

Formal, Executable and Explainable Runtime Monitoring of Spoken Air Traffic Control Operational Procedures

Roberto Luvini, Giacomo Longo, Alessandro Armando, and Enrico Russo

Abstract—Air traffic control procedures are executed through spoken exchanges between controllers and pilots. These interactions are essential to the safety of air transportation: failures in their execution can create severe operational hazards, as evidenced by past fatal accidents. Assessing whether an instruction has been followed requires relating what was said to the aircraft concerned, its state, and the obligations that pilots must meet. We present a runtime verification framework that monitors such procedures by checking controller-pilot exchanges, surveillance data, and onboard observations. The framework parses radio communications into events linked to the entities they concern and merges them with surveillance and onboard observations into a time-stamped trace. The ICAO-derived obligations as formalized as temporal formulas with explicit time bounds and evaluated over execution traces. Every violation is reported along with the breached obligations and the observations that support the verdict. With real traffic, the complete pipeline reaches an F1 of 0.85 against blind human-annotated violations; in 1,495 synthetic situations derived from two public corpora, the monitor logic returns the expected verdict in every case. In two historical accidents reconstructed from official investigation reports, the monitor identifies the same procedural deviations documented by the investigators.

Index Terms—Air traffic control, Runtime verification, Formal methods, Methods for safety, Air transportation.

I. INTRODUCTION

TRANSPORTATION procedures coordinate vehicle operators and control centers. In aviation, these roles correspond to pilots and air traffic control (ATC): controllers issue clearances and instructions, pilots acknowledge and execute them, and both rely on standardized phraseology. These exchanges create obligations that must be interpreted, tracked, and verified as the situation evolves.

For ATC, determining whether an instruction has been followed cannot rely on the transcript alone. It requires interpreting what was said, by whom, to whom it was addressed, whether it was acknowledged, and whether the aircraft state evolved accordingly. This involves combining speech content, roles, aircraft identity, system state, and timing, as well as reconstructing the obligations created by the radio exchange and verifying them against operational evidence. When a deviation occurs, the controller must be able to identify which obligation failed and which observations support that

conclusion. Consequently, the safety-critical activity of determining whether ongoing pilot-controller interactions satisfy all applicable obligations ultimately relies on human operators.

In this paper, we make this reasoning explicit through four monitoring requirements: *observation grounding* ($\mathbf{R}_1$), which links each utterance to the aircraft and operational entity it concerns, such as a runway or vertical clearance; *multi-source integration* ($\mathbf{R}_2$), which relates the radio exchange to surveillance and onboard observations; *temporal and timeliness reasoning* ($\mathbf{R}_3$), which tracks and relates acknowledgments, active obligations, deadlines, and superseded clearances; and *evidence-backed verdicts* ($\mathbf{R}_4$), which connect each deviation to the corresponding failed obligation and observations.

These requirements reflect routine checks in ATC, but meeting them continuously across multiple aircraft and radio exchanges under stringent real-time constraints is increasingly challenging as traffic grows and staffing often remains constrained. Worldwide scheduled operations reached 37 million departures in 2024 [1]. In Europe, ATC capacity and staffing accounted for more than half of the 22.4 million minutes of en-route delay in 2024, the worst level since 2001 [2]. Such workload pressures occur in a safety-critical domain, where runway collisions remain a major risk [3] and procedural deviations have contributed to documented accidents, including the cases examined in §III.

Automated assistance could help address this burden, but only if spoken exchanges can be checked against the evolving operational state. Advances in speech transcription and Large Language Model (LLM) based semantic parsing now make real-time extraction of operational events from radio communications feasible and automated support for spoken ATC procedure monitoring an active research direction. However existing approaches cover the above requirements only in part.

We close this gap with a novel runtime verification framework that turns spoken radio exchanges into entity-linked operational events, combines them with surveillance and onboard state in a formal model of the procedure, and evaluates the resulting trace to produce time- and evidence-linked verdicts. Encoding the procedures as temporal formulas gives them a mathematically precise semantics and makes compliance machine-checkable.

This paper presents the following key contributions:

- A temporal formalism for encoding spoken operational procedures as formal specifications with metric time bounds and precedence among obligations; to the best of our knowledge, the first temporal-logic formalization of controller and pilot obligations.
- An automated trace-construction pipeline that transforms spoken radio exchanges and heterogeneous operational

This work has been submitted to the IEEE for possible publication. Copyright may be transferred without notice, after which this version may no longer be accessible. Manuscript received ...; R. Luvini (Corresponding author) and A. Armando are with the Department of Informatics, Bioengineering, Robotics, and Systems Engineering (DIBRIS), University of Genoa, Italy. Email: S4979483@studenti.unige.it; alessandro.armando@unige.it. G. Longo and E. Russo are with CASD - University School of Advanced Defense Studies, Rome, Italy. Email: {name.surname}@unicas.it.

data into unified execution traces.

- A verification framework that instantiates and evaluates these specifications over operational traces, producing verdicts linked to the violated obligation and supporting observations.
- An empirical evaluation on real traffic and synthetic situations, showing that the framework applies to genuine controller-pilot communications and, in particular, detects rare safety-critical configurations under controlled conditions.
- A validation of the framework on two documented accidents, Überlingen (2002) and Comair 5191 (2006), reconstructed from the official investigation reports, showing that it identifies the reported procedural deviations with a time margin before each impact.

The framework supports three operational uses: (i) *real-time assistance*, running alongside operations to flag anomalies as they arise; (ii) *post-operation review*, auditing recorded traces for safety assessments; and (iii) *controller training*, providing active feedback on procedural compliance. Its deployment as an operational technology could significantly improve aviation safety by reducing the risk that procedural deviations go undetected, a recurring contributing factor in serious accidents.

Structure of the paper. The rest of the paper is organized as follows. In §II, we provide background on ATC procedures and on temporal-logic specifications over finite traces. §III presents two motivating examples for the requirements. In §IV, we introduce our formal model for procedural trace verification and, in §V, the architecture that implements it. We evaluate the framework in §VI, discuss the results in §VII, compare with the related work in §VIII, and conclude the paper in §IX.

II. BACKGROUND

A. Air traffic control procedures and monitoring sources

In ATC, a controller is responsible for preventing collisions between aircraft and for maintaining an orderly flow of traffic [4, §2.2, §2.3.1]. The controller and the pilots communicate by voice over a radio channel, exchanging spoken messages in real time as each flight progresses. On this channel, each aircraft is addressed by its radio identifier, namely the *call sign*, typically an operator designator followed by a flight number [5], e.g., AZA545. These exchanges are not free-form. ICAO Doc 4444 [6] codifies both the procedure and the phraseology used to express clearances, instructions, and acknowledgments [6, Ch. 12]. A *clearance* authorizes an aircraft to proceed under conditions specified by the controller, whereas an *instruction* directs the pilot to perform a specific action [6, Ch. 1], such as adopting a particular heading, speed, or assigned level. Here, a *level* denotes the aircraft's assigned vertical position and is stated as a flight level (FL), referenced to a standard pressure datum [6, Ch. 1], e.g., FL120. The pilot acknowledges the safety-related parts of these messages by reading them back to the controller, who listens and corrects any discrepancy the readback reveals [6, §4.5.7.5].

Instruction compliance cannot be assessed from the radio exchange alone. It requires information on the aircraft state,

e.g., position and speed, obtained either by external observation, as in radar surveillance, or automatically transmitted by automatic dependent surveillance-broadcast (ADS-B) [7, Ch. 1]. Thus, the radio exchange specifies the intended action, whereas surveillance provides the observed behavior of the aircraft. Compliance is the comparison of the two.

Operational monitoring today relies on surveillance-derived safety nets: short-term conflict alert warns the controller of a predicted infringement of separation minima within its warning time [8], and, on the airport surface, advanced surface movement guidance and control systems alert the controller to runway incursions [9].

Aircraft behavior may also be constrained by onboard safety systems. The traffic alert and collision avoidance system (TCAS) is a transponder-based airborne collision-avoidance system defined by International Civil Aviation Organization (ICAO) [10]. It tracks nearby aircraft and, independently of ATC, issues a resolution advisory (RA) instructing the pilot to climb or descend when a collision risk becomes imminent.

B. Temporal-logic specifications over finite traces

Linear temporal logic (LTL) is a formal language for describing properties of how a system behaves over time, and for checking whether a given behavior satisfies them [11]. Its basic facts are *atomic propositions* (atoms, for short): elementary statements that are either true or false at a single point of the behavior. A *temporal formula* combines atomic propositions through the boolean connectives and a set of *temporal operators* that relate points of the behavior across time. The standard operators are *globally* (G), a property holding at the current point and every later one; *next* (X), a property holding at the immediately following point; *until* (U), one property holding until a second occurs; *weak until* (W), the same without requiring the second to occur; and *eventually* (F), a property holding at the current point or some later one.

A formula is evaluated over a *trace*, a sequence of observations produced by the behavior. Each observation records which atomic propositions hold at that point. Formally, this evaluation is defined by a *satisfaction relation*, which states whether a formula holds at a given *position* of the trace. Temporal operators evaluate the sequence of observations based on their relative position, establishing the constraints imposed by a temporal property. Classical LTL is interpreted over infinite traces, whereas linear temporal logic on finite traces (LTL_f) [12] evaluates temporal specifications over finite traces, where the sequence of observations has a last position.

The temporal operators of LTL and LTL_f constrain observations occurrence and ordering, but not the elapsed time between them. Metric temporal logic (MTL) extends temporal logic with explicit time bounds on temporal requirements [13]. Runtime verification (RV) checks an observed behavior against a temporal-logic specification [14], with monitors for data values, finite traces, and onboard use [15]–[18].

III. CASE STUDIES

A. Selection and operational coverage

We use two well-documented accidents as validation scenarios. *Überlingen (2002)* involved two aircraft, at the same

level on crossing tracks; all 71 people on board died [19]. In *Comair 5191* (2006), an aircraft cleared to taxi to runway 22 at Lexington instead took off from the shorter, unlit runway 26 and overran it; 49 of the 50 people on board died [20].

We selected these two cases because they are complementary and well documented: they cover airborne and surface operations, combine radio, surveillance, and onboard observations in different proportions, and both stem from failures to meet procedural obligations. Each was reconstructed in detail by an authoritative investigation carried out by Germany's BFU and the U.S. NTSB, respectively.

B. Monitoring requirements exposed by the cases

The official reports first show the need for evidence-backed verdicts ($\mathbf{R}_4$): as post-hoc reconstructions, they link each procedural deviation to the governing obligation, aircraft, instant, and recorded observations. The analysis below shows why detecting these deviations requires $\mathbf{R}_1$ – $\mathbf{R}_3$.

Überlingen. The deviation is invisible in the transcript. It appears only when the controller's descent instruction and the collision-avoidance advisories are associated with the instructed aircraft ($\mathbf{R}_1$) and related to its observed vertical motion ($\mathbf{R}_2$). During the final minute, an advisory commanded a climb while the controller's standing instruction required descent [19]. Under ICAO collision-avoidance procedures, an active advisory takes precedence over controller instructions [6, §12.3.1.2] [10, §5.2.1.14]. So which obligation governed depends on whether each instruction was issued before or after the advisory became active ($\mathbf{R}_3$).

Comair 5191. The deviation becomes apparent only after associating the taxi clearance and its assigned runway with the correct aircraft ($\mathbf{R}_1$), and comparing that authorization with the runway actually occupied ($\mathbf{R}_2$). The procedural failure also depends on event order: the aircraft entered runway 26 and began its takeoff roll without an authorization for that runway. Detecting the deviation requires tracking whether the relevant authorization existed before runway entry and takeoff ($\mathbf{R}_3$).

IV. FORMAL MODEL FOR PROCEDURAL TRACE VERIFICATION

The procedures considered in this work involve interactions between air traffic controllers, pilots, aircraft states, and safety-critical onboard systems. At the operational level, they refer to domain-specific facts such as issued clearances, pilot readbacks, assigned flight levels, and runway authorizations, codified by the ICAO in Doc 4444, together with observed aircraft positions and RAs issued by the TCAS (see §II-A). For automated verification, these aeronautical concepts must be represented in a uniform, mathematically precise form.

To this end, we distinguish between the operational vocabulary and its formal abstraction. The operational vocabulary identifies the safety-relevant concepts that may appear in ATC procedures. The formal abstraction represents these concepts using standard temporal-logic entities: atomic propositions, finite timed traces, and temporal formulas (see §II-B).

In our setting, atomic propositions are parameterized by aeronautical entities (e.g., aircraft identifiers, flight levels, and

runways), timed traces collect the propositions that hold at each time instant, and temporal formulas encode the monitored procedural requirements. Moreover, the observed behavior is finite, since a flight, radio exchange, or surveillance recording eventually ends. We therefore adopt the finite-trace semantics of LTL_f . Because procedural requirements may also impose deadlines, we attach timestamps to observations and bound temporal operators by elapsed time, in the manner of MTL. This trace-based view connects the operational vocabulary to temporal formulas for procedural compliance. Based on this abstraction, we define the formal model used for procedural trace verification as

$$\mathcal{M} = \langle \mathcal{AP}, \Pi, \Phi, \models \rangle$$

where $\mathcal{AP}$ is a set of atomic propositions; Π is the set of finite timed traces; Φ is the finite-trace temporal language used to encode procedural requirements; and $\models$ is the satisfaction relation used to evaluate formulas at positions.

The following subsections detail each component.

A. Atomic propositions

The set $\mathcal{AP}$ collects the facts that can be observed and used in the temporal formulas. Its propositions are the ground atomic formulas of a many-sorted first-order vocabulary. The sorts are the kinds of aeronautical entities that the procedures refer to, such as aircraft, flight levels, headings, speeds, and runways, together with a sort *Agent* for the speakers and a sort *Utterance* for the spoken content; for a given recording, each entity sort comes with the finite set of constants naming the observed values, such as the call signs heard on the radio.

Facts asserted by a person are built around the predicate *Says*, of sort *Agent* $\times$ *Utterance*: the constants *atc* and *pilot* of sort *Agent* denote the two speaking roles, and each spoken entry of the operational vocabulary contributes a function symbol that builds the uttered content, such as *cmd_descend* of sort *Aircraft* $\times$ *Level* $\rightarrow$ *Utterance*. Facts established during trace construction, whether observed by surveillance or computed, are expressed by predicates over the same sorts, such as *descending* over *Aircraft*. Computed facts are obtained from the observed ones, spoken or surveillance-derived, through fixed relations; these relations, together with the vocabulary, define the framework's domain ontology.

A controller descent instruction, for instance, is a proposition of the form *Says*(*atc*, *cmd_descend*(*a*, *l*)), where *a* and *l* stand for constants of sorts *Aircraft* and *Level*: they are variables of the metalanguage, not of the language, which is ground. Propositions are grouped by procedural area, such as descent, approach, ground movement, or emergency handling. Since the vocabulary has finitely many symbols and each sort finitely many constants, there are finitely many ground atomic formulas, and $\mathcal{AP}$ collects all of them.

We use four families of propositions: (i) *speech propositions* represent controller and pilot utterances, including clearances, instructions, acknowledgments, and readbacks; (ii) *air-state propositions* represent surveillance-derived aircraft state, such as level, position, and vertical motion; (iii) *surface-state propositions* represent runway and taxiway occupancy;

(iv) *derived propositions* represent facts computed from the observed ones, such as whether a readback matches the corresponding instruction or an authorization remains active across observations, together with the resolution advisories we reconstruct from the aircraft trajectories or from the radio. Some propositions are inertial: trace construction records them as holding until a later observation ends them; others hold only at the observations that report them.

Example 1. A descent instruction is a controller utterance directing an aircraft to descend to an assigned flight level, with the required readback as described in §II-A. The corresponding instruction, readback, and aircraft-state facts have the forms

$Says(atc, cmd_descend(a, \ell)),$
 $Says(pilot, readback_descend(a, \ell)),$
 $descending(a)$

All three belong to the descent area. The spoken forms share a and ℓ , i.e., the same addressed aircraft and assigned level, and differ in the speaking role and in the utterance built, one carrying the controller's instruction and the other the pilot's readback. The state form mentions only the aircraft a , since it records the observed vertical motion of that aircraft; no agent asserts it, as it is obtained from surveillance. $\square$

B. Timed traces

A *timed trace* $\pi \in \Pi$ is a finite sequence of observations

$$\pi = (t_1, L_1), (t_2, L_2), \dots, (t_n, L_n)$$

where $n \geq 1$ is the length of the trace. Each pair (t_i, L_i) is the observation at position i : $t_i \in \mathbb{N}$ is the instant in seconds from the start of the recording, and $L_i \subseteq \mathcal{AP}$ is the set of propositions that hold at that instant.

Explicit timestamps are needed to check procedural deadlines stated in seconds: observations arrive at irregular intervals, so order alone does not determine elapsed time. They are monotone, $t_i \leq t_{i+1}$, so that observations follow their recorded order. Equal timestamps are admitted, since independent sources may report at the same instant.

Example 2. Consider the descent facts from Example 1. Suppose the controller instructs AZA545 to descend to FL120, the crew reads the instruction back eight seconds later, and surveillance then reports the aircraft descending. These observations form the trace π

$(0, \{Says(atc, cmd_descend(AZA545, FL120))\}),$
 $(8, \{Says(pilot, readback_descend(AZA545, FL120))\}),$
 $(20, \{descending(AZA545)\})$

The controller's instruction holds at position 1, at $t_1 = 0$. Its readback holds at position 2, at $t_2 = 8$. The descent reported by surveillance, recorded as the air-state proposition $descending(AZA545)$, holds at position 3, at $t_3 = 20$. Thus, the trace records both the order of the three facts and their unequal elapsed times, all measured in seconds from the start of the recording. $\square$

C. Temporal-logic patterns and ATC formula families

A temporal formula $\varphi \in \Phi$ is built from propositions $p \in \mathcal{AP}$, boolean connectives $\neg, \wedge, \vee, \rightarrow$, and temporal operators over trace positions (see §II-B).

The formulas follow five recurring logical patterns, each capturing a common procedural dependency such as response, compliance, precedence, or exception handling. In the patterns, α and β stand for propositional formulas: boolean combinations of propositions in $\mathcal{AP}$, with no temporal operators.

Several of these patterns use an outer $\mathbf{G}$ to make the requirement apply at every trace position.

A *bounded-response* pattern has the form

$$\mathbf{G}(\alpha \rightarrow \mathbf{F}_{\leq \delta} \beta)$$

requiring every trigger α to be followed by a position satisfying β within δ seconds. The metric operator $\mathbf{F}_{\leq \delta}$ is the only operator that uses timestamps. The other temporal operators reason over the order of trace positions.

An *invariant pattern* has the form

$$\mathbf{G}(\alpha \rightarrow \beta)$$

requiring every position that satisfies α to satisfy β as well.

A *precedence pattern* has the form

$$\neg \alpha \mathbf{W} \beta$$

requiring that no action α occur before an authorization β . Weak until is used because a trace in which the action never occurs should not be reported as a violation merely because no authorization appears.

A *pending-response* pattern has the form

$$\mathbf{G}(\alpha \rightarrow \neg \mathbf{X} \neg (\neg \alpha \mathbf{W} \beta))$$

requiring that, after a trigger α , no further trigger of the same kind appear before the response β . The guard $\neg \mathbf{X} \neg$ is weak next: it coincides with strong next except at the final trace position, where it avoids reporting a violation solely because no next position exists.

Finally, a *bounded-prohibition* pattern has the form

$$\mathbf{G}(\alpha \rightarrow \neg \mathbf{F}_{\leq \delta} \beta)$$

requiring that, whenever α holds, no position satisfying β occur within δ seconds.

A pattern is not itself a formula of Φ : it denotes the set of formulas of that form. With α and β so restricted, no formula of one pattern belongs to another. The formulas of one pattern differ only in the propositional formulas for α and β and, in the bounded patterns, in the time bound δ ; a pattern thus mentions no specific aircraft, value, or utterance.

Specializing these patterns to the ATC domain gives the families of formulas below. The set is extensible: new procedures are codified as further instances of the same five patterns.

Each ATC family selects, within one of the patterns above, the form of each propositional formula and, where required, the operational time bound. A variable of the metalanguage, such as a or ℓ , occurring in more than one form of the same family denotes the same constant in each. The normative sources fix the obligations but not their deadlines,

TABLE I
FAMILIES WITH PATTERN, FORMULA, REPRESENTATIVE FORMS OF α AND β , AND SOURCE.

Family	Pattern	α	β	Source
readback	bounded-response $\mathbf{G}(\alpha \rightarrow \mathbf{F}_{\leq \delta} \beta)$	$Says(atc, cmd_descend(a, \ell))$	$Says(pilot, readback_descend(a, \ell))$	[6, §4.5.7.5] [5, §5.2.1.9.2.1]
state-consistency		$Says(pilot, readback_descend(a, \ell))$	$descending(a)$	[6, §4.5.7.5.1.1]
RA		$ra_climb(a)$	$climbing(a)$	[10, §4.1.4.2]
runway-incursion	invariant $\mathbf{G}(\alpha \rightarrow \beta)$	$on_runway(a, r)$	$line_up_authorized(a, r)$	[6, §7.6.3.1.1.2]
advisory-priority		$ra_active(a)$	$\neg Says(atc, cmd_descend(a, \ell))$	[6, §7.6.3.1.2.1] [6, §15.7.3.2]
clearance-precedence	precedence $\neg \alpha \mathbf{W} \beta$	$landed(a)$	$landing_cleared(a)$	[6, §7.10.2] [6, §6.5.6]
repeated-instructions	pending-response $\mathbf{G}(\alpha \rightarrow \neg \mathbf{X} \neg (\neg \alpha \mathbf{W} \beta))$	$Says(atc, cmd_descend(a, \ell))$	$Says(pilot, readback_descend(a, \ell))$	[6, §4.5.7.5]
advisory-direction	bounded-prohibition $\mathbf{G}(\alpha \rightarrow \neg \mathbf{F}_{\leq \delta} \beta)$	$ra_climb(a)$	$descending(a)$	[10, §4.1.4.4]

except for the five-second resolution-advisory response [10, §4.1.4.2]. The remaining bounds, such as the thirty-second readback window, are operational choices; pilot response-time statistics [21] provide the empirical basis for selecting them. Table I summarizes the families considered in this work. For each family, it reports the pattern, its formula, representative forms of α and β , and the normative source from which the procedural requirement is derived.

The bounded-response rows cover three delayed obligations: *readback* links controller instructions to pilot readbacks, *state-consistency* links acknowledged maneuvers to the observed aircraft state, and *resolution-advisory* links advisories to the expected pilot response.

The invariant rows cover authorization and contingency states: *runway-incursion* checks that runway occupancy has a matching authorization, while *advisory-priority* blocks immediate controller instructions during active advisories except for deferrals such as “when ready”.

The *clearance-precedence* row covers actions that require prior clearance, for example landing or establishing on final approach. The *repeated-instructions* row prevents a second instruction of the same kind before the first one has been read back. The *advisory-direction* row covers bounded prohibitions: after a climb advisory, descent is forbidden within the operational window, and vice versa for descend advisories, unless a later advisory reversing the original one, recorded as a derived proposition, is present.

Example 3. For the instruction of Example 2, the *readback* family contains the formula

$$\mathbf{G}(Says(atc, cmd_descend(AZA545, FL120)) \rightarrow \mathbf{F}_{\leq 30} Says(pilot, readback_descend(AZA545, FL120)))$$

The formula is evaluable over the trace of Example 2. $\square$

D. Evaluation semantics

The satisfaction relation $\models$ defines how formulas are evaluated over timed traces. For a trace $\pi = (t_1, L_1), \dots, (t_n, L_n)$, a position i , and a formula φ , the judgment $\pi, i \models \varphi$ states that φ is satisfied at position i . Each formula is evaluated on the trace, yielding its *verdict*.

Atomic propositions are interpreted by the sets $L_i: \pi, i \models a$ iff $a \in L_i$. Boolean connectives have their standard semantics at position i . The temporal operators used in the formulas are defined as follows:

$$\begin{aligned} \pi, i \models \mathbf{X}\varphi &\Leftrightarrow i < n \text{ and } \pi, i+1 \models \varphi, \\ \pi, i \models \mathbf{G}\varphi &\Leftrightarrow \forall j (i \leq j \leq n \Rightarrow \pi, j \models \varphi), \\ \pi, i \models \varphi \mathbf{U} \psi &\Leftrightarrow \exists k (i \leq k \leq n \wedge \pi, k \models \psi \\ &\quad \wedge \forall j (i \leq j < k \Rightarrow \pi, j \models \varphi)), \\ \pi, i \models \varphi \mathbf{W} \psi &\Leftrightarrow \pi, i \models \varphi \mathbf{U} \psi \\ &\quad \vee \forall j (i \leq j \leq n \Rightarrow \pi, j \models \varphi), \\ \pi, i \models \mathbf{F}_{\leq \delta} \varphi &\Leftrightarrow \exists k (i \leq k \leq n \wedge t_k \leq t_i + \delta \\ &\quad \wedge \pi, k \models \varphi). \end{aligned}$$

The operator $\mathbf{X}$ is strong next: it requires a successor position and is therefore false at the end of the trace. The operator $\mathbf{G}$ is universal over the suffix starting at i , so a single later counterexample falsifies it. Until, $\mathbf{U}$, requires a witness position where ψ holds, with φ holding at every intervening position. Weak until, $\mathbf{W}$, admits the same witnessed case, but also the case in which ψ never occurs and φ holds until the trace ends. The bounded finally operator $\mathbf{F}_{\leq \delta}$ is the metric part of the semantics: its witness must occur no later than δ seconds after the current timestamp. $\mathbf{X}$ and $\mathbf{U}$ enter only through $\neg \mathbf{X} \neg$ and the definition of $\mathbf{W}$, respectively.

Every formula is evaluated at the first position of the trace, so its outermost operator ranges over all positions. We call the procedure that applies $\models$ to a trace the *monitor*: it runs either offline, over a completed recording, or online, on a still-growing trace, issuing a verdict as new observations arrive. Offline, over a recording that extends past the deadlines of the obligations triggered in it, the verdict is two-valued: by the end every obligation has been met or missed, so each formula is definitively true or false.

While the trace is still open, a formula may not yet have a definitive verdict: a time-bounded obligation triggered at position i whose response has not yet been observed is false under $\models$, since no witness exists yet, although one may still arrive by $t_i + \delta$. The monitor therefore evaluates each formula under the three-valued semantics of runtime verification [14], in its finite-trace form [16]: on an open trace, a formula is true if every finite timed trace extending it, including the trace as it

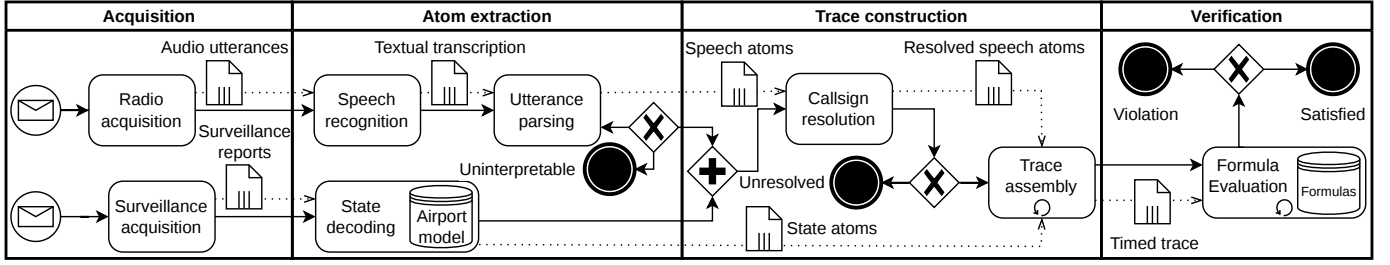


Fig. 1. Framework architecture and workflow.

stands, satisfies the formula, false if every such extension violates it, and *inconclusive* otherwise. The pending obligation is thus inconclusive until the trace extends beyond $t_i + \delta$, since an extension may still supply the witness in time; past that instant no extension can meet the deadline, and the formula is false. Operationally, the monitor reports inconclusive obligations as not-yet-violated, so that no violation is announced before the response is due.

For formulas under $\mathbf{G}$, the verdict can become true only once the recording ends, since a further observation may falsify their bodies. From then on, the only extension is the trace itself, and the three-valued semantics coincides with $\models$.

When a formula is violated, the monitor also returns the position responsible and the propositions holding there. For globally scoped formulas, this is the earliest failing position; for precedence formulas, the action position with no prior authorization. This evidence links each reported violation to the observed facts underlying it, making the verdict explainable.

Verdicts are relative to the captured trace. A reported missing authorization may indicate either that no authorization was issued or that it was issued before the trace begins. Distinguishing between these cases requires observations outside the available trace.

Example 4. The formula of Example 3 is evaluated on the trace of Example 2. At position 1, the instruction proposition holds, so the implication requires a matching readback within thirty seconds. The readback occurs at $t_2 = 8$, within the bound from $t_1 = 0$. At positions 2 and 3 the instruction proposition is absent, so the implication is true there. Since the outer $\mathbf{G}$ ranges over all positions, the formula is satisfied.

If the same trace were still open just after the instruction, with no readback observed yet, the formula would be inconclusive, reported as not-yet-violated, until the trace extended beyond $t_1 + 30$; only then would a violation be reported. $\square$

V. VERIFICATION FRAMEWORK

The model in §IV assumes that atoms and a timed trace are given. The architecture in Fig. 1 adds four stages: *acquisition* collects heterogeneous operational data, *atom extraction* converts them into the formal vocabulary of §IV-A, *trace construction* integrates the atoms into a unified timed trace, and *verification* evaluates the temporal formulas over it. The figure shows offline verification over a completed recording. In the online configuration, the same stages incrementally process a growing trace.

A. Acquisition and atom extraction

The first two stages transform radio and surveillance signals into ground atoms of $\mathcal{AP}$ (see §IV-A).

Acquisition receives the two raw input signals, each on its own timeline. Radio acquisition captures the voice exchanged between controllers and pilots and outputs the *audio utterances*. Surveillance acquisition collects the aircraft-state messages broadcast through ADS-B and outputs the stream of *surveillance reports*. The acquisition stage is modular and can incorporate additional state streams, such as primary radar or aircraft telemetry.

Atom extraction turns the acquired signals into ground atoms: speech atoms are derived from the audio utterances, and state atoms from the surveillance reports. Speech recognition transcribes each utterance to text; it is a replaceable front end [22], [23]. Utterance parsing uses a language model to map each transcription to the speech-atom schema vocabulary. Unmatched transcriptions are discarded as *uninterpretable*.

State decoding outputs two classes of *state atoms*. *Air-state atoms* record aircraft level, position, and motion from surveillance reports. *Surface-state atoms* relate reported positions to the airport layout to determine runway and taxiway occupancy.

B. Trace construction

Trace construction converts speech and state atoms into the timed trace π introduced in §IV-B. Call sign resolution associates each speech atom with an aircraft and normalizes transcribed or abbreviated call signs against surveillance identities [22]. An unconfirmed call sign is retained only when used consistently by both controller and pilot. Unresolved speech atoms are discarded. Conversely, state atoms need no resolution: surveillance already tags them with the aircraft.

Trace assembly orders the resolved speech atoms and the state atoms by the time at which the corresponding events occurred, so speech-recognition and decoding delays do not affect the timestamps used by the temporal constraints. The trace is incrementally extended with new and derived atoms of $\mathcal{AP}$: readbacks are matched with the instructions they acknowledge, and clearances are propagated until the action is completed or revoked. Offline, the trace closes when the recording ends. Online, it remains open.

C. Verification

Formula evaluation receives the timed trace π and the temporal formulas of Φ (§IV-C). For each family, the monitor

selects the formulas whose propositions mention the aircraft and values occurring in π , and evaluates them according to §IV-D.

A formula is evaluated over the timed trace by structural recursion, with the metric operator comparing timestamps against its deadline. The monitor directly implements the satisfaction relation $\models$, as in monitors for metric parametric specifications [15]. Since the outermost operator ranges over all positions and each temporal operator scans at most a suffix of the trace, restricted to its temporal window when the operator is metric, evaluating a formula has worst-case quadratic complexity in the trace length.

In the online configuration, evaluation is also performed between observations by extending the trace with a virtual empty observation at the current instant. This allows elapsed deadlines to be detected without waiting for a subsequent event. A violation is reported only if a formula is false under both $\models$ and a second evaluation in which unexpired metric obligations are treated as satisfied. Formulas for which the two evaluations disagree remain inconclusive.

Verification produces one verdict per formula, two-valued once the recording ends. For an open trace, formulas the observations do not yet decide carry the inconclusive verdict; obligations whose deadlines have not yet expired are reported as not-yet-violated. Any formula that evaluates to false leads to the *violation* outcome; consistent with the explainability mechanism of §IV-D, each violation identifies the trace position responsible and the propositions holding there. If no formula is violated, the outcome is *satisfied*.

VI. EVALUATION

A. Implementation and experimental setup

We implement the architecture of Fig. 1 as follows. The acquisition stage can rely on standard radio and ADS-B receiver solutions [24], [25]. In our evaluation, we bypass live acquisition and provide recorded inputs directly (see below). Speech recognition uses faster-whisper [26], a CTranslate2 reimplement of Whisper [27], with a model fine-tuned for air traffic control speech [28]. The utterance parser is an LLM performing structured extraction. Inference uses llama.cpp [29], which constrains the output to a JSON-schema grammar; decoding is greedy (temperature 0) with a fixed seed.

State decoding and the airport model are implemented in Python. The airport layout is derived from OpenStreetMap aeroway data [30] and processed with the Shapely geometry library [31]. Callsign resolution, trace assembly, and formula evaluation are implemented as bespoke Python modules.

For utterance parsing, the online configuration uses qwen2.5-7b [32] for low latency, while offline analysis uses qwen3.8-27b [33]. We evaluate both thinking and no-thinking modes where available. The former generates intermediate reasoning and the latter answers directly.

All experiments run locally on a single workstation (NVIDIA RTX 4090, AMD Ryzen 9 7900, 64 GB RAM, Fedora Linux 44). LLM and Automatic Speech Recognition (ASR) inference run on the GPU, while the remaining framework components run on the CPU.

B. Datasets and vocabulary coverage

We feed the pipeline with two public corpora of recorded traffic, ATCO2 [34] and TartanAviation [35]. To our knowledge, they are the only publicly available corpora that pair controller–pilot communications with recoverable aircraft state, reflecting the limited availability of ATC recordings due to legal and regulatory constraints [34].

ATCO2 contains communications without aircraft state. Its openly released one-hour test set provides manually verified transcriptions from seven airports across Europe and Australia, covering tower, ground, approach, and radar operations. We retrieve the corresponding decoded ADS-B state vectors from the OpenSky Network [36], for the relevant regions and recording intervals, and align them with the radio communications. TartanAviation provides synchronized radio communications and ADS-B trajectories from a towered airport (Allegheny County Airport, KAGC [37]) and a non-towered airport (Pittsburgh–Butler Airport, KBTP [38]). Since the monitored formulas require controller-issued instructions, we restrict the analysis to KAGC. We form the ground-truth set by annotating in full the busiest hour of each of three days of 2022 KAGC traffic, blind to the monitor’s output and using both audio and surveillance. The set contains every procedural violation present in these three hours: 12 violations, mainly involving readbacks. ATCO2 transcriptions are passed directly to the parser, whereas TartanAviation audio, for which no manual transcriptions are available, is transcribed by the speech-recognition front end.

We measure vocabulary coverage as the fraction of atom schemas exercised by the corpora, counting a schema when at least one instance appears in the traces. It is computed over routine procedures only, since exceptional events such as emergencies, runway incursions, and collision scenarios are absent from the corpora. Across both corpora, 37.6% of the routine schemas are exercised.

C. Evaluation protocol

The evaluation assesses implementation correctness on synthetic situations and agreement with expert judgment on real traffic. We additionally use the case studies (§III) as qualitative validation.

Implementation correctness. We test the monitor implementation on synthetic situations whose expected verdicts are known by construction. Safety-critical configurations are rare in recorded traffic and cannot be deliberately induced in live traffic; we therefore build each situation from a real corpus event satisfying the trigger of a monitored formula, and apply a controlled perturbation like removal of the required response, a deadline exceeded by half a second, or action without the required clearance. The advisory families are exercised separately through the Überlingen reconstruction of §VI-E.

Agreement with expert judgment. We evaluate the monitor on real traffic against the ground-truth set of §VI-B. Verdicts are scored per event: multiple alerts generated for the same situation across successive surveillance samples count as a single detection.

D. Performance

The 255 events satisfying formula triggers yield 1,495 synthetic situations, 664 compliant and 831 violating, exercising 50 forms across five of the eight families of Table I. The monitor returns the expected verdict in every case. Table II reports precision, recall, and F1 of the pipeline of Fig. 1 when using the qwen3.8-27b, gemma4-26b-a4b [39], and qwen2.5-7b as parsers, scored against the three-hour ground-truth set. Precision is the fraction of reported violations that are correct, recall is the fraction of true violations that are reported, and F1 is their harmonic mean. The no-thinking qwen3.8-27b achieves the highest F1 and recall.

TABLE II
PERFORMANCE BY PARSER MODEL

Parser	Precision	Recall	F1
qwen3.8-27b, no-thinking (NT)	0.79	0.92	0.85
qwen3.8-27b, thinking (T)	0.78	0.58	0.67
gemma4-26b-a4b, no-thinking (NT)	0.40	0.33	0.36
gemma4-26b-a4b, thinking (T)	0.67	0.33	0.44
qwen2.5-7b, no-thinking	0.57	0.67	0.62

Table III summarizes the computational cost of each stage using the minimum, first quartile (Q_1), median, third quartile (Q_3), and maximum over the measured samples. Each stage is timed separately, without concurrent execution of the other stages. Here, *Large LLM* denotes the no-thinking qwen3.8-27b and *Small LLM* the qwen2.5-7b of Table II. Speech recognition is reported as inverse real-time factor (RTFx), defined as $t_{\text{audio}}/t_{\text{processing}}$ [40]. Speech recognition consistently operates faster than real time, with limited variability across recordings. Parsing accounts for the largest share of the computational cost, whereas formula evaluation contributes a smaller share despite its wider distribution.

TABLE III
PER-STAGE PROCESSING COST.

Stage	Min	Q_1	Median	Q_3	Max Unit
Speech recognition	32.1	41.4	45.1	47.5	48.7 RTFx
Atom parse (Small LLM)	0.266	0.282	0.296	0.339	1.46 s/utt.
Atom parse (Large LLM)	1.16	1.34	1.58	1.75	4.94 s/utt.
Formula evaluation	0.000003	0.037	0.119	0.245	2.54 s/event

Fig. 2 shows the per-utterance parse latency CDF. The no-thinking distributions are narrow, with medians of about 0.3 s (qwen2.5-7b), 0.46 s (gemma4-26b-a4b), and 1.6 s (qwen3.8-27b). The thinking distributions have medians of about 3.9 s (gemma4-26b-a4b) and 38 s (qwen3.8-27b), with tails up to 134 s and 83 s, respectively.

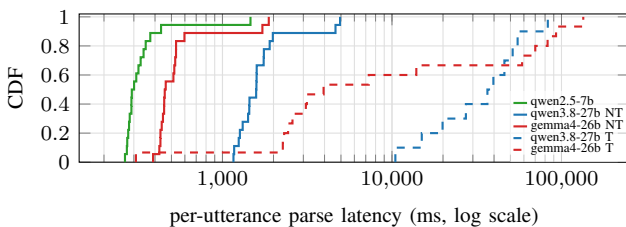


Fig. 2. Per-utterance parse latency.

Fig. 3 relates accuracy to latency. The no-thinking qwen3.8-27b sits at the highest F1 at moderate latency, the qwen2.5-7b at the lowest latency with reduced F1, and the thinking configurations at much higher latency and lower F1. The 7B therefore suits the online monitor, while the no-thinking qwen3.8-27b serves as the offline reference parser.

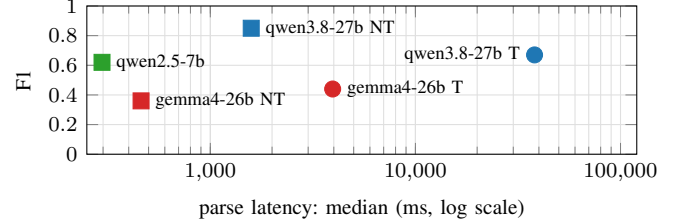


Fig. 3. Accuracy versus parse latency.

E. Documented accidents

The two accidents introduced in §III are replayed through the monitor, with their radio and surveillance state reconstructed from the official investigation reports. The cases lie outside the corpora from §VI-B and as such carry no accuracy metrics. They test the monitor on rare safety-critical situations, with the official reports as reference: in each case we check whether the firings agree, in content and timing, with the events the investigators documented, and whether the detection instants leave a margin usable for a real-time alert.

We report detection instants on the wall clock of the investigation reports. The monitor itself runs on session-relative time and places each violation at the instant its formula is decided over the reconstructed trace, once the deadline following the triggering observation has elapsed.

Überlingen. On the reconstructed radio, avionics, and surveillance state, the monitor flags the same sequence the BFU documented [19]. At 21:34:56 the Tupolev's collision-avoidance system commands a climb while surveillance shows it descending under the controller's standing instruction; *advisory-direction* and *resolution-advisory* are confirmed at 21:35:01, once the reaction window elapses, and the monitor also flags that the advisory is never reported to the controller, as ICAO requires of a crew deviating in response to one [6, §15.7.3.3, Note]. At 21:35:03 a second descent is issued while the advisory is active, firing *advisory-priority*. Each of these firings lies in the avionics and surveillance state, not on the radio, and those at 21:35:01 and 21:35:03 precede the 21:35:32 collision by thirty-one and twenty-nine seconds, respectively (Fig. 4). Thus the monitor reproduces the precedence reasoning the BFU documented.

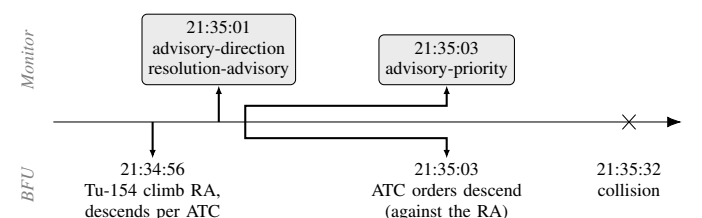


Fig. 4. Überlingen: monitor detections and BFU-documented events (UTC).

Comair Flight 5191 (Comair). On the reconstructed radio and position track, the monitor flags the sequence the NTSB documented [20]. At 06:05:15 the first officer reports ready with the wrong call sign, “Comair one twenty one” for one ninety one, firing the *callsign-blunder* phraseology check. The authorization on record is runway 22, named in the taxi clearance; the takeoff clearance names no runway. *takeoff-clearance-no-runway* fires at 06:05:18, six seconds before the hold-short crossing. The formula encodes the current requirement of a runway designator in the takeoff clearance [6, §12.3.4.11]; this requirement was not in force at the time of the accident [20]. As the track crosses onto the surface the monitor registers runway entry at 06:06:00 and fires *runway-incursion* and its precursor check *hold-short-bust*. This multi-source detection compares the runway authorized on the radio with the runway the aircraft actually occupies. Both firings are decided once the reconstruction confirms surface entry at 06:06:00, and so trail the documented hold-short crossing at 06:05:24; that earlier instant is not resolved in this reconstruction. *wrong-runway-takeoff*, a specialization of the *runway-incursion* family, fires at 06:06:16, as ground speed enters the takeoff-roll regime, nineteen seconds before the 06:06:35 impact in Fig. 5. Thus the monitor identifies the wrong-runway takeoff while the roll is still in progress.

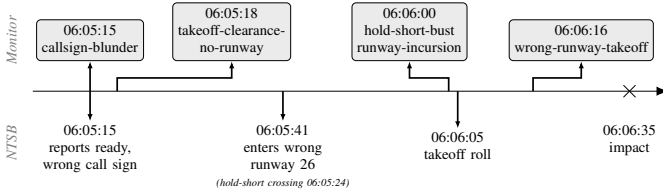


Fig. 5. Comair: monitor detections and NTSB-documented events (EDT).

VII. DISCUSSION

Performance figures. The F1 of 0.85 on real traffic reflects the whole chain that recovers observations from speech: the conditions of the radio channel, the speech-recognition model, and the parser. Hold the formulas and the recordings fixed, and the score moves with the parser alone (Table II); each figure also carries the quality of the transcriptions it reads. Advances in speech recognition and language models therefore carry straight over to monitoring, with the formulas and their semantics untouched. The computational requirements of the system are compatible with deployment on standard workstation hardware. The full pipeline executes on a single machine, with parsing as the dominant computational cost. The lightweight parser, *qwen2.5-7b*, operates with sub-second latency and could support real-time monitoring; the slower but more accurate parser, *qwen3.8-27b*, suits offline analysis of recorded operational data.

Operational implications. The evaluation shows that the four monitoring requirements can be addressed jointly: operational traffic exercises grounding ($\mathbf{R}_1$), the accident reconstructions combine radio with surveillance or onboard observations ($\mathbf{R}_2$), and both require temporal ordering and deadlines ($\mathbf{R}_3$); every reported violation remains linked to the failed obligation and its supporting observations ($\mathbf{R}_4$). Beyond this requirement

coverage, the results support the three uses outlined in §I. For real-time assistance, early detection can support risk mitigation by identifying deviations while the operational state is still evolving, as in the reconstructed accidents. Operationally, such early detection provides a basis for risk mitigation. While the aircraft’s dynamics and controllability remain beyond the monitor’s purview, identifying a deviation while the operational state is still evolving creates a window for potential intervention, particularly when a loss of situational awareness is a primary trigger or a contributing factor. For post-operation review, each violation is linked to the failed obligation and supporting observations, providing an immediate procedural account before the completion of a formal investigation. The same evidence can support training by providing explicit feedback on procedural deviations in recorded sessions, where the more accurate offline parser can be used without real-time latency constraints. Nevertheless, operational deployment requires human-in-the-loop evaluation to study how the tool integrates with existing ATC workflows and systems, and how its alerts, including false positives, should be presented and calibrated under controller workload.

VIII. RELATED WORK

Existing work addresses complementary parts of spoken ATC procedure monitoring. Table IV summarizes the closest representative approaches with respect to the four requirements introduced in §I. A requirement is met (●) when the approach represents it and uses it in the monitoring decision, partially met (◐) when it is supported only in a restricted or indirect form, and not addressed (○) when absent from the method’s inputs, model, or verdict.

On the surveillance side, Reynolds et al. [41] compare observed tracks with trajectories expected from active clearances. Trajectory timing provides a restricted form of $\mathbf{R}_3$, while radio exchanges are not an input ($\mathbf{R}_1$, $\mathbf{R}_2$), and alerts identify the aircraft but not the breached obligation or supporting observations ($\mathbf{R}_4$). Thus, $\mathbf{R}_3$ is partially met, while the other requirements are not addressed.

On the voice side, a common ontology [47] structures transcribed controller and pilot utterances by command type, call sign, and values. Building on it, Helmke et al. [23] compare pilot readbacks with controller instructions, linking utterances to the addressed aircraft and instructed values ($\mathbf{R}_1$). Surveillance restricts candidate call signs during recognition but does not enter the compliance check, which stops at the readback rather than extending to the maneuver actually flown ($\mathbf{R}_2$); timing is limited to a response timeout ($\mathbf{R}_3$); and the output flags discrepant values rather than the underlying obligation ($\mathbf{R}_4$). $\mathbf{R}_1$ is met; $\mathbf{R}_2$, $\mathbf{R}_3$, and $\mathbf{R}_4$ are partially met.

TABLE IV
REQUIREMENT COVERAGE: RELATED WORK VS. THIS WORK.

	[41]	[23]	[42]	[43]	[44], [45]	[46]	This work
$\mathbf{R}_1$ grounding	○	●	●	○	○	●	●
$\mathbf{R}_2$ integration	○	◐	●	○	○	○	●
$\mathbf{R}_3$ timeliness	◐	◐	◐	●	●	○	●
$\mathbf{R}_4$ verdicts	○	◐	◐	◐	◐	◐	●

● met ◐ partially met ○ not addressed

Combining voice and surveillance, Lin et al. [42] ground spoken instructions and evaluate them together with the ADS-B track, meeting $\mathbf{R}_1$ and $\mathbf{R}_2$ in a single compliance decision. Their pipeline implements three checks: repetition, conflict detection, and conformance with the instructed maneuver, the latter measured against a trajectory predicted from the instruction. Repetition and conformance use fixed validity windows, but instructions do not supersede one another, providing only partial $\mathbf{R}_3$. Warnings identify the triggered check, recognized text, aircraft, and time, but not the breached obligation, partially addressing $\mathbf{R}_4$.

Among RV monitors (§II-B), Lima et al. [43] explain each verdict with a proof at the formula level. Their specification language expresses deadlines and ordering, meeting $\mathbf{R}_3$. $\mathbf{R}_4$ is only partially met, since the explanation remains at the formula level rather than linking the verdict to the underlying operational observations. The trace is assumed given as logical facts, so $\mathbf{R}_1$ and $\mathbf{R}_2$ are not addressed.

In neighboring transportation domains, Maierhofer et al. [44] formalize interstate driving norms in MTL and evaluate them over recorded trajectories, while Krasowski et al. [45] do the same for the International Regulations for Preventing Collisions at Sea (COLREGS). Metric operators encode rule time bounds ($\mathbf{R}_3$), and violations identify the affected rule, partially addressing $\mathbf{R}_4$. Because the trajectory is the single observation source, $\mathbf{R}_1$ and $\mathbf{R}_2$ are not addressed.

Interpretation can also be delegated to a language model. Lall et al. [46] prompt models to judge protocol compliance in maritime training transcripts. Utterances are linked to checklist items rather than identified entities or values, providing only partial $\mathbf{R}_1$; the transcript is the sole input, so $\mathbf{R}_2$ is not addressed. Time selects the excerpts presented to the model without a procedural deadline, so $\mathbf{R}_3$ is not addressed. Correctness is assessed against expert annotation; reported items include an instant and quoted span, partially addressing $\mathbf{R}_4$.

In summary, existing approaches cover different subsets of the requirements, while our framework addresses them jointly.

IX. CONCLUSION AND FUTURE WORK

We presented an executable and explainable framework for monitoring spoken ATC procedures. It grounds communications in the aircraft and values they concern, integrates them with surveillance and onboard observations in a timed trace, and evaluates ICAO-derived obligations with explicit deadlines and precedence. Violations are linked to the failed obligation and supporting observations.

Future work will extend the set of monitored formulas to cover local procedural conventions, validate the framework on broader traffic corpora, and assess its integration into operational ATC. We also plan to evaluate the approach in other transportation domains.

REFERENCES

- [1] International Civil Aviation Organization, "ICAO Safety Report, 2025 Edition: State of Global Aviation Safety," International Civil Aviation Organization (ICAO), Montréal, Canada, Tech. Rep., 2025, data year 2024. [Online]. Available: https://www.icao.int/sites/default/files/sp-files/safety/Documents/ICAO_SR_2025.pdf
- [2] EUROCONTROL Performance Review Commission, "Performance Review Report (PRR) 2024: An Assessment of Air Traffic Management in Europe," EUROCONTROL, Brussels, Belgium, Tech. Rep., Mar. 2025. [Online]. Available: <https://www.eurocontrol.int/sites/default/files/2025-03/eurocontrol-performance-review-report-2024.pdf>
- [3] European Union Aviation Safety Agency, "Annual Safety Review 2025," European Union Aviation Safety Agency (EASA), Cologne, Germany, Tech. Rep., Aug. 2025, published 26 August 2025; ISBN 978-92-9210-288-3. [Online]. Available: <https://www.easa.europa.eu/en/document-library/general-publications/annual-safety-review-2025>
- [4] International Civil Aviation Organization, *Air Traffic Services*, 15th ed., Montréal, Quebec, Canada, July 2018, Annex 11 to the Convention on International Civil Aviation, AN 11.
- [5] —, *Aeronautical Telecommunications – Volume II: Communication Procedures including those with PANS status*, 7th ed., Montréal, Quebec, Canada, July 2016, Annex 10 to the Convention on International Civil Aviation, AN 10-2.
- [6] —, *Procedures for Air Navigation Services – Air Traffic Management (PANS-ATM)*, 16th ed., Montréal, Quebec, Canada, November 2016, Doc 4444, AN/501.
- [7] —, *Aeronautical Telecommunications – Volume IV: Surveillance and Collision Avoidance Systems*, 5th ed., Montréal, Quebec, Canada, July 2014, Annex 10 to the Convention on International Civil Aviation, AN 10-4.
- [8] *EUROCONTROL Specification for Short Term Conflict Alert*, 1st ed., EUROCONTROL, Nov. 2007, EUROCONTROL-SPEC-0108.
- [9] International Civil Aviation Organization, *Advanced Surface Movement Guidance and Control Systems (A-SMGCS) Manual*, 1st ed., Montréal, Quebec, Canada, 2004, Doc 9830.
- [10] —, *Airborne Collision Avoidance System (ACAS) Manual*, 2nd ed., Montréal, Quebec, Canada, 2012, Doc 9863, AN/461.
- [11] A. Pnueli, "The temporal logic of programs," in *Proceedings of the 18th Annual Symposium on Foundations of Computer Science (FOCS)*. IEEE Computer Society, 1977, pp. 46–57.
- [12] G. De Giacomo and M. Y. Vardi, "Linear temporal logic and linear dynamic logic on finite traces," in *Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence (IJCAI)*. Beijing, China: AAAI Press, 2013, pp. 854–860.
- [13] R. Koymans, "Specifying real-time properties with metric temporal logic," *Real-Time Systems*, vol. 2, no. 4, pp. 255–299, 1990.
- [14] A. Bauer, M. Leucker, and C. Schallhart, "Runtime verification for LTL and TLTL," *ACM Transactions on Software Engineering and Methodology*, vol. 20, no. 4, pp. 1–64, 2011.
- [15] D. Basin, F. Klaedtke, S. Müller, and E. Zălinescu, "Monitoring metric first-order temporal properties," *Journal of the ACM*, vol. 62, no. 2, pp. 15:1–15:45, 2015.
- [16] H. Kallwies, M. Leucker, and C. Sánchez, "General anticipatory monitoring for temporal logics on finite traces," in *Proceedings of the 23rd International Conference on Runtime Verification (RV)*, ser. Lecture Notes in Computer Science, vol. 14245. Springer, 2023, pp. 106–125.
- [17] T. Reinbacher, K. Y. Rozier, and J. Schumann, "Temporal-logic based runtime observer pairs for system health management of real-time systems," in *Proceedings of the 20th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS)*, ser. Lecture Notes in Computer Science, vol. 8413. Springer, 2014, pp. 357–372.
- [18] P. Moosbrugger, K. Y. Rozier, and J. Schumann, "R2U2: Monitoring and diagnosis of security threats for unmanned aerial systems," *Formal Methods in System Design*, vol. 51, no. 1, pp. 31–61, 2017.
- [19] Bundesstelle für Flugunfalluntersuchung, "Investigation report AX001-1-2/02," Bundesstelle für Flugunfalluntersuchung (BFU), Braunschweig, Germany, Tech. Rep., May 2004, Überlingen mid-air collision, 1 July 2002; English translation, German version authentic. [Online]. Available: https://www.bfu-web.de/EN/Publications/FinalReports/2002/Report_02_AX001-1-2_Ueberlingen_Report.pdf
- [20] National Transportation Safety Board, "Attempted takeoff from wrong runway, comair flight 5191, bombardier CL-600-2B19, N431CA, lexington, kentucky, august 27, 2006," National Transportation Safety Board, Washington, DC, USA, Tech. Rep. NTSB/AAR-07/05, 2007, adopted 26 July 2007. [Online]. Available: <https://www.ntsb.gov/investigations/AccidentReports/Reports/AAR0705.pdf>
- [21] M. Lutz, G. B. Chatterji, and H. R. Idris, "Characterization of response times based on voice communication and traffic surveillance data," in *Proceedings of the AIAA AVIATION 2022 Forum*, Chicago, IL, USA, 2022.
- [22] J. Zuluaga-Gomez, A. Prasad, I. Nigmatulina, P. Motlicek, and M. Kleinert, "A virtual simulation-pilot agent for training of air traffic

- controllers,” *Aerospace*, vol. 10, no. 5, 2023. [Online]. Available: <https://www.mdpi.com/2226-4310/10/5/490>
- [23] H. Helmke, K. Ondřej, S. Shetty, H. Arilfusson, T. S. Simiganoschi, M. Kleinert, O. Ohneiser, H. Ehr, J. Zuluaga-Gomez, and P. Smrž, “Readback error detection by automatic speech recognition and understanding – results of HAAWAI project for Isavia’s enroute airspace,” in *Proceedings of the 12th SESAR Innovation Days (SID)*, Budapest, Hungary, 2022.
- [24] charlie foxtrot, T. Lemiech, and M. H. Wong, “Rtlds-airband: Multichannel am/nfm demodulator,” <https://github.com/rtl-airband/RTLSDR-Airband>, 2025, accessed: 2026-08-21.
- [25] S. Sanfilippo, “dump1090: A simple mode s decoder for rtlds devices,” <https://github.com/antirez/dump1090>, 2012, accessed: 2026-08-21.
- [26] SYSTRAN, “faster-whisper: Faster Whisper transcription with CTranslate2,” <https://github.com/SYSTRAN/faster-whisper>, 2023, accessed 18 June 2026.
- [27] A. Radford, J. W. Kim, T. Xu, G. Brockman, C. McLeavey, and I. Sutskever, “Robust speech recognition via large-scale weak supervision,” in *Proceedings of the 40th International Conference on Machine Learning (ICML)*, ser. Proceedings of Machine Learning Research, vol. 202. PMLR, 2023, pp. 28 492–28 518.
- [28] jacktol, “whisper-medium.en fine-tuned for air traffic control,” <https://huggingface.co/jacktol/whisper-medium.en-fine-tuned-for-ATC>, 2024, hugging Face model; Whisper medium.en fine-tuned on ATCO2 and UWB-ATCC, WER 15.08%.
- [29] G. Gerganov *et al.*, “llama.cpp: LLM inference in C/C++,” <https://github.com/ggml-org/llama.cpp>, 2023, accessed 18 June 2026.
- [30] OpenStreetMap contributors, “OpenStreetMap,” <https://www.openstreetmap.org>, 2026, accessed 22 August 2026.
- [31] S. Gillies *et al.*, “Shapely: Manipulation and analysis of geometric objects,” <https://github.com/shapely/shapely>, 2007, accessed 22 August 2026.
- [32] Qwen Team, “Qwen2.5 technical report,” *arXiv preprint arXiv:2412.15115*, 2025.
- [33] —, “Qwen3.8,” <https://github.com/QwenLM/Qwen3.8>, 2026, alibaba Group; accessed 21 August 2026.
- [34] J. Zuluaga-Gomez, K. Veselý, I. Szöke, A. Blatt, P. Motlicek, M. Kocour, M. Rigault, K. Choukri, A. Prasad, S. S. Sarfjoo, I. Nigmatulina, C. Cevenini, P. Kolčárek, A. Tart, J. Černocký, and D. Klakow, “ATCO2 corpus: A large-scale dataset for research on automatic speech recognition and natural language understanding of air traffic control communications,” *Journal of Data-centric Machine Learning Research (DMLR)*, vol. 2, pp. 1–45, 2024, also available as arXiv:2211.04054.
- [35] J. Patrikar, J. Dantas, B. Moon, M. Hamidi, S. Ghosh, N. Keetha, I. Higgins, A. Chandak, T. Yoneyama, and S. Scherer, “Image, speech, and ADS-B trajectory datasets for terminal airspace operations,” *Scientific Data*, vol. 12, 2025, preprint arXiv:2403.03372, “TartanAviation”.
- [36] M. Schäfer, M. Strohmeier, V. Lenders, I. Martinovic, and M. Wilhelm, “Bringing up OpenSky: A large-scale ADS-B sensor network for research,” in *Proceedings of the 13th International Symposium on Information Processing in Sensor Networks (IPSN)*. IEEE, 2014, pp. 83–94.
- [37] SkyVector, “Allegheny county airport (KAGC),” 2026, official FAA airport data. [Online]. Available: <https://skyvector.com/airport/AGC/Allegheny-County-Airport>
- [38] —, “Pittsburgh/butler regional airport (KBTP),” 2026, official FAA airport data. [Online]. Available: <https://skyvector.com/airport/BTP/Pittsburgh-Butler-Regional-Airport>
- [39] Gemma Team, Google DeepMind, “Gemma 4,” https://ai.google.dev/gemma/docs/core/model_card_4, 2026, accessed 18 June 2026.
- [40] V. Srivastav, S. Zheng, E. Bezzam, E. Le Bihan, N. R. Koluguri, P. Żelasko, S. Majumdar, A. Moumen, and S. Gandhi, “Open ASR leaderboard: Towards reproducible and transparent multilingual and long-form speech recognition evaluation,” *arXiv preprint arXiv:2510.06961*, 2025.
- [41] T. G. Reynolds and R. J. Hansman, “Conformance monitoring approaches in current and future air traffic control environments,” in *Proceedings of the 21st Digital Avionics Systems Conference (DASC)*, vol. 2. Irvine, CA, USA: IEEE, 2002, pp. 7C1–1–7C1–12.
- [42] Y. Lin, L. Deng, Z. Chen, X. Wu, J. Zhang, and B. Yang, “A real-time ATC safety monitoring framework using a deep learning approach,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 21, no. 11, pp. 4572–4581, 2020.
- [43] L. Lima, A. Herasimau, M. Raszkyk, D. Traytel, and S. Yuan, “Explainable online monitoring of metric temporal logic,” in *Proceedings of the 29th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS)*, ser. Lecture Notes in Computer Science, vol. 13994. Springer, 2023, pp. 473–491.
- [44] S. Maierhofer, A.-K. Rettinger, E. C. Mayer, and M. Althoff, “Formalization of interstate traffic rules in temporal logic,” in *Proceedings of the IEEE Intelligent Vehicles Symposium (IV)*, 2020, pp. 752–759.
- [45] H. Krasowski and M. Althoff, “Temporal logic formalization of marine traffic rules,” in *Proceedings of the IEEE Intelligent Vehicles Symposium (IV)*, 2021, pp. 186–192.
- [46] V. Lall and Y. Liu, “Prompt-and-check: Using large language models to evaluate communication protocol compliance in simulation-based training,” in *Proceedings of the 2025 International Conference on Cyberworlds (CW)*. IEEE, 2025, pp. 358–361.
- [47] H. Helmke, M. Slotty, M. Poiger, D. Ferrer Herrero, O. Ohneiser, N. Vink, A. Cerna, P. Hartikainen, B. Josefsson, D. Langr, R. Garcia Lasheras, G. Marin, O. G. Mevatne, S. Moos, M. N. Nilsson, and M. Boyero Perez, “Ontology for transcription of ATC speech commands of SESAR 2020 solution PJ.16-04,” in *Proceedings of the 37th IEEE/AIAA Digital Avionics Systems Conference (DASC)*. London, UK: IEEE, 2018, pp. 1–10.



Roberto Luvini is an MSc student in Computer Engineering (Software Platforms and Cybersecurity) at the University of Genoa, Italy. His research interests include cybersecurity and air traffic operations. He has been admitted to a PhD programme in Cybersecurity and Artificial Intelligence at the University School of Advanced Defense Studies (CASD), where he will continue his research in this area.



Giacomo Longo is a Contract Researcher at the University School of Advanced Defense Studies (CASD), Rome, Italy. He holds a Ph.D. in Artificial Intelligence for Security from Sapienza University of Rome and a Master’s degree in Computer Engineering from the University of Genoa. His research interests include the cybersecurity of cyber-physical and transportation systems (maritime, avionic, and industrial control systems), wireless security, and digital forensics.



Alessandro Armando is a Professor of Computer Security at the University of Genoa. His research focuses on the application of automated reasoning techniques to the security of distributed systems. He has coordinated and led research teams in several national and European projects. He formerly served as Director of the CINI National Cybersecurity Laboratory and currently chairs the Scientific Committee of the SERICS Foundation.



Enrico Russo received his M.Sc. in Computer Science and Ph.D. in Computer Science and Systems Engineering at the University of Genoa in 2001 and 2021. He joined the University School of Advanced Defense Studies (CASD), Rome, Italy, as an Assistant Professor in 2026. His research focuses on the cybersecurity of cyber-physical systems, particularly in the transportation domain, leveraging cyber range capabilities for security testing and assessment.