;
endrewards

/////////////////////////////////////////////////////////////////////////////////////////
```