An Agent-Based Approach for Accident Analysis in Safety Critical Domains: A Case Study on a Runway Incursion Incident

Tibor Bosse and Nataliya M. Mogles(✉)

Agent Systems Research Group, Vrije Universiteit Amsterdam,
de Boelelaan 1081, 1081 HV Amsterdam, The Netherlands
{tbosse,nm.mogles}@few.vu.nl

Abstract. This paper introduces an agent-based approach to analyze the dynamics of accidents and incidents in aviation. The approach makes use of a number of elements, including formalization of a real world scenario, agent-based simulation of variations of the scenario, and formal verification of dynamic properties against the (empirical and simulated) scenarios. The scenario formalization part enables incident reconstruction and formal analysis of it. The simulation part enables the analyst to explore various hypothetical scenarios under different circumstances, with an emphasis on error related to human factors. The formal verification part enables the analyst to identify scenarios involving potential hazards, and to relate those hazards (via so-called interlevel relations) to inadequate behavior on the level of individual agents. The approach is illustrated by means of a case study on a runway incursion incident, and a number of advantages with respect to the current state-of-the-art are discussed.

Keywords: Aviation · Incidents · Agent-based simulation · Verification · Interlevel relations

1 Introduction

Within aviation, analyzing the exact causes of accidents and incidents is a nontrivial task. Even if detailed flight data from the 'black box' are available, it is usually still difficult to come up with a clear analysis, for the simple reason that the causes of incidents cannot be attributed to a single point of failure of one individual entity. Instead, most incidents in aviation are found to be caused by a complex interplay of processes at various levels of the socio-technical system, involving pilots, air traffic controllers, technical systems, and their interaction. For example the famous accident in 2009 of Air France Flight 447 is still under investigation and seems to have been the consequence of a rare combination of factors. On May 31, 2009, this flight disappeared

Parts of this article appeared in the Proceedings of the Twenty-Fifth International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems (IEA/AIE'12) and in the Proceedings of the Fifth International Conference on Agents and Artificial Intelligence (ICAART'13).

N.T. Nguyen (Ed.): TCCI XVII 2014, LNCS 8790, pp. 66–88, 2014.
DOI: 10.1007/978-3-662-44994-3_4

somewhere over the Atlantic Ocean, during a route from Rio de Janeiro to Paris. The crash was the deadliest accident in the history of Air France, killing all 228 people on board. This accident seems to have been the consequence of a rare combination of factors, like inconsistent airspeed sensor readings, the disengagement of the autopilot, and the pilot pulling the nose of the plane back despite stall warnings.[1]

For the analysis of accidents and incidents in aviation, roughly two streams can be distinguished in the literature, namely *accident analysis* and *risk analysis*. Whilst the former has the goal to determine the cause of an accident that actually took place, the latter aims to assess the likelihood of the occurrence of future accidents. Hence, although both streams have similar purposes, a main difference is that accident analysis attempts to identify one specific combination of hazardous factors, whereas risk analysis basically explores a whole range of such factors, and the associated risks. Nevertheless, most of the existing approaches are used for both streams.

Traditionally, accident and incident analyses are done via fault and event trees, graphical representations of Boolean logic relations between success and failure types of events. However, although widely used, there is an increasing awareness that fault and event trees have serious limitations, especially when it comes to analysing dynamic systems with time-dependent interactions (see [6] for a more extensive argumentation). More recently, alternative approaches have been developed, such as FRAM [7] and STAMP [11]. While these approaches have proved successful in various case studies, they still have some drawbacks. In particular, FRAM lacks a formal semantics, which makes a computational analysis of complex non-linear processes impossible. STAMP does have a formal basis, but takes an aggregated, organisational perspective (based on system dynamics), which hinders an analysis at the level of individual agents (such as pilots and air traffic controllers), and their underlying mental processes.

As an alternative, the current paper presents an approach for analysis of aviation incidents that takes a multi-agent perspective, and is based on formal methods. The approach is an extension of the approach introduced in the work of Bosse and Mogles [4], which was in turn inspired by Blom, Bakker, Blanker, Daams, Everdij and Klompstra [1]. Whereas this approach mainly focuses on the analysis of existing accidents (also called *accident analysis* or *retrospective analysis*), the current paper also addresses analysis of potential future accidents (called *risk analysis* or *prospective analysis*). This is done by means of a multi-agent simulation framework that addresses both the behaviour of individual agents (operators, pilots) as well as their mutual communication, and interaction with technical systems. By manipulating various parameters in the model, different scenarios can be explored. Moreover, by means of automated checks of dynamic properties, these scenarios can be assessed with respect to their likelihood of the occurrence of accidents. The approach is illustrated by a case study on a runway incursion incident at a large European airport in 1995.

The remainder of this paper is structured as follows. In Sect. 2, the modelling approach used in the paper is presented. In Sect. 3, the scenario used within the case study is described. Section 4 introduces the agent-based model to simulate this (and similar)

[1] http://en.wikipedia.org/wiki/Air_France_Flight_447

scenarios, and Sect. 5 presents the simulation results. Section 6 addresses formal analysis of the model and its results, and Sect. 7 concludes the paper with a discussion.

2 Modeling Approach

In this section, first an overview of the modeling paradigm underlying the proposed methodology is given. After that, the modeling language and the methodology used for accident analysis are introduced.

2.1 Agent-Based Modeling

Agent-oriented approaches have been widely used for modeling complex socio-technical systems [8]. The essence of agent-based modeling is the agent-oriented world view that implies that the world consists of active, purposeful agents that interact to achieve their objectives. There is still much debate, however, about what exactly constitutes an agent or agenthood. The majority of researchers agree on the following definition of an agent proposed in [15]: 'An agent is an encapsulated (computer) system that is situated in some environment and that is capable of flexible, autonomous action in that environment in order to meet its design objectives'. The agent-based approach for modeling is characterized by flexibility of outcomes and the strong possibility of system's emergent behavior that cannot be predicted from behavior of its individual components, or agents. The class of agent models that has beliefs (what the agent knows about the world and other agents), desires (what the agent wants or which goals he has) and intentions (what the agent intends to do) have been used in a wide variety of applications including air traffic control, process control and transportation [8]. Such a Belief-Desire-Intention (BDI) paradigm [13] is powerful for the representation of cognitive states of agents and will be adopted in the current work in order to model internal cognitive states of human agents that may be crucial for understanding of human performance in critical domains and in air traffic in particular. Instead, entities with less autonomy (like aircraft and runways) will not be modeled using the BDI approach; these entities will simply be represented as static objects that are part of the environment with which the agents interact.

2.2 Temporal Trace Language

To model the different aspects of aviation operations from an agent perspective, an expressive modeling language is needed. On the one hand, qualitative aspects have to be addressed, such as observations, beliefs, and actions of human operators. On the other hand, quantitative aspects have to be addressed, such as the locations and speeds of aircraft. Another requirement of the chosen modeling language is its suitability to express both the basic mechanisms of aviation operations (for the purpose of simulation), as well as more global properties of these operations (for the purpose of logical analysis and verification). For example, basic mechanisms of aviation operations involve decision functions for individual agents (e.g., an operator may decide to give runway clearance, and a pilot to abort a take-off procedure in case of an emergency). Instead, examples of global properties address the overall safety of an operation, such as "no collisions take place".

The predicate-logical Temporal Trace Language (TTL) introduced in the work of Bosse, Jonker, van der Meij, Sharpanskykh and Treur [2] fulfils all of these desiderata. It integrates qualitative, logical aspects and quantitative, numerical aspects. This integration allows the modeler to exploit both logical and numerical methods for analysis and simulation. Moreover it can be used to express dynamic properties at different levels of aggregation, which makes it well suited both for simulation and logical analysis.

The TTL language is based on the assumption that dynamics can be described as an evolution of states over time. The notion of state as used here is characterised on the basis of an ontology defining a set of physical and/or mental (state) properties that do or do not hold at a certain point in time. These properties are often called *state properties* to distinguish them from dynamic properties that relate different states over time. A specific state is characterised by dividing the set of state properties into those that hold, and those that do not hold in the state. Examples of state properties are 'aircraft A moves with speed S', or 'Air Traffic Controller C provides runway clearance to aircraft A'. Real value assignments to variables are also considered as possible state property descriptions.

To formalise state properties, ontologies are specified in a (many-sorted) first order logical format: an *ontology* is specified as a finite set of sorts, constants within these sorts, and relations and functions over these sorts (sometimes also called signatures). The examples mentioned above then can be formalised by n-ary predicates (or proposition symbols), such as, moves_with_velocity(A, S) or communicate_from_to(C, A, runway_clearance). Such predicates are called *state ground atoms* (or *atomic state properties*). For a given ontology Ont, the propositional language signature consisting of all ground atoms based on Ont is denoted by APROP(Ont). One step further, the *state properties* based on ontology Ont are formalised by the propositions that can be made (using conjunction, negation, disjunction, implication) from the ground atoms. Thus, an example of a formalised state property is moves_with_velocity(A, S) & communicate_from_to(C, A, runway_clearance). Moreover, a *state* S is an indication of which atomic state properties are true and which are false, i.e., a mapping S: APROP (Ont) $\rightarrow$ {true, false}. The set of all possible states for ontology Ont is denoted by STATES(Ont).

To describe dynamic properties of complex processes such as in aviation, explicit reference is made to *time* and to *traces*. A fixed time frame T is assumed which is linearly ordered. Depending on the application, it may be dense (e.g., the real numbers) or discrete (e.g., the set of integers or natural numbers or a finite initial segment of the natural numbers). Dynamic properties can be formulated that relate a state at one point in time to a state at another point in time. A simple example is the following (informally stated) dynamic property about the absence of collisions:

For all traces γ,
there is no time point t
on which a collision takes place.

A *trace* γ over an ontology Ont and time frame T is a mapping $\gamma : T \rightarrow$ STATES (Ont), i.e., a sequence of states γ_t ($t \in T$) in STATES(Ont). The temporal trace language TTL is built on atoms referring to, e.g., traces, time and state properties. For example, 'in trace γ at time t property p holds' is formalised by state(γ, t) |= p. Here |= is a predicate symbol in the language, usually used in infix notation, which is comparable to the Holds-predicate in situation calculus. *Dynamic properties* are expressed by temporal statements built using the usual first-order logical connectives (such as $\neg$, $\wedge$, $\vee$, $\Rightarrow$) and quantification ($\forall$ and $\exists$; for example, over traces, time and state properties). For example, the informally stated dynamic property introduced above is formally expressed as follows:

$\forall\gamma$:TRACES $\neg\exists$t:TIME
state(γ, t) |= collision

In addition, language abstractions by introducing new predicates as abbreviations for complex expressions are supported.

To be able to perform (pseudo-)experiments, only part of the expressivity of TTL is needed. To this end, the executable LEADSTO language described in [3] has been defined as a sublanguage of TTL, with the specific purpose to develop simulation models in a declarative manner. In LEADSTO, direct temporal dependencies between two state properties in successive states are modelled by *executable dynamic properties*. The LEADSTO format is defined as follows. Let α and β be state properties as defined above. Then, $\alpha \twoheadrightarrow_{e,\ f,\ g,\ h} \beta$ means:

If state property α holds for a certain time interval with duration g,
then after some delay between e and f
state property β will hold for a certain time interval with duration h.

Based on TTL and LEADSTO, two dedicated pieces of software have recently been developed. First, the LEADSTO Simulation Environment [3] takes a specification of executable dynamic properties as input, and uses this to generate simulation traces. Second, to automatically analyse the resulting simulation traces, the TTL Checker tool [2] has been developed. This tool takes as input a formula expressed in TTL and a set of traces, and verifies automatically whether the formula holds for the traces.

2.3 Accident/Incident Analysis Methodology

Based on the agent-based modelling paradigm and TTL, the current paper proposes the following 7-step methodology for formal analysis of aviation incidents:

(1) *Development of formal ontology*: to develop the state ontology Ont introduced above, all relevant sorts, constants, functions and predicates have to be specified for the domain under investigation, enabling the modeller to describe the relevant aspects of the world (e.g., pilots, controllers, aircraft, actions, communications, mental states, and so on). This step is addressed in Sect. 4.1 of the current paper.
(2) *Formalisation of real world scenarios in terms of traces*: for each scenario, express the different events using the formal ontology developed in step 1), and allocate a time stamp to them. This step is described in Sect. 4.2.

(3) *Specification of local executable dynamic properties* of agents involved in the ATM system of the scenario under consideration: identify the relevant executable dynamic properties and express them in LEADSTO. This step is applied in Sect. 4.3.
(4) Perform *dynamic simulations* of the scenario: identify parameters and/or initial settings that might be crucial for occurring of an incident or an accident, manipulate the parameters and observe the behavior of the agents and the emergent behavior of the whole system. This step enables the analyst to observe a variety of alternative developments of the scenario (simulation traces) under investigation. Section 5 is devoted to this methodological step.
(5) *Specification of non-local dynamic properties at different levels*: identify non-local dynamic properties that are relevant for the domain, and express them in TTL. Section 6.1 is dedicated to this step.
(6) *Specification of interlevel relations between dynamic properties*: the dynamic properties identified in step (3) and (5) may be classified according to different levels of aggregation of the aviation domain. For instance, some properties may apply to the air traffic organisation as a whole (e.g., 'no incident will occur'), whereas others apply to the level of individual agents (e.g., 'agent A will only communicate correct information'). In this step, logical relationships between dynamic properties at different levels are established, to ensure that conjunctions of properties at one level imply properties at higher levels. This step is addressed in Sect. 6.2.
(7) *Verification of properties against (real life and simulated) traces*: using the TTL Checking Tool mentioned above, dynamic properties at different levels are checked automatically against the traces generated in step 2) and 4), allowing the analyst to find out what exactly went wrong in the scenarios under investigation. This step is described in Sect. 6.3.

Note that this approach is in principle not restricted to the domain of aviation. In fact, it is completely domain-independent, as long as it is applied to systems that consists of multiple interacting agents, and of which it is possible to obtain empirical data in the form of scenario descriptions. Nevertheless, the main purpose of the current paper is to study the applicability of this approach to the domain of aviation. Hence, in the remainder of the paper, the 7 steps are illustrated by means of the runway incursion case study.

3 Case Study

One of the possible approaches in analyzing the behavior of complex systems in aviation is by identification and formal analysis of case studies. This type of analysis allows researchers to acquire possible underlying information about incidents or almost-incidents within the air traffic domain. Our interest was mainly focused on incidents where a small mistake of one or multiple actors could have led to severe consequences at the level of the whole system, but was corrected by another actor and thus the possible accident was prevented. This focus on incidents was motivated by the fact that the numerous descriptions of air traffic accidents that could be found in the published literature are one-sided, as these cases cover just a small top of the iceberg of all risky situations occurring daily in air traffic interactions. However, it is not so easy to get access to these incidents, as they are mostly company confidential and not available for broad publications, or they are not officially reported at all. To obtain such a case study, it was decided to perform a semi-structured interview with an available expert, a two years retired pilot of a European civil aviation company.

The following subsections provide the overview of an interview that was performed with the available expert and the description of the extracted incident that has been selected for the formal analysis.

3.1 Interview

The interview with a retired pilot of a civil aviation company took place on May 12, 2011 and lasted approximately 1 hour and 15 minutes. It was a semi-structured interview with a predefined set of questions concerning the incidents that the pilot or any of his colleagues had experienced during his flight career. In the beginning of the interview it was clearly announced to the interviewee that we were interested in the cases within air traffic where a small local mistake could have led to severe global consequences, but was corrected before an actual accident would occur. The interviewee was asked to recall such incidents. This question contained the following subquestions:

1. Who was involved in the incident?
2. What was the cause of the problem?[2]
3. How was the problem solved?
4. What were the consequences?
5. Was the situation familiar to you from trainings or procedures?

During the interview a case study was identified that describes an incident where, due to the mistake of a pilot of one taxiing aircraft, two aircraft were taking off almost simultaneously from crossing runways. After the correct intervention of the air traffic controllers from the ATC Tower, and adequate decision making, coordination and action of the pilots of one of the aircraft, a collision was prevented. This incident is described (in an anonymised manner) in the following section.

3.2 Runway Incursion Incident Description

The runway incursion incident took place during the departure of an Airbus A310 of a civil aviation company from one large airport in Europe. A summary of the scenario is provided below. A schematic overview of the situation is provided in Fig. 1.

[2] Possible causes that might be relevant include failure of technical systems, miscommunication, fatigue, high or low workload (restricted Situation Awareness or decreased vigilance), strong positive or negative emotions, power influences, (dis)trust in colleagues or computer systems, little experience, negligence of the existing procedures, organisational management etc.

The Airbus was preparing for the departure: the pilot-in-command was sitting on the left and the co-pilot on the right seat in the cockpit and they were ready to start taxiing. They were supposed to taxi to runway 03 in the north-east direction. The Airbus received permission to taxi and started taxiing to its runway. Approximately at the same time, a military Hercules aircraft that was ready for the departure as well received permission to taxi in the north-west direction from its parking gate. The Hercules was supposed to take off from runway 36 that crossed with runway 03 that was designated for the Airbus. Both aircraft were taxiing to their runways. During the taxiing, the Airbus received its flight route from the air traffic controllers. Some time later, when the Airbus was near the runway designated for taking off, it switched from the taxiing radio frequency to the frequency of the Tower and received permission to line up on the assigned runway. The Hercules was still at the taxiing radio frequency and also received permission to line up, while at the same time the Airbus received permission to take off at the radio frequency of the Tower. However, due to unknown reasons,[3] *the Hercules pilot interpreted his permission for lining up as permission for taking off and started taking off on runway 36. As a result of this mistake of the pilot of the Hercules, two aircraft were taking off simultaneously on crossing runways, and none of the crews were aware of that. The air traffic controllers in the Tower observed the conflicting situation and communicated a 'STOP' signal to the pilot-in-command of the Airbus, while the Airbus was still on the ground (but at high speed). The pilot had to make a quick decision about the termination of the take-off as there is a point in this process that one cannot safely do this anymore. After having analysed the situation, the pilot-in-command of the Airbus gave a command to the co-pilot (who controlled the aircraft) to abort the take-off and start braking on the runway. During braking, the crew of the Airbus saw the Hercules flying close in the air above their own aircraft at a distance of about 5 m. A serious collision was prevented.*

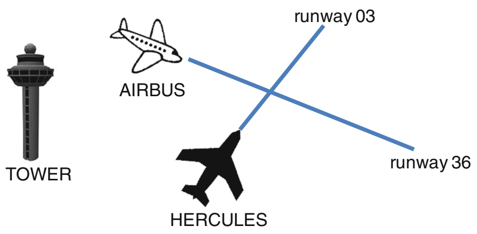

Fig. 1. Schematic overview of the case study.

[3] This misinterpretation might be explained by the fact that the pilot of the Hercules got used to the routine procedure of taxiing from the same military parking place at this airport and perhaps also of taking off from the same runway. And in many past cases, the line up procedure was often immediately followed by taking off, as permissions for lining up and taking off were sometimes given simultaneously.

4 Agent-Based Model

The following subsections describe, respectively, the formal ontology for the case study, a formalized trace of the case study based on this ontology, and a set of executable dynamic properties (or rules) that can be used to simulate the scenario (and variations of it).

4.1 Formal Ontology

As the first step towards the formalization of the incident identified during the interview, formal domain ontology was developed in TTL. In Tables 1 and 2, an overview of the ontology elements is shown, including the relevant sorts and subsorts relations, elements (constants) of sorts, and logical predicates over sorts.

Table 1. Domain ontology: sorts and elements.

SORT	ELEMENTS
AGENT	{tower sub-sorts: PILOT, AIRCRAFT}
PILOT	{airbus_pilot, hercules_pilot}
AIRCRAFT	{hercules, airbus}
ROADWAY	sub-sorts: RUNWAY, TAXIWAY, STARTINGPOINT, CROSSINGPOINT
RUNWAY	{runway_03, runway_36}
TAXIWAY	{taxiway_1, taxiway_2}
STARTINGPOINT	{startingpoint_1, startingpoint_2}
CROSSINGPOINT	{crossing_point(runway_03), crossing_point(runway_36)}
ACTION	{start_taxiing, start_line_up, start_take_off, take_off_from, stop_take_off}
VELOCITY	{low, high, very_high}

As shown in the first three rows of Table 1, the model consists of five active agents that play a role in the scenario (see also Fig. 1): Tower, Airbus Aircraft, Hercules Aircraft, Airbus Pilot and Hercules Pilot. In addition, there are elements of the environment that influence the agents' behavior in the model, such as runways, taxiways and other locations.

4.2 Formal Trace

The informal scenario described in Sect. 3 was formalized using the ontology presented in the previous subsection. A time point was assigned to each event of the case study under consideration.

The time points in the trace indicate the relative timing of the events. The trace was visualized in the LEADSTO software environment [3], as shown in Fig. 2. The states that hold in the world are represented on the vertical axis and the time line on the horizontal axis. The dark lines on the right indicate time intervals within which the given states are true.

Table 2. Domain ontology: logical predicates.

PREDICATE	DESCRIPTION
Communication	
communicate_from_to(A:Agent, B: Agent, C:Action, R:Roadway)	agent A communicates permission for action C on roadway R to agent B
incoming_communication(A: Agent, C:Action, R:Roadway)	agent A receives permission for action C on roadway R
Internal states of agents	
observation(A:Agent, I:Info_El)	agent A observes information element I from the world
belief(A:Agent, I:Info_El)	agent A believes that information element I is true in the world
expectation(A:Agent, C:Action)	agent A has expectation for action C
Actions of agents	
move_from_to(R1: Roadway, R2: Roadway)	action of moving from roadway R1 to roadway R2
performed(A:Agent, C:Action)	agent A performs action C
set_velocity(A:Aircraft, V:Velocity)	aircraft A acquires velocity V
take_off_from(R:Runway)	take-off is performed from runway R
stop_take_off(R:Runway)	take-off from runway R is aborted
Positions of agents	
is_at_position(A:Agent, R:Roadway)	agent A is on roadway R
is_adjacent_to(R1:Roadway, R2:Roadway)	roadway R1 is adjacent to roadway R2
crossing_ways(R1:Roadway, R2:Roadway	roadways R1 and R2 cross
is_half_way(A:Agent,R:Roadway))	agent A is half way on roadway R
in_air(A:Aircraft)	aircraft A is in air
Other information elements used within predicates	
is_available(R:Roadway)	roadway R is available
is_pilot_of(A:Agent, B:Aircraft)	agent A is a pilot of aircraft B
has_role(A:Agent)	an agent has role A
start_taxiing	start taxiing
start_line_up	permission to line up
start_take_off	permission to take off
velocity(A:Aircraft, V:Velocity)	aircraft A has velocity V
has_priority_over(A:Aircraft, B:Aircraft)	aircraft A has priority over aircraft B
not_in_conflict(A1:Agent, A2: Agent)	agent A1 is not in conflict with agent A2
similarity(A1:Action, A2:Action)	action A1 is similar to action A2
velocity(A:Aircraft, V:Velocity)	aircraft A has velocity V
collision(A:Aircraft, B:Aircraft)	aircraft A collides with Aircraft B

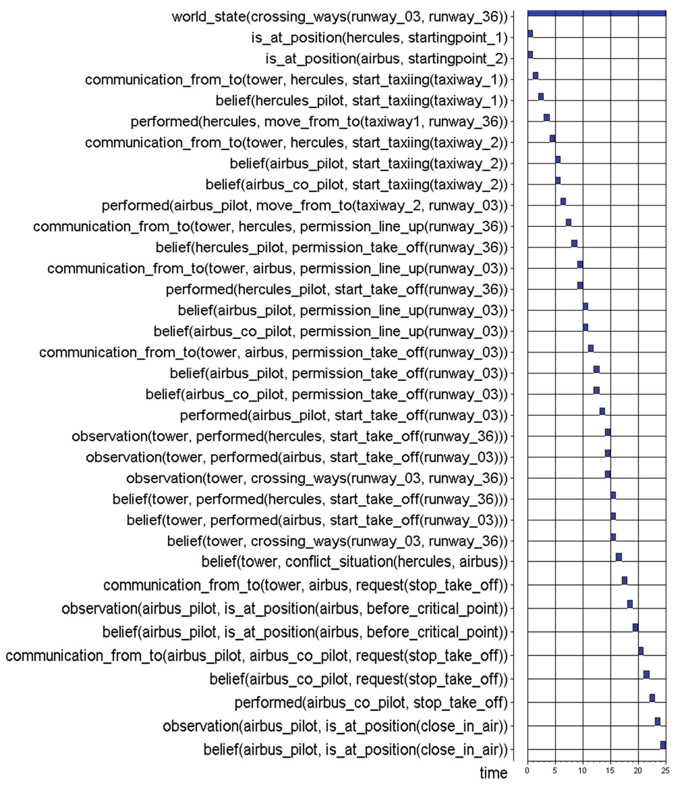

Fig. 2. Formalised empirical trace of the runway incursion incident in LEADSTO.

4.3 Executable Dynamic Properties

This subsection presents an agent-based simulation model of the runway incursion scenario, which consists of a number of executable dynamic properties (EPs) in LEADSTO. This model can be used both to reproduce the trace as shown in the previous subsection, but also (by slightly changing initial parameter settings) to generate a variety of alternative traces. The executable properties can be subdivided into four different categories, namely properties related to (1) belief formation, (2) communicative action generation, (3) physical action generation, and (4) transfer.

Below some examples of properties in formal LEADSTO notation per category are given (for simplicity, the time parameters have been left out). Note that most properties are applied to all agents. Only some of the properties (e.g., EP2, EP6 and EP16) are specific to a particular agent role (e.g., Tower or Pilot).

4.3.1 Belief Formation

Belief formation properties specify how agents create beliefs about the world on the basis of the observations or communications they receive. For instance, EP1 states that, if an agent observes no other agents at a certain roadway, it concludes that this roadway is available.

Belief formation properties may also represent erroneous behavior, e.g. related to cognitive biases such as the ATC expectation bias,[4] which is related to the well known confirmation bias [12]. For example, EP5 states that, if an agent receives an instruction I1, while it has a strong expectation to receive a similar, but slightly different instruction I2, it will believe that it actually did receive I2. This property can be used to model the fact that the Hercules pilot interpreted his permission for lining up as permission for taking off.

EP1 - Belief formation on roadway availability
observation(A:Agent, not_at_position(B:Agent, R:Roadway))
↠ belief(A:Agent, is_available(R:Roadway))

EP5 - Communication misinterpretation
incoming_communication(A:Agent, I1:Action, R:Roadway)
& belief(A:Agent, similarity(I1: Action, I2: Action))
& I1 ≠ I2
& expectation(A:Agent, I2:Action)
↠ belief(A:Agent, I2:Action, R:Roadway)

4.3.2 Communicative Action Generation

These properties specify how agents derive actions to communicate to other agents, based on the beliefs they possess. For instance, EP2 determines when the Tower agent communicates a permission to start taxiing to the different aircraft, whereas EP16 when the Tower communicates a request to abort take-off.

EP2 - Tower: Taxiing request communication
belief(A:Agent, is_at_position(B:Aircraft, S: Startingpoint))
& belief(A:Agent, is_adjacent_to(T:Taxiway, S: Startingpoint))
& belief(A:Agent, is_available(T:Taxiway))
& belief(A:Agent, has_role(tower))
↠ communicate_from_to(A:Agent, B:Aircraft, start_taxiing(T:Taxiway))

EP16 - Tower: Take-off abort request communication
belief(tower, is_half_way(A:Aircraft, R1: Runway))
& belief(tower, is_half_way(B:Aircraft, R2: Roadway))
& belief(tower, crossing_ways(R1:Runway, R2:Roadway))
& belief(tower, velocity(B:Aircraft, high))
& not collision(A:Aircraft, B:Aircraft)
& B ≠ A
↠ communicate_from_to(tower, B:Aircraft, stop_take_off, R1:Runway)

4.3.3 Physical Action Generation

In addition to communicative actions, agents may also derive physical actions. An example of this is represented by property EP6, which determines that pilot agents may start taxiing when they believe this is appropriate.

EP6 - Pilot: Taxiing initiation
belief(P:Pilot, start_taxiing(T:Taxiway)
& is_a _pilot_of(P:Pilot, A:Aircraft)
& belief(P:Pilot, is_available(T:Taxiway))
& is_at_position(A:Aircraft, S:Startingpoint)
& belief(P:Pilot, is_adjacent_to(T:Taxiway, S:Startingpoint))
↠ performed(P:Pilot, move_from_to(S:Startingpoint, T:Taxiway))
& performed(P:Pilot, set_velocity(A:Aircraft, low))

[4] http://www.skybrary.aero/index.php/ATC_Expectation_Bias

4.3.4 Transfer

Finally, transfer properties represent correct transfer of information. For instance, EP3 states that information that is communicated from agent A to agent B is also received as such by agent B (of by the pilot of agent B, if agent B is an aircraft).

EP3 - Communication Transfer
communicate_from_to(A:Agent, B:Agent, I:Action, R:Roadway)
& is_pilot_of(P:Pilot, B:Aircraft)
↠ incoming_communication(P:Pilot, I:Action, R:Roadway)

To enhance readability, only a number of the executable properties per category have been listed. However, the full specification (using the notation of the LEADSTO simulation tool) can be found at http://www.cs.vu.nl/~tbosse/aviation.

5 Simulation Results

This section describes simulation results of the case study across three different scenarios. The first scenario represents the real situation as described in Sect. 3, and the other two scenarios simulate two hypothetical situations that would occur when the perceptions and the actions of the agents involved would slightly differ from the real case. These hypothetical situations were created by making small changes in some of the relevant parameters.

In the simulation traces depicted in Figs. 3, 4, 5, again a time line is represented on the horizontal axis and the states that hold in the world are represented on the vertical axis. For the sake of transparency, the atoms that represent *observations* and *beliefs* of the agents are not depicted in the traces.

5.1 Scenario 1: Intervention of Tower

The simulation trace of scenario 1 is shown in Fig. 3. This scenario simulates the real events of the case study. It represents the situation that the pilot of the Hercules aircraft misinterprets the information that is communicated to him by controllers in the Tower because of an incorrect expectation (see atom expectation(hercules_pilot, start_take_off) at the top of the trace that is true during the whole simulation), and consequently initiates take-off without take-off clearance (see atom performed(hercules_pilot, take_off_from (run-way_36)) that is true from time point 15–21).

There is no atom that states that take-off clearance from the Tower is communicated to the Hercules. At the same time, the clearance for take-off is given to the Airbus aircraft that almost simultaneously initiates take-off from the crossing runway at time point 20; see atom performed(airbus_pilot, take_off_from(runway_03)). Luckily, the Tower observes the conflict situation (this atom is not depicted in the trace) and communicates a "STOP" signal to the Airbus at time point 24. As a result, the pilot of the Airbus aborts the take-off at time point 27 and a severe collision is prevented by this action. As can be seen this scenario is almost identical (with the exception of some minor differences in terminology and timing) to the empirical trace shown in Fig. 2.

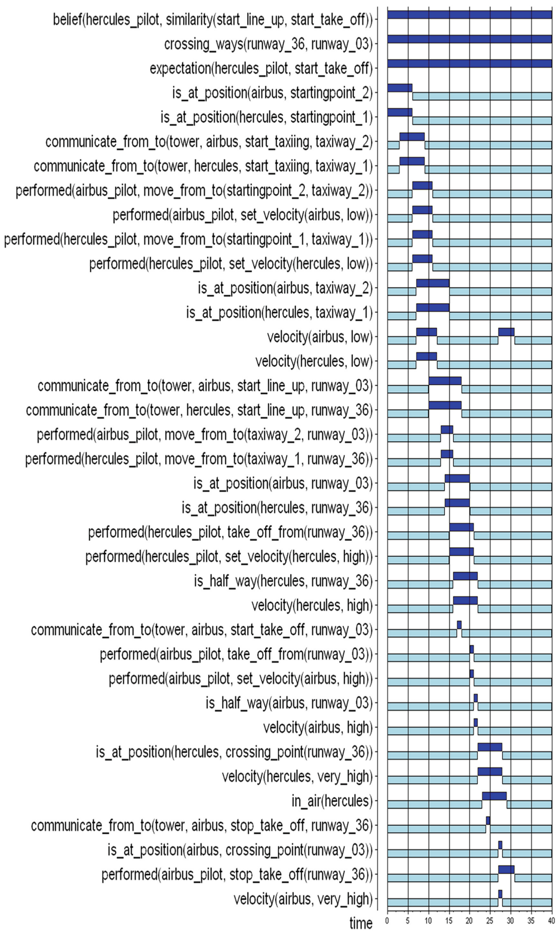

Fig. 3. Simulation results of scenario 1 - intervention of tower prevents severe collision.

Hence, it is an example of a case when a hazardous situation created by the wrong decision and action of one agent can be corrected by appropriate intervention of other agents.

5.2 Scenario 2: Nominal Behaviour

The simulation trace of scenario 2 is shown in Fig. 4. This trace represents an ideal scenario where all agents behave properly. In the initial settings of this hypothetical scenario the pilot of the Hercules has no erroneous expectation about the take-off clearance as in scenario 1. As a result, he performs line-up correctly and does not initiate any take-off, as shown in Fig. 4. After both aircraft have performed line-up on their runways at time point 14, permission to take off is communicated only to the Airbus (see atom communicate_from_to(tower, air-bus, start_take_off, runway_03))). Hence, in this scenario all agents behave according to the nominal prescriptions of the agent system. Consequently, no collision or hazardous situation occurs.

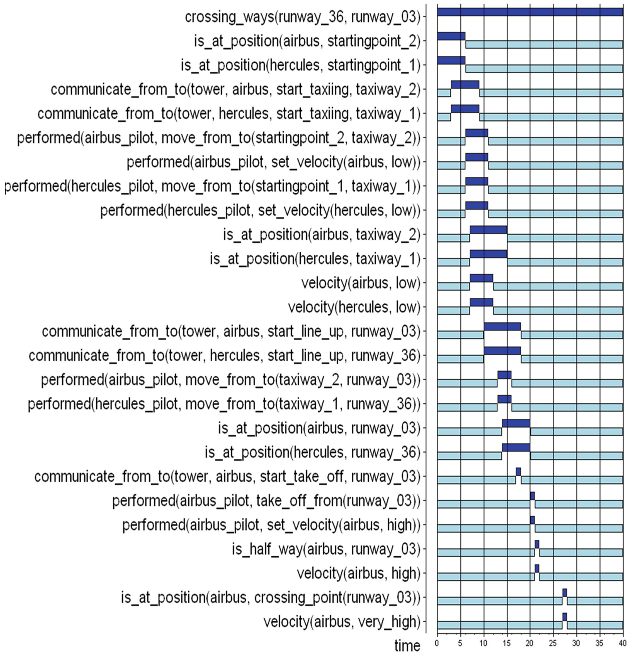

Fig. 4. Simulation results of scenario 2 - Hercules pilot does not make interpretation error.

5.3 Scenario 3: Collision

The simulation of scenario 3 is shown in Fig. 5. This scenario represents a situation when the pilot of the Hercules aircraft has erroneous expectations about the take-off clearance and initiates take-off while he should not (like in scenario 1). However, in this case the controllers in the Tower observe the conflict situation rather late, and therefore they do not have the time to interfere. As a result, both aircraft collide; see atom collision(hercules, airbus) at the end of the trace.

In this scenario the time parameters of the rule that generates the action to take off have been modified in such a way that this action is performed more quickly. This has important consequences for the opportunity of the Tower to interfere and prevent the collision. As can be seen in Fig. 5, the short duration of the take-off procedure leads to severe consequences as both aircraft perform take-off almost simultaneously on crossing runways.

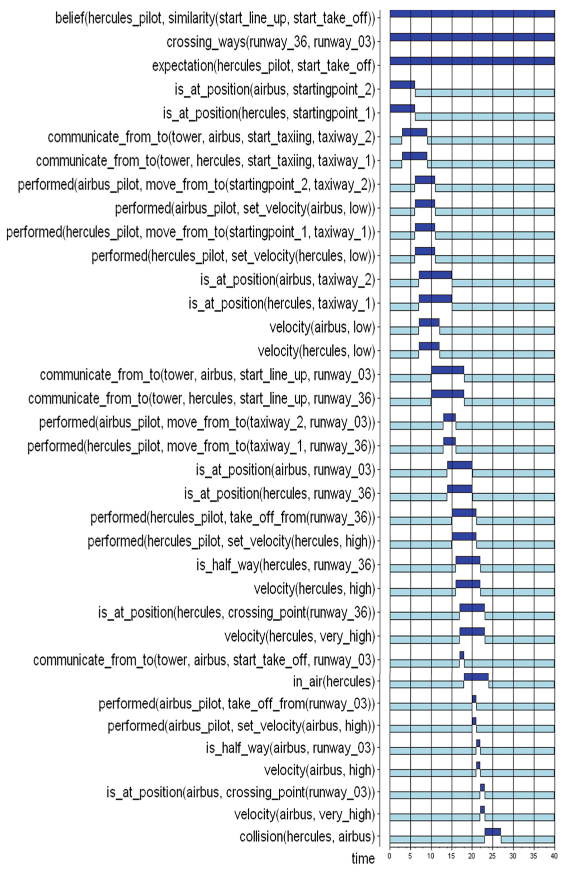

Fig. 5. Simulation results of scenario 3 - interpretation error by Hercules results in severe collision.

6 Formal Analysis

This section addresses formal analysis of the simulated traces. Section 6.1 addresses specification of (global) dynamic properties, Sect. 6.2 address specification of interlevel relations between dynamic properties at different aggregation levels, and Sect. 6.3 discusses some results of verification of properties against empirical and simulated traces.

6.1 Global Dynamic Properties

Various dynamic properties for the aviation domain have been formalized in TTL, a number of which are introduced below. All of these properties are related in some way to the occurrence of collisions. More specifically, Sect. 6.1.1 addresses properties that relate to the fact that 'there are never two simultaneous take-offs at crossing runways'. Section 6.1.2 addresses properties that relate to the fact that 'IF any of such simultaneous take-offs occur, THEN they will be corrected on time because one of the aircraft aborts its take-off'. It is easy to see that either one of these cases is sufficient to guarantee that no runway incursions will occur (assuming for simplicity that simultaneous take-offs are the only ways in which runway incursions can possibly occur). All properties in Sect. 6.1.1 are presented both in semi-formal and in formal (TTL) notation; to enhance readability, the properties in Sect. 6.1.2 are presented only in semi-formal notation.

Note that the properties presented below address processes at different aggregation levels, thereby distinguishing global properties about the entire scenario (indicated by GP), intermediate properties about input and output states of individual agents (indicated by IP), and local properties about mental processes of agents or about information/communication transfer between agents (indicated by LP). As will be explained in Sect. 6.3, this distinction enables the analyst to apply a diagnostic process in which the causes of global system failures can be related to more local fault and errors.

6.1.1 Absence of Simultaneous Take-Offs

GP1 - No simultaneous take-offs at crossing runways

There are no trace m, time points t1 and t2, agents a1 and a2, and runway r1 and r2 such that
agent a1 performs a take-off on runway r1 at time t1
and agent a2 performs a take-off on runway r2 at time t2
and runway r1 and r2 are crossing runways
and the difference between t1 and t2 is smaller than or equal to d[5].

```
¬ [∃m:TRACE ∃t1,t2:TIME ∃a1,a2:AGENT ∃r1,r2:RUNWAY
    state(m, t1) |= performed(a1, take_off_from(r1)) &
    state(m, t2) |= performed(a2, take_off_from(r2)) &
    state(m, t1) |= world_state(crossing_ways(r1, r2)) &
    | t1 - t2 | ≤ d ]
```

IP1 - No simultaneous permissions to take off at crossing runways

There are no trace m, time points t1 and t2, agents a1 and a2, and runway r1 and r2 such that
the tower gives agent a1 permission for take-off on runway r1 at time t1
the tower gives agent a2 permission for take-off on runway r2 at time t2
and runway r1 and r2 are crossing runways
and the difference between t1 and t2 is smaller than or equal to d.

```
¬ [∃m:TRACE ∃t1,t2:TIME ∃a1,a2:AGENT ∃r1,r2:RUNWAY
    state(m, t1) |= communicate_from_to(tower, a1, start_take_off(r1)) &
    state(m, t2) |= communicate_from_to(tower, a2, start_take_off(r2)) &
    state(m, t1) |= world_state(crossing_ways(r1, r2)) &
    | t1 - t2 | ≤ d ]
```

[5] Many of the properties given in this section contain some parameters d and e. These should be seen as constants, of which the value can be filled in by the modeller.

IP2 - Each take-off is preceded by a corresponding permission
For all traces m, time points t1, agents a, and runways r
if agent a performs a take-off on runway r at time t
then there was a time point t2 with t1-d ≤ t2 ≤ t1 on which
the tower gave agent a permission for take-off on runway r.

∀m:TRACE ∀t:TIME ∀a:AGENT ∀r:RUNWAY
state(m, t1) |= performed(a, take_off_from(r)) ⇒
[∃t2:TIME state(m, t2) |= communicate_from_to(tower, a, start_take_off(r)) &
t1-d ≤ t2 ≤ t1]

LP1 - Each take-off is preceded by a corresponding belief
For all traces m, time points t1, agents a, and runways r
if agent a performs a take-off on runway r at time t
then there was a time point t2 with t1-d ≤ t2 ≤ t1 on which
agent a believed that it had permission for take-off on runway r.

∀m:TRACE ∀t:TIME ∀a:AGENT ∀r:RUNWAY
state(m, t1) |= performed(a, take_off_from(r)) ⇒
[∃t2:TIME state(m, t2) |= belief(a, start_take_off(r)) &
t1-d ≤ t2 ≤ t1]

LP2 - Each belief about permissions is preceded by a corresponding communication
For all traces m, time points t1, agents a, and runways r
if agent a believes that it has permission for take-off on runway r at time t
then there was a time point t2 with t1-d ≤ t2 ≤ t1 on which
the tower gave agent a permission for take-off on runway r.

∀m:TRACE ∀t:TIME ∀a:AGENT ∀r:RUNWAY
state(m, t1) |= belief(a, start_take_off(r)) ⇒
[∃t2:TIME state(m, t2) |= communicate_from_to(tower, a, start_take_off(r)) &
t1-d ≤ t2 ≤ t1]

6.1.2 Correction of Simultaneous Take-Offs

GP2 - All simultaneous take-offs are corrected on time
For all traces m, time points t1 and t2, agents a1 and a2, and runways r1 and r2,
if agent a1 performs a take-off on runway r1 at time t1
and agent a2 performs a take-off on runway r2 at time t2
and runway r1 and r2 are crossing runways
and the difference between t1 and t2 is smaller than or equal to d
then there is a time point t3 with t1 ≤ t3 ≤ t1+e and t2 ≤ t3 ≤ t2+e on which either agent a1 or agent a2 aborts take-off.

IP3 – For all simultaneous take-offs that are observed an abort request is communicated
For all traces m, time points t1 and t2, agents a1 and a2, and runways r1 and r2,
if at time t1the tower observes that agent a1 performs a take-off on runway r1
and at time t2 the tower observes that agent a2 performs a take-off on runway r2
and runway r1 and r2 are crossing runways
and the difference between t1 and t2 is smaller than or equal to d
then there is a time point t3 with t1 ≤ t3 ≤ t1+e and t2 ≤ t3 ≤ t2+e on which the tower communicates either to agent a1 or to agent a2 a request to abort take-off.

IP4 - All received abort requests are followed
For all traces m, time points t1, agents a1 and a2, and runways r1,
if at time t1agent a1 receives from agent a2 a request to abort take-off from runway r1
then there is a time point t2 with t1 ≤ t2 ≤ t1+d on which agent a1 indeed aborts take-off from r1.

LP3 - All simultaneous take-offs are observed
For all traces m, time points t1 and t2, agents a1 and a2, and runways r1 and r2,
if agent a1 performs a take-off on runway r1 at time t1
and agent a2 performs a take-off on runway r2 at time t2
and runway r1 and r2 are crossing runways
and the difference between t1 and t2 is smaller than or equal to d
then there are two time points t3 and t4 with t1 ≤ t3 ≤ t1+e and t2 ≤ t4 ≤ t2+e on which the tower observes both take-offs.

LP4 - All communicated abort requests are received
For all traces m, time points t1, agents a1 and a2, and runways r1,
if at time t1agent a1 communicates to agent a2 a request to abort take-off from runway r1
then there is a time point t2 with t1 ≤ t2 ≤ t1+d on which this request is received from a1 by 2.

LP5 - All observed take-offs are converted into corresponding beliefs
For all traces m, time points t1, agents a1, and runways r1,
if at time t1 the tower observes that agent a1 performs a take-off on runway r1
then there is a time point t2 with t1 ≤ t2 ≤ t1+d on which the tower believes that agent a1 performs a take-off on runway r1.

LP6 – For all beliefs on simultaneous take-offs an abort request is communicated
For all traces m, time points t1 and t2, agents a1 and a2, and runways r1 and r2,
if at time t1the tower believes that agent a1 performs a take-off on runway r1
and at time t2 the tower believes that agent a2 performs a take-off on runway r2
and runway r1 and r2 are crossing runways
and the difference between t1 and t2 is smaller than or equal to d
then there is a time point t3 with t1 ≤ t3 ≤ t1+e and t2 ≤ t3 ≤ t2+e on which the tower communicates either to agent a1 or to agent a2 a request to abort take-off.

LP7 - All received requests are converted into corresponding beliefs
For all traces m, time points t1, agents a1 and a2, and runways r1,
if at time t1agent a1 receives from agent a2 a request to abort take-off from runway r1
then there is a time point t2 with t1 ≤ t2 ≤ t1+d on which agent a1 believes that it should abort take-off from r1.

LP8 - All believed requests are followed
For all traces m, time points t1, agents a1, and runways r1,
if at time t1agent a1 believes that it should abort take-off from runway r1
then there is a time point t2 with t1 ≤ t2 ≤ t1+d on which agent a1 indeed aborts take-off from r1.

6.2 Interlevel Relations

A number of logical relationships have been identified between properties at different aggregation levels. An overview of all identified logical relationships relevant for GP1 is depicted as an AND-tree in Fig. 6.

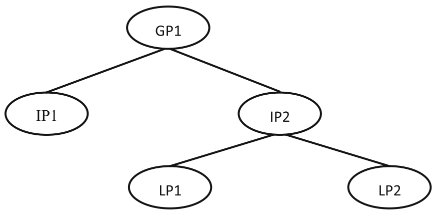

Fig. 6. AND-tree of interlevel relations between dynamic properties related to GP1.

The relationships depicted in this figure should be interpreted as semantic entailment relationships. For example, the relationship at the highest level expresses that the implication IP1 & IP2 => GP1 holds, whereas the relationship at the lower level expresses that LP1 & LP2 => IP2 holds. A sketch of the proof for the first implication is as follows (for simplicity reasons abstracting from time constrains):

Suppose that IP1 and IP2 hold. Then, according to IP1, no two permissions to take off at crossing runways will be communicated simultaneously. Moreover, since take-offs are only performed immediately after a corresponding permission has been communicated (as guaranteed by IP2), no simultaneous take-offs are performed at crossing runways. This confirms GP1.

Such logical relationships between dynamic properties can be very useful in the analysis of (both simulated as well as empirical) scenarios, especially when used in combination with the TTL Checker Tool mentioned earlier. For example, for the empirical trace 1, checking GP1 pointed out that this property was not satisfied. As a result, by a refutation process (following the tree in Fig. 6 top-down) it could be concluded that either IP1 or IP2 failed (or a combination of them). When, after further checking, IP2 was found to be the cause of the failure, the analysis could proceed by focusing on LP1 and LP2. Eventually, LP1 was found satisfied, whereas LP2 failed. Thus, (part of) the source of the incident could be reduced to failure of LP2, i.e., there was an agent (namely the pilot of the Hercules) that believed to have the permission to take off, whilst this was not communicated by the tower. A discussion with our domain expert confirmed that this was indeed the case. One level deeper, such local properties can even be related to executable properties. For instance, the failure of LP2 can be explained because the Hercules pilot applied property EP5. A full connection of local properties to executable properties is beyond the scope of this paper, but a detailed discussion can be found in [10].

Similar to Fig. 6, an AND-tree representing all identified logical relationships relevant for GP2 is shown in Fig. 7.

Note that the example scenario provided here is mainly meant as an illustration of the approach. In addition to this relatively simple case, similar trees of interlevel relations are being constructed that involve more properties at multiple levels. For such more complex cases, the diagnostic process is economic in the sense that, when a certain property holds, the entire subtree under this property does not have to be examined.

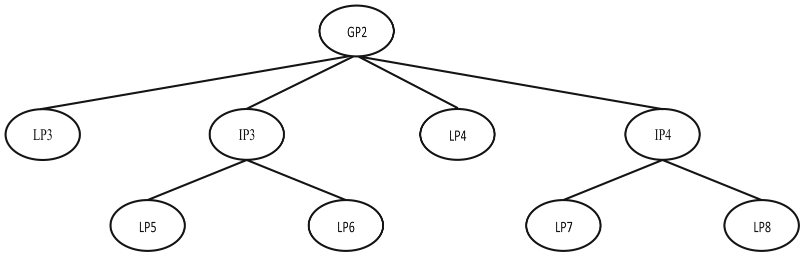

Fig. 7. AND-tree of interlevel relations between dynamic properties related to GP2.

6.3 Checking Results

Using the TTL Checker, all dynamic properties introduced in Sect. 6.1 have been checked against the three simulation traces discussed in Sect. 5 as well as the empirical trace discussed in Sect. 4.2. The results of these checks are shown in Table 3 (where ‘X’ denotes ‘satisfied’). As can be seen from the table, scenario 2 is indeed a nominal case in which all expected properties hold. In contrast, in scenario 1, two simultaneous take-offs at crossing runways occur (since GP1 fails), which can eventually be related to an incorrectly derived belief of permission for take-off (failure of LP2). However, since the situation is corrected on time (GP2 succeeds), no collision occurs in this scenario. In scenario 3, GP1 also fails, but in addition GP2 fails, which can be related partly to failure of LP3 (the simultaneous take-offs are observed, but too late) and to failure of LP6 (once the tower believes that there are simultaneous take-offs, it is too late to communicate an abort request). As a result, the collision is not prevented. As can be seen, the same system properties failed for the empirical trace as for scenario 1, which makes sense because these scenarios are identical.

Table 3. Checking dynamic properties against (simulated and empirical) traces.

Property	Scenario 1	Scenario 2	Scenario 3	Empirical
GP1	–	X	–	–
IP1	X	X	X	X
IP2	–	X	–	–
LP1	X	X	X	X
LP2	–	X	–	–
GP2	X	X	–	X
IP3	X	X	–	X
IP4	X	X	X	X
LP3	X	X	+/−	X
LP4	X	X	X	X
LP5	X	X	X	X
LP6	X	X	–	X
LP7	X	X	X	X
LP8	X	X	X	X

7 Discussion

In this paper, an agent-based approach was introduced for analysis of the dynamics of accidents and incidents in aviation. Although agent-based modeling and simulation has been widely applied to study complex systems (see, e.g. [8]), it has not yet been commonly accepted within the domain of aviation safety (see [6]). The presented approach makes use of a number of elements, including formalization of a real world scenario, agent-based simulation of variations of the scenario, and formal verification of dynamic properties against the (empirical and simulated) scenarios. The scenario formalization part enables incident reconstruction and formal analysis of it. The simulation part enables the analyst to explore various hypothetical scenarios under different circumstances, with an emphasis on error related to human factors. The formal verification part enables the analyst to identify scenarios involving potential hazards, and to relate those hazards (via interlevel relations) to inadequate behavior on the level of individual agents. The approach was illustrated by means of a case study on a runway incursion incident.

To obtain information about this incident, an interview with a domain expert was conducted. In the current paper, this interview was used as the only source of information to construct the formal ontology and the traces. Nevertheless, for more complex case studies it may be interesting to consider more extensive knowledge elicitation techniques, involving a larger number of experts. Some example case studies in which such techniques were applied in the aviation domain are reported in [5] and [9].

The approach introduced in the current paper in principle addressed both *retrospective* and *prospective* analysis of scenarios. In particular, the possibility to check properties against formalized empirical traces is an adequate way to analyze past scenarios. This has been illustrated by the analysis of the empirical trace shown in Sect. 4.2, which was a formalization of a real world scenario. In contrast, the possibility to run simulations and to check properties against the simulated traces enables analysts to study future scenarios as well. This has been illustrated by the analysis of the simulation traces in Sect. 5, most of which addressed hypothetical future scenarios. Nevertheless, one should keep in mind that the scope of the prospective analysis is restricted to those elements that are part of the formal ontology.

For a more quantitative type of agent-based dynamic risk analysis, often Monte Carlo methods are applied; see e.g. the work of Blom et al. [1] or Stroeve, Blom and Bakker [14]. These methods are very useful for quantitative collision risk estimations, but one of their disadvantages is lack of transparency due to the complex stochastic relations between the elements of the agent-based models that are used. In contrast, the approach presented in this paper is highly transparent; it provides a visible trace of risk related events that can be analyzed manually or automatically with the help of special tools. Moreover, the roles of the agents involved in risk creation and reduction (as well as their underlying cognitive processes, like the influence of biased reasoning) are clear from the trace, while in dynamic quantitative risk models used for Monte Carlo simulations this is usually not the case. The complexity of Monte Carlo methods makes it also difficult for the non-specialist to understand the implications of actions and thus makes a public debate of issues a problem. However, a disadvantage of the method proposed in this paper is that it cannot provide a precise risk estimation as is provided by Monte Carlo methods. In follow-up research, we therefore intend to explore the possibilities to combine our approach with elements from Monte Carlo methods.

Another promising direction for future work would be to automate more steps in the analysis method. For example, the ontology developed in step 1 of the methodology could make use of standard templates for common agent-based concepts, such that it does not have to be designed by hand for any new domain. Similarly, templates for dynamic properties could be developed, as well as computer-supported techniques to automate parts of step 4 (e.g., systematically generating large numbers of simulation runs) and step 7 (e.g., systematically checking multiple properties against large sets of traces).

As mentioned earlier, agent-based modeling and simulation has been widely applied outside the aviation domain, and the state-of-the-art in this area is extensive. A comparison with other (agent-based) dynamic modeling approaches is therefore outside the scope of this article; for this purpose, the interested reader is referred to [2, 3].

Acknowledgements. This work was performed under the auspices of the SESAR WP-E research network ComplexWorld. It is co-financed by Eurocontrol on behalf of the SESAR Joint Undertaking. The authors are grateful to the retired airline pilot who participated in the interview for his useful input on the case study, and to Jan Treur for a number of fruitful discussions.

References

1. Blom, H.A.P., Bakker, G.J., Blanker, P.J.G., Daams, J., Everdij, M.H.C., Klompstra, M.B.: Accident risk assessment for advanced air traffic management. In: Donohue, G.L., Zellweger, A.G. (eds.) Air Transport Systems Engineering, pp. 463–480. AIAA, Washington, D.C. (2001)
2. Bosse, T., Jonker, C.M., van der Meij, L., Sharpanskykh, A., Treur, J.: Specification and verification of dynamics in agent models. Int. J. Coop. Inf. Syst. **18**(1), 167–193 (2009)
3. Bosse, T., Jonker, C.M., van der Meij, L., Treur, J.: A language and environment for analysis of dynamics by simulation. Int. J. Artif. Intell. Tools **16**(3), 435–464 (2007)
4. Bosse, T., Mogles, N.M.: Formal analysis of aviation incidents. In: Jiang, H., Ding, W., Ali, M., Wu, X. (eds.) IEA/AIE 2012. LNCS, vol. 7345, pp. 371–380. Springer, Heidelberg (2012)
5. Bosse, T., Treur, J., Mogles, N.M., Stroeve, S.H., Blom, H.A.P., Sharpanskykh, A.: Model constructs validation. SESAR Joint Undertaking. Technical report E.02.10-MAREA-D3.1 (2013)
6. Everdij, M.H.C.: Review of techniques to support the EATMP safety assessment methodology. Report for EEC Safety Methods Survey Project, volume I and II (2004)
7. Hollnagel, E.: Barriers and Accident Prevention. Ashgate, Aldershot (2004)
8. Jennings, N.R.: On agent-based software engineering. Artif. Intell. **117**, 277–296 (2000)
9. de Jong, H.H., Blom, H.A.P., Stroeve, S.H.: How to identify unimaginable hazards? In: 25th International System Safety Conference (ISSC 2007), Baltimore, USA (2007)
10. Jonker, C., Treur, J.: Compositional verification of multi-agent systems: a formal analysis of pro-activeness and reactiveness. Int. J. Coop. Inf. Syst. **11**, 51–92 (2002)
11. Leveson, N.: A new accident model for engineering safer systems. Saf. Sci. **42**, 237–270 (2004)
12. Nickerson, R.S.: Confirmation bias: a ubiquitous phenomenon in many guises. Rev. Gen. Psychol. **2**(2), 175–220 (1998)
13. Rao, A.S., Georgeff, M.P.: BDI-agents: from theory to practice. In: Lesser, V. (ed.) Proceedings of the International Conference on Multiagent Systems, pp. 312–319 (1995)
14. Stroeve, S.H., Blom, H.A.P., Bakker, G.J.: Systemic accident risk assessment in air traffic by Monte Carlo simulation. Saf. Sci. **47**, 238–449 (2009)
15. Wooldridge, M.: Agent-based software engineering. IEE Proc. Softw. Eng. **144**(1), 26–37 (1997)